표정 연기 코칭 AI 구현

목차

1. Part 5 개요

1-1. 전체 프로젝트 설명

2022 OUTTA AI 부트 캠프의 Part 5, 최종 프로젝트의 주제는 “표정 연기 코칭 AI 구현”이다. 표정 연기 코칭 AI 구현은 말 그대로 표정 연기 연습에 사용해볼 수 있는 간단한 인공지능 모델을 구현해보는 프로젝트이다. 프로젝트를 진행하며 여러 파일에 나누어 코드를 작성하면, 궁극적으로 하나의 프로그램을 구현하게 된다. 프로그램은 자신이 연기하고 싶은 일상생활 속의 한 마디나 영화 대사 등을 입력한 후, 웹캠을 바라보며 표정 연기를 하면 표정 연기의 점수를 화면에 출력해주게 된다. 프로그램 구현은 크게 두 파트로 나뉘어 진행되는데, 첫 번째는 입력한 문장에 담겨 있는 감정을 7가지(공포, 놀람, 분노, 슬픔, 중립, 행복, 혐오)로 분류하는 것, 두 번째는 웹캠을 통해 입력받는 표정에 담긴 감정을 앞서 언급한 7가지의 감정으로 분류하는 것이다. 각각은 자연어 처리와 컴퓨터 비전의 활용으로 진행되며, 프로그램은 문장으로부터 추출된 감정과 표정으로부터 추출된 감정이 비슷할수록 높은 점수를 화면에 출력해주게 된다.

막막하게 느껴질 수 있지만, 그동안《인공지능 교육단체 OUTTA와 함께하는! 머신러닝 첫 단추 끼우기》와《인공지능 교육단체 OUTTA와 함께하는! 머신러닝 첫 단추 끼우기》에서 배운 내용들, 그리고 중간미션에서 진행하였던 내용을 잘 되짚어본다면, 어느새 프로그램이 조금씩 완성되어갈 것이다. 이번 프로젝트를 통해 PyTorch 등의 라이브러리에 조금이나마 친숙해지고, 자신만의 ‘표정 연기 코칭 AI’를 구현하며 2022 OUTTA AI 부트 캠프를 잘 마무리하길 바란다.

1-2. NLP: Bert 모델로 입력 문장을 7가지 감정으로 분류하기

구글이 2017년 〈Attention is all you need〉 논문에서 발표한 ‘트랜스포머’ 모델은 머신러닝, 특히 자연어처리 과제 수행에서 매우 높은 성능을 자랑한다. 2021년 스탠퍼드 연구진은 그것을 일명 ‘파운데이션 모델(foundation model)’이라 일컫기도 하였다. **Bert**(Bidirectional Encoder Representations from Transformers)는, 구글이 트랜스포머 모델을 발표한 다음 해인 2018년, 트랜스포머의 인코더-디코더 구조 가운데 인코더 부분을 활용하여 새로이 개발한 language representation 모델이다. 당시 Bert는 기존의 NLP 11개의 task, 특히 질의응답 및 언어추론과 같이 자연어의 문맥을 반영하는 task에서 최고 성능을 보이며 그 효과성이 입증되었다.

본 캠프의 NLP 파트에서는 Bert 모델의 적용 및 응용에 대해 살펴보고자 한다. 최종 미션에서는 중간미션에서의 감정 이진 분류에서 더 확장하여, 주어진 입력 문장을 일곱 가지 감정 중 하나로 분류하는 태스크를 진행해볼 것이다. 학습을 위해 활용하는 데이터는 포티투마루가 제공하여 AIHUB 사이트에서 다운받을 수 있는 ‘한국어 SNS 대화 데이터셋’이며, 학습 모델은 Bert 기반 모델을 사용할 것이다. 중간미션과 차별화되는 지점은 크게 세 가지로 정리할 수 있다. 첫째, 미리 제공된 인터페이스를 사용하지 않고 Bert 모델에 대한 파인튜닝을 직접 실행해볼 것이다. 이 부분은 스켈레톤 코드를 참조하면서 진행할 것이다. 둘째, 전처리 부분은 중간미션과 거의 동일하게 진행하되 평가 비중이 줄어들었다. 셋째, 분류의 가짓수가 다양해진만큼 모델 예측의 정확도가 다소 낮다. 따라서, 주어진 ‘sns 데이터의 감정 7진 분류’라는 태스크에 최적화된 모델을 찾기 위해 참가자 스스로 다양한 시도를 해보

고 모델의 성능을 점차 향상시키는 것이 최종 미션의 목표이다. 이를 위해 미션 초반에 신경망 성능 향상을 위한 툴을 멘토와 함께 학습하고, 부가적인 내용은 직접 구글링해볼 것이다. 이후, 참가자 나름의 방식으로 신경망 성능 향상을 위한 다양한 툴을 선택하여 자율적으로 모델에 적용하고, 최적의 하이퍼파라미터 조합을 직접 탐색하는 과정을 포함할 것이다.

1-3. Vision: 얼굴 사진을 7가지 감정으로 분류하기

컴퓨터 비전 파트에서는, 1-1에서 언급했듯 웹캠으로부터 촬영된 영상을 입력으로 받아, 인물의 표정을 7가지 감정으로 분류하는 작업을 진행한다. 웹캠으로부터 영상을 받고, 화면에 각종 ui를 표시하는 것은 두 번째 중간미션 중 'OpenCV 사용법 익히기'를 활용하여 진행하게 된다. 감정의 분류는 두 번째 중간미션 중 'FER2013 데이터셋 탐색하기'에서 다루었던 FER2013 데이터셋을 분류하는 CNN 기반 모델을 학습시켜 진행하게 된다. 학습시킬 모델로는 세 번째 중간미션에서 다루었던 'Mini Xception' 모델을 사용해도 좋으며, Inception, Xception, 혹은 새로운 모델을 공부하여 사용해도 좋다.

모델 중간에 새로운 계층을 추가 및 변경하거나, 채널 수, 학습률, batch 크기, epoch 수 등 다양한 하이퍼파라미터를 조정해보며 모델을 학습시켜보자. 모델이 잘 학습된다면, 웹캠에 인식된 얼굴 표정으로부터 감정을 잘 추출해내는 것을 확인할 수 있을 것이다.

2. 주의사항

주의사항을 어길 시 발생하는 문제는 미션 참가자의 책임으로 간주한다.

- 랜덤 시드를 주어진 2022 값으로 반드시 고정하여야 한다.
- 주어진 데이터 전처리 과정을 변형하지 않는다. (즉, 전처리 과정을 변경해 모델의 성능을 향상시키는 것은 불가능하다)
- 제공된 압축 파일의 파일 구조를 변경하지 않는 것을 권장한다. 변경하여 사용해도 무방하나, 그 경우 파일을 import하여 사용하는 등의 코드에서 경로에 신경써 변경해주어야한다. (파일 구조에 대해서는 4-1의 설명을 참고)
- 모델 학습을 위해서는 GPU를 사용 가능한 Colab을 사용하는 것을 추천한다. 개인 GPU가 있는 경우에는 Colab을 사용하지 않아도 괜찮다. 단, Google Colab을 사용하는 경우, 기본적인 실행은 Google Colab에서 하되, OpenCV 라이브러리를 사용하는 main_stream_final.ipynb, vision_main_final.ipynb 파일은 Anaconda Jupyter Notebook을 사용하여 실행하여야 한다.
- OUTTA에서 Training 용으로 제공한 데이터만을 이용해 학습을 진행한다. 위반 사항이 적발될 시 부정행위로 간주된다.

- (NLP) 기본 모델은 제공된 KoBert 모델을 사용하며, Layer의 종류 및 개수, Pretrained model 종류는 바꾸지 않는다. 코드 상으로는 Bert 모델을 사용하고, from_pretrained 함수를 활용해서 KoBert 모델을 받아오는 구조이다.
- (NLP) 즉, KoBert 외의 모델 (ALBERT, RoBERTa등) 을 이용하는 것은 불가능하다

- (Vision) 기본 모델은 Mini Xception을 사용해도 좋고, 그 외의 모델(Inception, Xception, 혹은 2022 OUTTA AI 부트 캠프에서 다루지 않은 모델)을 사용해도 좋다. Layer의 종류 및 개수 또한 변경 가능하다.
- (Vision) 즉 NLP Part와 다르게, Pretrained Model 을 가져와 Fine-tuning 하는 것도 가능하다. (출처 명시 필요) 단, Pretrained Model을 Fine-tuning 없이 그대로 가져와 사용하는 것은 불가능하다.

3. 셋업

3-1. 압축 파일 다운로드

1) 제공된 압축 파일을 원하는 local 경로에 다운받는다.
2) Colab을 사용하여 모델 학습을 진행하는 경우, local 경로에 다운받은 폴더의 압축을 풀고, 그대로 구글 드라이브에 업로드한다.

3-2. Colab 설정

2022 OUTTA AI 부트 캠프의 중간 미션을 잘 완료했다면, 이미 Colab을 위한 기본적인 셋업이 진행되어 있을 것이다. 주의사항에서 언급했듯, Colab을 사용하여 모델 학습을 진행하는 것을 추천하며, OpenCV를 사용하는 파일은 Colab에서 실행이 불가능함에 주의하길 바란다.

3-3. Anaconda 가상환경 생성

3-1을 통해 다운받은 압축파일의 압축을 해제하고, anaconda promt에 다음 순서에 따라 명령어를 입력하여, 가상환경을 생성한다.

1) 가상환경 (final) 생성

conda create -n final python=3.8

이제, 아래 명령을 실행하여 생성한 가상환경을 활성화하자. 2) 부터의 과정은 활성화 상태에서 진행하면 된다.

conda activate final

2) keras, tensorflow 설치

pip install keras==2.8.0 tensorflow==2.8.0

3) numpy 설치
1), 2)를 잘 수행했다면 이미 설치되어 있을 것이다.

conda list

를 통해 numpy가 이미 설치되었는지 확인해보고 넘어가자.

4) jupyter, notebook, matplotlib, pandas, scikit-learn 설치

conda install jupyter notebook matplotlib pandas scikit-learn

5) opencv 설치

conda install -n final -c conda-forge opencv

6) Pytorch 설치

(Windows, Linux) *conda install -n final pytorch torchvision cpuonly -c pytorch*
(MacOS) *conda install -n final pytorch torchvision -c pytorch*

7) import_ipynb, mxnet, gluonnlp, tqdm, sentencepiece, openpyxl, transformers==3.0.2 설치

pip install import_ipynb mxnet gluonnlp tqdm sentencepiece openpyxl transformers==3.0.2

이때, 실행 도중 다음의 에러 메시지가 뜰 수 있다. 무시하고 넘어가면 된다.

```
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the
following dependency conflicts.
tensorflow 2.8.0 requires numpy≥1.20, but you have numpy 1.16.6 which is
incompatible.
tensorboard 2.8.0 requires requests<3,≥2.21.0, but you have requests
2.18.4 which is incompatible.
pandas 1.4.3 requires numpy≥1.18.5; platform_machine != "aarch64" and
platform_machine != "arm64" and python_version < "3.10", but you have
numpy 1.16.6 which is incompatible.matplotlib 3.5.1 requires numpy≥1.17,
but you have numpy 1.16.6 which is incompatible.
```

8) protobuf 버전 다운그레이드

conda install protobuf==3.19.1 --force-reinstall

9) numpy 버전 업그레이드

pip install numpy --upgrade

이때, 실행 도중 다음의 에러 메시지가 뜰 수 있다. 무시하고 넘어가면 된다.

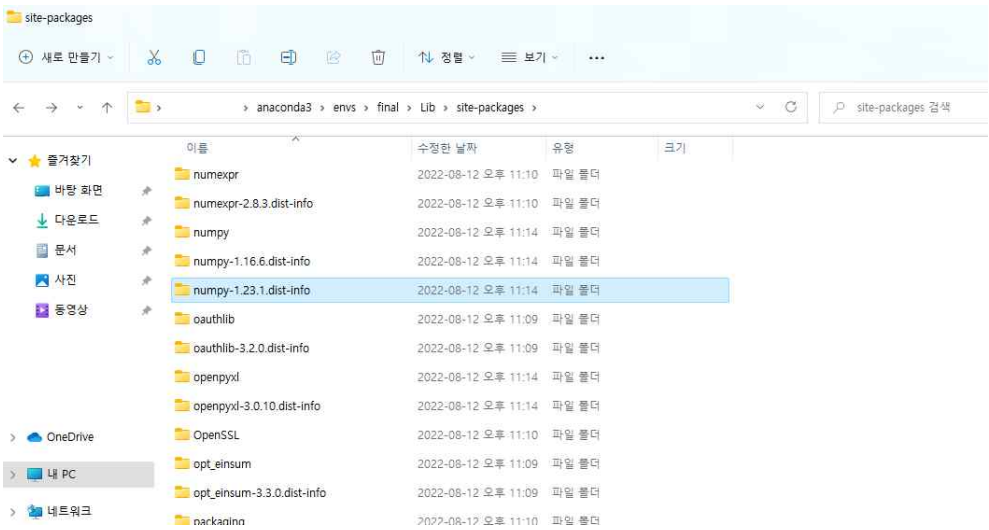
```
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the
following dependency conflicts.
tensorboard 2.8.0 requires requests<3,⩾2.21.0, but you have requests
2.18.4 which is incompatible.
scipy 1.7.3 requires numpy<1.23.0,⩾1.16.5, but you have numpy 1.23.1
which is incompatible.
mxnet 1.7.0.post2 requires numpy<1.17.0,⩾1.8.2, but you have numpy 1.23.1
which is incompatible.
```

만약 마지막 단계에서 numpy 버전 업그레이드가 진행되지 않고 아래와 같이 에러가 발생하는 경우, 다음과 같이 해결해 볼 수 있다.

> 9-1, 9-2, 9-3 내용은 멘토팀이 최종 미션을 테스트 해 보며 에러를 해결한 방식으로, 사용하는 운영 체제 및 anaconda 버전에 따라 해결법이 상이할 수 있으며 아래 방법으로 100% 해결이 되지 않을 수도 있다. 아래 방법을 시도해 본 후에도 에러가 해결되지 않는 경우 구글링을 통해 에러 메시지를 검색해 보자.

9-1) 에러 메시지 속, 폴더 경로를 찾아 파일 탐색기에서 연다. 위 에러의 경우,
c:₩₩users₩₩*(사용자명)*₩₩anaconda3₩₩envs₩₩final₩₩lib₩₩site-packages₩₩numpy-1.23.1.dist-info
디렉토리를 찾으면 된다.

파일 탐색기에서 폴더를 찾아보자.



9-2) 서로 다른 2개 버전의 numpy 가 설치되어 있는 것을 확인할 수 있다. 여기서 **가장 낮은 버전을 제외**하고 나머지를 모두 삭제하자. 즉, 위 사진의 경우 'numpy-1.16.6.dist-info' 폴더만 남기고 **'numpy-1.23.1.dist-info' 폴더를 삭제**하는 것이다.

9-3) 이후, Anaconda prompt 로 돌아와 다시 numpy 버전을 업그레이드하자.

pip install numpy --upgrade

```
(final) C:₩Users₩frogy>pip install numpy --upgrade
Requirement already satisfied: numpy in c:₩users₩[redacted]₩anaconda3₩envs₩final₩lib₩site-packages (1.16.6)
Collecting numpy
  Using cached numpy-1.23.1-cp38-cp38-win_amd64.whl (14.7 MB)
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.16.6
    Uninstalling numpy-1.16.6:
      Successfully uninstalled numpy-1.16.6
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviou
r is the source of the following dependency conflicts.
tensorboard 2.8.0 requires requests<3,>=2.21.0, but you have requests 2.18.4 which is incompatible.
scipy 1.7.3 requires numpy<1.23.0,>=1.16.5, but you have numpy 1.23.1 which is incompatible.
mxnet 1.7.0.post2 requires numpy<1.17.0,>=1.8.2, but you have numpy 1.23.1 which is incompatible.
Successfully installed numpy-1.23.1
```

업그레이드가 정상적으로 진행된다.

이로써 프로젝트 진행을 위한 Anaconda 가상 환경 구축이 완료되었다.

3-4. 코드 작성 완료 후, 발생할 수 있는 에러

전체 코드를 잘 작성했더라도, main_stream_final.ipynb 실행에서 에러가 발생할 수 있다. 발생하는 에러의 종류 및 에러 발생 여부는 실행 환경에 따라 조금씩 다를 수 있기에, 유동적으로 대처하는 것이 필요하다.

1) state_dict 불러오기 과정에서의 Runtime Error

코드 작성을 완료한 상태에서, main_stream_final.ipynb를 실행하는 도중 state_dict를 불러오는 과정에서 아래와 같은 Runtime Error가 발생할 수 있다.

```
File ~₩anaconda3₩envs₩final₩lib₩site-packages₩torch₩nn₩modules₩module.py:1604, in Module.load_state_dict(self, state_dict, s
trict)
   1599         error_msgs.insert(
   1600             0, 'Missing key(s) in state_dict: {}. '.format(
   1601                 ', '.join('"{}"'.format(k) for k in missing_keys)))
   1603 if len(error_msgs) > 0:
-> 1604     raise RuntimeError('Error(s) in loading state_dict for {}:₩n₩t{}'.format(
   1605                        self.__class__.__name__, "₩n₩t".join(error_msgs)))
   1606 return _IncompatibleKeys(missing_keys, unexpected_keys)

RuntimeError: Error(s) in loading state_dict for BertClassifier:
        Unexpected key(s) in state_dict: "bert.embeddings.position_ids".
```

이 경우 nlp_model_final.ipynb 파일의 get_model 함수를 정의한 셀에서, model.load_state_dict 함수에 두번째 인수로 strict=False를 추가하면 된다. 셀 내에 model.load_state_dict 함수는 두 개이며, 이 두 함수 모두에 아래와 같이 추가해주면 된다.

```
def get_model(device, cuda_available):
    PATH = "model.pt"
    model = BertClassifier()
    if cuda_available:
        # PATH에 저장된 모델을 불러오기
        model.load_state_dict(torch.load(PATH), strict=False)
        # 불러온 모델을 device에 올리기
        model = model.to(device)
    else:
        # PATH에 저장된 모델을 불러오기 및 불러온 모델을 device에 올리기
        model.load_state_dict(torch.load(PATH, map_location=device),
strict=False)
    return model
```

2) 무시해도 프로그램 실행에 영향을 주지 않는 에러

main_stream_final.ipynb의 첫 번째 셀을 실행하는 과정에서 아래와 같은 에러들이 출력될 수 있다. 그러나 무시하더라도 프로그램의 실행에는 영향을 주지 않는다.

```
ERROR: Could not install packages due to an OSError: [WinError 5] 액세스가 거부되었습니다: 'C:##Users##[redacted]
##anaconda3##envs##final##Lib##site-packages##numpy##~~ibs##libopenblas.FB5AE2TYXYH2IJRDKGDGQ3XBKLKTF4
3H.gfortran-win_amd64.dll'
Consider using the `--user` option or check the permissions.
```

```
ERROR: pip's dependency resolver does not currently take into account all the packages that are installe
d. This behaviour is the source of the following dependency conflicts.
tensorboard 2.8.0 requires requests<3,>=2.21.0, but you have requests 2.18.4 which is incompatible.
scipy 1.7.3 requires numpy<1.23.0,>=1.16.5, but you have numpy 1.23.1 which is incompatible.
mxnet 1.7.0.post2 requires numpy<1.17.0,>=1.8.2, but you have numpy 1.23.1 which is incompatible.
```

```
C:₩Users₩jessi₩anaconda3₩envs₩final₩lib₩site-packages₩scipy₩__init__.py:146: UserWarning: A NumPy version
>=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.1
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

3) 그 외의 에러

간혹, jupyter notebook에서 .ipynb 파일에 작성된 함수를 import하지 못하는 경우가 있다.

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
Input In [2], in <cell line: 2>()
      1 import import_ipynb
----> 2 from nlp_main_final_sol import predict_sentence
      3 from vision_main_final_sol import predict_video
      4 import numpy as np

ImportError: cannot import name 'predict_sentence' from 'nlp_main_final_sol' (nlp_main_final_sol.ipynb)
```

이런 경우에는 **jupyter notebook을 껐다가 켜보는 것을 추천한다.** Jupyter Notebook을 재시작한 후 해당 Cell을 다시 실행시켜 보자. 정상적으로 실행이 될 수 도 있고, 또 다른 에러가 발생할 수도 있다. 또 다른 에러가 발생하는 경우 아래 부분을 참고하거나 구글링을 통해 해결해 보

자. 또한, 실행 환경마다 다른 에러가 발생할 가능성이 있기에, 본 문서에 언급되지 않는 새로운 에러가 발생한다면 에러 메시지를 구글링하여 해결해보자.

4) main_stream_final.ipynb 파일을 실행하는 과정에서 Tensorflow 로딩 에러

4) 내용은 멘토팀이 최종 미션을 테스트 해 보며 에러를 해결한 방식으로, 사용하는 운영 체제 및 anaconda 버전에 따라 해결법이 상이할 수 있으며 아래 방법으로 100% 해결이 되지 않을 수도 있다. 아래 방법을 시도해 본 후에도 에러가 해결되지 않는 경우 구글링을 통해 에러 메시지를 검색해 보자.

main_stream_final.ipynb 파일에서, 아래 Cell을 실행할 때 Tensorflow를 불러오는 부분에서 에러가 발생할 수 있다.

```
In [1]:  1 import import_ipynb
         2 from nlp_main_final_sol import predict_sentence
         3 from vision_main_final_sol import predict_video
         4 import numpy as np
```

```
File ~₩anaconda3₩envs₩final₩lib₩site-packages₩tensorflow₩python₩client₩pywrap_tf_session.py:19, in <module>
     17 # pylint: disable=invalid-import-order,g-bad-import-order, wildcard-import, unused-import
     18 from tensorflow.python import pywrap_tensorflow
---> 19 from tensorflow.python.client._pywrap_tf_session import *
     20 from tensorflow.python.client._pywrap_tf_session import _TF_SetTarget
     21 from tensorflow.python.client._pywrap_tf_session import _TF_SetConfig

ImportError: SystemError: <built-in method __contains__ of dict object at 0x000002187DC3A3C0> returned a result with an error set
```

이 에러는 Tensorflow 버전이 잘못 설치되어서 발생하는 에러이다.

먼저, 실행중인 모든 Jupyter Notebook을 종료하고 Anaconda prompt를 재실행하자. 두 번째로, 아래와 그림과 같이 현재 설치되어 있는 tensorflow를 제거한다.

conda activate final
pip uninstall tensorflow

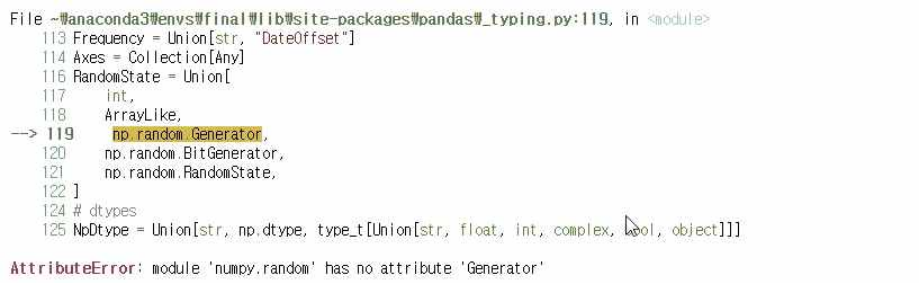
```
(base) C:₩Users₩frogy>conda activate final

(final) C:₩Users₩frogy>pip uninstall tensorflow
Found existing installation: tensorflow 2.8.0
Uninstalling tensorflow-2.8.0:
  Would remove:
    c:₩users₩frogy₩anaconda3₩envs₩final₩lib₩site-packages₩tensorflow-2.8.0.dist-info₩*
    c:₩users₩frogy₩anaconda3₩envs₩final₩lib₩site-packages₩tensorflow₩*
    c:₩users₩frogy₩anaconda3₩envs₩final₩scripts₩estimator_ckpt_converter.exe
    c:₩users₩frogy₩anaconda3₩envs₩final₩scripts₩import_pb_to_tensorboard.exe
    c:₩users₩frogy₩anaconda3₩envs₩final₩scripts₩saved_model_cli.exe
    c:₩users₩frogy₩anaconda3₩envs₩final₩scripts₩tensorboard.exe
    c:₩users₩frogy₩anaconda3₩envs₩final₩scripts₩tf_upgrade_v2.exe
    c:₩users₩frogy₩anaconda3₩envs₩final₩scripts₩tflite_convert.exe
    c:₩users₩frogy₩anaconda3₩envs₩final₩scripts₩toco.exe
    c:₩users₩frogy₩anaconda3₩envs₩final₩scripts₩toco_from_protos.exe
Proceed (Y/n)? y
  Successfully uninstalled tensorflow-2.8.0
```

다음으로, tensorflow 버전 2.7.0을 설치하자.

pip install tensorflow==2.7.0

Tensorflow 설치가 완료되면 다시 Jupyter Notebook을 실행해 보자.

5) main_stream_final.ipynb 파일을 실행하는 과정에서 numpy 모듈 로딩 에러



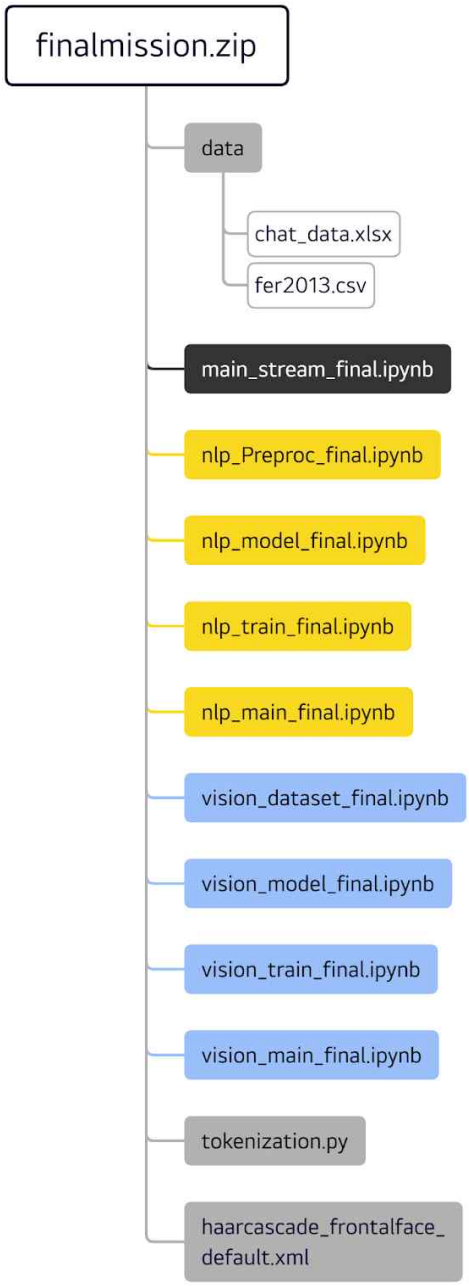
5) 내용은 멘토팀이 최종 미션을 테스트 해 보며 에러를 해결한 방식으로, 사용하는 운영 체제 및 anaconda 버전에 따라 해결법이 상이할 수 있으며 아래 방법으로 100% 해결이 되지 않을 수도 있다. 아래 방법을 시도해 본 후에도 에러가 해결되지 않는 경우 구글링을 통해 에러 메시지를 검색해 보자.

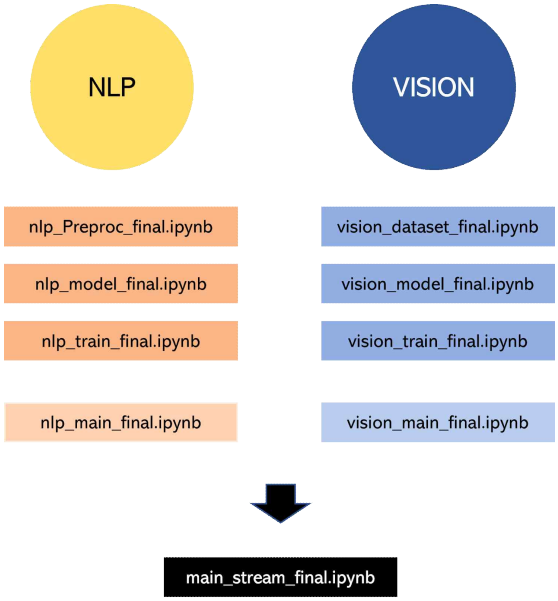
Jupyter Notebook 종료 후, Anaconda Prompt를 재시작하고 과정 3-3 의 9-1), 9-2), 9-3)을 순차적으로 시행해 보자.

4. 스켈레톤 코드 목록

4-1. 전체 스켈레톤 코드 목록

최종미션에서 제공하는 fer2013.csv 파일은 중간미션에서 사용한 fer2013.csv 데이터셋과 구조만 동일하고 구성이 다른 파일임에 유의해야 한다.
이름이 같다고 해서 기존에 가지고 있던 데이터셋 파일을 그대로 사용하여 학습을 시키면 정확도 측면에서 불이익이 있을 수 있다.





4-2. NLP 파트 구성 코드 목록

1) 총 4가지의 .ipynb파일

- nlp_Preproc_final.ipynb: 한국어 SNS 데이터셋을 전처리
- nlp_model_final.ipynb: 분류를 위한 Bert 모델 생성, 파인튜닝 및 하이퍼파라미터 설정
- nlp_train_final.ipynb: 모델의 학습 진행
- nlp_main_final.ipynb: nlp_Preproc_final.ipynb, nlp_model_final.ipynb에서 정의한 함수 및 학습이 완료된 모델을 호출하여, 학습한 모델이 새로운 입력 시퀀스에 대한 감정 7진 분류 시행하고 그 출력으로 7차원 벡터를 반환

2) .ipynb 파일의 연결 방식

각 파일에서 정의한 함수를 호출하는 방법: import_ipynb를 설치 후 함수를 import

- 설치

(Anaconda) 셋업 단계에서 이미 수행하였기에, 별도로 import_ipynb를 설치할 필요 없다.
(Colab) 첫 번째 셀에 *!pip install import_ipynb* 를 입력하여 실행한다.

- nlp_train_final.ipynb에서 nlp_Preproc_final.ipynb, nlp_model_final.ipynb에서 정의한 함수 호출

```
import import_ipynb, torch
from nlp_Preproc_final import data_to_token_ids, token_ids_to_mask
from nlp_model_final import get_model
from tokenization import KoBertTokenizer
```

- nlp_main_final.ipynb에서 nlp_Preproc_final.ipynb, nlp_model_final.ipynb에서 정의한 함수를 호출

```
import import_ipynb, torch
from Preproc_final import data_to_token_ids, token_ids_to_mask
from model_final import get_model
```

4-3. Vision 파트 구성 코드 목록

1) 총 4가지의 .ipynb파일

- vision_dataset_final.ipynb : FER2013 데이터셋의 Custom Dataset 및 Dataloader 정의. (중간미션 FER2013 Dataset 참고.) ※ test용 dataloader는 점수에 무관하다. 따라서 정의해도, 하지 않아도 괜찮다.
- vision_model_final.ipynb : 이미지 분류를 위한 모델 클래스 정의.
- vision_train_final.ipynb : 모델의 학습 진행.
- vision_main_final.ipynb : 얼굴을 탐지하고, 감정을 예측하여, 문장에서 느껴진 감정과 비교한 표정 연기 점수를 화면으로 띄워주는 기능.

2) .ipynb 파일의 연결 방식

각 파일에서 정의한 함수를 호출하는 방법: import_ipynb를 설치 후 함수를 import

- 설치

(Anaconda) 셋업 단계에서 이미 수행하였기에, 별도로 import_ipynb를 설치할 필요 없다.
(Colab) 첫 번째 셀에 *!pip install import_ipynb* 를 입력하여 실행한다.

- vision_train_final.ipynb에서 vision_dataset_final.ipynb, vision_model_final.ipynb에서 정의한 클래스를 호출

```
import import_ipynb
from vision_model_final import Mini_Xception
from vision_dataset_final import create_train_dataloader,
create_val_dataloader, create_test_dataloader
```

- vision_main_final.ipynb에서 vision_model_final.ipynb에서 정의한 클래스를 호출

```
import import_ipynb
from vision_model_final import Mini_Xception
```

4-4. NLP-Vision 연결 코드

1) main_stream_final.ipynb : 특정 입력("0")이 들어오기 전까지, 아래의 과정을 반복. 제공한 파일 그대로 사용하여도 무방.

- 연기할 문장을 입력받는다.
- 입력받은 문장에 대한 감정 7진 분류 결과를 불러온다. (nlp_main_final.ipynb의 predict_sentence 함수 이용)
- 문장과 표정에서 추출한 각 감정, 그리고 표정 연기에 대한 점수(코사인 유사도)를 화면 상에 시각화한다. (vision_main_final.ipynb의 predict_video 함수 이용)

2) .ipynb 파일의 연결 방식

각 파일에서 정의한 함수를 호출하는 방법: import_ipynb를 설치 후 함수를 import

- 설치

(Anaconda) 셋업 단계에서 이미 수행하였기에, 별도로 import_ipynb를 설치할 필요 없다.
(Colab) OpenCV가 포함되어, Colab에서 실행 불가한 파일이다.

- main_stream_final.ipynb에서 nlp_main_final.ipynb, vision_main_final.ipynb에서 정의한 함수를 호출

```
import import_ipynb
from nlp_main_final import predict_sentence
from vision_main_final import predict_video
```

4-5. 그 외 파일 설명

1) data 폴더 내의 데이터셋
- chat_data.xlsx : (nlp) 한국어 SNS 대화 데이터셋 (포티투마루 제공, AIHUB)
- fer2013.csv : (vision) Facial Emotion Recognition용 데이터셋 (중간미션 FER2013 데이터셋 탐색하기.ipynb 참고)

2) tokenization.py : (nlp) 본 task에서 활용하는 KoBertTokenizer가 저장된 파일

3) haarcascade_frontalface_default.xml : (vision) 얼굴 탐지를 위한 CascadeClassifier를 사용하기 위해 필요한 파일 (중간미션 OpenCV 사용법 익히기.ipynb 참고)

5. NLP 파트 코딩 가이드라인

5-1. nlp_Preproc_final.ipynb

한국어 SNS 대화 데이터셋을 전처리

1) function

- data_processing: id, Sentence, Emotion(label)의 세 가지로 구성되어 있는 데이터에서 필요한 부분만 추출 (sentence, label)
- data_to_token_ids: 하나의 입력 문장을 토큰화하여 각 토큰의 id를 반환
    - ▶ special token인 [CLS]와 [SEP]를 추가
    - ▶ KoBERTTokenizer의 tokenize 함수를 활용하여, 문장을 토큰화
    - ▶ 생성한 토큰을 숫자 형태의 id로 저장
    - ▶ pad_sequences 함수를 활용하여, 최대 길이(max length)에 미치지 못하는 문장에 한하여 문장의 빈칸에 padding 처리
- token_ids_to_mask: 토큰 id에서 0보다 큰 숫자만 유효하도록 하는 mask 리스트 생성
- tokenize_processed_data: 전체 데이터셋을 토큰화하여 토큰 id를 반환, mask 반환
- split_into_train_validation: 전체 데이터셋을 train용과 validation용 데이터로 분리
- data_to_tensor: torch의 tensor를 활용하여, 데이터를 텐서로 변환
- tensor_to_dataloader: 텐서로 변환한 데이터를 데이터로더에 입력
- preproc: 앞서 정의한 함수를 활용하여 일련의 전처리 과정 수행
- main: 다른 파일에서 import하기 전, preproc.ipynb 내에 정의된 함수가 성공적으로 구현되어 실행되는지 확인

5-2. nlp_model_final.ipynb

분류를 위한 Bert 모델에 파인튜닝 진행 및 모델 하이퍼파라미터 설정

1) class

- BertClassifier: 감정 7진 분류 task를 위한 BertClassifier 클래스 생성
    - ▶ 사전 훈련된 Bert 모델로서 monologg의 kobert을 사용한 후, 하나의 linear layer를 추가하여 파인튜닝 진행

2) function

- BertModelInitialization: BertClassifier 클래스를 통해 Bert 모델을 생성하여 저장
- get_model: 저장된 Bert 모델을 불러와 device에 등록
- get_model_with_params: model의 학습 과정에 활용되는 신경망 성능 향상을 위한 다양한 툴 및 하이퍼파라미터 설정

▶ Optimizer 종류, Scheduler 종류, epoch 수, learning rate 등 → 참가자의 자율적 조절 가능

- main: 다른 파일에서 import하기 전, nlp_model_final.ipynb 내에 정의된 함수가 성공적으로 구현되어 실행되는지 확인

5-3. nlp_train_final.ipynb

모델 학습 및 최종 성능 (테스트 정확도) 출력
과적합 방지를 위한 툴(Early stopping, Dropout, Regularization 등), batch-size 등은 자율적으로 조절 가능하다.

- accuracy: 모델의 예측 정확도 계산

1) random.seed: 재현을 위한 랜덤시드를 2022로 고정
2) 데이터셋 및 토크나이저 지정
3) 데이터셋을 전처리하여 train, validation, test 각각의 데이터로더에 입력
4) BertModel 생성
5) device 설정
6) train 및 validation
    - 미리 정의한 get_model_with_params 함수를 활용하여 train에 대한 model, Optimizer, Scheduler, epoch 수 등을 지정
    - 그래디언트 초기화
    - 학습 실행
    - 학습된 모델을 특정 경로(PATH)에 저장
    - ▶ 중간 미션과의 차이점: 중간미션에서는 forward 함수가 loss까지 모두 계산해준 반면, 최종미션에서는 7차원 벡터인 logit까지만 계산해주기 때문에, nlp_train_final.ipynb 내에 loss를 계산하는 로직(forward 함수를 통해 반환받은 7차원 벡터에서 가장 큰 값의 인덱스를 정답 label과 비교)이 별도로 포함됨

5-4. nlp_main_final.ipynb

- nlp_Preproc_final.ipynb, nlp_model_final.ipynb에서 정의한 함수 및 학습이 완료된 모델을 호출하여, 학습한 모델이 새로운 입력 문장에 대한 감정 7진 분류 시행

6. Vision 파트 코딩 가이드라인

6-1. vision_dataset_final.ipynb

학습에 사용하기 위해, FER2013 데이터셋을 불러와서 Custom Dataset으로 정의해야 한다. 또한, 데이터의 항목별 사용 목적에 따라 train / validation / test (선택사항) 용 데이터셋을 load 하는 데 사용할 함수를 정의해야 한다. 중간미션에 해당하는 FER2013 데이터셋 탐색하기.ipynb를 잘 공부했다면 스켈레톤 코드를 쉽게 채울 수 있을 것이다. (이 파일을 작성할 때는 코드의 표절 검사는 우려하지 않아도 된다.)

1) class

- FER2013(Dataset) : FER2013의 Custom Dataset.

2) function

- create_train_dataloader : train용 dataloader 함수
- create_val_dataloader : validation용 dataloader 함수
- create_test_dataloader : test용 dataloader 함수 (작성하지 않아도 점수에 무관하다.)

6-2. vision_model_final.ipynb

학습에 사용하게 될 모델을 클래스 형태로 구현하여 담는 파일이다. 앞서 중간미션에서는 FER2013 데이터셋 분류 학습을 위해, Mini Xception 모델을 클래스 형태로 구현하여 .ipynb 파일로 작성해본 바 있다. 프로젝트에 있어서 이 모델을 그대로 가져다 써도 되며, 앞서 언급했듯 새로운 모델을 구현하여 사용해도 된다. 새로운 모델을 구현하면서 클래스의 이름을 바꾸게 될 경우, 스켈레톤 코드의 vision_main_final.ipynb와 vision_train_final.ipynb에서 import 해오는 모델의 이름 또한 변경해주어야 함을 기억하자.

새로운 모델로 사용해볼 만한 것들로, 이하 링크의 사이트를 참고해보는 것도 좋을 것이다. 우리가 학습에 사용할 FER2013 데이터셋 분류를 진행한 다양한 모델의 정확도를 나타낸 것이다. 참고로, 중간 미션 자료에 언급했듯, Mini Xception의 최대 정확도는 66% 정도였다.

https://paperswithcode.com/sota/facial-expression-recognition-on-fer2013

하이퍼 파라미터들을 조금씩 변경해보거나, layer를 추가하거나 변경하는 등 다양한 시도를 해보며 정확도를 높이는 데 집중해보자. 과적합(overfitting)을 막기 위해서는, 5-3에서 제시된 다양한 기법들을 참고하는 것이 도움이 될 것이다.

6-3. vision_train_final.ipynb

vision_model_final에서 만든 모델을 import하여 dataset을 통해 모델을 훈련하고 평가하는 파일이다. 이 모델을 훈련할 때, 매 epoch당 checkpoint로 학습한 모델을 저장하여 이를 'vision_main_final.ipynb'에서 활용하게 된다. 따라서 checkpoint을 저장할 폴더를 생성하고 그곳에 따로 저장해놓아야 한다.(그래야 'main_stream_final.ipynb'에서 NLP와 vision이 합쳐질 때 마지막으로 학습한 모델을 사용할 수 있다.) 또 tensorboard을 활용해 epoch당 손실과 정확도의 변화 경향을 시각적으로 확인할 수 있는데 이를 위해 tensorboard을 저장할 폴더도 만들어주면 좋다. 물론 이 부분은 선택사항이며 구현하지 않아도 된다.
train에서 필수적으로 구현해야 할 사항은 다음과 같다.

1) function

모델이 데이터셋을 학습할 train 함수와, 이후 정확도를 산출할 validation 함수를 짜야 한다. 이 부분은 첫번째 중간미션에서 마지막에 MNIST 데이터셋을 학습하는 코드를 짜면서 구현했던 부분이다. 그 양식을 참고하여 구현하면 편할 것이다. 이 두 함수를 활용할 main 함수도 짜주어야 한다.

- main: 중간미션에서 행했던 것처럼 모델을 학습시키고 이후에 with torch.no_grad()에 정확도를 산출하는 코드를 짜 매 epoch당 정확도를 얻는 작업을 train과 validation 함수에서 진행할 것이다. main 함수에는 이 두 함수를 이용해 모델을 훈련하고 정확도를 얻고, 그 모델을 checkpoint로 저장하는 코드를 구현하면 된다.
- train: model, dataloader, epoch을 인자로 받는 함수이다. 이 epoch을 인자로 받을 지의 여부는 선택사항이나 model, dataloader는 반드시 인자에 포함시켜 코드를 구현해야 한다. 그 외의 인자를 추가하는 것은 자유이다. 이 함수는 본인이 원하는 손실함수와 옵티마이저를 정의한 후 데이터셋으로 모델을 훈련하는 함수이다.
- validation: model, dataloader, epoch을 인자로 받는 함수로 이 함수에서는 정확도를 산출해야 한다. 자세한 내용은 주석에 설명되어 있으니 참고하여 순서대로 코드를 구현해나가면 된다.

6-4. vision_main_final.ipynb

얼굴을 탐지하고, 감정을 예측하여, 문장에서 느껴진 감정과 비교한 표정 연기 점수를 화면으로 띄워주는 기능은 다음과 같은 여섯 단계에 따라 구현할 수 있다.

준비) 얼굴 탐지를 위한 CascadeClassifier 와 감정 분류를 위해 학습시킨 vision_model의 모델을 모두 불러온다. (vision_model의 모델을 불러올 때는, vision_main에서 사용할 학습된 가중치의 경로를 잘 지정하여야 함에 유의하자.)
0) 웹캠을 통해 실시간 영상을 불러온다.
1) CascadeClassifier(혹은 다른 모델)을 통해 얼굴을 탐지한다.
2) Mini_Xception(혹은 다른 모델)을 통해 얼굴 표정으로부터 7차원 감정 벡터를 추출한다.
3) 2)에서 구한 감정 벡터의 순서를 재정렬한다.

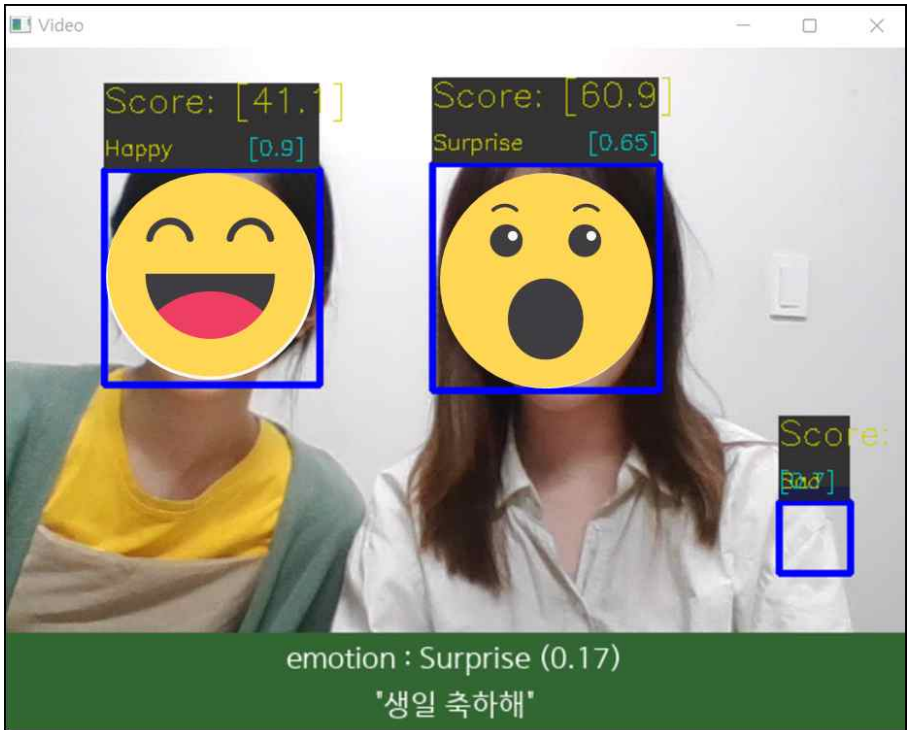
4) 입력받은 문장에서 추출한 7차원 감정 벡터와의 코사인 유사도를 계산한다.
5) OpenCV 라이브러리를 사용하여, 문장과 표정에서 추출한 각 감정, 그리고 표정 연기에 대한 점수(코사인 유사도)를 화면 상에 시각화한다.

위의 단계들을 구현하기 위해 작성할 핵심 함수는 predict_video 함수이며, 3), 4), 5) 번을 위해 sort_vec, cos_sim, get_label_emotion 함수를 추가로 작성하여야 한다.

1) function

- sort_vec : 7차원 감정 벡터의 순서를 일치시키기 위한 함수
  자연어처리 모델의 학습에 사용한 데이터셋은 감정 라벨을 '가나다' 순으로 index와 대응시켰고, 컴퓨터비전 모델의 학습에 사용한 데이터셋은 감정 라벨을 'ABC' 순으로 index와 대응시켰다. 서로 다른 라벨 순서를 가지고 있기에, 7차원 감정 벡터의 라벨 순서를 일치시키기 위해 이 함수가 필요하다.
- get_label_emotion : label 값에 대응되는 감정의 이름 문자열을 구하기 위한 함수
  7차원 감정 벡터에서 가장 높은 확률을 가진 감정의 '이름' 문자열을 화면에 띄우기 위해서는 이 함수가 필요하다. OpenCV 라이브러리에서 한글을 출력하는 것은 불편하기 때문에 영어 문자열을 출력하도록 하였다.
- cos_sim : 두 벡터 A, B의 코사인 유사도를 구하기 위한 함수
  코사인 유사도에 대해서는『(OUTTA와 함께하는!) 자연어처리 첫 단추 끼우기』에서 배운 바 있다. 두 벡터 A, B가 얼마나 유사한지를 나타내는 척도로써 코사인 유사도를 사용하기 위해 이 함수가 필요하다.
- predict_video : 실시간 웹캠 영상에, 문장과 표정에서 추출한 각 감정, 그리고 표정 연기에 대한 점수(코사인 유사도)를 시각화하기 위한 함수
  vision_main_final.ipynb의 핵심 기능이다. 중간미션에 해당하는 OpenCV 사용법 익히기.ipynb를 잘 공부하고 사용법을 충분히 연습했다면 스켈레톤 코드를 쉽게 채울 수 있을 것이다.

이하의 그림은 OpenCV 라이브러리를 통해 화면 상에 띄워야 하는 모든 정보를 나타내고 있다. 여러 명의 얼굴이 인식되었다면 제시한 그림과 같이 여러 명의 표정 연기 점수가 모두 나타나도록 해야 한다. (각자 원하는 디자인으로 꾸며도, 제시한 그림과 동일한 디자인으로 구현해도, 모두 괜찮다.)



출력해야 하는 정보:

- 원래의 한국어 문장.
- 한국어 문장에서 최고 확률을 나타낸 감정과 그 확률
- 탐지한 얼굴을 나타내는 네모 박스
- 탐지한 얼굴에서 최고 확률을 나타낸 감정과 그 확률
- 탐지한 얼굴의 표정 연기 점수

7. 평가 기준

PPT 자료 중 'Grading Policy' 를 참고하기 바란다.

8. 제출 안내

PPT 자료 중 'Submission' 를 참고하기 바란다.

9. 참고자료

[1] Arriaga, Octavio et al. "Real-time Convolutional Neural Networks for emotion and gender classification." ArXiv abs/1710.07557 (2019): n. pag.
[2] Amr Elsersy(amrelsersay@gmail.com), 〈Emotion-Recognition〉, 《GitHub》, 2021.04.30., 〈https://github.com/AmrElsersy/Emotions-Recognition〉, 2022.06.01.
[3] Rubem Winastwan, 〈Text Classification with BERT in PyTorch〉, 《Towards Data Science》, 2021.11.10., 〈https://towardsdatascience.com/text-classification-with-bert-in-pytorch-887965e5820f〉, 2022.06.01.
[4] 알디노, 〈[에어] 7가지 감정의 한국어 대화, 'KOBERT'로 다중 분류 모델 만들기 (파이썬/Colab)〉, 《Tistory》, 2021.05.27., 〈https://www.dinolabs.ai/m/271〉, 2022.06.01.