**오픈소스전문프로젝트**

**- #06 -**

**날짜 : 2020/06/09**

**조원 : 장재원, 조창권, 윤정환, 이석범**

목차

1. 데이터 처리
2. 데이터 분석
3. 데이터 분석 결과
4. 개선사항
5. 활용

**1. 데이터 처리**

**COLOR To GRAYSCALE**

****

눈을 뜨고 감은 상태를 분류할 때 색은 중요하지 않기 때문에 제거

3채널을 1채널로 변환하였기 때문에 학습할 때 연산량 감소

**Crop**

C:\Users\JungHwan\Desktop\Dev\Language\Python\project1\異붿텧.jpg****

다양한 크기를 가진 정면 사진에서 눈 부분을 24x24로 crop

**Filtering**



위의 이미지처럼 연관성이 떨어지거나 주변 픽셀들의 값 차이가 적어서 훈련에 방해가 될 데이터들은 제거

**Labeling**

**C:\Users\JungHwan\Desktop\Dev\Language\Python\project1\dataset_B_Eye_Images\openRightEyes\sb_open_r.jpg** 눈을 뜬 이미지는 1

C:\Users\JungHwan\Desktop\Dev\Language\Python\project1\dataset_B_Eye_Images\closedRightEyes\closed_eye_0059.jpg_face_2_R.jpg 눈을 감은 이미지는 0

|  |  |  |
| --- | --- | --- |
|  | **Training Set** | **Test Set** |
| **Label 1** | **2100** | **900** |
| **Label 0** | **2100** | **900** |

**Data Augmentation**

인식률을 높이기 위해서 원본 데이터에 다양한 연산을 적용해서 만든 새로운 데이터를 포함하여 데이터를 부풀렸다.

다양한 연산: 상하/좌우로 이동, 회전, 크기 조정 등

**Scaling**

입력으로 사용될 픽셀 값의 범위는 0~255로 학습시킬 때 큰 수치이기 때문에 0~1로 매핑하여 스케일링을 해주었다.

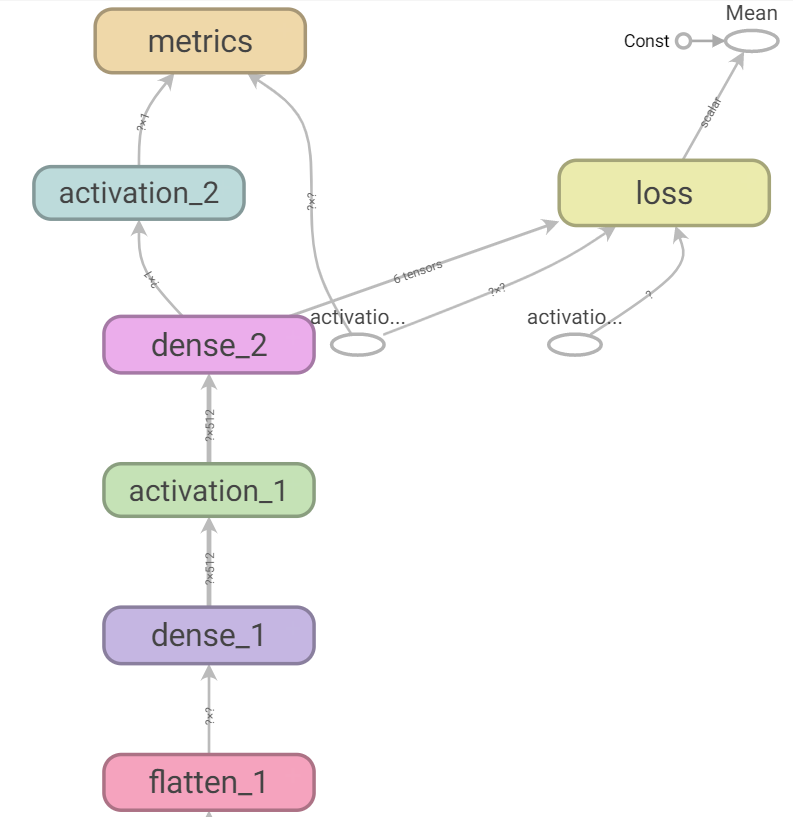
**2. 데이터 분석**

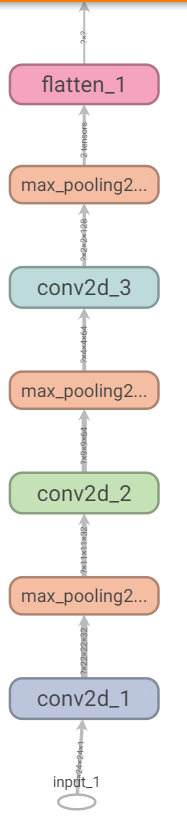
눈을 떴는지 감았는지 분류를 하기 위해서는 이미지의 특징(패턴)을 찾아야 한다.

이 특징을 찾기 위해서 CNN 기법을 이용해 이미지의 데이터를 분석할 것이다.

아래와 같이 네트워크를 구성

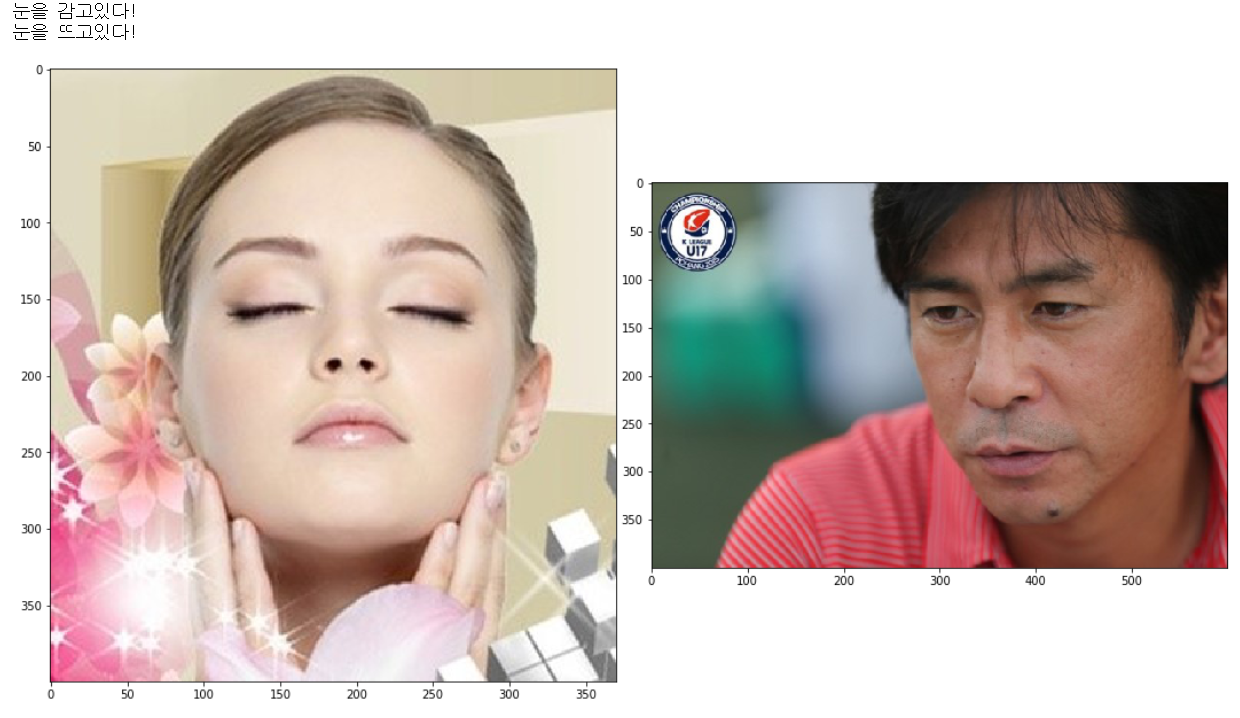
* **loss(binary\_crossentropy), optimizer(adam)**
* **24 x 24 x 1 (input)**
  + **Convolution Layer:필터 수(32), 커널 크기(3x3), 활성화 함수(relu)**
* **24 x 24 x 32**
  + **MaxPooling Layer:풀 크기(2)**
* **12 x 12 x 32**
  + **Convolution Layer:필터 수(64), 커널 크기(3x3), 활성화 함수(relu)**
* **12 x 12 x 64**
  + **MaxPooling Layer:풀 크기(2)**
* **6 x 6 x 64**
  + **Convolution Layer:필터 수(128), 커널 크기(3x3), 활성화 함수(relu)**
* **6 x 6 x 128**
  + **MaxPooling Layer:풀 크기(2)**
* **3 x 3 x 128**
  + **Flatten Layer**
* **1152**
  + **Dense Layer:출력 뉴런 수(512)**
* **512**
  + **활성화(relu)**
* **512**
  + **Dense Layer:출력 뉴런 수(1)**
* **1**
  + **활성화(sigmoid)**
* **1 (output)**



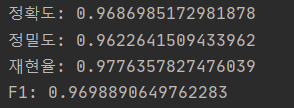


그리드 서치방식처럼 다양하게 네트워크를 구성하여 가장 좋은 성능을 내는 네트워크 구성을 택하였습니다.

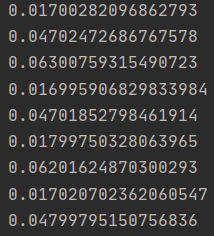
**3. 데이터 분석 결과**



훈련 데이터에 없는 데이터(사진)로 분류를 한 결과입니다.



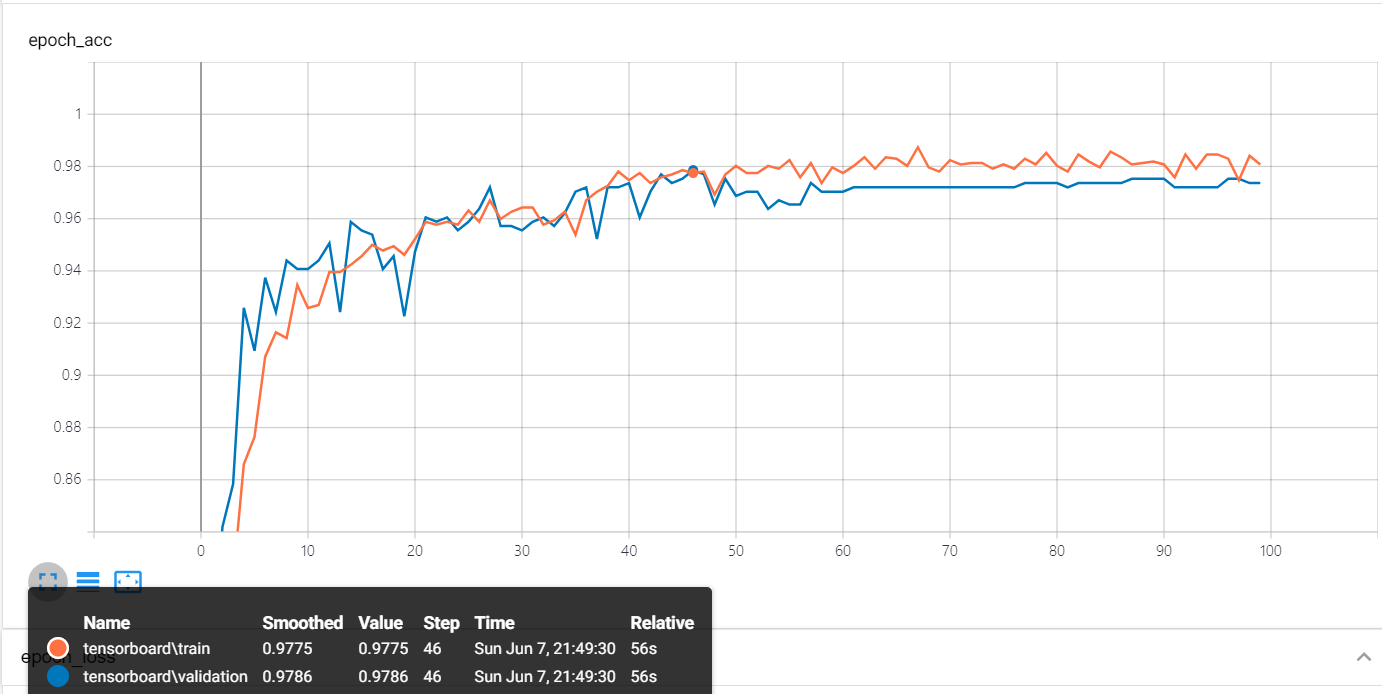
테스트 셋(1800개)를 모델을 통해 분류를 했을 때 나오는 평가 지표들입니다

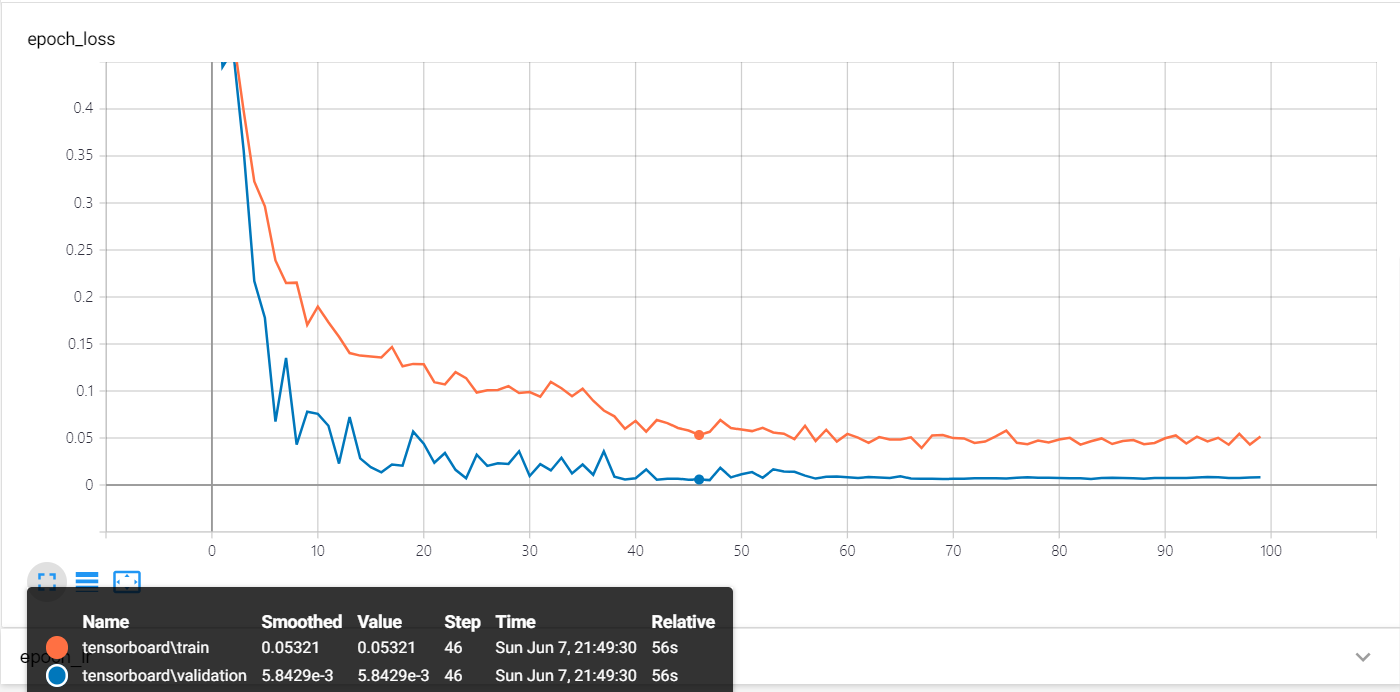


정면 사진 데이터에 전처리 작업을 진행하고 분류 하는데 걸린 시간

1000번을 수행하여 걸린 시간을 1000으로 나누어 평균을 내면 약 0.03초로 1초당 약 30개의 데이터를 처리할 수 있는 속도입니다.

**4. 개선사항**

****

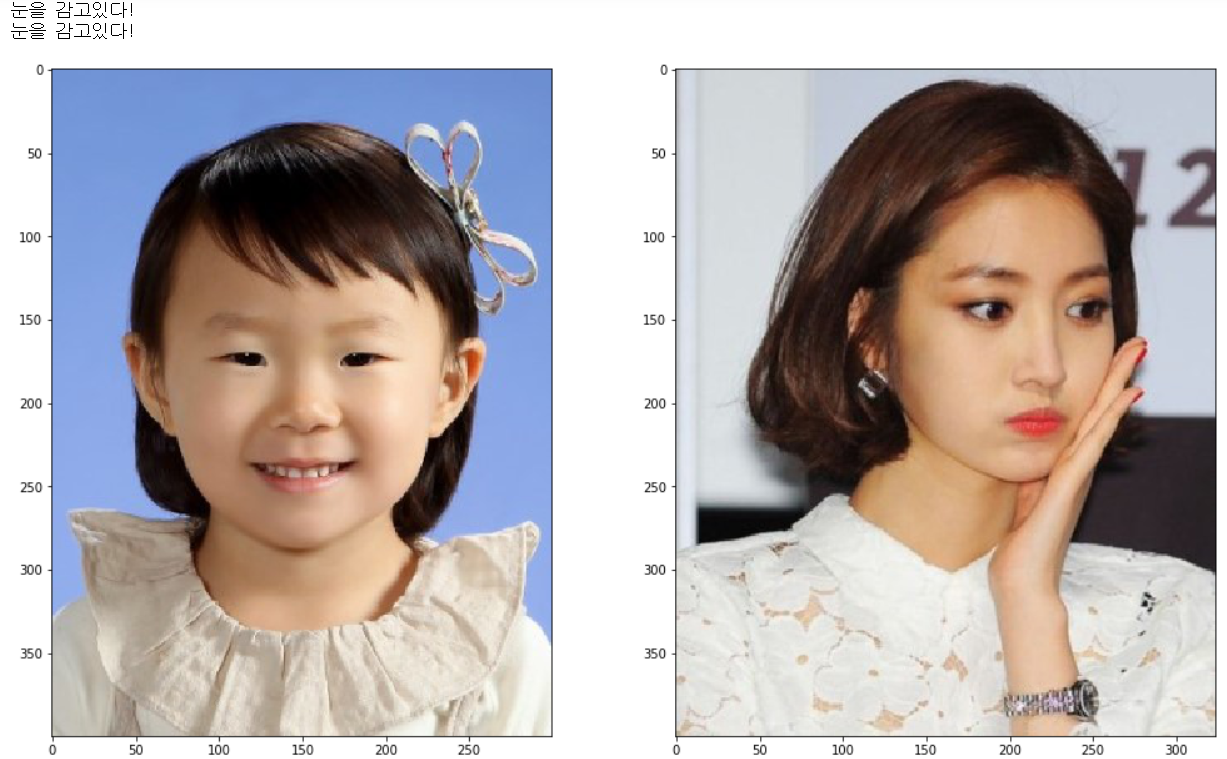
****

더 좋은 성능을 내기 위해 Hyper Parameter를 최적화하였다.

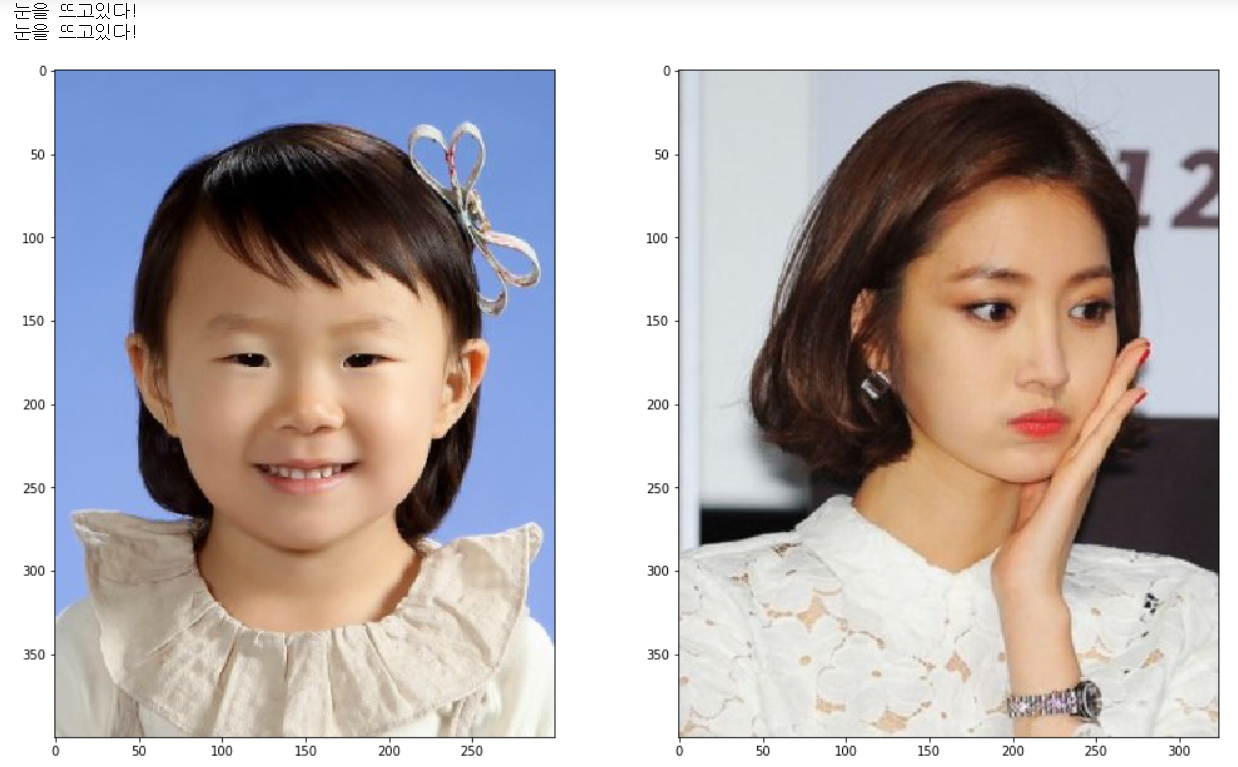
epoch를 1번부터 100번까지 하면서 나온 로그에서 가장 정확도가 높으면서 손실이 작게 측정된 46으로 epoch를 설정

* Epoch가 46일 때 훈련 정확도(0.9775), 검증 정확도(0.9786)

Optimizer로 Adam을 사용하기 때문에 Learning Rate는 따로 설정하지 않음



눈이 작거나 큰 사람들 또는 시선이 정면이 아닌 경우 분류 정확도가 떨어진다.

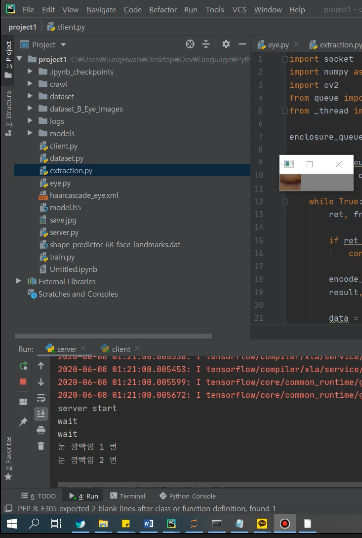
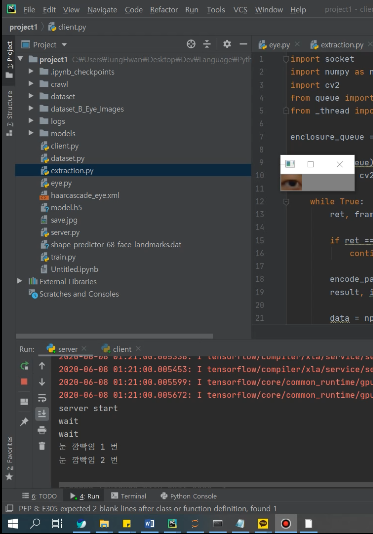


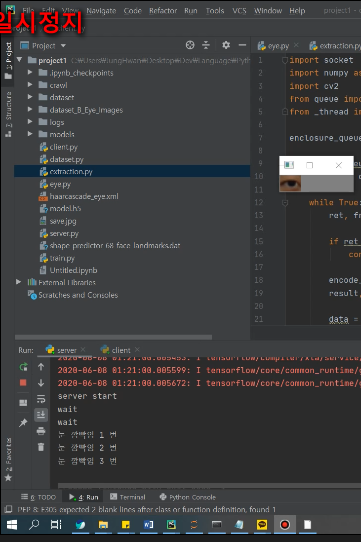
더 많고 다양한 데이터를 수집하여 훈련

원본 데이터를 변형하여 만든 새로운 데이터를 증가

* 정확도가 증가한 것을 확인할 수 있다.

**5. 활용**





주 목적인 눈 깜빡임 횟수 카운트가 잘되는지 테스트를 위해 스마트폰이 아닌 노트북에 서버 프로그램과 클라이언트 프로그램을 구현하여 테스트 해보았습니다.

휴대폰 카메라 대신 웹 캠을 사용해 카메라 데이터를 받아 전송하여 처리하는 식으로 구현해보았습니다.