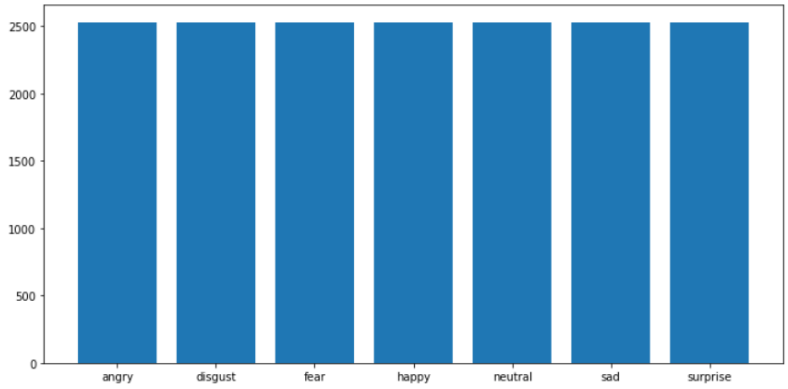
1. dataset 준비하기

- 각 사진들의 개수를 균일하게 맞추어 주기 위해, angry 기준으로 각 감정마다 2531장의 사진을 준비



- 이미지 파일 가져와서 이름 재정의 <- DataFrame을 만들기 위해

  (수업 시간의 Oxford Pet Dataset과 비슷한 형태로 데이터 전처리 해주기 위함)

- 각 감정에 해당하는 사진들에 감정을 labeling

  => 사진 이름(file_name)과 감정(emotion)을 묶어서 DataFrame 형태로 저장

	file_name	emotion
0	angry_000	angry
1	angry_001	angry
2	angry_002	angry
3	angry_003	angry
4	angry_004	angry
...	...	...
17712	surprise_995	surprise
17713	surprise_996	surprise
17714	surprise_997	surprise
17715	surprise_998	surprise
17716	surprise_999	surprise

17717 rows × 2 columns

- 교차 검증 시행 -> train_data와 test_data로 분리 -> 5개로 분리하여 1개는 valid_set으로 사용, 나머지(4개)는 train_set으로 사용

  -> shuffle = True : train_set와 valid_set로 정의된 데이터 셋들이 계속 변경되면서 학습-검증을 반복

  (기출문제와 모의고사의 관계로 비유할 수 있죠)

- fold 값까지 지정 후 DataFrame에 반영

	file_name	emotion	fold
0	angry_000	angry	5
1	angry_001	angry	5
2	angry_002	angry	1
3	angry_003	angry	4
4	angry_004	angry	2
...	...	...	...
17712	surprise_995	surprise	5
17713	surprise_996	surprise	2
17714	surprise_997	surprise	5
17715	surprise_998	surprise	5
17716	surprise_999	surprise	3

17717 rows × 3 columns

- DataFrame을 csv 파일로 저장 -> 나중에 데이터 가져올 때 활용

2. Model 정의/학습/평가

1) Tensorflow Hub 이용

- 어디서나 미세 조정 및 배포 가능한 '선행 학습된' 머신러닝 모델의 저장소
- 몇 줄의 코드만으로 BERT 및 Faster R-CNN과 같은 학습된 모델을 '재사용' 할 수 있음

▶ **EfficientNetV2(Imagenet) 활용**

- 모델 compile
  - input_shape = [None,48,48,3] (4차원 구조/앞은 기억 잘 안나구...뒤의 3개는 순서대로 h, w, color)
  - optimizer : Adam 사용
  - 손실함수(loss function) : 'sparse_categorical_crossentropy' => 정수 형태로 label 전달
  - metrics: 'accuracy'

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer (KerasLayer)    (None, 1280)              5919312

 dense (Dense)               (None, 7)                 8967

=================================================================
Total params: 5,928,279
Trainable params: 5,867,671
Non-trainable params: 60,608
_________________________________________________________________
```

- Albumentation(이미지 변형/증폭)

  -> 수업 내용을 그대로 활용함(굳이 바꿀 필요가 없어 보였음..)

- DataGenerator

· DaraFrame에서 감정의 경우 str(문자열)로 저장이 되어 있음

  -> dict 형태로 감정(key)-정수(value) 형태로 묶어서 모델에 학습시킬 수 있는 형태로 변경

```
csv_path = './feelings_skfold2.csv'

LABEL_INT_DICT = np.unique(pd.read_csv(csv_path)['emotion'])
pprint(LABEL_INT_DICT) # 데이터의 타입과 형태 등도 같이 보여준다.(조금 더 예쁘게 출력해준다?)
LABEL_STR_DICT = {k:v for v,k in enumerate(LABEL_INT_DICT)}
pprint(LABEL_STR_DICT)# Keras의 Sequential model 이용
```

LABEL_INT_DICT

```
array(['angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'surprise'],
      dtype=object)
```

LABEL_STR_DICT

```
{'angry': 0,
 'disgust': 1,
 'fear': 2,
 'happy': 3,
 'neutral': 4,
 'sad': 5,
 'surprise': 6}
```

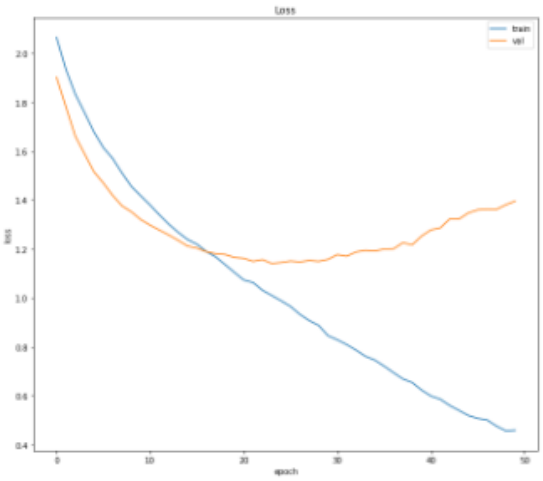
· 대부분의 내용은 수업 내용에서 변경 x -> get_data( ) 부분만 조건 수정

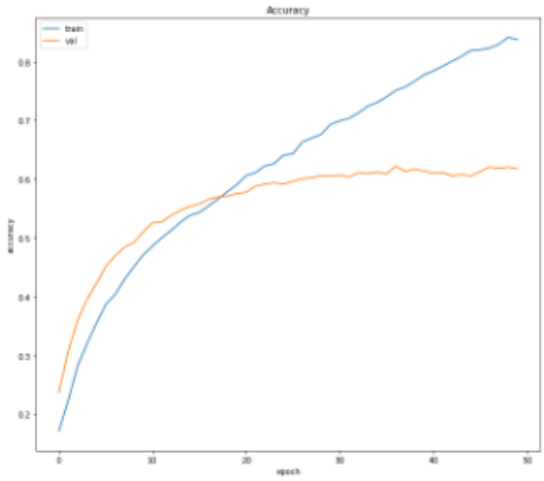
```
def get_data(self, data):
    batch_x = [] #사진
    batch_y = [] # label (감정)

    for _, r in data.iterrows():
        file_name = r['file_name']
        img_folder = r['emotion'] # type = np.str_

        image = cv2.imread(f'datasets/{img_folder}/{file_name}.jpg',cv2.IMREAD_GRAYSCALE)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image = cv2.resize(image, (self.image_size, self.image_size))

        if self.mode == 'train':
            image = image.astype('uint8') # 정수로 형변환
            image = self.transform(image=image)

        image = image.astype('float32') # 실수로 다시 형변환
        image = image / 255. # 0~1 사이의 값을 가지도록


        emotion = str(img_folder) # 사진에 해당하는 감정 찾기(angry, fear, ... )
        emotion = LABEL_STR_DICT[emotion] # 감정(str) -> label(정수)

        batch_x.append(image)
        batch_y.append(emotion)

    return batch_x, batch_y
```

· train_generator, valid_generator 객체 생성

=> batch_size = 128, img_size = 48

· 학습 결과(train-validation)





=> 자세한 내용은 ipynb 파일 참조

- testing

· 새로운 이미지 파일을 가져와서 testing

· 감정별로 2개씩 사진을 폴더에 저장 -> 파일 경로를 순서대로 가져가 사진 파일을 읽고, 모델이 해당 사진에 드러나는 감정을 예측 (angry, disgust, fear, happy, neutral, sad, surprise 순서)

· 이미지 전처리 시 계속 차원 unmatching 문제가 발생해서, keras 모듈의 전처리기 사용 (preprocess_input)

· sad로 다 찍어버리는...불상사가..발생...

```
model이 예측한 1번째 사진의 감정은 sad입니다.
model이 예측한 2번째 사진의 감정은 sad입니다.
model이 예측한 3번째 사진의 감정은 neutral입니다.
model이 예측한 4번째 사진의 감정은 sad입니다.
model이 예측한 5번째 사진의 감정은 neutral입니다.
model이 예측한 6번째 사진의 감정은 sad입니다.
model이 예측한 7번째 사진의 감정은 sad입니다.
model이 예측한 8번째 사진의 감정은 sad입니다.
model이 예측한 9번째 사진의 감정은 sad입니다.
model이 예측한 10번째 사진의 감정은 sad입니다.
model이 예측한 11번째 사진의 감정은 sad입니다.
model이 예측한 12번째 사진의 감정은 sad입니다.
model이 예측한 13번째 사진의 감정은 sad입니다.
model이 예측한 14번째 사진의 감정은 sad입니다.
```

▶ ResNet50 활용

- ImageNet과 전체적인 구조(Albumentation, DataLoader, testing)는 동일함

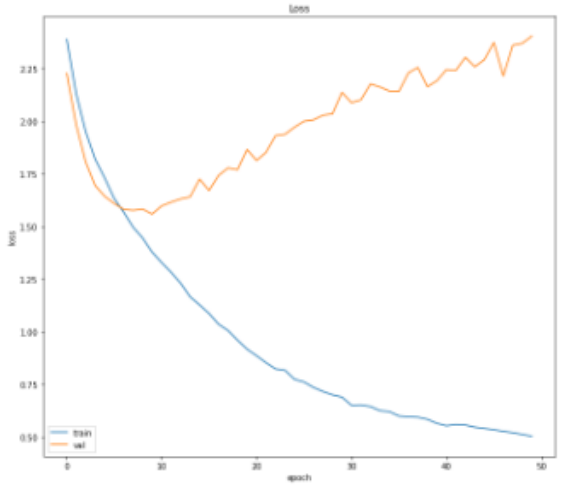
- batch_size = 128, batch_size = 64로 학습을 두 번 시행

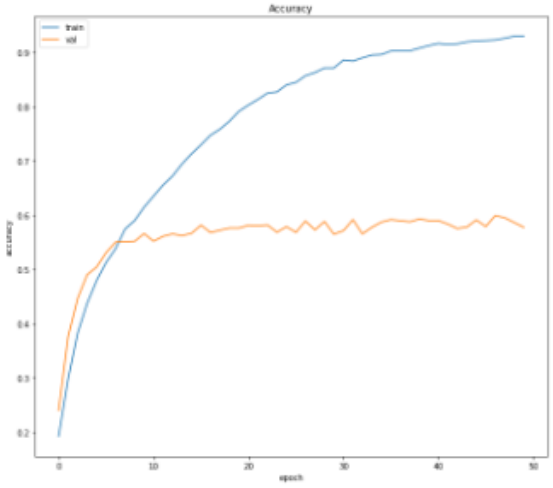
(각각 들어가는 건지, 아님 앞의 학습 결과가 뒤의 학습에 영향을 미치는지...

내용 정리하다 문득 든 생각이라, 강사님께 확인 받아야 할 부분인 듯 해요.)

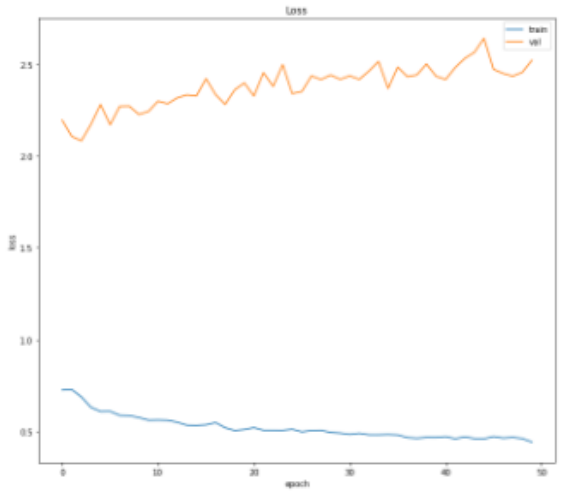
· 학습 결과(train-validation)

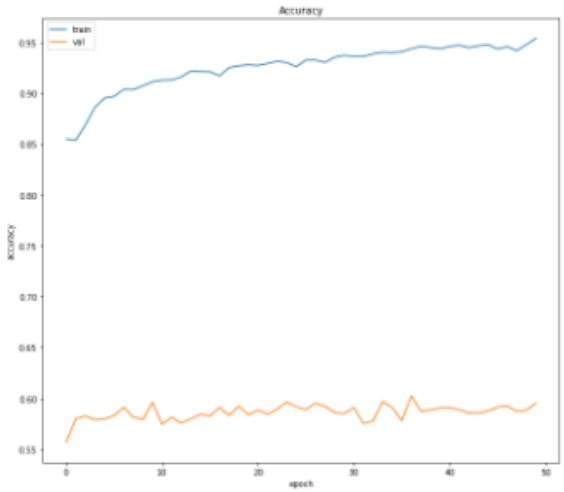
i) batch_size = 128 (history)





ii) batch_size = 64 (history2)





- testing

· 이 아이는요..neutral로 다 찍어버리는...불상사가..발생...

```
model이 예측한 1번째 사진의 감정은 neutral입니다.
model이 예측한 2번째 사진의 감정은 neutral입니다.
model이 예측한 3번째 사진의 감정은 neutral입니다.
model이 예측한 4번째 사진의 감정은 neutral입니다.
model이 예측한 5번째 사진의 감정은 neutral입니다.
model이 예측한 6번째 사진의 감정은 neutral입니다.
model이 예측한 7번째 사진의 감정은 neutral입니다.
model이 예측한 8번째 사진의 감정은 neutral입니다.
model이 예측한 9번째 사진의 감정은 neutral입니다.
model이 예측한 10번째 사진의 감정은 neutral입니다.
model이 예측한 11번째 사진의 감정은 neutral입니다.
model이 예측한 12번째 사진의 감정은 neutral입니다.
model이 예측한 13번째 사진의 감정은 neutral입니다.
model이 예측한 14번째 사진의 감정은 neutral입니다.
```

2) 직접 layer를 쌓는 방식

- 수연님이 모델을 만드신 방법과 비슷한 방식

- kaggle의 예제 코드 보면서 layer를 훨씬 더 deep하게 쌓아봤습니다.

- 4개의 layer + 2개의 연결된 레이어를 쌓음

  => 필터의 수를 64 → 128 → 512 → 512 → Flatten(평탄화) → 256 →512로 반복적으로 증가/감소

(사실 저도 왜 이렇게 하는진 모르고..움...ㅎㅎ 캐글의 예제 코드 응용했습니다..)

· 모델 구조 확인>

https://www.kaggle.com/code/soumyadeepp/face-emotion-recognition/notebook

(모델 시각화하는 코드 jupyterdptj 안되더라고요..ㅠ)

· input_shape = (48,48,1) :순서대로 width, height, color(흑백)

· optimizer = Adam

· 손실함수(loss function): categorical_crossentropy => 원-핫 인코딩된 형태로 label 전달

· metrics: 'accuracy'

- Albumentation(이미지 변형/증폭)

  -> 수업 내용을 그대로 활용함

- Dataloader

  · DaraFrame에서 감정의 경우 str(문자열)로 저장이 되어 있음

  -> dict 형태로 감정(key)-정수(value) 형태로 묶어서 저장

  -> value 값에 해당하는 각 감정의 번호를 원-핫 인코딩 형태로 변환

  (기억 안나서 걍 직접 해줌..ㅎㅎ..)

```
array(['angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'surprise'],
      dtype=object)
====================================
{'angry': 0,
 'disgust': 1,
 'fear': 2,
 'happy': 3,
 'neutral': 4,
 'sad': 5,
 'surprise': 6}
====================================
{'angry': [1, 0, 0, 0, 0, 0, 0],
 'disgust': [0, 1, 0, 0, 0, 0, 0],
 'fear': [0, 0, 1, 0, 0, 0, 0],
 'happy': [0, 0, 0, 1, 0, 0, 0],
 'neutral': [0, 0, 0, 0, 1, 0, 0],
 'sad': [0, 0, 0, 0, 0, 1, 0],
 'surprise': [0, 0, 0, 0, 0, 0, 1]}
```

- DataGenerator: 1)과 동일한 방식

  · get_data 부분에서 label의 값을 원-핫 인코딩된 ndarray 형태로 제공하기 위해 바꿔줌

  · batch_size = 128, img_size = (48,48)

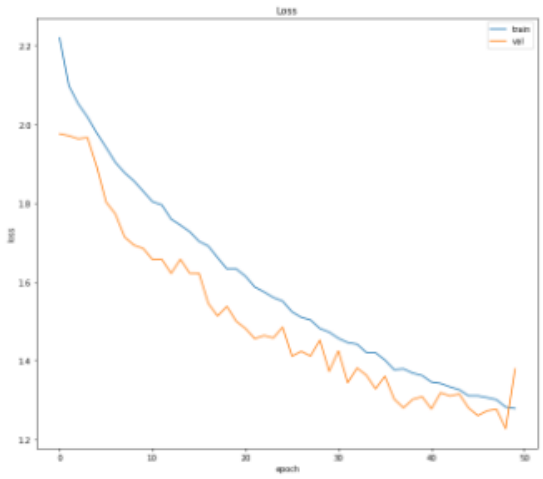
- 콜백함수 지정

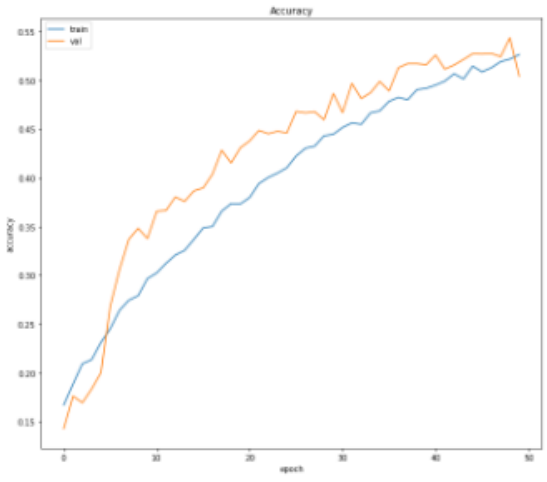
  · Early_stopping

  · Reduce_on_plateau

  → 과적합 방지

· 학습 결과(train-validation)





- testing

val_accuracy가 제일 낮은 모델인데...이렇게 보면 이놈이 제일 잘 맞추네요..?(띠용..)

```
model이 예측한 1번째 사진의 감정은 neutral입니다.
model이 예측한 2번째 사진의 감정은 angry입니다.
model이 예측한 3번째 사진의 감정은 happy입니다.
model이 예측한 4번째 사진의 감정은 sad입니다.
model이 예측한 5번째 사진의 감정은 surprise입니다.
model이 예측한 6번째 사진의 감정은 surprise입니다.
model이 예측한 7번째 사진의 감정은 happy입니다.
model이 예측한 8번째 사진의 감정은 happy입니다.
model이 예측한 9번째 사진의 감정은 happy입니다.
model이 예측한 10번째 사진의 감정은 neutral입니다.
model이 예측한 11번째 사진의 감정은 happy입니다.
model이 예측한 12번째 사진의 감정은 sad입니다.
model이 예측한 13번째 사진의 감정은 surprise입니다.
model이 예측한 14번째 사진의 감정은 surprise입니다.
```

* **포트폴리오/ppt 정리 방향**

- 제가 작성해놓은 순서대로 언급하면 될 것 같아요

  (약간 점점 발전하는 느낌이라..ㅎ)

  (val_accuracy는 사실 반대인 것이 아이러니..^^)

- 맨 마지막에 callback 함수를 넣어준 거 언급하기..?