[이미지 정리하기]

1. **이미지 넘버링**(이미지 파일의 이름을 변경)
   - 나중에 재활용하기 위해 **함수**를 만들어 둔다.
   - 각각의 감정에 대하여 각각 폴더 경로를 잡아준다.
     ex> Oxford Pet Dataset 분석 프로젝트에서 'img_path'와 같은 거
   - 각각의 파일 경로를 읽어와서, 감정에 따라 각각 "감정_번호" 형태로 변경
     ex> "happy_001"
   - 코드

```
[ ]    elif 'neutral' in files[0]:
          for i,f in enumerate(files):
              os.rename(f, os.path.join(path+"/neutral", 'neutral_' + '{0:03d}.jpg'.format(i)))
          neutral = glob.glob(path+"/neutral"+'/*')
          print("neutral {}번째 이미지까지 성공".format(i+1))

       elif 'sad' in files[0]:
          for i,f in enumerate(files):
              os.rename(f, os.path.join(path+"/sad", 'sadr_' + '{0:03d}.jpg'.format(i)))
          sad = glob.glob(path+"/sad"+'/*')
          print("sad {}번째 이미지까지 성공".format(i+1))

       elif 'surprise' in files[0]:
         for i,f in enumerate(files):
           os.rename(f, os.path.join(path+"/surprise", 'surprise_' + '{0:03d}.jpg'.format(i)))
         surprise = glob.glob(path+"/surprise"+'/*')
         print("surprise {}번째 이미지까지 성공".format(i+1))
```

```
[ ]  # 파일의 이름을 재정의 해주는 함수 정의
     def rename(files):

       if 'angry' in files[0]: # 파일 경로에 angry라는 단어가 포함되어 있으면
          for i,f in enumerate(files): # i: 순번, f: 파일
              os.rename(f, os.path.join(path+"/angry", 'angry_' + '{0:03d}.jpg'.format(i)))
          angry = glob.glob(path+"/angry" + '/*')
          print("angry {}번째 이미지까지 성공".format(i+1))

       elif 'disgust' in files[0]:
          for i,f in enumerate(files):
              os.rename(f, os.path.join(path+"/disgust", 'disgust_' + '{0:03d}.jpg'.format(i)))
          disgust = glob.glob(path+"/disgust"+'/*')
          print("disgust {}번째 이미지까지 성공".format(i+1))

       elif 'fear' in files[0]:
          for i,f in enumerate(files):
              os.rename(f, os.path.join(path+"/fear", 'fear_' + '{0:03d}.jpg'.format(i)))
          fear = glob.glob(path+"/fear"+'/*')
          print("fear {}번째 이미지까지 성공".format(i+1))

       elif 'happy' in files[0]:
         for i,f in enumerate(files):
              os.rename(f, os.path.join(path+"/happy", 'happy_' + '{0:03d}.jpg'.format(i)))
         happy = glob.glob(path+"/happy"+'/*')
         print("happy {}번째 이미지까지 성공".format(i+1))
```

- 이미지 개수, 각각의 감정 사진이 전체에서 차지하는 비율 확인하기
- 만약 이미지가 불균형하게 존재하는 경우, 데이터의 양을 맞춰준다.
  (저희가 배웠던 방법 있었는데..왜 기억이 안날까요..'^';;)
  (데이터 개수 몇 개인지 확인해봤는데, angry도 지금 엄청 적네요..ㅠ
  이 부분을 미처 확인하지 못했는데, 이 부분은 데이터 수집이 더 들어가야 할 것 같아요.)

2. **이미지 시각화하기**
   - 파일이 제대로 읽어지는지 확인하기 위해 matplotlib을 이용해 시각화해본다.
   - resize, cvtColor(BGR2RGB) 등은 DataLoader를 구성할 때 하면 될 듯 하다.
   - 넘버링이 잘 되었는지 몇 개만 찍어보면서(각 감정 당 2~3개 정도) 확인한다.
     -> 필요한 모듈:
        random: 랜덤으로 몇 개의 이미지만을 가져오기 위해서
        matplotlib.pyplot(m * n 형태로 찍어보기 위해서)
        cv2: 이미지 읽어오기
   - 시각화를 해보는 김에, cvtColor와(BGR2RGB)와 resizing, grayscale을 적용
     (이미지를 실제로 가공하는 것은 DataLoader를 만들 때 구현하려 함)

3. **train_data와 test_data를 나누어주기**
   - sklearn 모듈을 이용하여 split 하는 것이 더 빠르기는 하다.
     (코랩 파일 다시 읽어봤는데...너무 새롭네요..^-^;;;)
     명령어: from sklearn.model_selection import train_test_split
   - 현재 준비된 dataset에서 train_data, test_data를 자체적으로 분리하여 정의해야 한다.
     즉, 현재 dataset에는 train_data와 test_data에 대한 개념이 없다고 봐야한다.
     -> 직접 나눠주는 편이 좋아보임(데이터 셋을 정리하며 폴더도 같이 정리해준다.)
   - 각각의 감정에 해당하는 dataset을 train과 test로 split
     -> 상대적으로 비율이 적은 test_data(0.2, 20%)를 이용
     -> 파일을 복사해주는 모듈인 shutil 모듈을 이용(ctrl+c -> ctrl+v)
   - 분류된 파일을 각각 train 폴더와 test 폴더에 저장
     (폴더는 '미리' 생성 -> 경로 잡아주기)

4. **split 해준 train_data와 test_data의 이름을 다시 넘버링**
   - random을 이용하여 랜덤 이미지가 test_data로 들어감 -> 사진 번호가 섞임
     -> 위에서 정의한 rename() 함수를 호출하여 다시 naming 해주기
   - 3에서 분류한 대로 train_data와 test_data 경로 각각 잡아주고, 다시 naming

5. **이미지 데이터를 모델이 학습할 수 있는 형태로 바꿔주기**
   - CNN 모델을 만들기 위해서는 이미지를 일정한 크기로 resize를 해주어야 한다.
     (DataLoader 만들 때 설정해주면 될 듯 하다.)
   - 이미지 데이터의 형태(type)를 ndarray 형태(5개월차 코랩 2번)나
     tensor 형태로 바꾸어 주어야 한다.(5개월차 코랩 3번 파일)

   -1) ndarray로 형태 변환 -> 라벨링 -> 모델에 학습
      * 5개월차 코랩 2번 파일에 비슷한 내용 있음(*2_텐서플로우와 케라스(2)*)

      * 참고한 사이트: https://m.blog.naver.com/jc3661/222350754899
      * 각각의 이미지의 범주에 대해 라벨링을 할 때, 원핫 인코딩 형태를 사용한 듯 하다.
      * train_data를 array 형태로 변환하고, array 형태로 변환 후 train_data를 255로
        나누어서 0~1 사이의 값으로 scaling을 해줌
        (저흰 일괄적으로 grayscale을 적용할 예정이니, ~~굳이 안해도 되는 과정 같기도...~~)

   - 2) torchvision 이용
      * 5개월차 코랩 3번 파일에 비슷한 내용 있음(*3_파이토치(2)*)

      * 참고한 사이트: https://inhovation97.tistory.com/37
      * transforms.Compose()를 이용해 train_data를 tensor 형태로 변환
      * 픽셀이 자동으로 255로 나눠어 scaling이 자동으로 진행된다.

(아직 저희 프로젝트에 적용해 볼 수 있는 방식으로 코드를 바꿔보진 못했어요..ㅠㅠ
1,2번 중 어떤 방식으로 데이터 변환을 할지 정해지면 수업 자료랑 사이트 참고해서
저희 프로젝트에 어떻게 적용할지 고민해 보겠습니다..!)

==========

● **모델링 과정 고민해보기**

(여기서부턴 저의 주절거림이 시작됩니다..^~^)
- Oxford Pet Dataset 가지고 지금 공부하는 것처럼 순서대로 따라서 해보면 될 것 같다.
- 수업시간에 한 코드를 기본으로 하고, 각각의 상황에 맞춰서 조금씩 변형하기
- Dataloader 만들기
  -> 이미지에 대한 전처리를 마저 하면 될 듯 하다
  -> resizing, augmentation 등
- 모델을 구현하는 방법에는 크게 3가지 방법 정도가 있는데, 어떠한 방식으로 해볼
  것인지 결정하기

  1) 기존 클래스의 옵션 변경하기(*keras.Sequential( )*)
  2) 함수 형태로 layering 진행하기(x 변수 가지고 계속 넘기는 거)
  3) 직접 클래스를 구현한다.(*keras.Model* 상속받아서)
  ~~(4명이니.. 한분은 포폴 정리하고 나머지 3명이 하나씩 해보는 것도..?!)~~