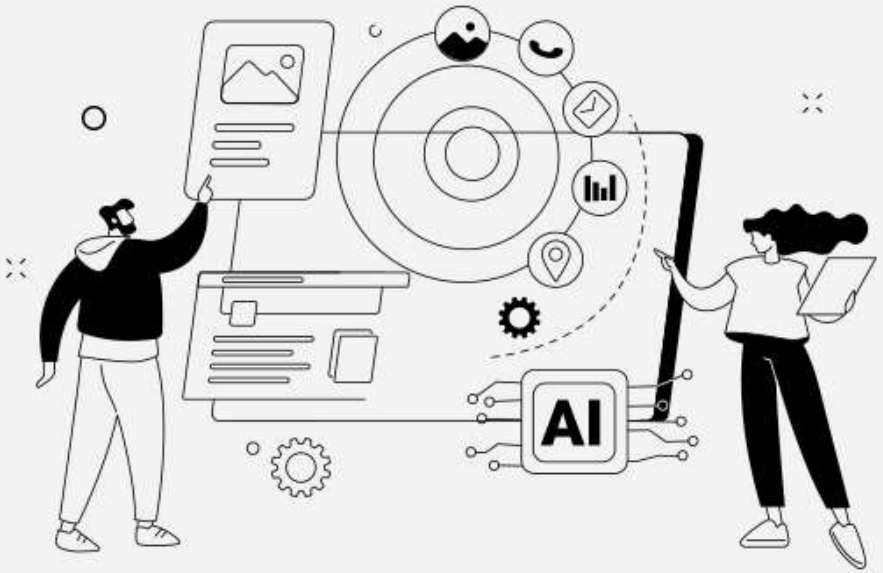
2022 데이터 크리에이터 캠프

Data Creator Camp



대학부- DCC (제갈민, 정재현, 이예지, 주윤나, 채승민)

과학기술정보통신부 NIA 한국지능정보사회진흥원

[Mission #1] 01. EDA_폴더 별 데이터 수

[CLASS별 데이터의 개수]

- Bar plot을 사용하여 결과값 출력함.

→ 클래스 별로 데이터 개수의 불균형 존재함.

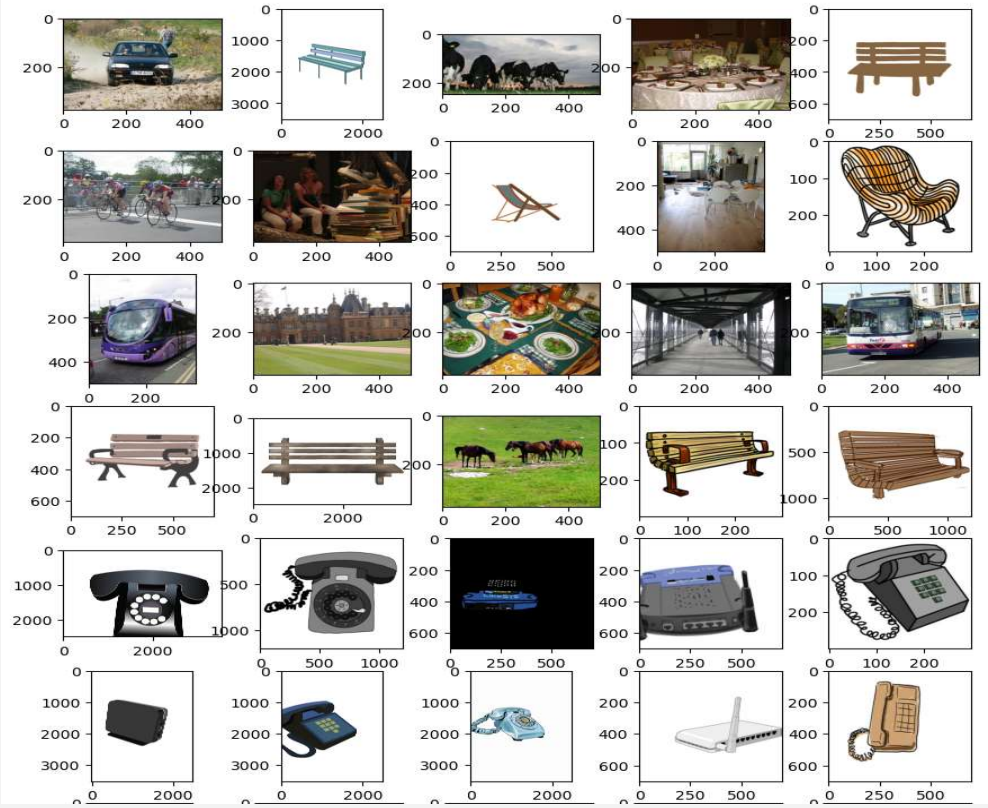
- 각 클래스 별로 데이터의 개수를 정확히 파악하고, 데이터 개수가 크게 차이 날 경우 조정이 필요하기 때문에 진행함.

- 클래스 별로 차이가 많이 나면 (Data Imbalance) F1 score 평균값을 떨어트리기 때문에 데이터 개수가 적은 클래스를 늘림.

→ Data Imbalance의 문제점을 해결해야 함.

[Mission #1] 02. EDA_각 클래스 별 랜덤 이미지 출력

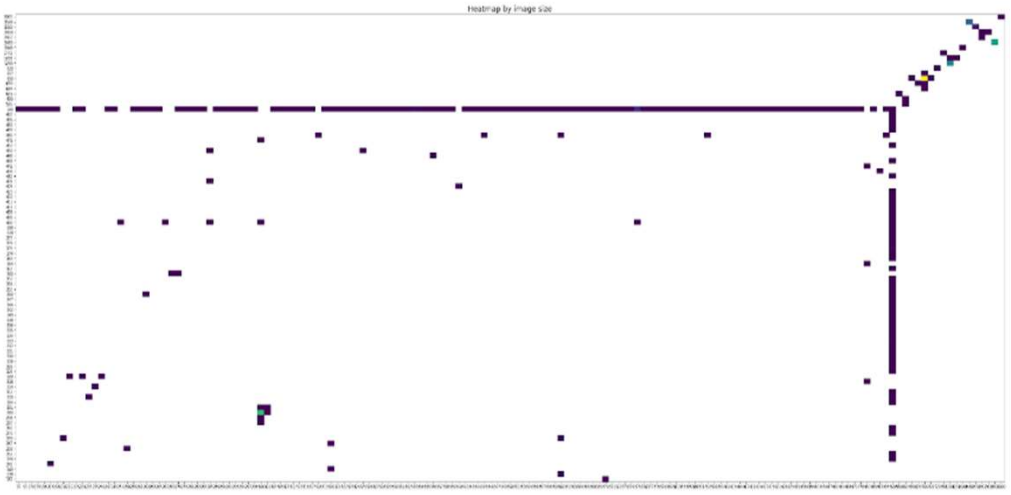
[각 클래스 별 랜덤 이미지 출력]



- PIL을 사용하여 20개의 클래스 별로 랜덤 이미지와 라벨 10개 추출
- 랜덤 이미지 출력 결과 이미지 종류와 크기가 달랐음.
- 임의로 추출하여 어떤 형태의 이미지 있는지 파악
  → {Real Image}, {Sketch Image/ Pictogram/ Illust}
- 이 작업을 통해 먼저 Real Image를 제거해야 함을 알 수 있음.

[Mission #1] 03. 전체 이미지 크기 별 개수 분포

[해상도별 데이터의 개수]



- 랜덤 이미지 출력 결과 이미지 크기가 달랐음.

→ 전체 이미지 크기 별 개수 분포 시각화

→ Heatmap을 이용하여 결과값 출력

- 이후에 resize를 3x224x224로 설정

→ 이미지들의 사이즈가 다 다르기 때문에 Neural Network 에 넣으려면 이미지 크기를 통일해야 함

[Mission #1 수행 결과]

1. 클래스당 개수

클래스마다 데이터 수가 불균형함. -> Augmentation

2. 이미지 크기의 분포

이미지 크기가 각기 다르게 존재 -> 3 x 224 x 224로 설정

3. 이미지의 포맷 및 특성

Real Image, Sketch Image, Pictogram, Illust 가 섞여 있음.

→ Real Image 제거

[Mission #2] 01. Data Cleansing_데이터 전처리

<데이터 전처리>

- 잘못 수집된 영상(real image)과 정상적으로 수집된 영상(picture image)은 다른 특징을 가지고 있음을 eda를 통해 파악함
- 이미지에 대하여 PCA를 진행(100개의 feature 추출)
- L2_3에서 먼저 여러가지 방법론을 적용하여 최적의 Cleansing 기법 찾기

① DBSCAN을 통한 접근

DBSCAN 알고리즘의 노이즈가 잘못 수집된 영상일 것이라고 생각했고, 이에 robust한 DBSCAN적용

-DBSCAN 사용 후 한계 및 문제점-

A. DBSCAN의 parameter(Eps, min_samples)를 특정짓기 어려움

A. 여러가지 수치로 parameter를 설정하고 진행했으나 이미지 분류상 오류가 다수 발견

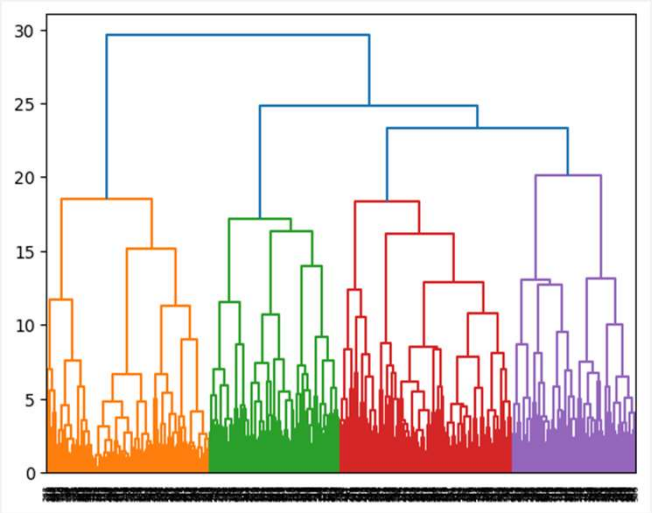
[Mission #2] 02. Data Cleansing_ Agglomerative Clustering 수행

② 병합군집(Agglomerative Clustering)을 통한 접근

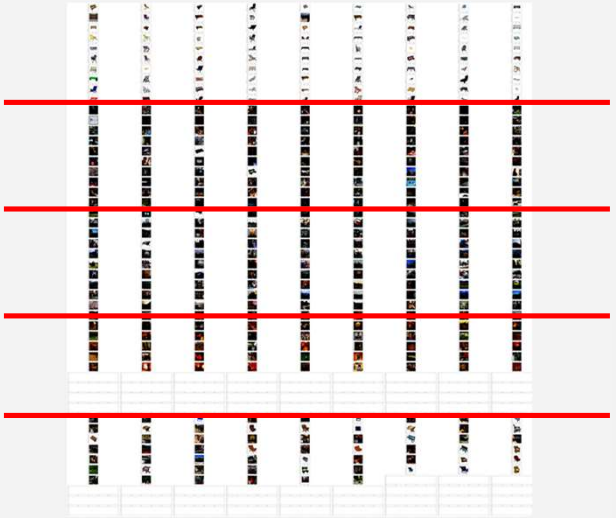
● EDA를 통해 Real, Sketch, Pictogram, Illust 이미지가 있음을 알 수 있었다.

	집단의 개수 = 5
집단 수 설정 이유	최종 보라색 두 집단의 거리가 최대
결과/분석	1= Sketch Image 2, 3, 4.1 = 거의 Real Image 4.2 = 혼합 4.1과 4.2집단이 합쳐지기까지의 거리가 상당히 크다. → 4.2에 Sketch Image도 포함되어 있기 때문 Sketch, Pictogram, Illust가 한 군집, 나머지는 Real Image 로 처음의 예상과 달랐다.

Cluster별 이름 : 1=주황, 2=초록, 3=빨강, 4.1=보라(왼) 4.2=보라(오)



각 집단 별로 이미지를 출력한 결과 위에서부터 cluster 1,2,3,4.1,4.2



[Mission #2] 02. Data Cleansing_ Agglomerative Clustering 수행

② 병합군집 (Agglomerative Clustering)을 통한 접근

- EDA를 통해 Real, Sketch, Pictogram, Illust 이미지가 있음을 알 수 있었다.

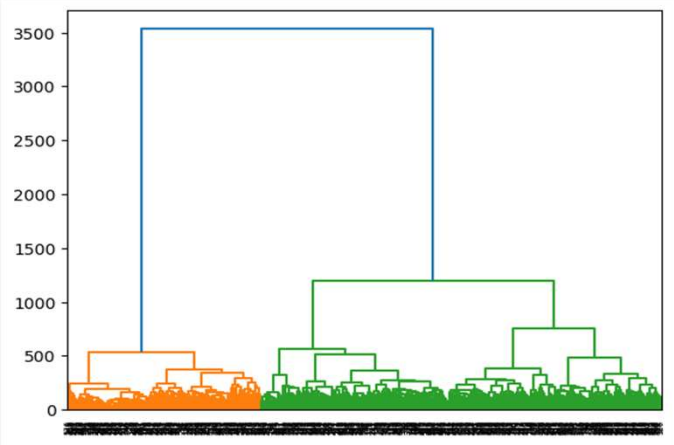
	집단의 개수 = 2
집단 수 설정 이유	K=5분석 결과, real, 기타 이미지로 구분되었기 때문이다.
결과/분석	전반적으로는 리얼이미지와 기타 이미지 (스케치, 일러스트, 픽토그램)가 잘 구분 됐으나 몇몇 오류 발생 멘토링 과정에서 PCA를 적용하지 말고 Raw Data로 구분하자는 idea 를 얻음

실제 이미지 출력

[Mission #2] 02. Data Cleansing_
Agglomerative Clustering 결과

② 병합군집(Agglomerative Clustering)을 통한 접근

	Raw Data로 집단의 개수 = 2
결과/분석	→ 두 집단의 명확한 차이 발생 (마지막 두 집단이 합쳐지기까지의 거리가 약 5 ->2000 로 증가)



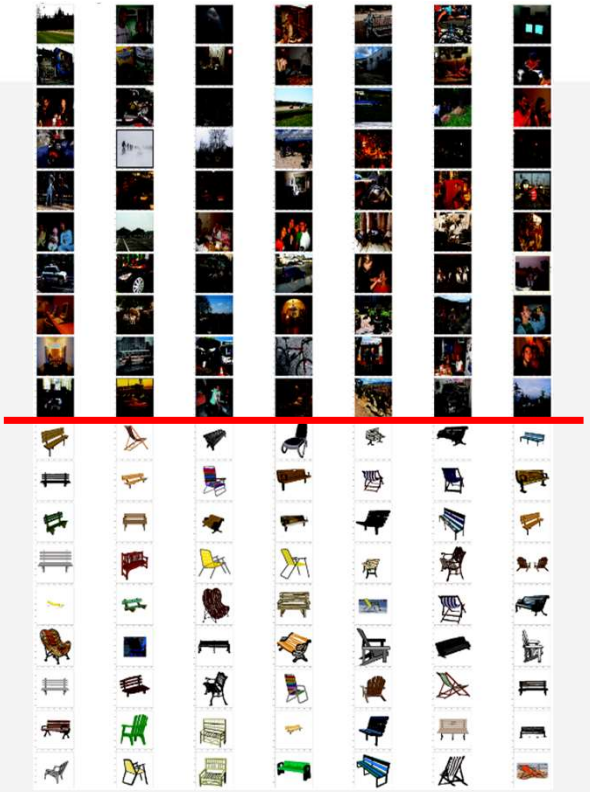
←Raw Data 로 진행한 덴드로그램

K=2일때, agglomerative clustering 분류 결과,
PCA를 적용했을 때보다 더 잘 구분함.

→ 따라서 **agglomerative clustering**을
모든 class의 raw data에 적용해서
Real Image를 제거하는 Data Cleansing을 진행 !

[Data Cleansing 이후 폴더별 이미지의 개수 분포]

L2_25, L2_33은 이미지가 매우 많고,
L2_12, L2_3등은 이미지가 매우 적다.



Raw Data로 K=2일 때의
병합군집 결과, 이미지 출력

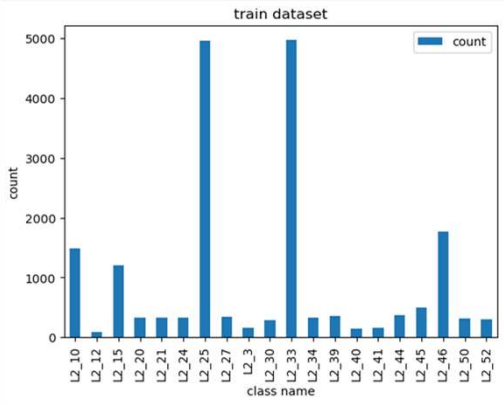
[Mission #2] 03. Data Cleansing_train, test split

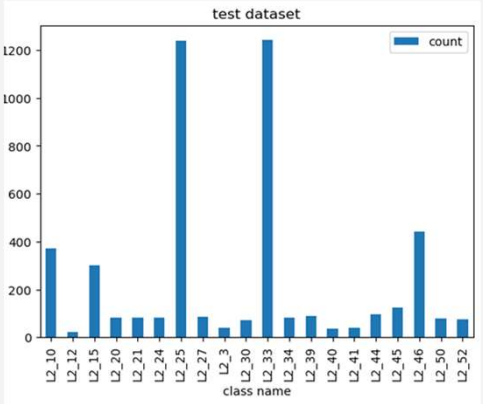
전체 이미지 23454개
Train : Test : Validation = 6 : 2 : 2 class 당 234개의 test image필요

- 그러나 L2_12, L2_3에게는 너무 많은 test image가 필요함 (data imbalance 때문)
- Class 별로 20%를 test image로 사용하고 이를 새로운 폴더에 저장

[폴더로 저장한 이유]

1. Pytorch transformer를 사용하여 모든 학습 데이터(14072개)를 augment하면 class별 불균형 문제 해결 불가
2. 매번 데이터를 나누면, train을 위한 시간이 많이 소요
3. Pytorch의 imagefolder 메소드를 편리하게 사용하기 위하여



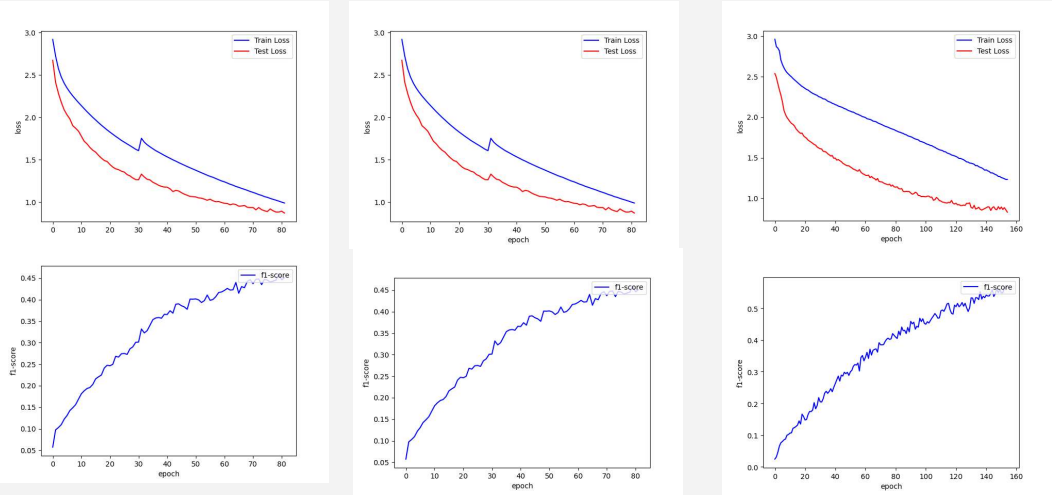


Mission #2 정리

- PCA를 적용하지 않은 Raw Data로 진행하는 것이 효과적
- DBSCAN보다 K=2일 때, Agglomerative Clustering으로 Cleansing을 진행하는 것이 합리적
- Cleansing 진행 후 Class별로 6:2:2로 train, test, validation data를 할당

[Mission #3] hyperparameter tuning – 모델 선정, VGG16 vs ResNet18

Model	Test01	Test02	Test06 (VGG 16사용)
Dataset	augmented_dataset	Random_augmented_dataset	Random_augmented_dataset
Batch size	256	256	128
LR	0.0003	1e-6	1e-6



- 현재 실험은 시간과 컴퓨팅 파워가 제한적

→ **영상/이미지 관련 classification** 수행에서 우수한 결과를 내면서도 layer가 깊지 않은 VGG16과 ResNet18을 선정하여 비교.

- ResNet18을 사용한 test01과 02보다 VGG16을 사용한 test06의 성능이 더 좋았음.
- 향후 실험에서는 **VGG16을 사용하면서 하이퍼파라미터 값 조정** 및 실험에 적합한 데이터셋 탐색

[Mission #3] hyperparameter tuning; Test01, 02 - lr 조절

	Test01	Test02
model	VGG16	VGG16
dataset	train_val_test_dataset	train_val_test_dataset
Batch_size	128	128
lr	0.001	0.01

- 멘토링 데이 때 **멘토님의 조언에 따라 초기 batch size 128로 설정**. 이후 조정함.

- Learning rate를 조정하였을 때 Train loss가 줄어들고 있는지 확인하고자 하였음.
- test02의 결과 train loss가 2.35에서 줄어들지 않았음.
- **결과** 이후 실험에서 1e-6으로 learning rate를 줄이게 됨.

Test 01 결과

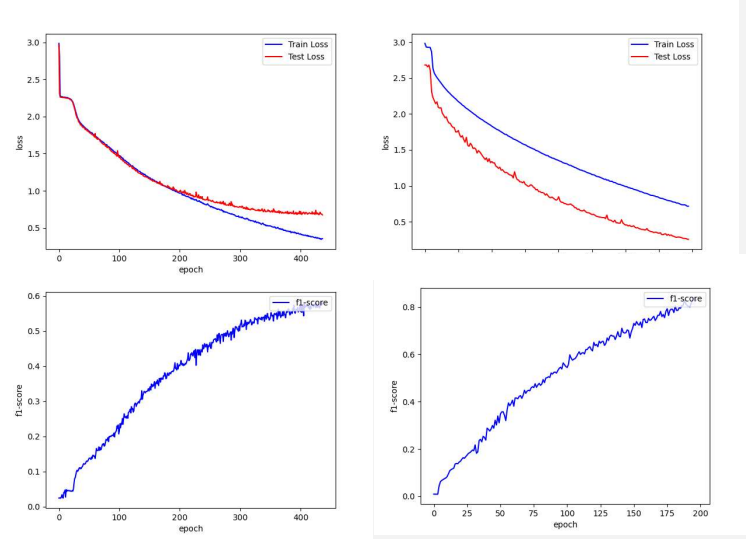
```
| epoch: 48 | accuracy: 26 | precision: 0.0195578231292517 | recall: 0.07142857142857142 | f1-score: 0.030707610146862487 |
| epoch: 49 | accuracy: 26 | precision: 0.013392857142857142 | recall: 0.0625 | f1-score: 0.022058823529411766 |
| epoch: 50 | accuracy: 26 | precision: 0.020634920634920634 | recall: 0.06666666666666667 | f1-score: 0.03151515151515152 |
```

Test 02 결과

```
epoch:  429 | accuracy:  26 | precision: 0.01955782 | recall: 0.07142857 | f1-score: 0.03070761 | test loss: 2.27326431 | train loss: 2.3524345
epoch:  430 | accuracy:  26 | precision: 0.02579365 | recall: 0.08333333 | f1-score: 0.03939393 | test loss: 2.27233625 | train loss: 2.3487314
epoch:  431 | accuracy:  26 | precision: 0.01785714 | recall:    0.0625 | f1-score: 0.02777777 | test loss: 2.27283615 | train loss: 2.3560508
```

[Mission #3] hyperparameter tuning; Test03, 04 - dataset 및 batch size 변경, **실험오류** 발견

Model	Test03	Test04
Dataset	train_val_test_dataset	augmented_dataset
Batch size	128	256
LR	1e-6	1e-6



Batch size가 달라졌을 때의 결과를 보고자 test03과 04를 진행하였는데, 이상하게도 f1-score가 상당히 높다는 점을 발견함.

-> Valid dataset과 Train dataset을 비교해보니, 중복되는 데이터 존재, test03, 04의 경우 실험이 잘못되었다고 판단함.

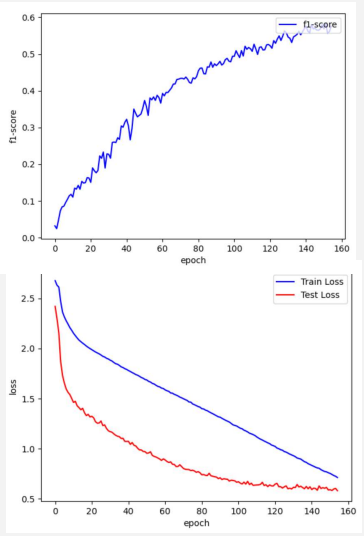
➔ **train, test로만 나눠진 final 폴더에서 Augmentation을 진행했기에 오류 발생. Augmentation을 새로 진행하고 Test 실행**

[Mission #3] hyperparameter tuning; Test07, 08, 09 - focal loss 도입

Model	Test07	Test08	Test09
Dataset	Random_augmented_dataset_v2	Random_augmented_dataset	Random_augmented_dataset_v2
Batch size	128	128	128
LR	1e-6	1e-6	1e-6

```
+----------------------------------------+
| epoch:                            134 |
| f1-score:                  0.56909968 |
| accuracy:                         76% |
| precision:                 0.61971644 |
| recall:                    0.57106663 |
| test loss:                 0.89729725 |
| train loss:                1.22260872 |
| time spent (epoch):           0:03:56.25 |
| time spent (total):           8:53:52.02 |
+----------------------------------------+
```

Test 07 결과



Test 09 결과

Test08부터 focal_loss 사용

Focal loss를 사용하여 Test07과 같은 테스트 test09를 진행.

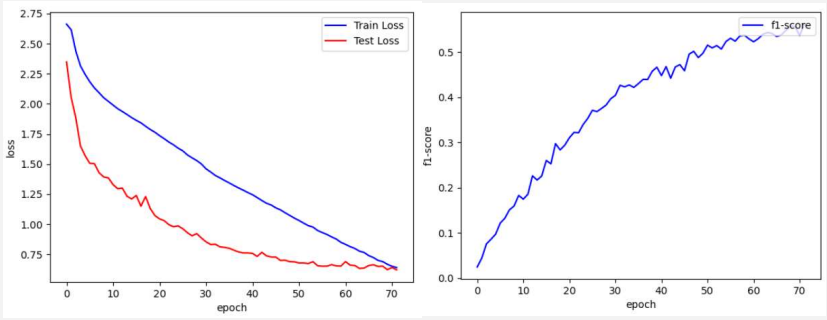
-> Focal loss 사용 이유: data imbalance에 특화된 loss임

-> test07보다 focal loss를 적용한 test09의 결과가 더 좋게 나와 이후에도 focal loss 사용

[Mission #1] data imbalance 문제 해결_dataset 수정 및 변경

Model	Test09	Test10	Test11
Dataset	Random_augmented_dataset_v2	Random_augmented_dataset_v3	Random_augmented_dataset_v4
Batch size	128	128	128
LR	1e-6	Epoch 당 다르게 부여	Epoch 당 다르게 부여

Test10의 경우부터 최대 클래스의 데이터 수를 줄이고 최대한 data imbalance한 상태에서 실험 진행.



Test 10 실행 결과

epoch:	93
f1-score:	0.58044976
accuracy:	77%
precision:	0.62090986
recall:	0.58596489
test loss:	0.67164365
train loss:	0.35707638
time spent (epoch):	0:02:31.00
time spent (total):	3:55:51.57
epoch:	94
f1-score:	0.59226787
accuracy:	77%
precision:	0.62656174
recall:	0.60784949
test loss:	0.66175897
train loss:	0.35198387
time spent (epoch):	0:02:30.52
time spent (total):	3:58:22.09

Test 11 실행 결과

<결과 분석>

1. Test10의 f1-score가 test09보다 높게 나와 test11에서 더 데이터를 제거해봤으나 더 낮은 f1-score가 나옴
   ➔ 이후 실험부터 **random_augmented_dataset_v3**로 실험 진행
2. Test10의 결과가 가장 좋음 → **test10의 모든 변인 유지**
3. **Accuracy가 높음에도 test loss가 높을 수 있음을 발견함**

[Mission #3] hyperparameter tuning; Test12, 50_test01, 50_test02 실험

Test 명	Test12	50_test01	50_test02
Model	VGG 16	ResNet 50	ResNet 50
Dataset	Random_augmented_dataset_v3	Random_augmented_dataset_v3	Random_augmented_dataset_v3
Batch size	128	128	128
LR	Test 10처럼 epoch마다 lr를 다르게 줌	1e-6	2e-6, 1e-6 에서 10 epoch 씩 학습, 나머지는 1e-7에서 학습

Augmentation, loss의 방법을 바꿔도 data imbalance한 문제로 모델을 변경해보고자 시도

Deep Residual Learning for Image Recognition (Kaiming He 외 2인,2015)에 소개되어 있는 ResNet 50을 사용했으며 torchvision의 코드를 사용하여 ResNet50 class도 만듦

Imbalance한 데이터에서 batch 데이터를 가져올 때, balance하게 가져오기 위한 샘플링 방법인 **WeightedRandomSampler**을 사용해서 테스트를 진행

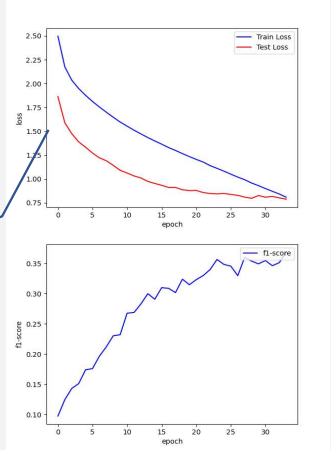
[Mission #3] hyperparameter tuning;
Test12, 50_test01, 50_test02 실험 결과 및 모델 설명

epoch:	82
f1-score:	0.58772429
accuracy:	77%
precision:	0.62420090
recall:	0.59941829
test loss:	0.65238190
train loss:	0.19781100
time spent (epoch):	0:02:47.60
time spent (total):	3:50:37.07

Test12

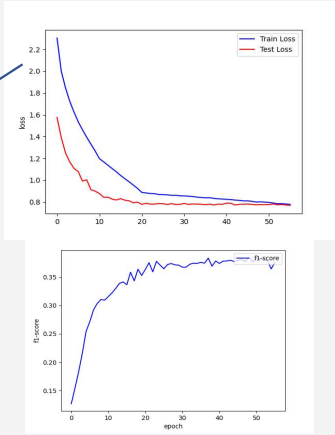
Test_loss 수렴값이 너무 커서 이후부터는 lr를 줄이고 학습하기로 결정

50_test01



train_loss가 줄어듦에도, test_loss가 줄어들지 않는다.(results폴더 확인) 따라서 learning rate의 문제가 아니라고 확신했고, ResNet101을 사용해 보기로 결정

50_test02



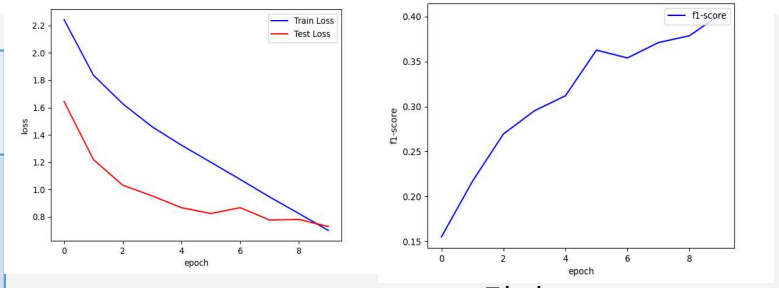
여러가지 시도에도 f1-score가 우리가 기대하는 수준에 미치지 못함

→모델의 layer가 영상의 feature를 잘 extract할 수 있을 만큼 깊지 않아 생기는 문제라고 판단

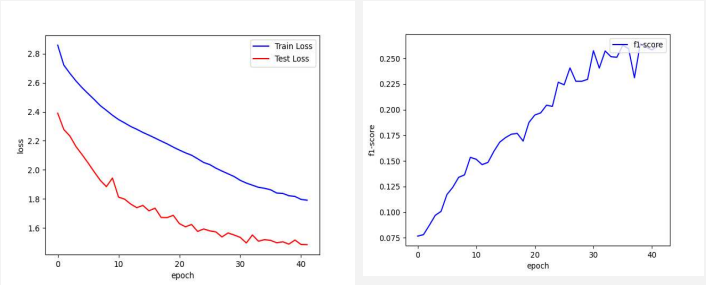
→ ResNet-50과 ResNet-101로 모델 변경을 시도

[Mission #3] hyperparameter tuning; 101_test01, 101_test02, 101_test03

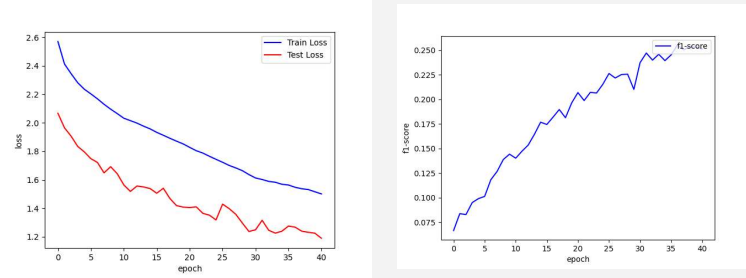
Test 명	101_test01	101_test02	101_test03
loss	Focal loss	Cross Entropy Loss	Focal loss
Model	ResNet101		
Dataset	Random_augmented_dataset_v3	Random_augmented_dataset_v4	
Batch size	128		
LR	Test 10처럼 epoch마다 lr를 다르게 줌		



101_test01 결과



101_test02 결과



101_test03 결과

"ResNet101이 VGG16보다 좋은 성능을 보이지 못함."

- → ResNet이 imbalance에 더 민감한 모델일 수 있다고 생각하여, random_augmented_dataset_v4로 시도했지만 결과는 좋지 못함
- → dropout 방법의 유무로 인해 VGG16이 더 우수했을 것이라고 결과를 분석
- → 결국, **VGG16 모델을 수정한 VGG16_V2를 시도함**

[Mission #3] hyperparameter tuning; VGG16_v2의 test01, 02

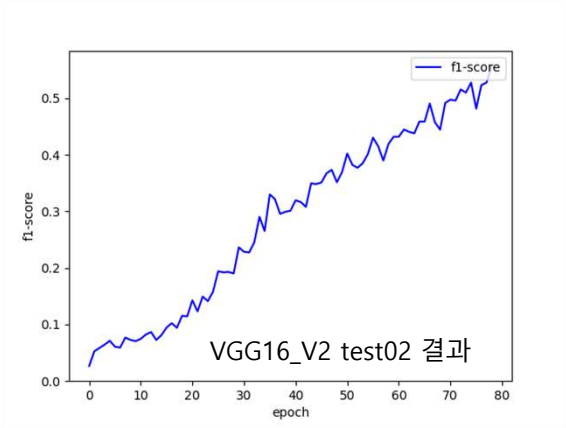
```
self.classifier = nn.Sequential(
    nn.Linear(in_features=25088, out_features=4096, bias=True),
    nn.ReLU(inplace=True),
    nn.Dropout(p=0.5, inplace=False),
    nn.Linear(in_features=4096, out_features=4096, bias=True),
    nn.ReLU(inplace=True),
    nn.Dropout(p=0.5, inplace=False),
    nn.Linear(in_features=4096, out_features=1024, bias=True),
    nn.ReLU(inplace=True),
    nn.Dropout(p=0.5, inplace=False),
    nn.Linear(in_features=1024, out_features=1024, bias=True),
    nn.ReLU(inplace=True),
    nn.Dropout(p=0.5, inplace=False),
    nn.Linear(in_features=1024, out_features=512, bias=True),
    nn.ReLU(inplace=True),
    nn.Dropout(p=0.5, inplace=False),
    nn.Linear(in_features=512, out_features=outputs, bias=True)
)
```

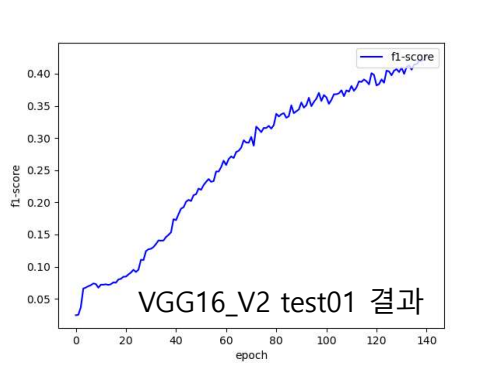
VGG16을 수정한 VGG16_V2의 코드

출력단이 20개만 필요한 우리 데이터에 적합하게 하기 위해 위와 같이 VGG16 마지막단의 fully connected layer를 수정

Test 명	Test01	Test02
Model	VGG16_V2	VGG16_V2
Dataset	Random_augmented_dataset_v3	Random_augmented_dataset_v5
Batch size	128	
LR	1e-6	

같은 조건 하에 학습 데이터를 90% 까지 늘린 dataset_v5를 만들어, 최종 classifier를 만듦





[Mission #3] 결과 및 전체 결과 정리

실험 과정:

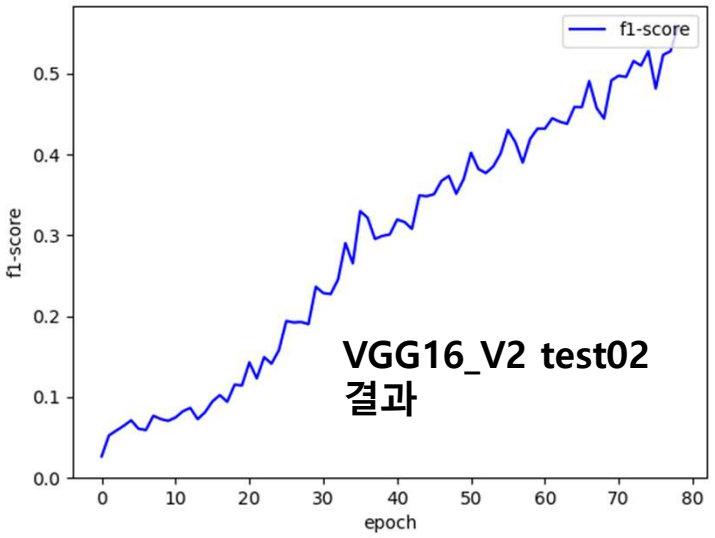
- 모델의 구조와 알려진 성능을 고려하여 실험에 사용할 모델을 선정(계획)하고 실험을 통해 최적의 모델을 찾아냄
- 동일한 데이터, 모델 조건 속에서 영상 처리, 신경망 네트워크 구조에 따라parameter 의 값을 계획, 결과 분석을 통한 hyperparameter 값 선정
- 데이터 imbalance 문제를 해결하기 위해 같은 hyperparameter 조건하에 이전 테스트 결과 및 데이터, 영상처리 특성에 따라 데이터 셋을 7번에 걸쳐 실험, 결과분석 및 피드백으로 수정

결과:

data imbalance 문제에 집중하여,

1. augmentation
2. random augmentation
3. undersampling, dropout
4. focal loss

등의 방법을 사용하여 문제를 해결!



컴퓨터 드로잉툴을 사용해 그린 일러스트레이션 영상을 효과적으로 분류할 수 있는 분류기를 VGG16_V2 test02 로 만들었다고 결론 내림

참고문헌

- https://www.kaggle.com/code/sungjunghwan/eda-images-view-processing
- Focal Loss for Dense Object Detection Focal loss (Tsung-Yi Lin 외, 2018)
- Deep Residual Learning for Image Recognition (Kaiming He 외 2인,2015)
- Transfer Learning for Illustration Classification (Manuel Lagunas외 1인, 2018)
- Introduction to Neural Networks in Java, Jeff Heaton, AbeBooks

감사합니다

2022 DATA CREATOR CAMP

과학기술정보통신부 NIA 한국지능정보사회진흥원