

표정인식을 통한 이모티콘 생성기



분과 B (데이터베이스·그래픽스 및 비전)
팀 이모티콘 만들조

201524536 이연걸

201524515 윤석현

201524455 문성욱

지도교수 감진규

목 차

1. 과제 개요	1
1.1. 대상 문제	1
1.2. 요구 사항	1
1.3. 과제 목표	2
2. 과제 배경 지식	3
2.1. 기술소개	3
2.1.1. 딥러닝	3
2.1.2. 컴퓨터 비전	3
2.2. 개발 환경	3
2.3. 주요 모듈	3
2.3.1. Tensorflow	3
2.3.2. OpenCV	3
2.3.3. Django	4
3. 과제 수행 내용	5
3.1. 초기 설계 문제점 및 개선된 설계	5
3.1.1. 초기 설계 문제점	5
3.1.2. 개선된 설계	5
3.2. 구현 과정 설명	6
3.2.1. 데이터 수집 파트 – 산업체 멘토링 결과 반영	6
3.2.2. 딥러닝 파트 – 산업체 멘토링 결과 반영	9
3.2.3 웹 파트	16
3.3. 구현 코드 설명	20

4. 성능 측정	25
4.1. 실험 환경	25
4.2. 실험 결과	26
5. 결론	28
5.1. 활용 방안	28
5.2. 향후 과제	28
6. 추진 체계 및 일정	29
6.1. 역할 분담	29
6.2. 개발 일정	29
7. 참고 문헌	31

1. 과제 개요

1.1. 대상 문제

최근 IT 및 컴퓨터 기술의 발전으로 사람이 판단하고 작업해야 하는 일을 대신해주는 AI들이 발명되고 있다. 하지만, 사람의 감정과 같은 모호한 부분은 바로 인식하기 쉽지 않다. 또한, 마스크를 썼을 경우 기존의 이모티콘 어플에서 인식하지 못하는 문제가 있다. 따라서 감정 인식과 함께 안경, 마스크 등을 포함한 얼굴 특징 인식이 가능한 이모티콘 생성기를 개발하게 되었다.



Figure 1 – 마스크를 쓴 경우 인식이 안되는 기존 어플 예시

1.2. 요구 사항

다음은 이모티콘 생성기를 개발하기 위한 요구사항을 순서대로 나열한 것이다.

1) 얼굴인식 및 추출

사진이나 영상으로부터 얼굴을 인식하고 그 얼굴 부분만 따로 추출할 수 있어야 한다.

2) 얼굴 특징 점 추출

추출한 얼굴사진으로부터 안경을 썼는지, 마스크를 썼는지 등의 얼굴의 특징점을 추출할 수 있어야 한다.

3) 이모티콘 생성을 위한 모델 구현

추출한 얼굴 특징점을 이용하여 어떤 종류의 안경을 썼는지, 마스크를 썼는 지와 피부 색 등 특징별로 분류할 수 있도록 하는 모델을 구현하고, 화남, 경멸, 역겨움, 공포, 행복, 무표정, 슬픔, 놀람 등의 감정을 인식하는 모델을 구현한다.

4) 웹을 통한 이모티콘 제작 및 출력

3)에서 생성된 모델을 통하여, 인풋 이미지와 가장 확률이 높은 감정과 특징점을 반영하여 이모티콘을 출력할 수 있어야 한다.

1.3. 과제 목표

이 과제를 수행하기위한, 과제 목표는 다음과 같다.

- 1) 사진에서 OpenCV와 미리 학습된 검출기를 이용하여 얼굴의 특징 점 추출
- 2) 코어알고리즘을 이용한 감정파악
- 3) 1)과 2)에서 도출된 결과를 사용하여 이모티콘 생성

2. 과제 배경 지식

2.1. 기술소개

2.1.1. 딥러닝

머신 러닝의 한 종류로, 컴퓨터가 여러 데이터를 이용해 마치 사람처럼 스스로 학습할 수 있게 하기 위해 인공 신경망을 기반으로 구축한 기술이다. 음성, 이미지 인식과 사진 분석 등 광범위하게 활용된다.

2.1.2. 컴퓨터 비전

카메라나 스캐너 등 영상 입출력 매체를 통하여 입력 받은 이미지나 영상에서 물체, 배경 등 물체와 주변 환경에 대한 데이터를 분석하여 유의미한 정보를 생성하는 기술이다. 즉, 컴퓨터의 시각적인 능력에 판단능력을 부여하여 이미지 데이터를 가공 및 해석하는데 사용된다.

2.2. 개발 환경

개발 환경은 다음과 같다.

1. 개발언어 : Python 3.7, HTML5.0, JavaScript, CSS, JSON
2. 개발도구 : django(Window Power Shell, Sublime Text), jupyter, Tensor flow

2.3. 주요 모듈

다음은 개발에 사용된 주요 모듈들이다.

2.3.1. Tensorflow

구글에서 만든 머신 러닝을 위한 오픈소스 라이브러리이다. 기본적으로 C++, python, JAVA, Go 등 다양한 언어를 지원하지만, python 개발에 최적화되어 있다. 또한 브라우저에서 실행가능한 시각화 도구인 TensorBoard 로 학습과정 추적이 용이한 장점이 있다.

2.3.2. OpenCV

Open Source Computer Vision의 약자로, 이미지와 동영상의 처리에 사용할 수 있는 오픈소스 라이브러리이다. C++을 기반으로 하고 있으며, real time image processing 에 초점을 둔 많은 내장함수를 포함하고 있다.

2.3.3. Django

장고는 python으로 작성된 open source web application framework로 MVC패턴을 따르고있다.

3. 과제 수행 내용

3.1. 초기 설계 문제점 및 개선된 설계

3.1.1. 초기 설계 문제점

초기 설계단계에서 어떤 모듈, 환경이 개발하기에 가장 적합한지 몰랐던 부분이 가장 큰 문제점이었다. 또한 이모티콘 생성과정에서 개개인의 얼굴 특징을 반영하기 위해 감정인식 모델 뿐만 아니라 얼굴 특징 점 모델(마스크, 안경 등)을 구하여 직접 학습시키는 과정을 추가했다.

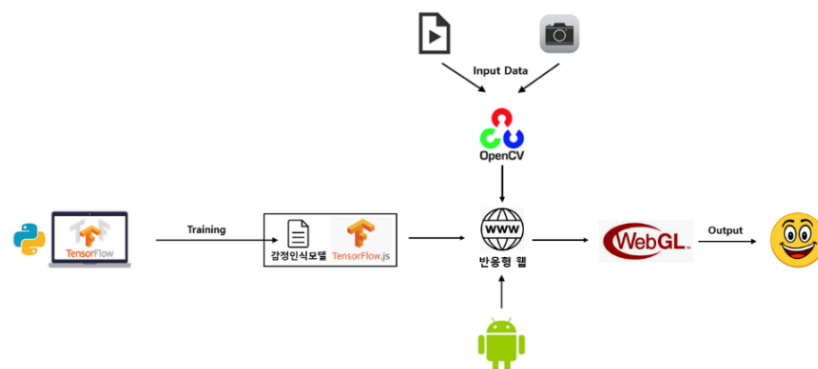


Figure 2 - 변경 전 시스템 구성도

3.1.2. 개선된 설계

Python을 이용해서 deep learning을 수행하였으며 OpenCV를 통해 이모티콘을 생성하였다. 그후에 모든 과정을 합친 어플리케이션을 Django를 통해 HTML 위에 올렸으며 PC, Mobile 모두 접속이 가능한 반응형 웹사이트 형태로 제작하도록 시스템 구성도를 변경했다.

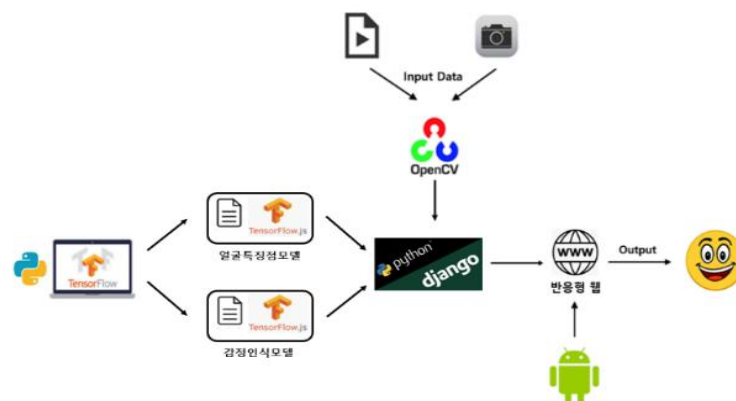


Figure 3 - 변경 후 시스템 구성도

3.2. 구현 과정 설명

3.2.1. 데이터 수집 파트 - 산업체 멘토링 결과 반영

안경과 관련된 데이터셋과 마스크와 관련된 데이터셋의 경우, 무료로 이용할 수 있는 kaggle 데이터셋을 다운받아서 사용하였다. 하지만, 대부분의 데이터셋이 백인 혹은 동양인으로 이루어져 있어서 피부색을 판단하는데 필요한 데이터가 현저히 부족하였다. 이 부분은 직접 검색을 통해 라벨링 과정을 거쳐 사용하였다.

Table 1 - 사용된 모델 별 데이터셋 현황

모델 별 데이터셋 종류	인식 대상	데이터 개수	학습 정확도
마스크 데이터셋	mask / No mask	3,833	95%
피부색 안경 데이터셋	피부색별 안경착용여부	633	80%
안경데이터셋 1-1 (Nomask)	안경 / 선글라스	480	97%
안경데이터셋 1-2(Nomask)	동그란안경 / 네모난안경	519	85%
안경데이터셋 1-3(Nomask, 흑인)	동그란안경 / 네모난안경 (흑인)	134	85%
안경데이터셋 2-1 (mask)	안경 착용 여부	2072	92%
안경데이터셋 2-2 (mask)	안경 / 선글라스	301	85%
안경데이터셋 2-3 (mask)	동그란안경 / 네모난안경	232	85%

마스크 데이터셋의 경우, 착용했을 때와 착용하지 않았을 때의 구분이 확연하다. 그와 더불어 데이터의 개수가 많기 때문에 학습 정확도가 95%를 상회했다. 그에 반해 대부분의 안경 데이터셋의 경우에는 데이터의 개수가 적고 인식대상의 차이가 미묘하기 때문에 학습 정확도가 85%내외를 유지했다. 안경 / 선글라스의 경우에는 차이가 확연하므로 정확도가 높았다.

한편, 초반 학습을 돌리는 과정에서 정확도가 높는데 의도한 대로 결과가 안 나오는 등 어려움이 많았는데 그 이유는 학습에 사용되는 데이터와 관련한 문제가 컸다. 첫째로는 데이터 분류의 문제이다. 초반 생각하기에는 선글라스와 동그란 안경, 네모난 안경, 안경을 착용하지 않은 경우까지 한번에 모델링하려고 했다. 하지만 모든 인종의 데이터셋을 각각 경우에 맞게 수집하는 것이 사실상 불가능했고 데이터셋이 적다 보니 정확도가 떨어질 뿐더러, 선글라스에도 동그란 선글라스, 네모난 선글라스 등이 포함되기 때문에 제대로 된 결과가 나오지 않았다. 따라서, 안경 착용 여부를 미리 확인한 뒤에, 안경과 선글라스에 대해서 분류하고 안경을 썼다면 동그란 안경인지 네모난 안경인지 분류하도록 바꾸었다.

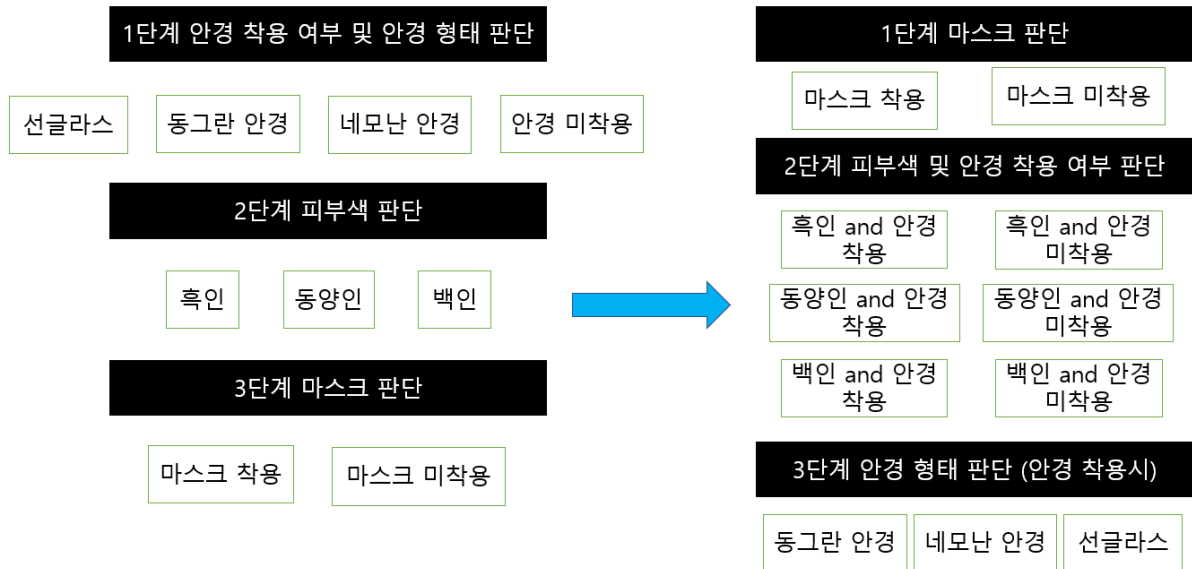


Figure 4 - 데이터 분류 순서의 변화를 보여주는 구조도



Figure 5 - 데이터 분류 상세 구조도

둘째, 데이터 자체의 정확도 문제이다. 아무래도 구글을 통하여 얻은 이미지는 얼굴만 검색했다고 하더라도 정제된 데이터가 아니기 때문에 이미지마다 뒷배경도 다르고 차이가 있다. 뿐만 아니라, 기존에 받은 데이터셋도 증명사진 같은 사진들이지만 얼굴만 필터링 된 사진이 아니다. 그래서 생각했던 것은 어차피 얼굴인식을 통해서 얼굴을 자르고 그 부분에 대하여 특징 점 인식과 감정인식을 진행할 것이기 때문에 얼굴인식 알고리즘을 활용해야 한다고 판단했다. 그 후, 알고리즘을 적용하여 데이터를 정제하고 활용하였다.



Figure 6 - 얼굴인식 알고리즘을 활용한 데이터 정제 예시

셋째, 데이터 수 부족 문제이다. 기존에 받았던 데이터셋은 대부분 백인과 동양인에 대한 파일들이었기 때문에, 흑인과 관련해서 피부색을 구분하고 안경 여부를 구분하는데 어려움이 많았다. 그리고 동그란 안경을 쓴 흑인, 선글라스를 쓴 흑인 등 흑인에 관련된 다양한 요소들을 지닌 데이터를 받기가 어려웠다. 따라서 몇가지 방법을 통하여 데이터 수를 증가시키고 다시 라벨링하는 과정을 거쳤다. 이들은 데이터 수가 현저히 적으므로, 좌우대칭과 좌표이동을 통해서 이미지 수를 증가시켰다. 그리고 피부색 구분을 위해서 밝기조절 등도 사용했다.

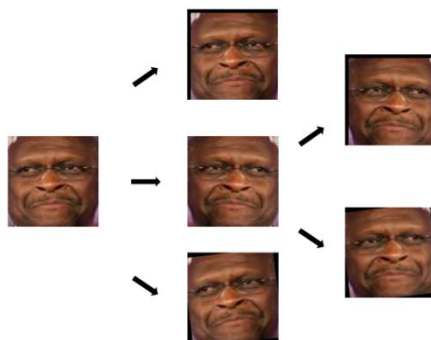
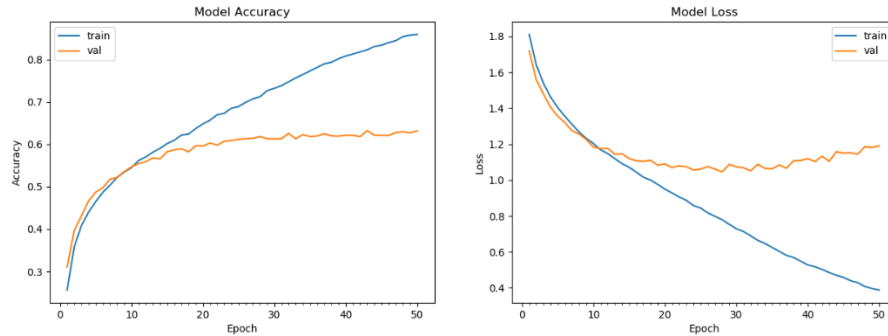


Figure 7 - 좌우대칭과 좌표이동을 통해 이미지 수를 증가시킨 예시

3.2.2. 딥러닝 파트 – 산업체 멘토링 결과 반영

감정 인식 같은 경우 데이터셋을 직접 모으기 힘들기 때문에 오픈소스를 활용하여 학습을 진행하였다. 학습 결과는 다음과 같다.



Graph 1 - 감정 인식 학습 결과

(출처 - <https://github.com/atulapra/Emotion-detection>)

특징 점 인식 같은 경우, 다음과 같이, 나누어 진행하였다.

Table 2 – 특징 점 인식 진행 상황

인식 대상	loss function	activation	optimizer
마스크 유무	binary_crossentropy	sigmoid	adamax
인종(백인,흑인,황인)과 안경 유무	categorical_crossentropy	softmax	adamax
안경 vs 선글라스 (마스크 미착용)	binary_crossentropy	sigmoid	adamax
동그란 안경 vs 네모난 안경 (마스크 미착용)	binary_crossentropy	sigmoid	adamax
동그란 안경 vs 네모난 안경 (마스크 미착용, 흑인 전용)	binary_crossentropy	sigmoid	adamax
안경 착용 여부 (마스크 착용)	binary_crossentropy	sigmoid	adamax
안경 vs 선글라스 (마스크 착용)	binary_crossentropy	sigmoid	adamax
동그란 안경 vs 네모난 안경 (마스크 착용)	binary_crossentropy	sigmoid	adamax

loss function을 살펴보면 단순 2개의 레이블을 비교하는 경우(마스크 유무, 안경 착용 여부 등)는, binary_crossentropy를 사용하였고, 2개 이상인 6개를 비교하는 인종(백인, 흑인, 황인)과 안경 유무 인식에 대해서는 categorical_crossentropy를 사용한 것을 볼 수 있다. 한편, 이와 비슷하게 활성화 함수도 categorical은 softmax를 사용하였고 binary는 sigmoid를 사용하였다.

대부분 과정이 비슷하기 때문에, 인종(백인,흑인,황인)과 안경 유무를 통해서 구현 과정을 설명하겠다. 이 신경망은 다음과 같은 은닉 계층 구성 정보로 지정되어 있다.

Table 3 – 인종, 안경 유무 인공지능망 은닉 계층 구성 정보

(인풋 사이즈는 128x128x3)

계층	정보	파라미터 수(#)
convolution layer	커널 사이즈 = 3x3, 채널 사이즈 = 32	896
maxpooling	보폭 사이즈 = 2x2	0
dropout	드롭 확률 = 0.5	0
convolution layer	커널 사이즈 = 3x3, 채널 사이즈 = 64	18496
maxpooling	보폭 사이즈 = 2x2	0
dropout	드롭 확률 = 0.5	0
convolution layer	커널 사이즈 = 3x3, 채널 사이즈 = 128	73856
maxpooling	보폭 사이즈 = 2x2	0
dropout	드롭 확률 = 0.5	0
flatten	평탄화	0
dense	출력 레이어 노드수 = 128, 활성화함수=relu()	3211392
dropout	드롭 확률 = 0.5	0
dense	출력 레이어 노드수 = 6, 활성화함수=softmax()	774

우선, 사이즈가 128x128이고 채널이 3인 이미지를 인풋으로 받는다. 합성곱 계층에서 커널이라는 작은 가중치 텐서를 이미지의 모든 영역에 반복 적용해 패턴을 찾아서 처리하게 된다. 참고로, 같은 커널이 모든 위치에 적용되기 때문에 한 곳에서 포착된 패턴이 다른 곳에도 이용된다. 커널 사이즈가 3x3이고 출력 채널 사이즈가 32이므로, 여기서 계산되는 파라미터 수는 커널 사이즈(3x3)과 입력 사이즈(3)과 채널 사이즈(32)를 곱하고 bias(32)를 더한, 896이 된다.

그후 maxpooling을 진행한다. 이 과정은 처리할 이미지 해상도를 줄이는 기능을 제공해 다양한 크기의 패턴을 단계적으로 처리한다. 즉, 합성곱 계층은 이미지 해상도를 유지하면서 채널 수를 늘리는 역할을 맡으며 풀링 계층은 채널 수를 유지하면서 이미지 해상도를 줄이는 역할이다.

이 과정을 3번반복한뒤, flatten과 dense과정을 거쳐서 여러 채널에 흩어져 있는 정보를 종합하여 문제 해결에 필요한 형태의 출력을 생성한다. 이 과정을 간단히 도식화 하면, 다음과 같다. [1]

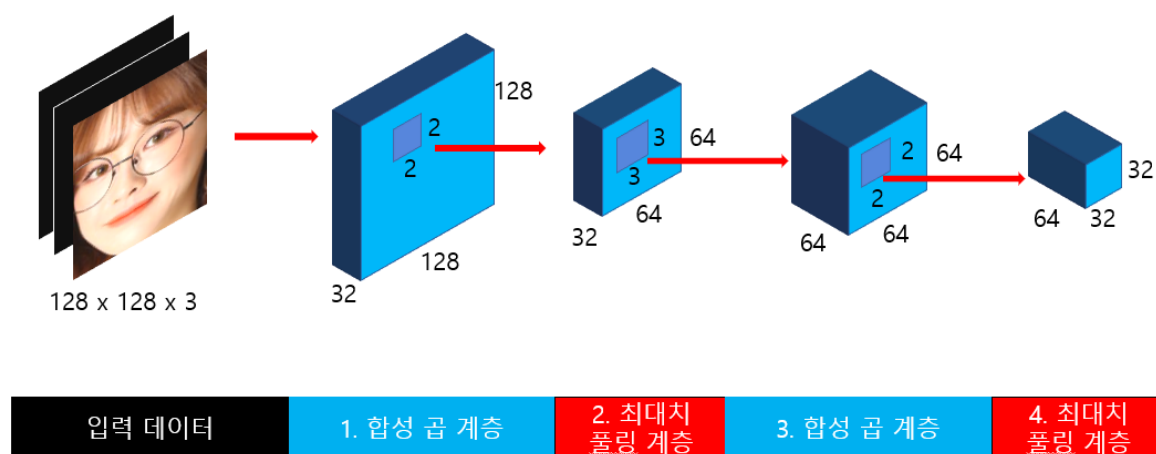


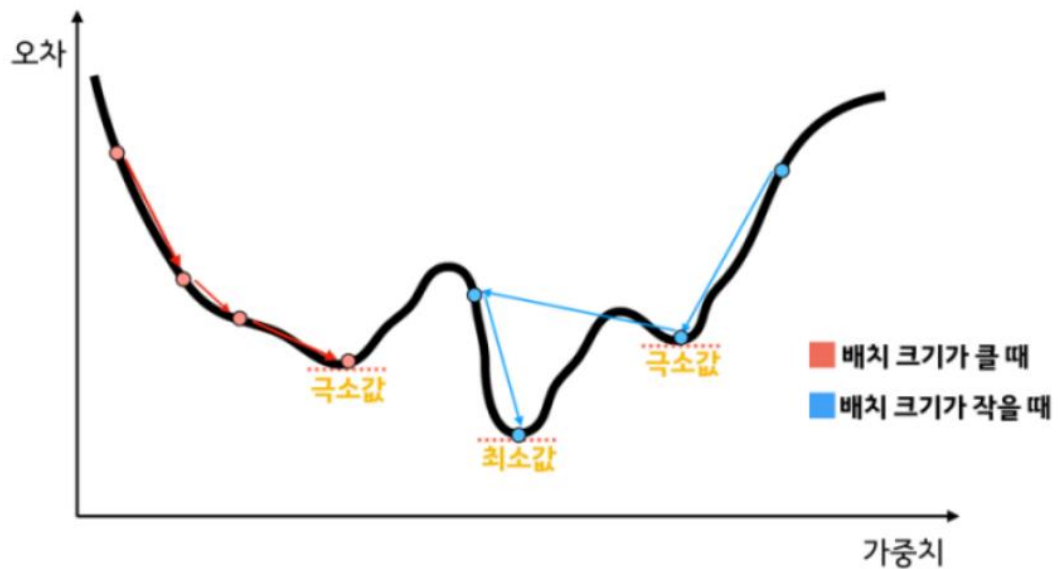
Figure 8 - 인공지능망 은닉 계층 도식화

여러 파라미터와 함수들을 다르게 주었을 때 결과가 어떻게 나오는지 알아보기 위하여 몇가지 기준을 두고 여러 동작을 실행해보았다. 처음으로는, 배치 사이즈를 다르게 주었다. 배치 사이즈란 데이터 중 한번에 네트워크에 넘겨주는 데이터의 수를 말한다. 예를 들면, 100개의 데이터가 있을 때 배치 크기가 1이면 모델은 1 에폭당 100번 훈련한다. 배치 크기가 10이면 10번, 배치 크기가 100이면 1번 훈련한다. 80에폭을 기준으로 한다면 배치 크기가 1이면 총 훈련 횟수는 8,000번, 배치 크기가 100이면 총 훈련 횟수는 80번이되는 것이다. 같은 약 500장의 데이터에 대해서 배치 사이즈를 1과 100으로 주고 에폭 50으로 학습을 돌린 결과 다음과 같이 나왔다.

Table 4 – 배치 사이즈에 따른 특성 표

배치 사이즈	1 에폭당 훈련 횟수	1에폭당 평균 걸리는 시간	정확도
1	506회	40~50초	76~82%
100	5회	약 20초	82~84%

배치 사이즈가 작다면, 1 에폭당 훈련 횟수가 늘어나고 당연히 총 훈련 횟수가 늘어난다. 따라서, 훈련 시간이 늘어남을 알 수 있다. 따라서 이론적으로는 보다 빠르게 최적의 성능에 도달할 수 있다는 이유로 배치 크기는 클수록 좋다는 결론을 도출할 수 있다.



Graph 2 - 최소값임을 보장할 수 없는 극소값 또는 안장점 근처에서 학습이 진행된다고 가정하면, 배치크기가 큰 상황에서는 이 구간을 빠져나오기가 어렵다. 반면, 배치 크기가 작을 때는 이 구간에서 빠져나오기가 훨씬 수월하다.

(출처 - <https://www.kakaobrain.com/blog/113>)

하지만, 위의 그림으로 알 수 있듯이 배치 크기가 클수록 좋다고 보긴 어렵기 때문에 적절한 배치 크기 탐색은 모델 훈련에서 매우 중요하다고 볼 수 있다.

다음으로는, 활성화 함수를 다르게 주었다.

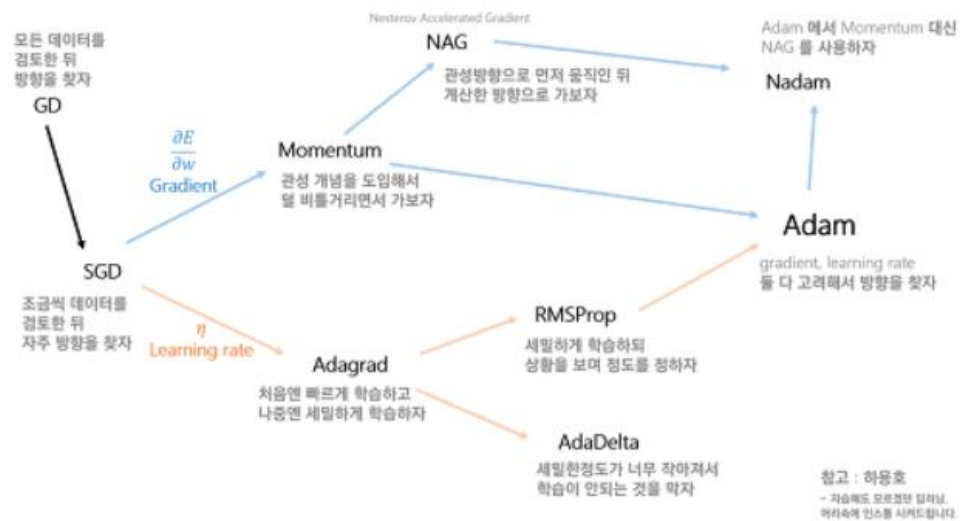


Figure 9 - Optimizer 발전 과정

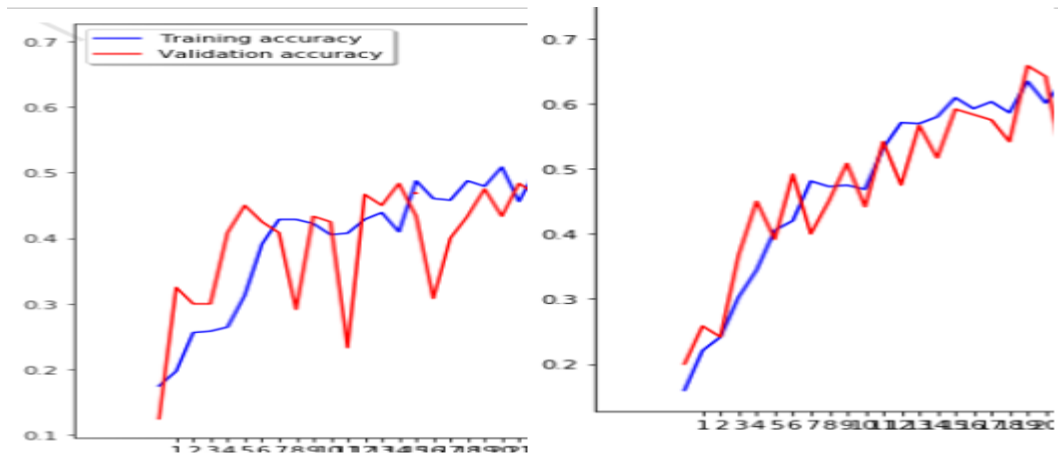
(출처 - <https://gomguard.tistory.com/187>)

우선, Momentum의 개념에 대해서 알아보자. Gradient descent기반의 알고리즘으로, 수식으로 표현하면 다음과 같다.

$$\begin{aligned} v &\leftarrow \alpha v - \eta \frac{\partial L}{\partial W} \\ W &\leftarrow W + v \end{aligned}$$

Figure 10 - Momentum 수식

가중치를 더해줄 때, v 를 더해주는데 이는 일종의 가속도(혹은 속도)같은 개념으로 가중치가 감소하던(혹은 증가하던)방향으로 더 많이 변화할 수 있도록 해준다.

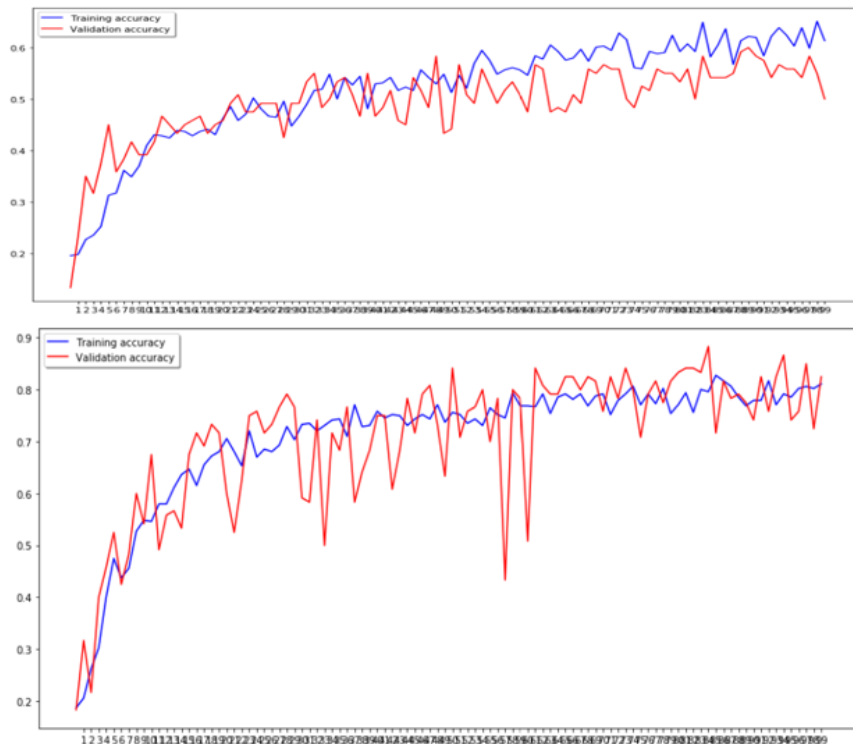


Graph 3 – SGD함수(왼쪽)와 Adam함수(오른쪽)의 학습 그래프

위 그래프를 보면, SGD함수보다 Momentum개념이 들어간 Adam함수가 학습이 증가하는 방향으로 기울기가 더 큰 것을 확인할 수 있다. (학습률과 가중치는 비례하기 때문이다.)

이제, Adagrad함수를 알아본다. 우선, Momentum과 NAG는 모든 가중치에 동일한 학습률을 적용해 왔다. 하지만 직관적으로 생각해보면 어떤 가중치는 빠르게 학습이 끝날 수도 있고, 다른 경우에는 더 큰 학습률 적용이 필요한 경우도 있다. 따라서 고안된 것이 Adagrad함수이다. 과거의 기울기를 제공하여 계속 더하면서 빈번히 업데이트가 일어나는 가중치의 학습률은 작게 하고, 반대의 경우는 크게 하는 방법이다.

하지만 Adagrad함수는 과거의 기울기를 제공하여 계속 더해가므로, 학습을 진행할수록 갱신 강도가 약해진다. 실제로 무한히 계속 학습한다면 어느 순간 갱신량이 0이 되어 전혀 갱신되지 않게 된다. 이 문제를 개선한 기법으로 RMSProp함수가 있다. RMSProp은 과거의 모든 기울기를 균일하게 더해가는 것이 아니라, 먼 과거의 기울기는 서서히 잊고 새로운 기울기 정보를 크게 반영한다. 이를 지수이동평균(Exponential Moving Average, EMA)라고 하여, 과거 기울기의 반영 규모를 기하급수적으로 감소시킨다.



Graph 4 - Adagrad함수(위)와 RMSProp함수(아래)의 정확도 비교 그래프

위의 그래프를 보면, Adagrad함수는 어느 순간부터 전혀 갱신이 되지 않음을 확인할 수 있고 RMSProp함수는 그 문제가 개선이된 것을 확인할 수 있다.

이러한 비교를 통해서, Momentum과 Adaptive learning rate를 함께 반영한 Adam함수의 extension인 Adamax함수를 주로 이용하여 구현하게 되었다. [2]

3.2.3 웹 파트

다음은 이모티콘 생성기를 위해 사용한 Django project의 구조이다. 참고로, Django는 MVT 패턴을 적용하여 효과적으로 웹 어플리케이션을 개발할 수 있는 프레임워크이다.

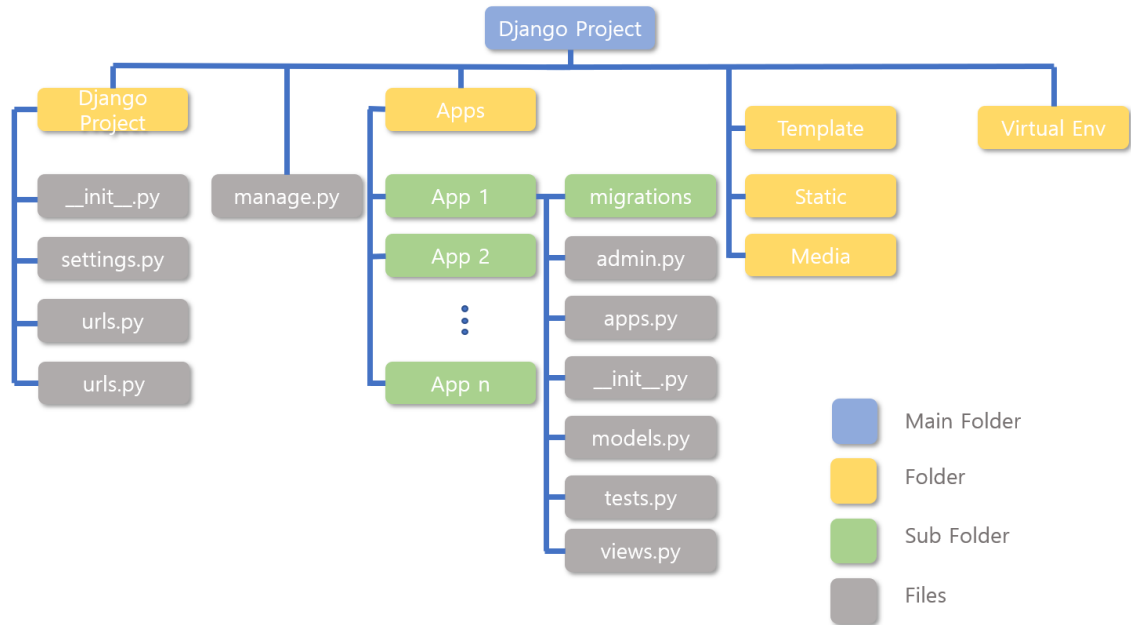


Figure 11 – Django project 구조도

웹에서는 크게 두가지의 기능을 구현하였고, 각각의 시나리오는 다음과 같다.

1) 웹 캠을 통한 사진입력

먼저 사용자로부터 웹 캠의 사용권한을 획득한 후 웹카메라로 사용자의 얼굴을 캡처한다. 캡처한 사진을 canvas태그에 저장하고 해당 사진을 사용자에게 보여준다. 업로드 버튼을 통해 사진데이터를 서버로 보낸다. 서버에서 이모티콘 생성에 관한 처리를 하고 생성된 이모티콘을 반환한다.

2) 이미지파일 업로드를 통한 사진입력

웹 캠 페이지에서 파일 첨부하기 버튼을 클릭하여 이미지 업로드 페이지로 이동한다. input태그를 이용하여 이미지 파일을 첨부한다. 업로드 버튼을 통해 사진데이터를 서버로 보낸다. 서버에서 이모티콘 생성에 관한 처리를 하고 생성된 이모티콘을 반환한다.

두가지 시나리오 모두 이미지파일을 업로드하는 기능이지만 서버에서 받는 이미지데이터형식이 달랐다. 웹 캠을 통하여 canvas태그에 담아 보냈을 경우는 base64 인코딩을 거친 후 전송하게 되어 서버에서도 base64 디코딩을 해주어야 byte데이터 -> RGB이미지 데이터로 변환 할 수 있다. input태그를 이용하여 이미지 파일을 업로드하면 base64 인코딩 과정 없이 바로 byte형식으로 데이터가 들어오게 된다. byte형식의 데이터는 openCV 라이브러리로 RGB이미지 데이터로 변환이 가능하다.

아래 그림은 웹 어플리케이션 홈 화면이다.

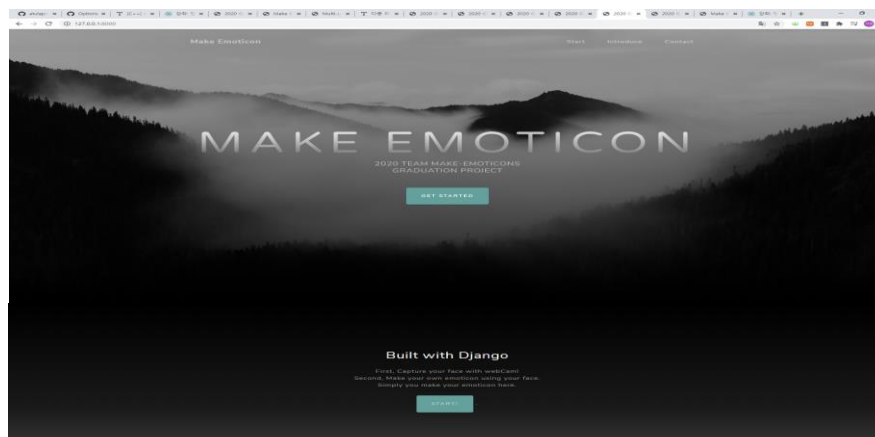


Figure 12 – 웹 어플리케이션 홈 화면

홈페이지는 반응형 웹페이지로 만들었다. 그에 따라 세로 폭이 좁아질 경우 해상도에 맞게 레이아웃이 변경되는 것을 볼 수 있다.

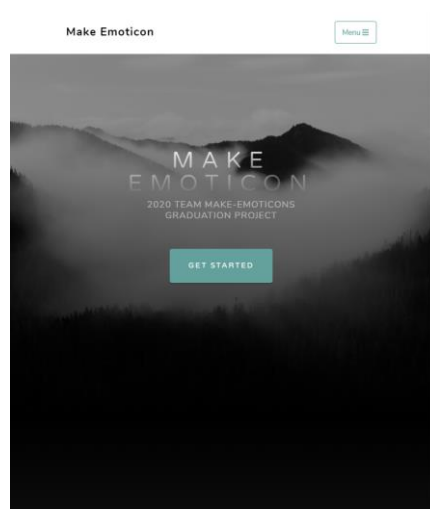


Figure 13 – 반응형 웹페이지임을 보여주는 예시

GET STARTED를 누를 시 아래 START로 화면이 이동하게 되며 START를 누를 시 웹 카메라 권한을 얻는다.

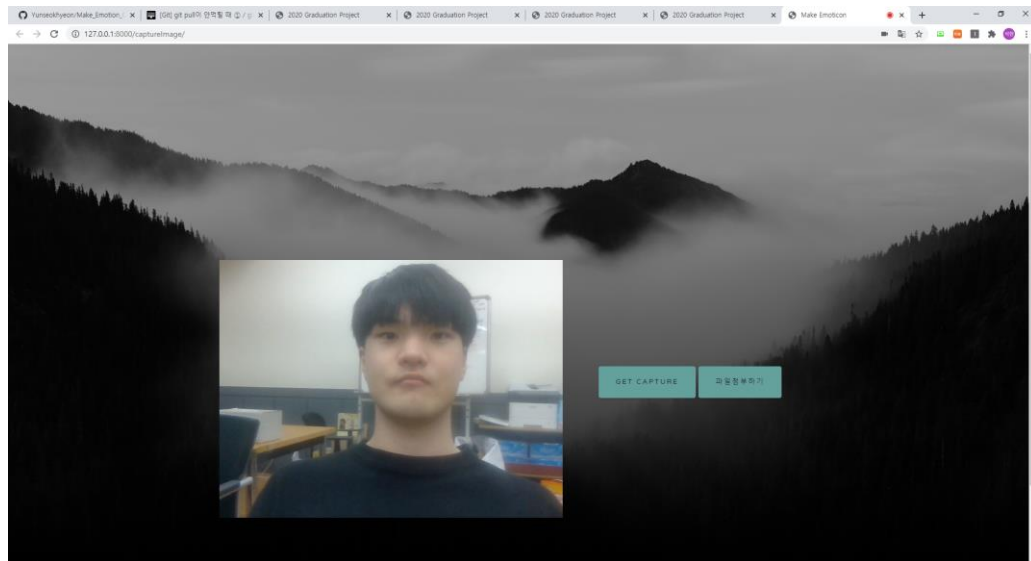


Figure 14 – 카메라 권한을 얻은 화면

GET CAPTURE 를 누를 시 캡처 화면을 우측에 보여주며 배터리 절약 및 효율성을 고려해 웹카메라는 종료한다. 이 후 제출 클릭 시 이모티콘 생성을 시작한다.

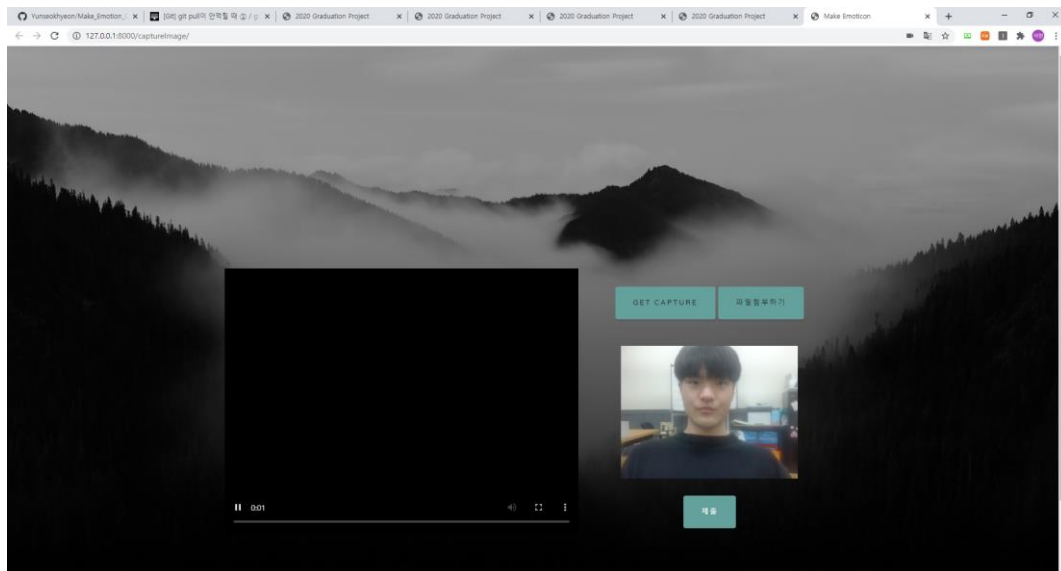


Figure 15 – 캡처 화면

Figure 14 에서 파일 첨부하기를 클릭 시 아래 화면으로 이동하며 사용자는 자신의 사진을 업로드 할 수 있다. 파일을 선택한 후 UPLOAD 시 이모티콘 생성을 시작한다.

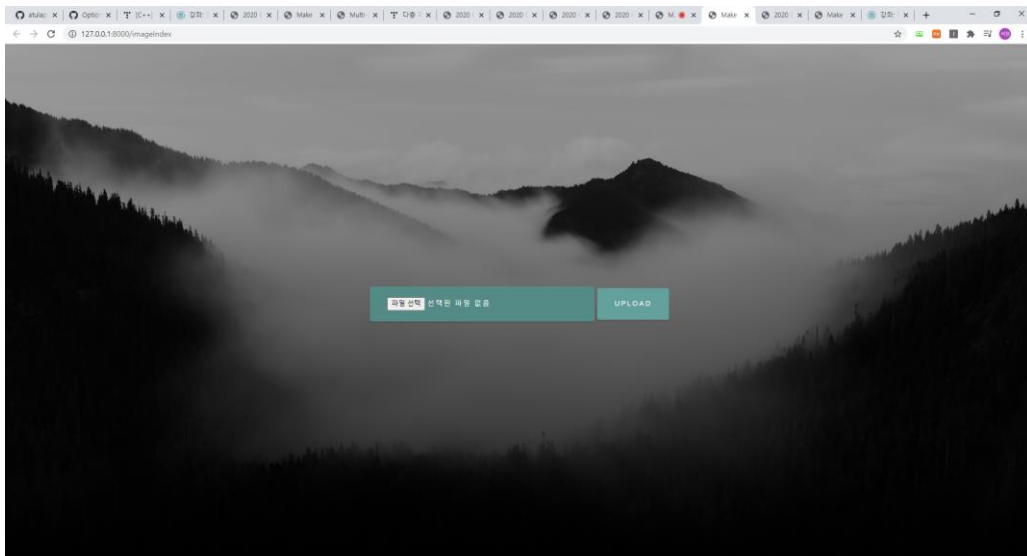


Figure 16 – 파일 업로드 화면

아래는 웹 어플리케이션의 최종 결과이다. 자신의 사진 혹은 업로드한 사진과 함께 생성된 이모티콘을 화면에 보여준다.

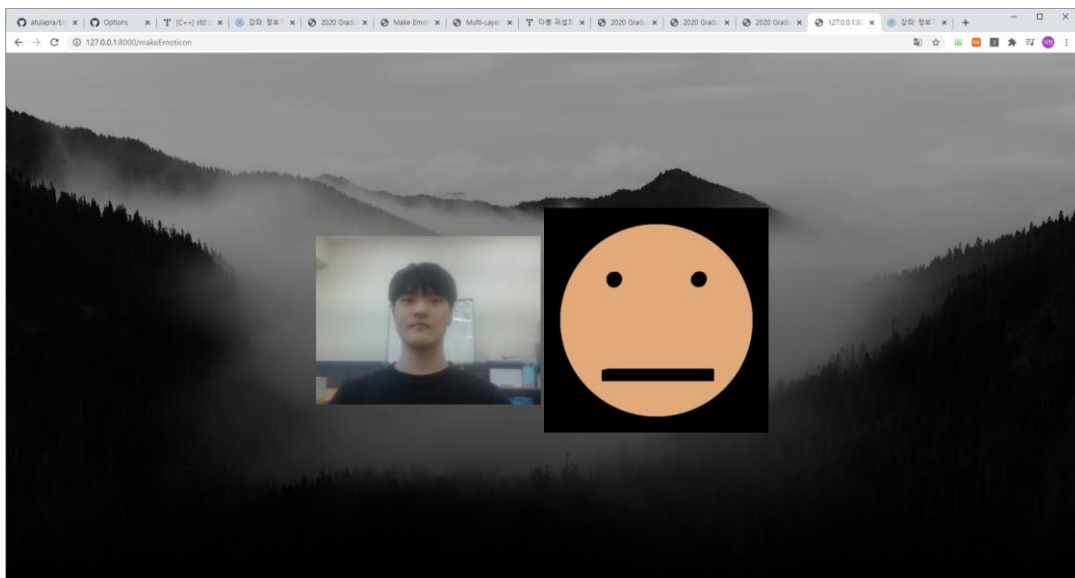


Figure 17 – 이모티콘 생성 결과

3.3. 구현 코드 설명

다음은 구현된 코드에 관한 의사 코드이다.

emoticon.py

```
# 학습된 모델 가져오기
감정모델
마스크모델
안경모델
피부색모델
선글라스모델
안경모양모델

# 분류될 감정리스트
감정리스트 = {"화남", "억겨움", "공포", "행복", "무표정", "슬픔", "놀람"}
피부색리스트 = {"아시아", "흑인", "백인"}
마스크리스트 = {"마스크착용", "마스크미착용"}
안경리스트 = {"안경착용", "안경미착용"}
안경모양리스트 = {"네모난안경", "동그란안경", "선글라스"}

# 메인로직
def 이모티콘만들기(입력이미지):
    결과={}
    # 초기 설정
    결과[피부색] = 아시아
    결과[감정] = 무표정
    결과[안경] = 미착용
    결과[안경모양] = []
    에러코드 = 정상

    얼굴이미지, 에러코드 = 얼굴인식(입력이미지)

    # 모델별로 쓰일 이미지 변환
    3채널데이터, 1채널데이터 = 이미지를 모델용데이터로 변환(얼굴이미지)
```

만약 에러코드가 얼굴이 한개 이상이라면

출력 : 사람이 한명이상

그렇지 않고 만약 에러코드가 얼굴이 없음이라면

출력 : 사람을 찾지못함

그렇지않다면

마스크 유무 판별

마스크모델 예측결과 = 마스크모델.예측(3채널데이터)

결과[마스크] = 마스크모델 예측결과

만약 마스크미착용이라면

피부색모델 예측결과 = 피부색모델.예측(3채널데이터)

결과[피부색] = 피부색모델 예측결과

안경모델 예측결과 = 안경모델.예측(3채널데이터)

결과[안경] = 안경모델 예측결과

만약 안경을 착용했다면

선글라스모델 예측결과 = 선글라스모델.예측(3채널데이터)

결과[선글라스] = 선글라스모델 예측결과

만약 선글라스를 착용했다면

결과[안경모양] = 선글라스

그렇지않고 선글라스를 미착용했다면

만약 흑인이라면

안경모양모델 = 흑인용안경모양모델

안경모양모델 예측결과 = 안경모양모델.예측(3채널데이터)

결과[안경모양] = 안경모양모델 예측결과

감정모델 예측결과 = 감정모델.예측(1채널데이터)

결과[감정] = 감정모델모양 예측결과

그렇지않고 마스크를 착용했다면

안경모델 예측결과 = 안경모델.예측(3채널데이터)

결과[안경] = 안경모델 예측결과

만약 안경을 착용했다면

선글라스모델 예측결과 = 선글라스모델.예측(3채널데이터)

결과[선글라스] = 선글라스모델 예측결과

만약 선글라스를 착용했다면

결과[안경모양] = 선글라스

그렇지않고 선글라스를 미착용했다면

안경모양모델 예측결과 = 안경모양모델.예측(3채널데이터)

결과[안경모양] = 안경모양모델 예측결과

이모티콘 생성영역

이모티콘 = 불러오기(이모티콘얼굴배경경로)

이모티콘 = 그림그리기(이모티콘얼굴배경, 결과[감정], 눈, 좌표값)

만약 마스크를 착용했다면

이모티콘 = 그림그리기(이모티콘얼굴배경, 결과[마스크], 흰색, 좌표값)

그렇지않다면

이모티콘 = 그림그리기(이모티콘얼굴배경, 결과[감정], 입, 좌표값)

만약 안경을 착용했다면

이모티콘 = 그림그리기(이모티콘얼굴배경, 결과[안경], 결과[안경모양], 좌표값)

반환 이모티콘, 에러코드

util.py

```
# 모델들이 사용할 수 있게 읽어온 이미지를 원하는 array로 변환
def 이미지를 모델용데이터로 변환(BGR이미지):

    RGB이미지 = RGB로 변환(BGR이미지)
    # 0~1사이의 값을 사용
    모델용데이터 = RGB이미지 / 255

    # 안경, 선글라스, 마스크 모델용 데이터
    3채널이미지 = 크기재설정(모델용데이터, 가로=128px, 세로=128px)
    3채널데이터 = 0차원추가(3채널이미지)

    # 감정 모델용 데이터
    1채널이미지 = 크기재설정(모델용데이터, 가로=48px, 세로=48px)
    1채널데이터 = 0차원추가(1채널이미지)

    반환 3채널데이터, 1채널데이터

# 이미지에서 얼굴인식하고 추출
def 얼굴인식(입력이미지):

    얼굴인식모델 = 불러오기(얼굴인식모델경로)
    이미지높이, 이미지넓이 = 높이넓이추출(입력이미지)

    Blob데이터 = Blob변환(입력이미지)
    얼굴인식모델.입력값설정(Blob데이터)

    # 인식결과리스트
    # 1x1xNx7 차원 행렬
    # N : N개의 얼굴 후보군
    # 7 : 7 개의 데이터
    # [2] : 신뢰도 / [3~6] : 사각형의 얼굴영역
    인식결과리스트 = 얼굴인식모델.예측하기()

    다음을 인식결과리스트의 처음부터 끝까지 반복:
```

현재신뢰도 = 인식결과리스트의 i번째 항목의 신뢰도

만약 현재신뢰도가 80% 이상이면

인식얼굴좌표데이터리스트 = 인식결과리스트의 i번째 항목의 좌표리스트
좌측상단x좌표, 좌측상단y좌표, 우측하단x좌표, 우측하단y좌표 = 정수화
(인식얼굴좌표데이터리스트)

얼굴좌표값 = 좌표보정(좌측상단x좌표, 좌측상단y좌표, 우측하단x좌표, 우측하단y좌표)

인식된얼굴 = 입력이미지[얼굴좌표값]
얼굴좌표리스트.추가(얼굴좌표값)

만약 인식된 얼굴이 한개 이상이라면

반환 입력이미지, 에러코드 : 얼굴이 한개 이상
그렇지 않고 만약 인식된 얼굴이 없다면
반환 입력이미지, 에러코드: 얼굴이 없음

반환 입력이미지[첫번째 얼굴좌표값], 에러코드

얼굴사진에 얼굴부위 붙이기

얼굴사진, 얼굴부위, 감정, 붙일 좌표

def 얼굴부위붙이기(얼굴사진, 감정과악세서리, 부위, 좌표X, 좌표Y):
부위이미지 = 불러오기(부위이미지경로)

높이, 넓이 = 높이넓이추출(부위이미지)

Blue채널, Green채널, Red채널, Alpha채널 리스트 = 분리하기(부위이미지)

투명도데이터 = 리스트3채널로설정(Alpha채널, Alpha채널, Alpha채널)

색깔데이터 = 리스트3채널로설정(Blue채널, Green채널, Red채널)

도화지 = 입력이미지

도화지 = 투명도적용하며 이미지붙이기(부위이미지)

4. 성능 측정

4.1. 실험 환경

학습을 진행한 환경과 코드를 실행하여 결과를 출력한 환경으로 나뉜다. 실험 환경은 다음과 같다.

운영 체제: Windows 10 Education 64비트 (10.0, 빌드 18362)
언어: 한국어 (국가별 설정: 한국어)
시스템 제조업체: SAMSUNG ELECTRONICS CO., LTD.
시스템 모델: 900X5N
BIOS: AMIBIOS Version P04AGA.037.170614.MK
프로세서: Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz (4 CPUs), ~2.7GHz
메모리: 8192MB RAM

Figure 18 - 딥 러닝학습을 진행한 실험 환경

macOS Mojave
버전 10.14.6

MacBook Pro (Retina, 13-inch, Early 2015)
프로세서 2.7 GHz Intel Core i5
메모리 16GB 1867 MHz DDR3
그래픽 Intel Iris Graphics 6100 1536 MB

Figure 19 - 코드를 실행한 실험 환경

4.2. 실험 결과

모든 모델링을 끝마친 뒤, 16장 테스트 데이터셋으로 실험 결과를 지켜보았다.

































																							
인종	백인	일치	인종	흑인	불일치	인종	디폴트	일치	인종	디폴트	일치	인종	디폴트	일치	인종	디폴트	일치	인종	디폴트	일치	인종	디폴트	일치
안경	X	일치	안경	선글라스	일치	안경	선글라스	일치	안경	선글라스	일치	안경	네모	불일치	안경	네모	불일치	안경	네모	불일치	안경	네모	불일치
마스크	X	일치	마스크	X	일치	마스크	O	일치	마스크	O	일치	마스크	O	일치	마스크	O	일치	마스크	O	일치	마스크	O	일치
감정	행복	일치	감정	행복	불일치	감정	X	일치	감정	X	일치	감정	X	일치	감정	X	일치	감정	X	일치	감정	X	일치
																							
인종	백인	일치	인종	디폴트	일치	인종	흑인	불일치	인종	백인	일치	인종	백인	일치	인종	백인	일치	인종	백인	일치	인종	백인	일치
안경	O	일치	안경	X	일치	안경	네모	불일치	안경	라운드	일치	안경	라운드	일치	안경	라운드	일치	안경	라운드	일치	안경	라운드	일치
마스크	X	일치	마스크	O	일치	마스크	X	일치	마스크	X	일치	마스크	X	일치	마스크	X	일치	마스크	X	일치	마스크	X	일치
감정	무표정	일치	감정	X	일치	감정	행복	일치	감정	무표정	일치	감정	무표정	일치	감정	무표정	일치	감정	무표정	일치	감정	무표정	일치
																							
인종	백인	일치	인종	백인	일치	인종	흑인	일치	인종	흑인	일치	인종	흑인	일치	인종	흑인	일치	인종	흑인	일치	인종	흑인	일치
안경	라운드	일치	안경	X	일치	안경	선글라스	일치	안경	네모	불일치	안경	네모	불일치	안경	네모	불일치	안경	네모	불일치	안경	네모	불일치
마스크	X	일치	마스크	X	일치	마스크	X	일치	마스크	X	일치	마스크	X	일치	마스크	X	일치	마스크	X	일치	마스크	X	일치
감정	공포	보류	감정	역겨움	일치	감정	무표정	일치	감정	행복	일치	감정	행복	일치	감정	행복	일치	감정	행복	일치	감정	행복	일치
																							
인종	흑인	일치	인종	흑인	일치	인종	황인	일치	인종	황인	일치	인종	황인	일치	인종	황인	일치	인종	황인	일치	인종	황인	일치
안경	X	일치	안경	라운드	불일치	안경	X	일치	안경	X	일치	안경	X	일치	안경	X	일치	안경	X	일치	안경	X	일치
마스크	X	일치	마스크	X	일치	마스크	X	일치	마스크	X	일치	마스크	X	일치	마스크	X	일치	마스크	X	일치	마스크	X	일치
감정	슬픔	보류	감정	화남	일치	감정	화남	일치	감정	화남	일치	감정	화남	일치	감정	화남	일치	감정	화남	일치	감정	화남	일치

Figure 20 - 16가지 테스트셋으로 실행한 결과

(인종의 경우, 피부색을 기준으로 하였다.)

위 결과를 바탕으로 정리해보면 마스크, 선글라스 착용과 피부색에 대해서는 높은 적중률을 확인할 수 있었다. 다만 아쉬웠던 점이 있다면, 수염에 대한 데이터셋이 고려되지 않아서 수염이 있을 시 흑인으로 판단하는 케이스가 있었다. 또한, 동그란 안경과 네모난 안경을 구분하려고 모델링을 했으나, 두 경우에 차이점이 크지 않아서 만족스럽지 못한 결과가 나왔다.

감정인식의 경우에는 기존의 데이터셋에 더하여 추가적인 라벨링을 거쳐서 모델링을 했다. 그러나 데이터셋의 높은 서양인 비율로 인해 동양인의 경우, 몇가지 감정에 대해서 모호한 결과가 나왔다.

감정인식으로 얻어지는 이모티콘은 다음과 같다.

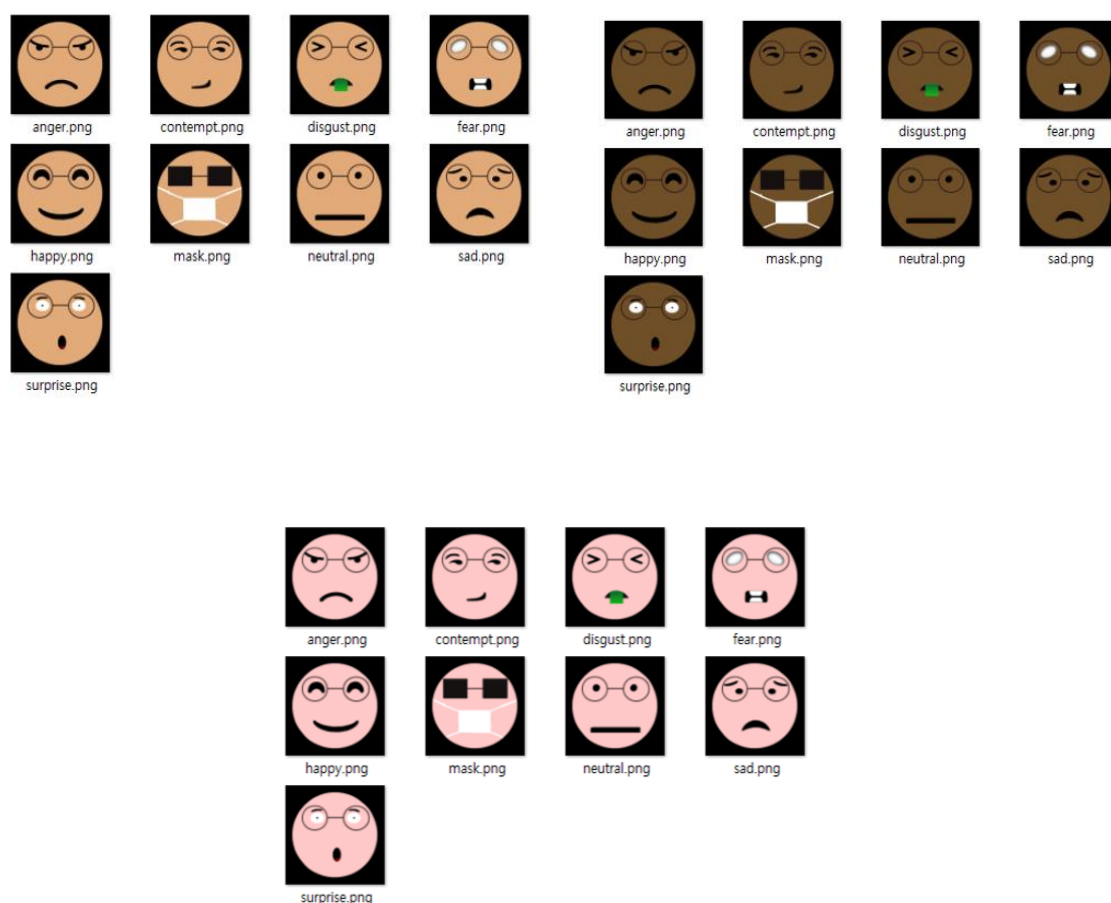


Figure 21 – 감정 별 이모티콘

5. 결론

5.1. 활용 방안

인공지능의 deep learning을 활용한 감정인식 및 분석 방법이 중요해지고 있는 추세입니다. 해당 기술을 어느 분야에 접목시키는지에 따라 활용 방안은 무궁무진합니다.

자동차에 접목시키면 운전자의 상태를 표정을 통해 파악할 수 있습니다. 해당 정보를 통해 교통사고가 나기 전, 후 운전자의 상태를 파악할 수 있으며 Accuracy를 높이면 신뢰성 있는 정보가 될 수 있습니다. 자율 주행 자동차에 접목시켜 운전자의 현재 상태를 파악할 수도 있습니다.

스트레스를 포함한 정신적 질환 예방에 사용될 수 있습니다. 사람이 포착할 수 없는 미묘하고 빠른 표정을 캐치할 수 있습니다. 캐치한 정보를 통해 환자의 상태를 쉽고 빠르게 파악할 수 있으며 후에 정신적 질환의 예방과 삶의 질 개선에 도움을 줄 수 있습니다.

기업에서 소비자를 대상으로 인간의 감정에 기반을 둔 새로운 비즈니스 기회 가능성을 키울 수 있습니다. 소비자의 특징을 통해 목표 소비자층을 타겟팅 하고, 감정 인식 기술을 통해 소비자의 감정 데이터를 수집, 분석, 추론할 수 있습니다. 해당 결과를 이용하여 광고, 게임, 교육, 판매, 홍보 등 다양한 분야에 활용될 수 있습니다.

5.2. 향후 과제

첫째, 비교적 낮은 감정 인식 정확도를 보완할 필요가 있다. 학습 데이터 셋이 현재 서양인에게 맞춰져 있는 점 때문에 비교적 낮은 정확성을 보이고 있고, 눈과 입 모양 등으로 판단을 하다보니 감정별로 모호한 부분이 많았다. 이에 대한 해결책은 동양인 혹은 흑인에 맞는 데이터 셋을 구해 적당한 알고리즘으로 학습시켜서 높은 정확도를 얻거나, 단순 외적 특징 점 외에 피부 근육 움직임 등 상세 요소를 추가하여 보완해야 하는 것이다.

둘째, 현재는 얼굴 외 특징점이 안경과 마스크에 국한되어 있다. 후에 모자나 귀걸이 등을 추가하여 차별성을 높이고 사용자의 특징을 잘 반영한다면 더 좋은 결과가 도출될 것이다.

6. 추진 체계 및 일정

6.1. 역할 분담

학번	이름	역할
201524536	이연걸	- Tensorflow를 이용한 딥러닝 모델 구성 - 표정인식 학습데이터 수집 - 부족한 데이터 수집, 가공 및 라벨링 작업
201524455	문성욱	- openCV를 활용한 얼굴 검출 오픈소스 검색 및 개발 - 이모티콘 생성 작업 - 전반적인 코드 정리
201524515	윤석현	- 웹 UI, 디자인 구성 - 웹 개발환경 구성(Django, 웹 연동) - 이모티콘 생성 작업 - 감정인식 오픈소스 검색 및 개발

6.2. 개발 일정

일정	5월	6월	7월	8월	9월
업무					
착수보고서					
Opencv를 이용한 얼굴 추출 및 특징 점 추출					
악세사리 착용 인식 학습데이터 수집 / 학습					
웹서버구축 및 웹 캠 연결					
중간보고서					
감정인식 모델 데이터 수집 / 학습					
이모티콘 생성기 개발					
호환성 테스트 / 보완					
최종보고서 제출 / 발표					

2020년 전기 산학협력 프로젝트 활동보고서

팀 명	이모티콘 만들조			
팀 원	이연결, 문성욱, 윤석현			
과 제 명	표정인식을 통한 이모티콘 생성기			
산 업 체 멘 토	기 업 명	위데이터랩㈜		
	성 명	권건우	직 위	대표이사
	연 락 처	010-6400-9127	E-MAIL	redpine71@wedatalab.com

세부 활동 내용	
<p>1. 산학협력 프로젝트 멘토링 내용</p> <p>ㄱ) 데이터 수집과 정제에 대한 조언</p> <ul style="list-style-type: none"> - 데이터를 잘 모으더라도 높은 성능의 GPU 서버가 필요할 것이라 조언. - 자체 학습을 시키기 위한 라벨링 권장 및 모델 튜닝을 위해 trial & error & tuning 권장. - 학생에게 배정될 수 있는 GPU 서버가 있으면 적극 이용 권장. <p>ㄴ) 딥러닝 모델 구성 부분과 OpenCV 알고리즘 구분이 필요</p> <ul style="list-style-type: none"> - 과제 핵심인 감성분류에 대해서 데이터 수집 및 딥러닝 모델 구성 파트와 OpenCV 알고리즘 파트를 잘 분리하여 진행하기를 조언. <p>ㄷ) 레포트 작성 방법 및 방향에 대한 조언</p> <ul style="list-style-type: none"> - 딥러닝이 주요 방법론이고 OpenCV는 보조 방법론이 되는 방향으로 레포트 작성을 권유받음. <p>2. 산학협력 프로젝트 멘토링 후기</p> <p>딥러닝 학습을 진행하는 부분에 있어서 데이터 수집 방법과 데이터 라벨링 방법 등에 대해서 큰 도움이 되었습니다.</p> <p>과제 진행 상태가 모호한 수준에 머물러 있을 때 구체적인 전략을 세울 수 있도록 도와주신 부분과 데이터 수집 방법, 라벨링 방법 등에 대해서 자세히 조언해 주셔서 과제를 원활하게 진행할 수 있었습니다.</p> <p>이모티콘 생성기 구현 과정에서 초기 OpenCV 알고리즘 부분과 딥러닝 부분이 섞여 있었는데 두 부분을 잘 분리할 수 있도록 조언해 주셨습니다.</p> <p>구현 가능성은 문제가 없다고 말씀해 주셨으며 기존 서비스보다 한 개라도 독창적인 서비스를 만들 것, 학습용으로 경험 삼아 해볼 것 등 조언을 해주셨습니다. 그에 따라 직접 학습시켜보는 소중한 경험과 기존 어플리케이션에는 없는 얼굴 특징점을 이모티콘에 적용하는 등 많은 결과를 얻어서 좋았습니다.</p>	
<p>권건우 <redpine71@wedatalab.com> 나에게 *</p> <p>이 고맙습니다. 변호사 사무실의 문화인물 기밀고사를 읽고 실적 처리를 하다보니 늦었습니다. 오늘 명의로 보내드리겠습니다.</p> <p>권건우 드림</p> <p>2020년 6월 25일 (목) 오후 1:28, 이연결 <2442012@gmail.com>님이 작성: ***</p>	<p>권건우 <redpine71@wedatalab.com> 나에게 *</p> <p>이명철님...항상 의견서 보내드립니다.</p> <p>교육 후원하는 인공지능 연합 프로젝트팀 같은 것 잘 알아보시고...</p> <p>물론 재미있는 주제인 것 같고...비즈니스 활용후회 없을 것 같습니다.</p> <p>대기업은 ktx 하고 서울의 문화유산은 프로나 하원의 지금은 일후나 시간을 못해서...저희회사 연구진들과 화상회의를 하도록 하시고</p> <p>권건우 드림</p>

7. 참고 문헌

- [1] Bang Jun Il, Park Dae Seo, Ahn Doo Heon, Kim Hwa Jong, "A Study on Modeling of Artificial Neural Network Using Python KERAS", *Proc. of Symposium of the Korean Institute of communications and Information Sciences 2018*, pp. 873-874, 2018. (in Korean)
- [2] Min-Jae Kang, "Comparison of Gradient Descent for Deep Learning", *Korea Academy Industrial Cooperation Society 2020*, pp. 189-194, 2020. (in Korean)