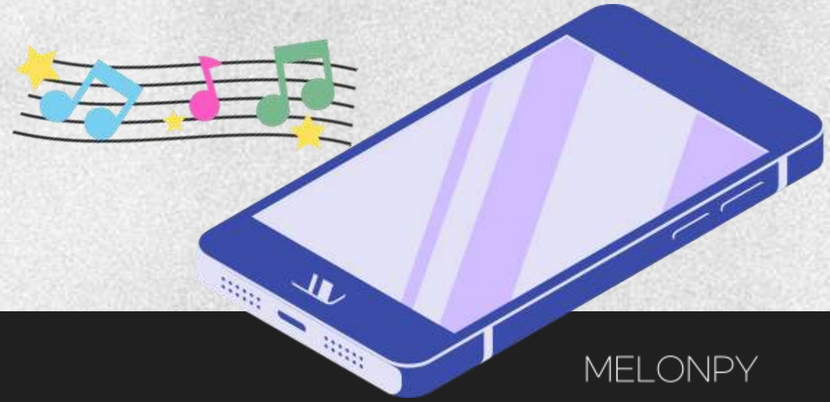
SPECIAL

TED PREMIUM_HIGHT_GUALITY LIMITED

WELL_MADE



CON-TENTS

김근태

허유나

오승준

이승수

01

요구사항 정의서

MELONPY

대분류	중분류	소분류	기능설명	우선순위	담당자
이미지 처리	1. 이미지 데이터 준비	1-1 이미지 수립	멜론 년도별 앨범이미지 크롤링	1	김근태
	2. 데이터 처리	2-1 이미지 사이즈	흑백 이미지 사이즈 48 * 48 조정	2	각자
		2-2 이미지 증폭	추가 수집 및 각도위치 조정 이미지 추가	2	각자
		2-3 데이터 정규화	데이터 정규화	2	각자
	3. 이미지 라벨링	3-1 시대별 라벨링	년도별 시대 라벨링	2	각자
분석	4. 모델링	4-1 이미지 데이터 분석	RGB 값 분석, 특징 추출	3	각자
		4-2 이미지 데이터 학습	딥러닝(CNN,DNN,전이학습 ...) 다양한 방법 사용	3	각자
		4-3 모델 검증	loss, accuracy 확인	4	각자
		4-4 예측 및 시각화	테스트 데이터 예측 및 과정 시각화	4	각자
	5. 결론	5-1 결론	결론 도출	5	각자

개발 환경

〉〉 O/S : Window

〉〉 언어 : Python

〉〉 라이브러리 : pandas, numpy, matplotlib, cv2
pillow, sklearn, keras

02

주제 선정 배경

MELONPY

앨범 이미지 시대별 특징 분석

- 시대별 앨범 이미지 분류
- 시대별 트렌드 분석

최종목표

시대별 특징을 첨가해서 새로운 앨범이미지를

만들어 주는 프로그램을 구현

02

주제

앨범

-

-

최종

MELONPY

앨범자켓만으로 장르 구분하는 AI 나온다

박혜섭 기자 입력 2020.09.10 17:03 댓글 0 좋아요 0

(사진=셔터스톡).

데이터 과학자들이 한데 모여 앨범자켓만으로 장르를 구별해내는 AI를 개발했다.

엔터프라이즈AI가 9일 보도한 바에 따르면 머신러닝 자동화 플랫폼을 제공하는 업체 데이터로봇의 소속 과학자들은 사내 AI 도구를 이용해 이러한 연구를 진행했다.

연구진은 연구를 위해 스포티파이의 오픈소스를 활용했다. 스포티파이는 세계 최대 온라인 음악 스트리밍 사이트로 사용자의 취향별로 음악을 셔플링 하는 데 탁월한 기능을 자랑한다.

03

데이터 출처 및 수집

MELONPY

데이터 출처

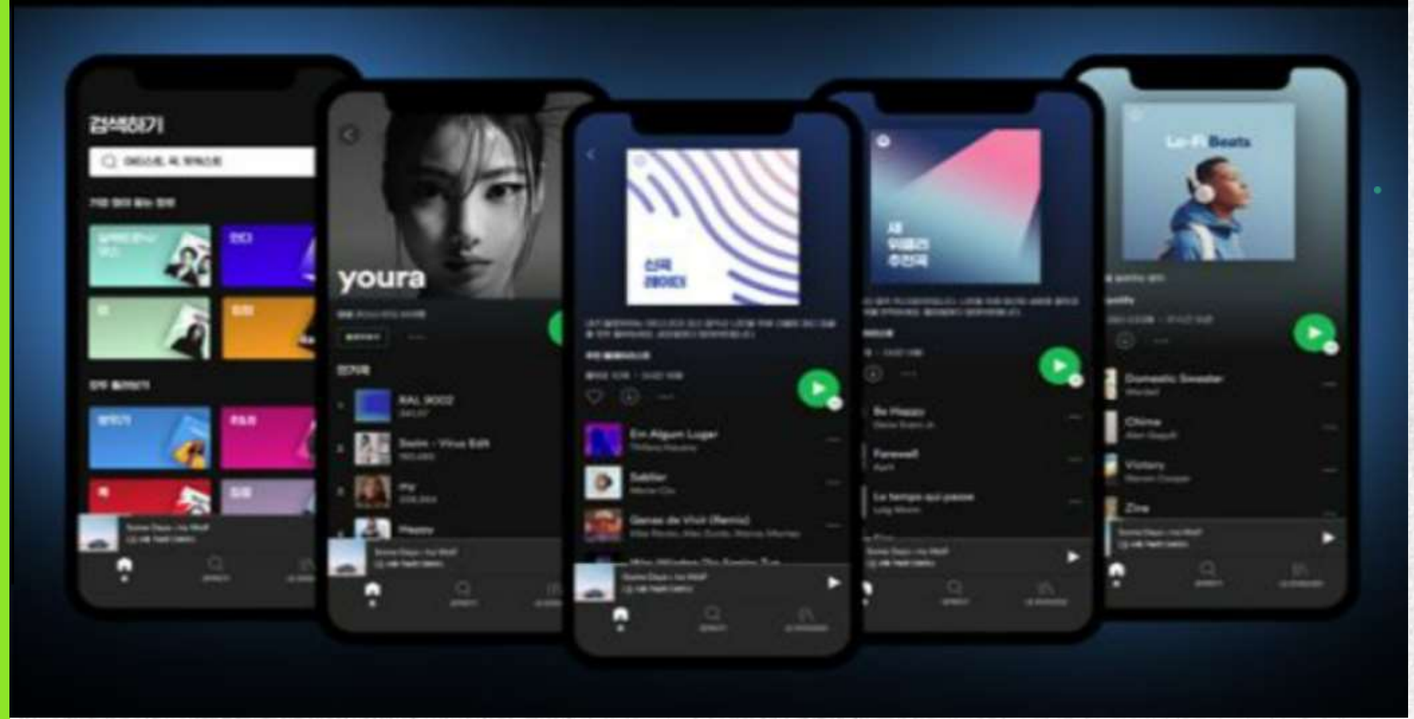
멜론 년도별 순위 top100

https://www.melon.com/chart/age/index.htm?chartType=AG&chartGenre=KPOP&chartDate=2000

데이터 수집방법

동적 크롤링

멜론 파이



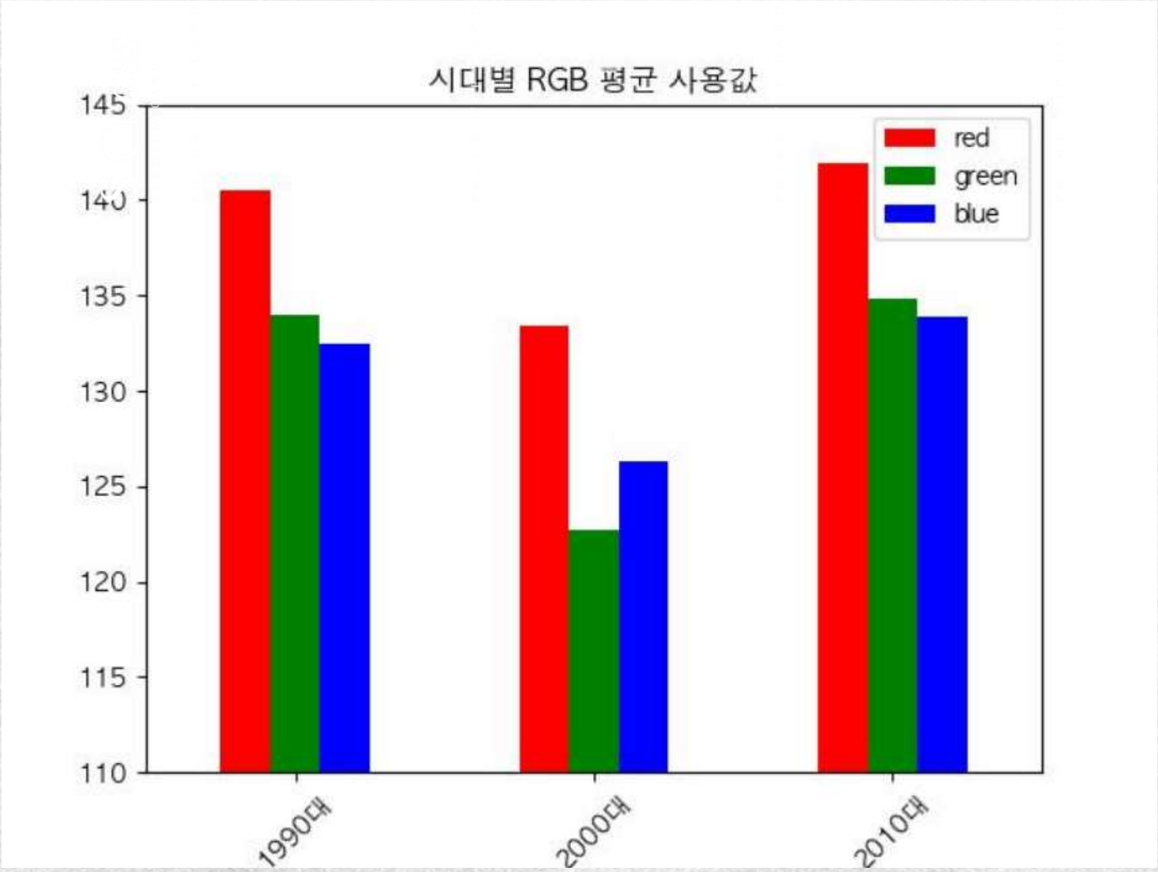
데이터 분석 및 결론

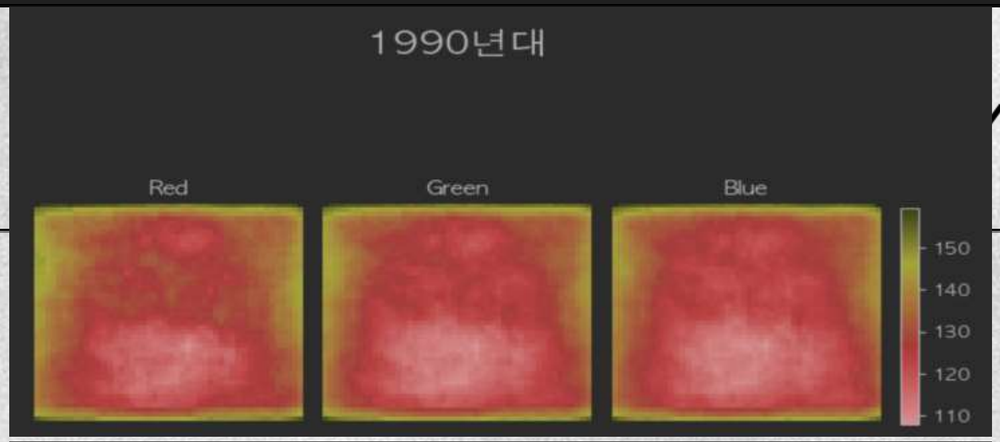
MELONPY

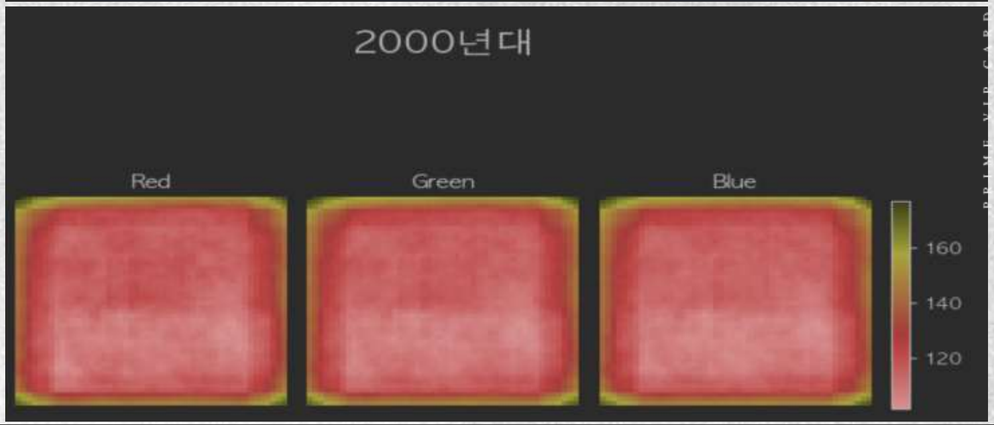
01

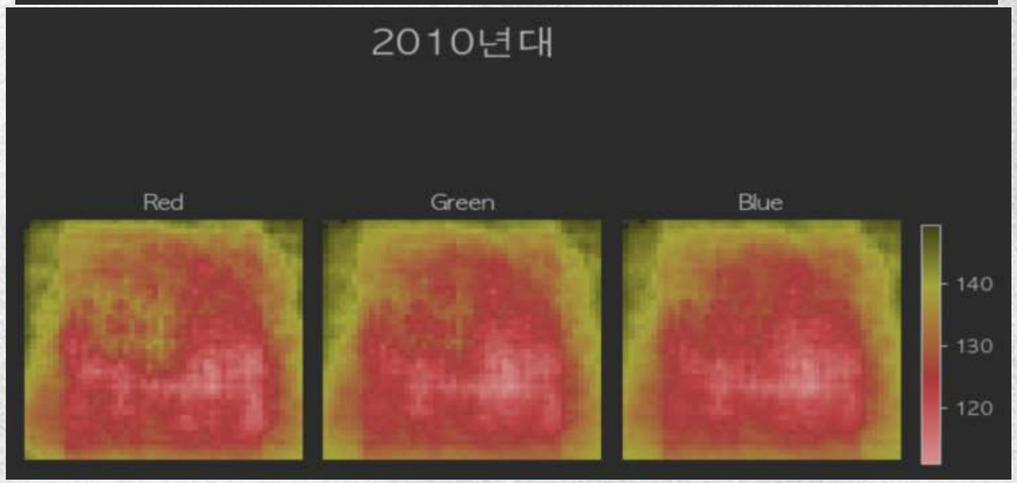
모델링 타당성 확인

시대 별 앨범 이미지 차이
1990년대, 2000년대, 2010년대



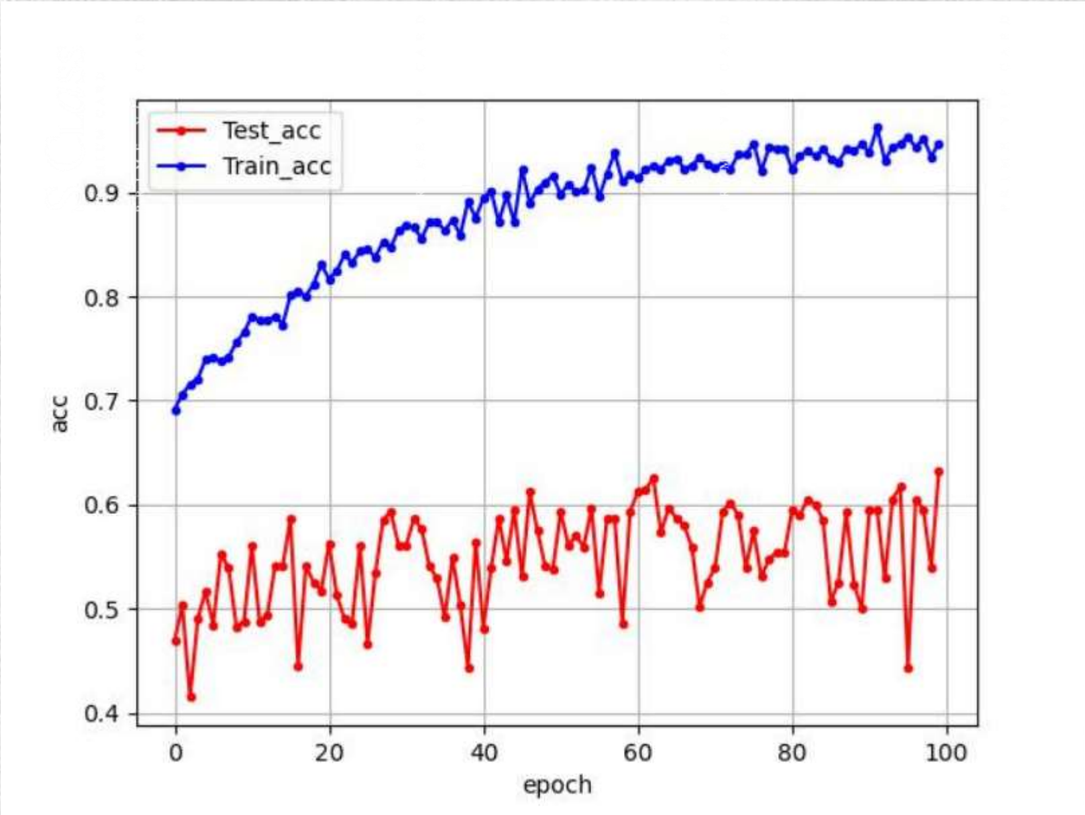


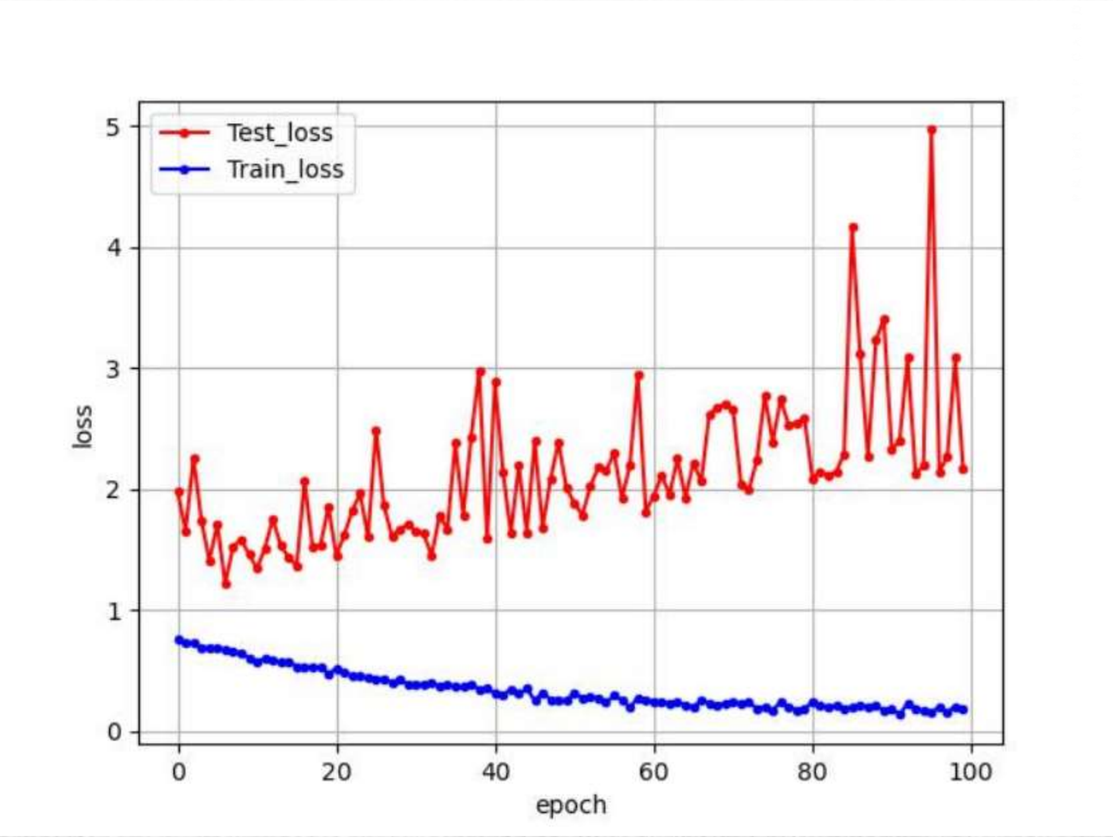


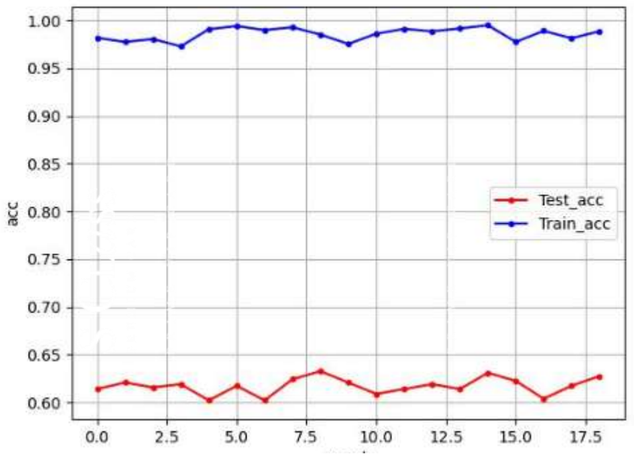


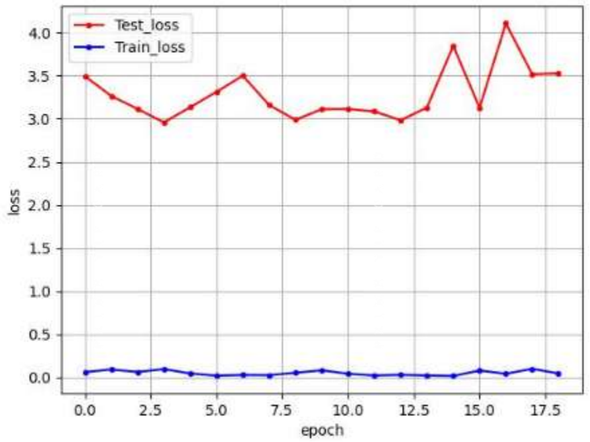
MELONPY

DNN

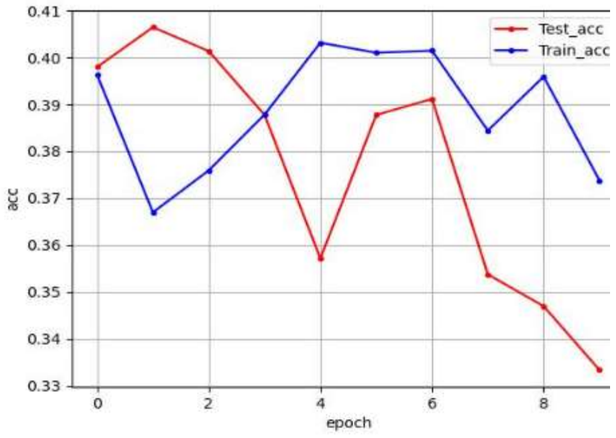

Test_acc
Train_acc
acc
epoch
0.4
0.5
0.6
0.7
0.8
0.9
0
20
40
60
80
100

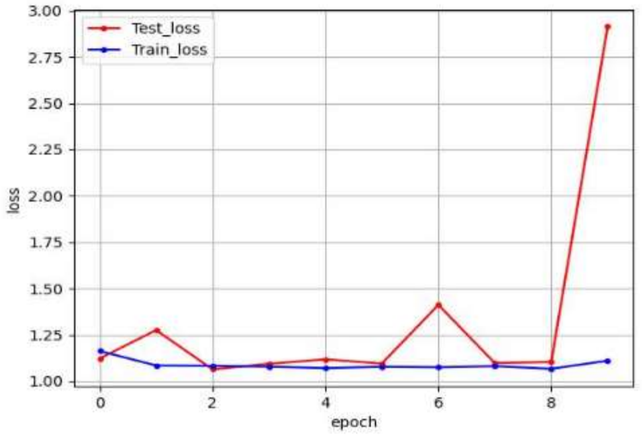

Test_loss
Train_loss
loss
epoch
0
1
2
3
4
5
0
20
40
60
80
100





기본

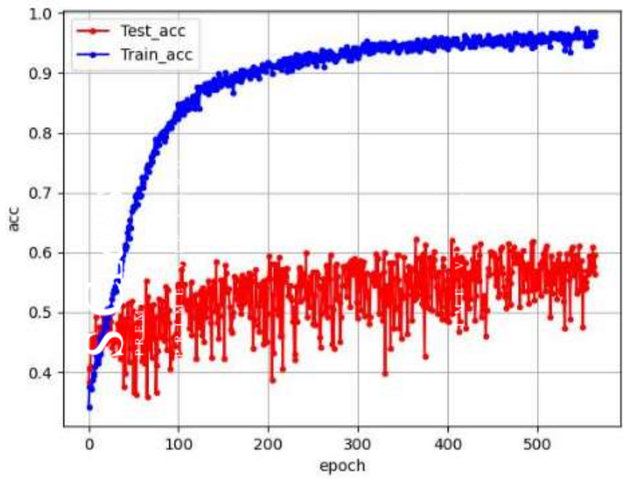


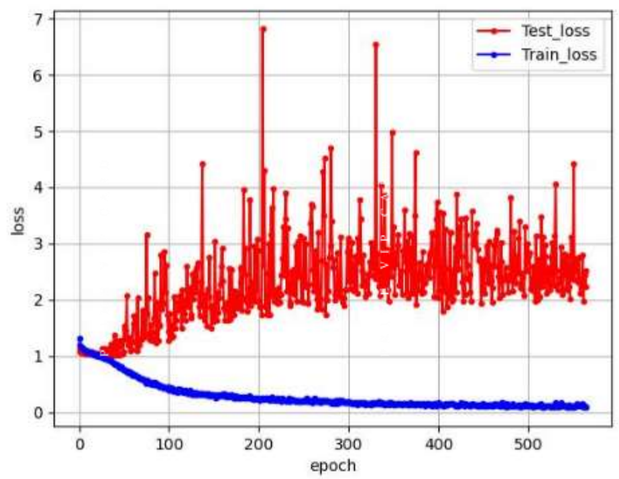


배치정규화 적용

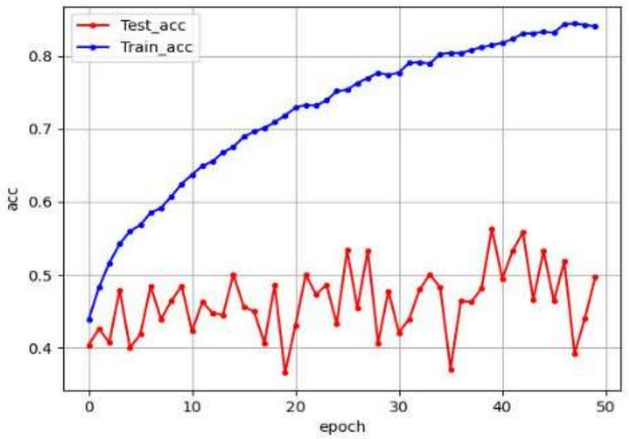
MELONPY

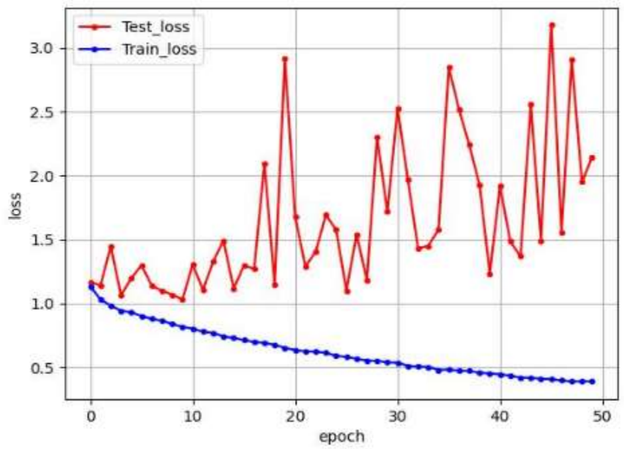
전이학습





ResNet50



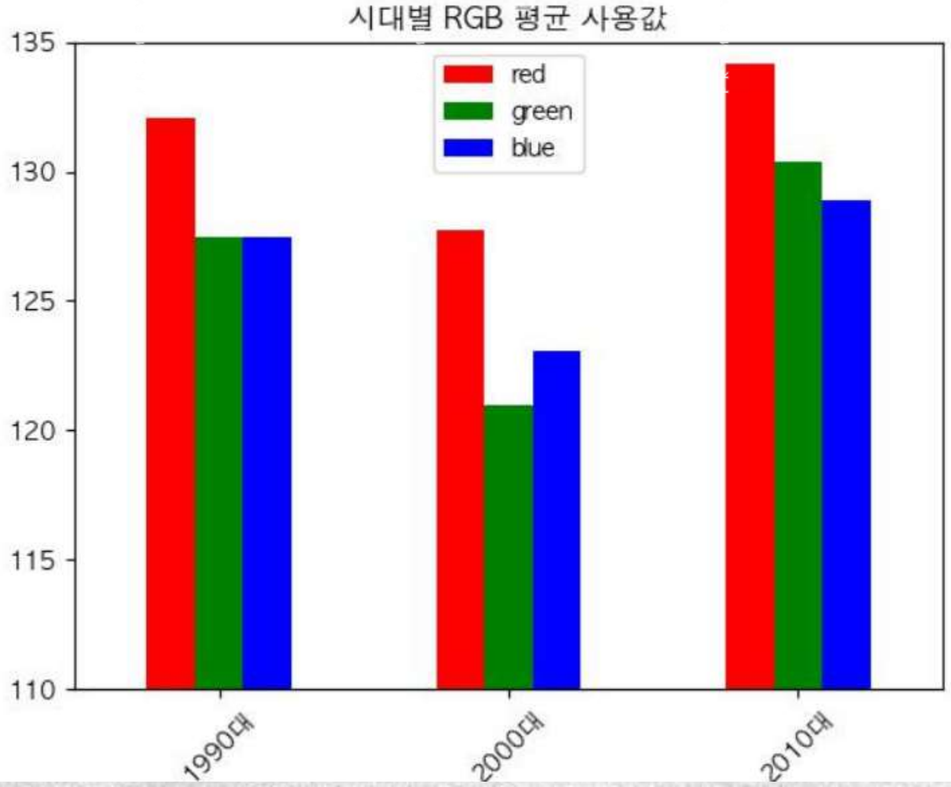


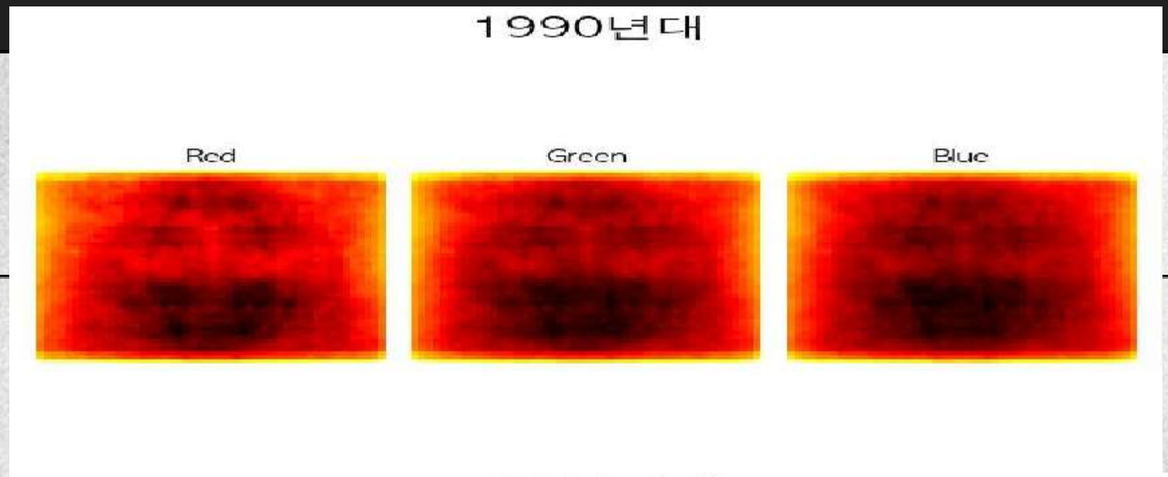
ResNet50 + 규제

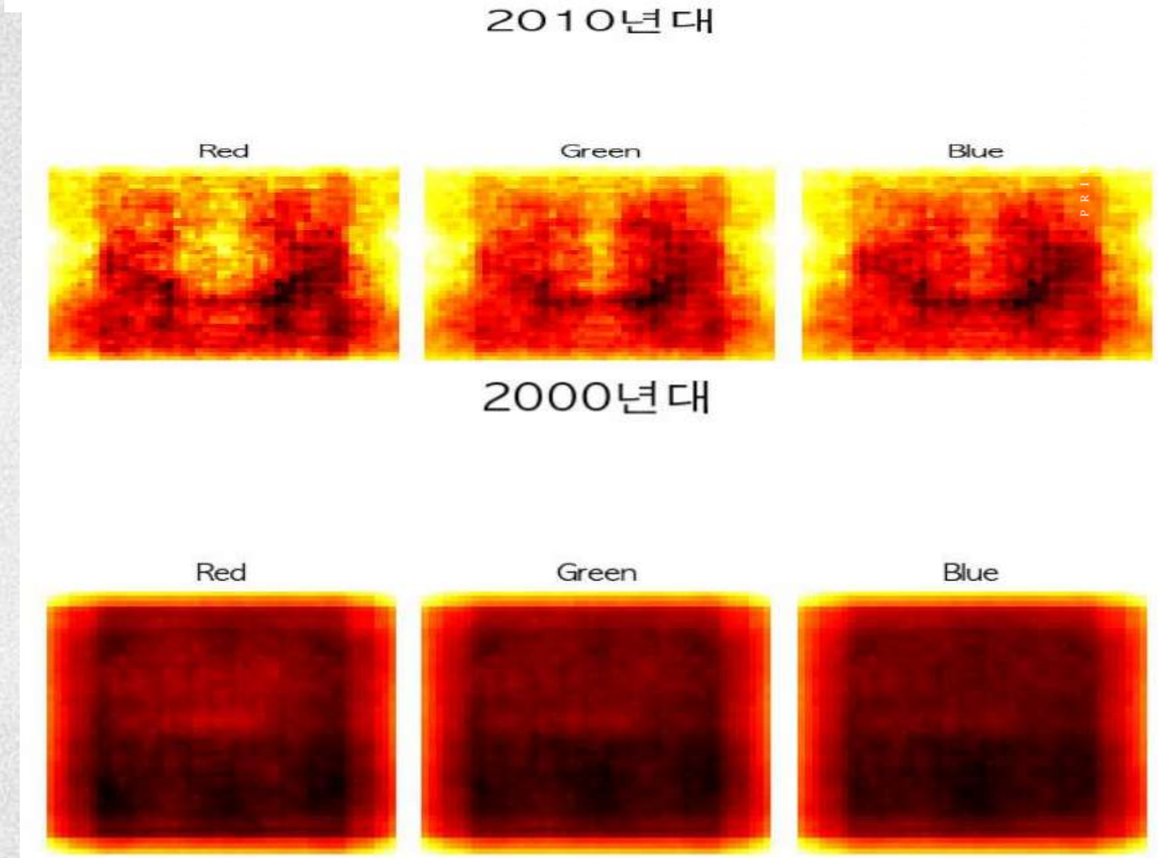
01

이미지 증강

시대 별 앨범 이미지 차이
1990년대, 2000년대, 2010년대



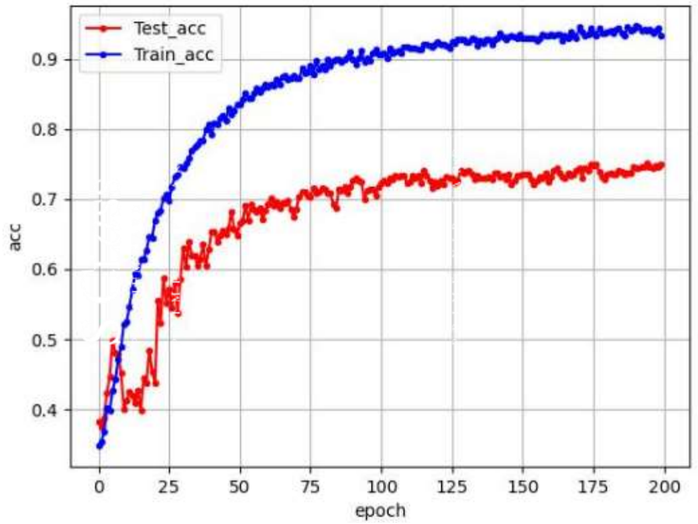


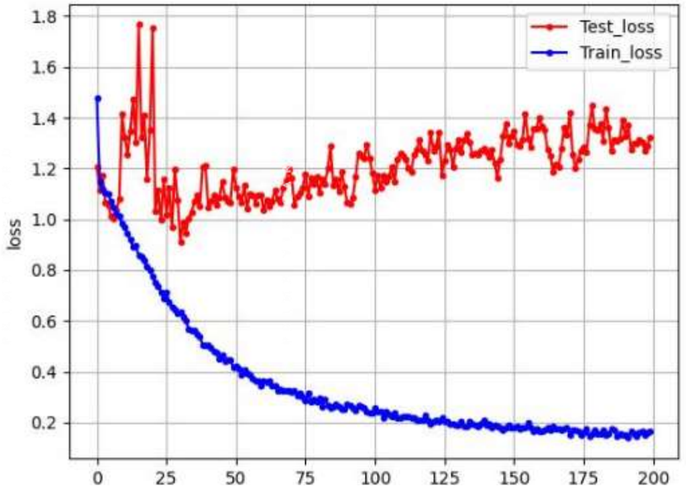


MELONPY

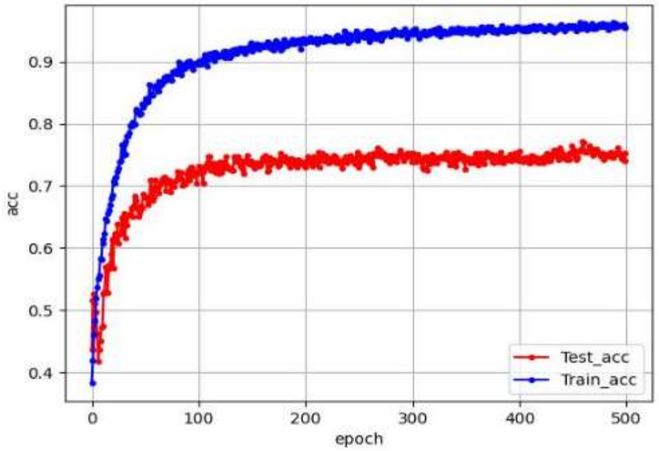
01

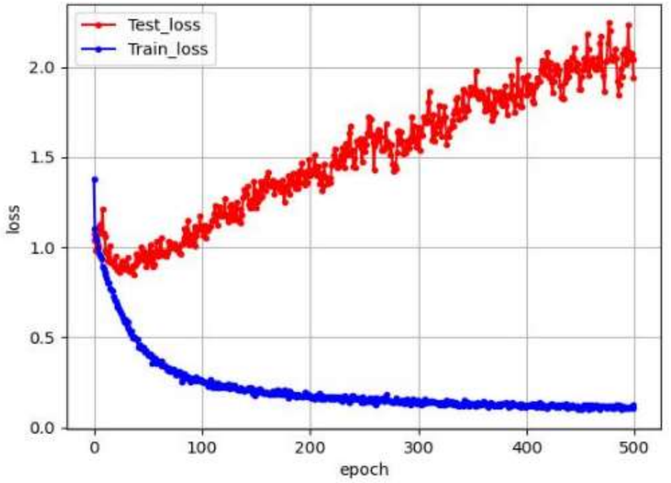
전이학습





VGG19

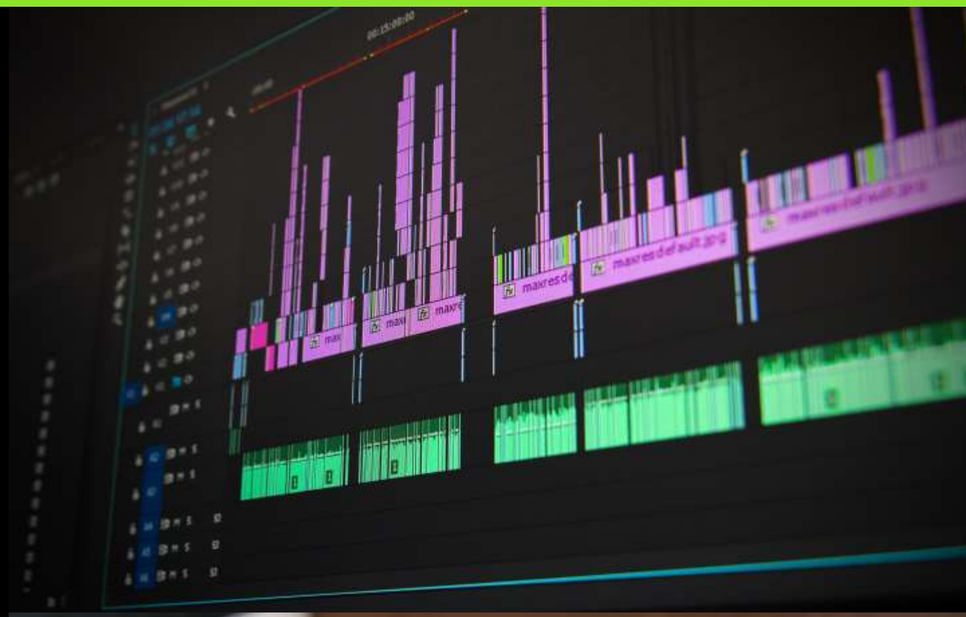


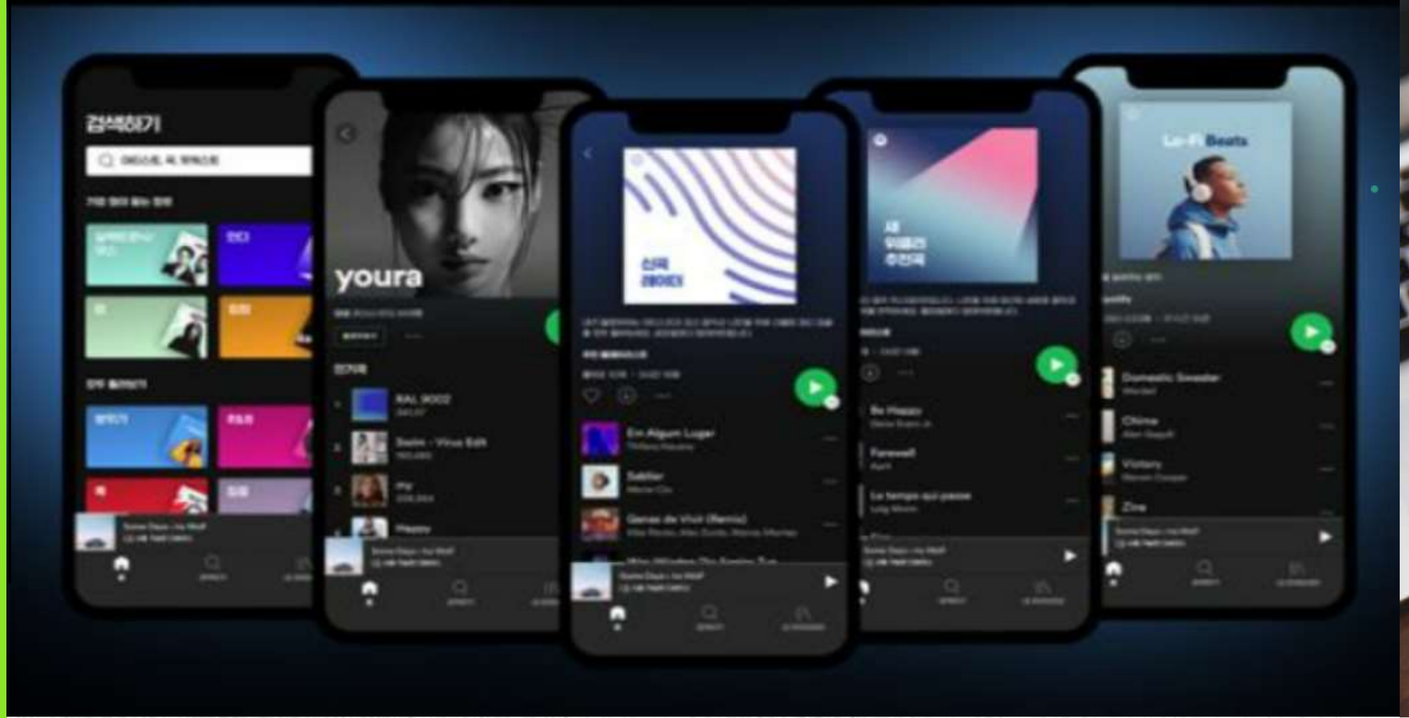


VGG19

MELONPY

멜론 파이





MELONPY

01

이미지 분류 모델

이미지 데이터 수 부족해서 확장하는 전처리

```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   horizontal_flip=True,
                                   width_shift_range=0.1,
                                   height_shift_range=0.1,
                                   rotation_range=30,
                                   vertical_flip=True
                                   )

test_datagen = ImageDataGenerator(rescale=1./255)
```

```
train_generator = train_datagen.flow_from_directory('./images/',
                                                    target_size=(48, 48),  # 이미지 크기 조정
                                                    batch_size=2,
                                                    class_mode='categorical',  # 분류 작업인 경우
                                                    shuffle=False  # 데이터를 무작위로 섞을지 여부
                                                    )
```

```
Found 2996 images belonging to 3 classes.
```

MELONPY

01

이미지 분류 모델

VGG16 불러오기

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 1, 1, 512)         14714688

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 64)                32832

 activation (Activation)     (None, 64)                0

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 3)                 195

 activation_1 (Activation)   (None, 3)                 0

=================================================================
Total params: 14747715 (56.26 MB)
Trainable params: 33027 (129.01 KB)
Non-trainable params: 14714688 (56.13 MB)
```

◀ 은닉층 추가

⇨ 정확도 안나와서 하이퍼파라미터 조정

MELONPY

01

이미지 분류 모델

kerastuner의 RandomSearch 사용

```
Epoch 1/100
169/169 [==============================] - 9s 25ms/step - loss: 0.9899 - accuracy: 0.4838 - val_loss: 0.9941 - val_accuracy: 0.4834
Epoch 2/100
169/169 [==============================] - 3s 18ms/step - loss: 0.9702 - accuracy: 0.4990 - val_loss: 0.9630 - val_accuracy: 0.5103
Epoch 3/100
169/169 [==============================] - 2s 14ms/step - loss: 0.9450 - accuracy: 0.5190 - val_loss: 0.9432 - val_accuracy: 0.5224
Epoch 4/100
169/169 [==============================] - 2s 14ms/step - loss: 0.9272 - accuracy: 0.5341 - val_loss: 0.9321 - val_accuracy: 0.5310
Epoch 5/100
169/169 [==============================] - 2s 14ms/step - loss: 0.9014 - accuracy: 0.5519 - val_loss: 0.9126 - val_accuracy: 0.5458
Epoch 6/100
169/169 [==============================] - 3s 16ms/step - loss: 0.8795 - accuracy: 0.5663 - val_loss: 0.9105 - val_accuracy: 0.5547
Epoch 7/100
169/169 [==============================] - 3s 17ms/step - loss: 0.8520 - accuracy: 0.5859 - val_loss: 0.9014 - val_accuracy: 0.5569
Epoch 8/100
169/169 [==============================] - 3s 16ms/step - loss: 0.8317 - accuracy: 0.5993 - val_loss: 0.8599 - val_accuracy: 0.5856
Epoch 9/100
169/169 [==============================] - 2s 14ms/step - loss: 0.8021 - accuracy: 0.6181 - val_loss: 0.8501 - val_accuracy: 0.5947
Epoch 10/100
...
Epoch 99/100
169/169 [==============================] - 3s 20ms/step - loss: 0.2323 - accuracy: 0.9029 - val_loss: 1.1882 - val_accuracy: 0.7418
Epoch 100/100
169/169 [==============================] - 2s 14ms/step - loss: 0.2154 - accuracy: 0.9094 - val_loss: 1.2146 - val_accuracy: 0.7290
```

```
1686/1686 [==============================] - 7s 4ms/step - loss: 1.2291 - accuracy: 0.7297

(1.2290809154510498, 0.7296951413154602)
```

MELONPY

02

이미지 분류 모델

MELONPY

예측결과

뉴진스 (2023)

```
1/1 [==============================] - 0s 8ms/step
[[0. 0. 1.]]
```

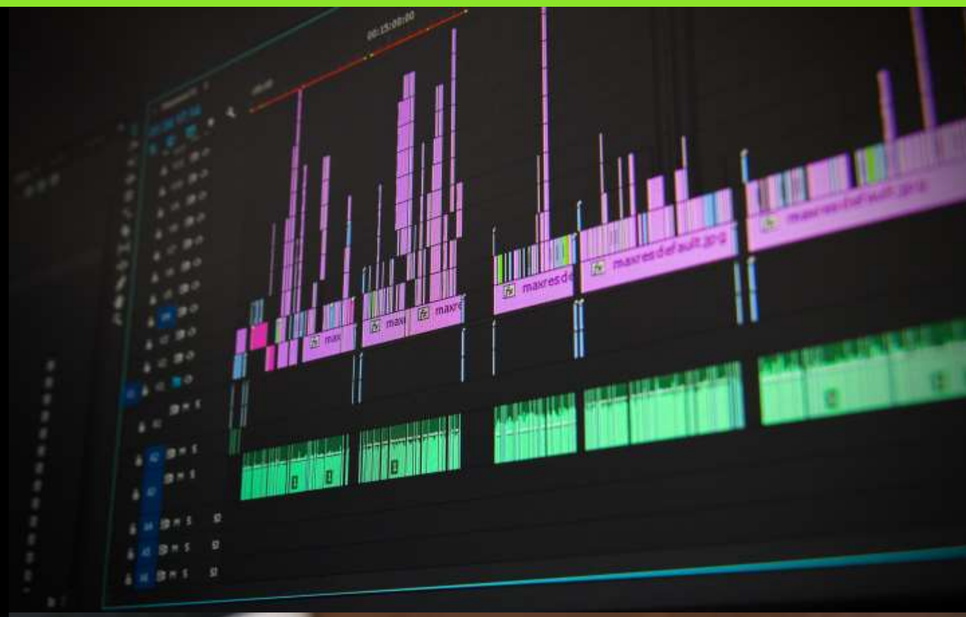
⇨ 2010년대로 예측됨

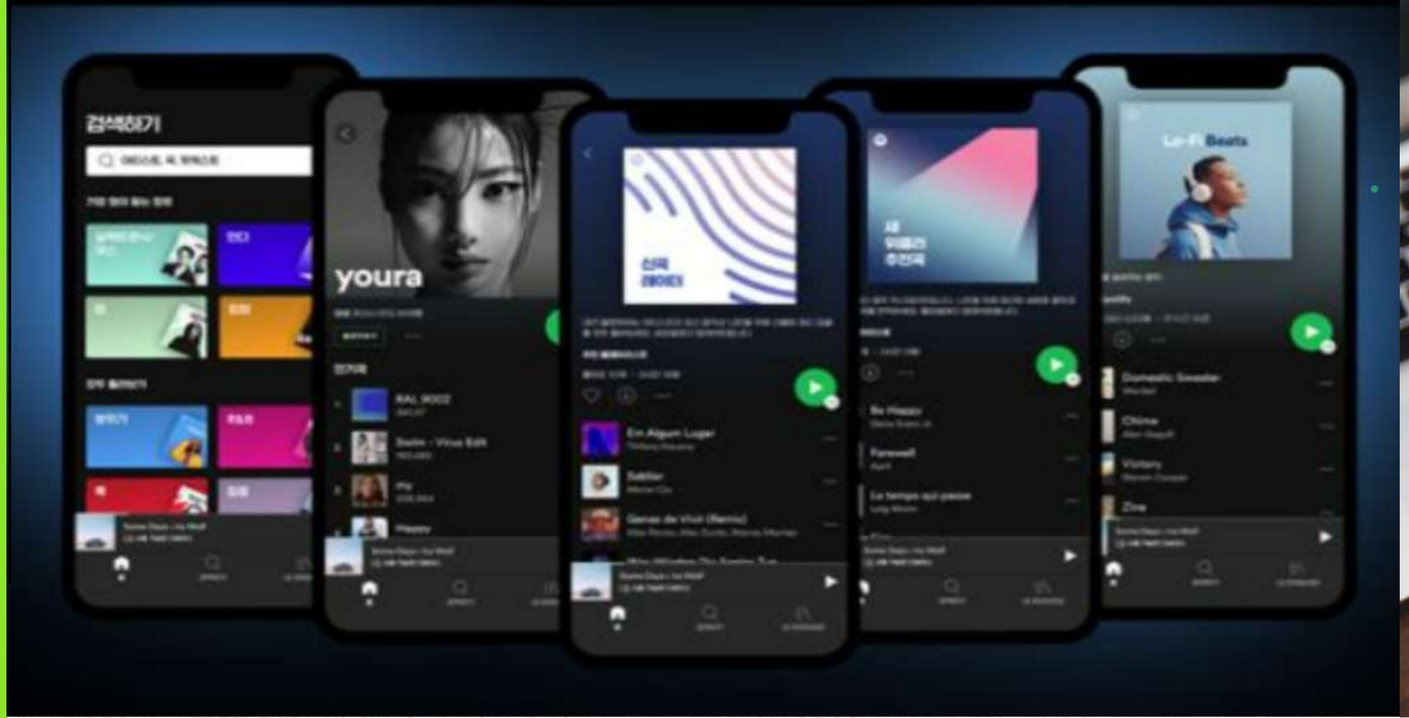
이상우 (1989)

```
1/1 [==============================] - 0s 33ms/step
[[1. 0. 0.]]
```

⇨ 1990년대로 예측됨

멜론 파이





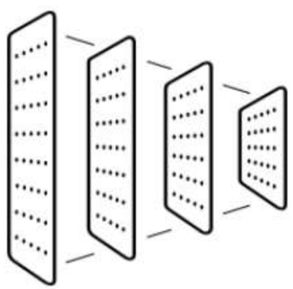
MELONPY

01

진행 과정 계획

MELONPY

분석 모델 설계 과정

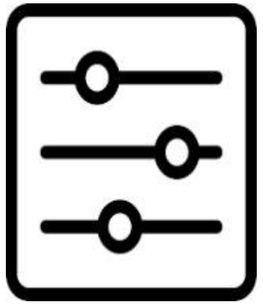


CNN 모델링

>>

전이 학습

>>



하이퍼 파라미터

>> CNN : 데이터를 경로에서 불러와 이미지와 라벨을 numpy형태로 저장
이미지 데이터를 정규화 하여 CNN모델에 적용.

>> 전이학습 : flow_from_directory 메서드를 이용해 이미지를 불러오고
VGG16 모델을 불러 전이 학습을 시킴.

예상 소요시간

>> 2023년 9월 18일 ~ 9월 20일

사용 모델링

>> CNN과 전이학습

예상 고충

>> 경로 에러 (불러오기)
>> 타입 에러

02 CNN과 전이학습 .summary()

MELONPY

CNN

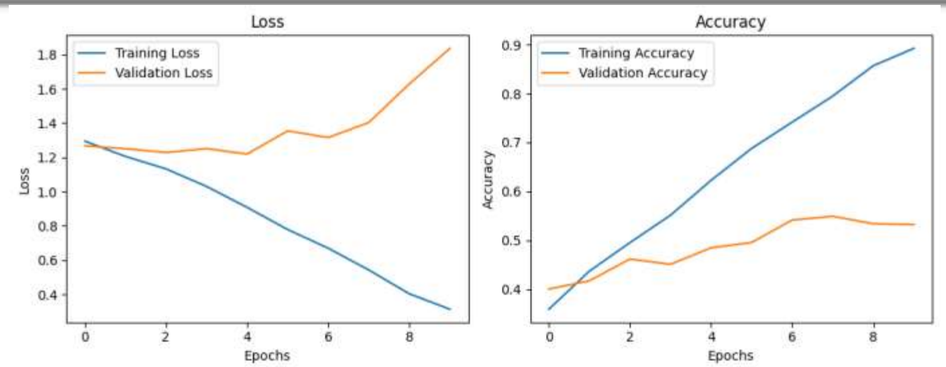
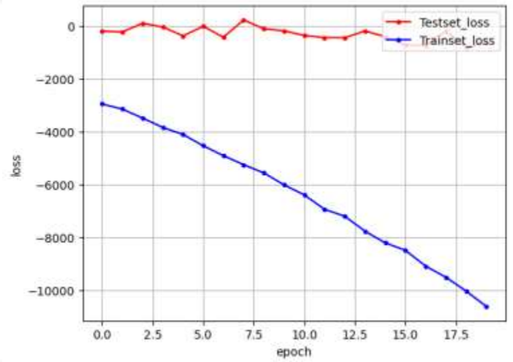
```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 46, 46, 32)        896

 max_pooling2d_4 (MaxPoolin  (None, 23, 23, 32)        0
 g2D)

 conv2d_5 (Conv2D)           (None, 21, 21, 64)        18496

 max_pooling2d_5 (MaxPoolin  (None, 10, 10, 64)        0
 g2D)

 flatten_2 (Flatten)         (None, 6400)              0

 dense_4 (Dense)             (None, 64)                409664

 dense_5 (Dense)             (None, 4)                 260

=================================================================
Total params: 429316 (1.64 MB)
Trainable params: 429316 (1.64 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

전이 학습(VGG16)

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 1, 1, 512)         14714688

 flatten_1 (Flatten)         (None, 512)               0

 dense_2 (Dense)             (None, 32)                16416

 activation_2 (Activation)   (None, 32)                0

 dropout_1 (Dropout)         (None, 32)                0

 dense_3 (Dense)             (None, 1)                 33

 activation_3 (Activation)   (None, 1)                 0

=================================================================
Total params: 14731137 (56.19 MB)
Trainable params: 16449 (64.25 KB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

02 CNN과 전이학습 결과 비교

MELONPY

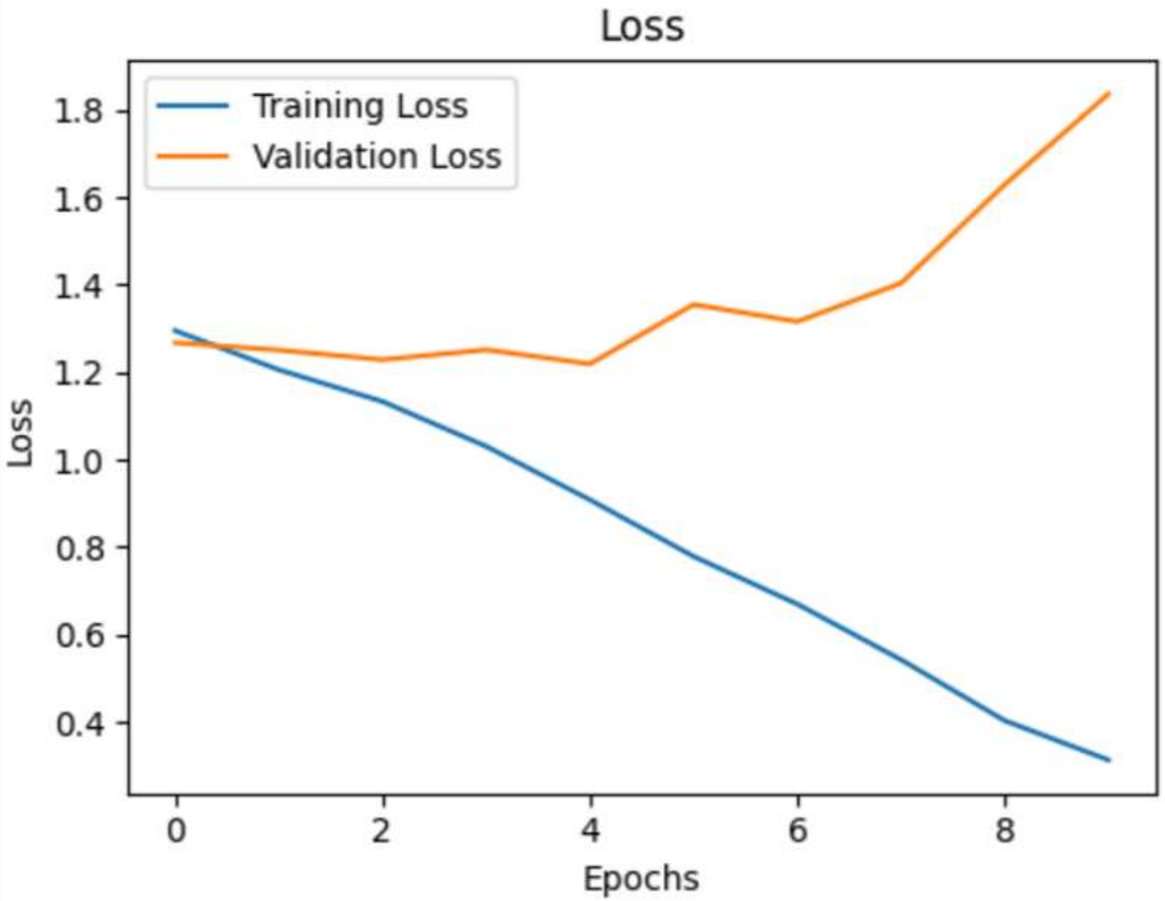
CNN	전이 학습(VGG16)
	
Epoch 10/10 - loss: 0.3140 - accuracy: 0.8922	Epoch 10/10 - loss: -6018.2666 - accuracy: 0.3034
- val_loss: 1.8356 - val_accuracy: 0.5322	- val_loss:-190.3043 - val_accuracy: 0.3400

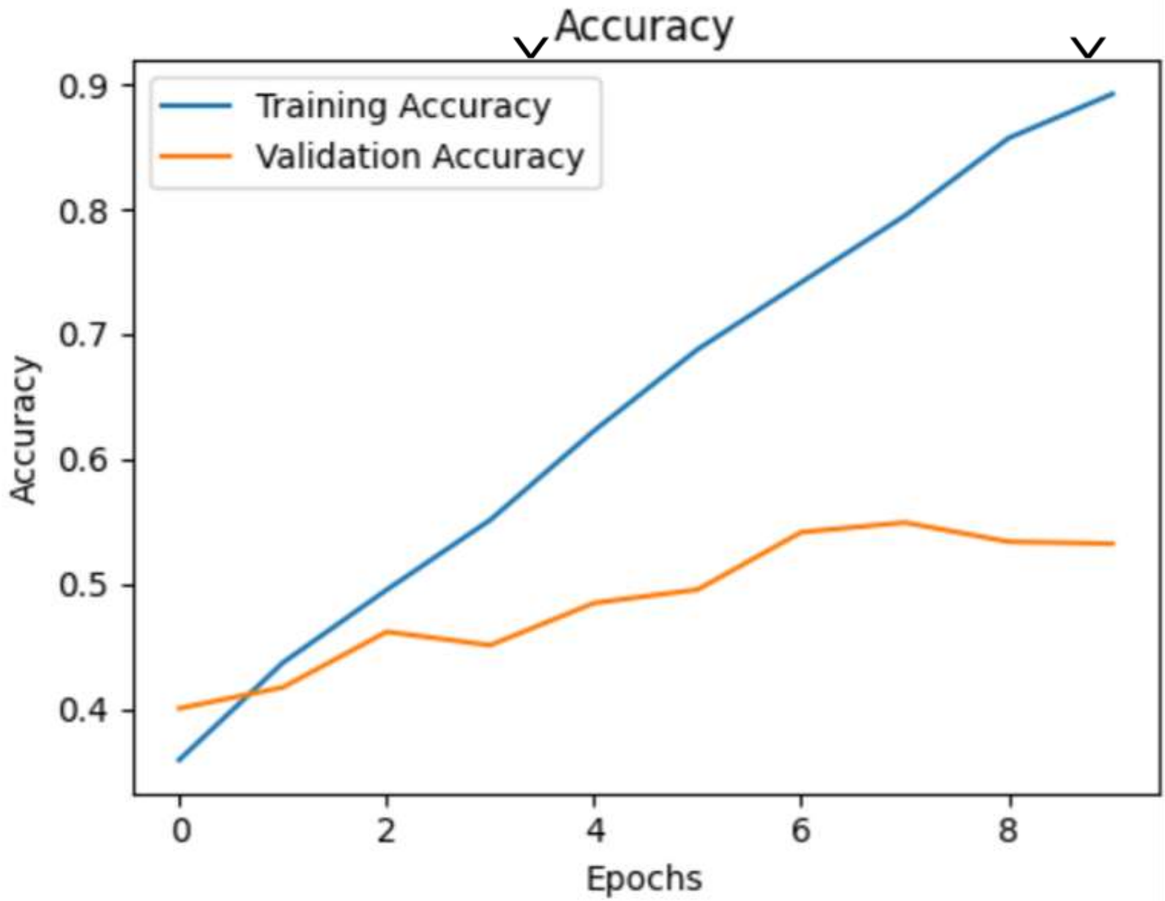
05

CNN 시각화

MELONPY

CNN의 결과 중 검증셋과 학습셋의 손실과 정확도를 시각화 하였읍니다,



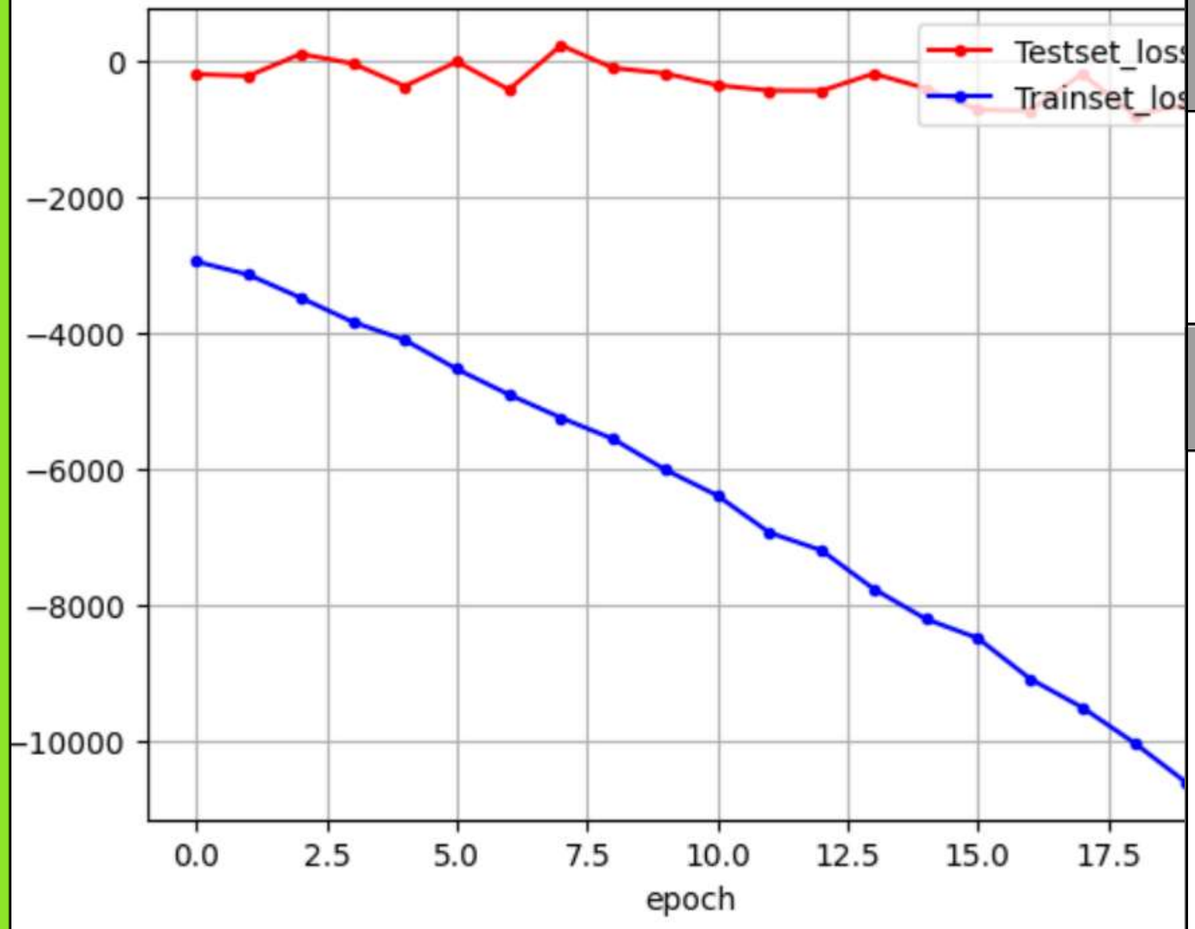


05

전이학습 시각화

MELONPY

전이학습의 결과 중 검증셋과 학습셋의 오차를 시각화 하였읍니다,



Testset_loss	Trainset_loss
오차 적음	오차 커짐

결과

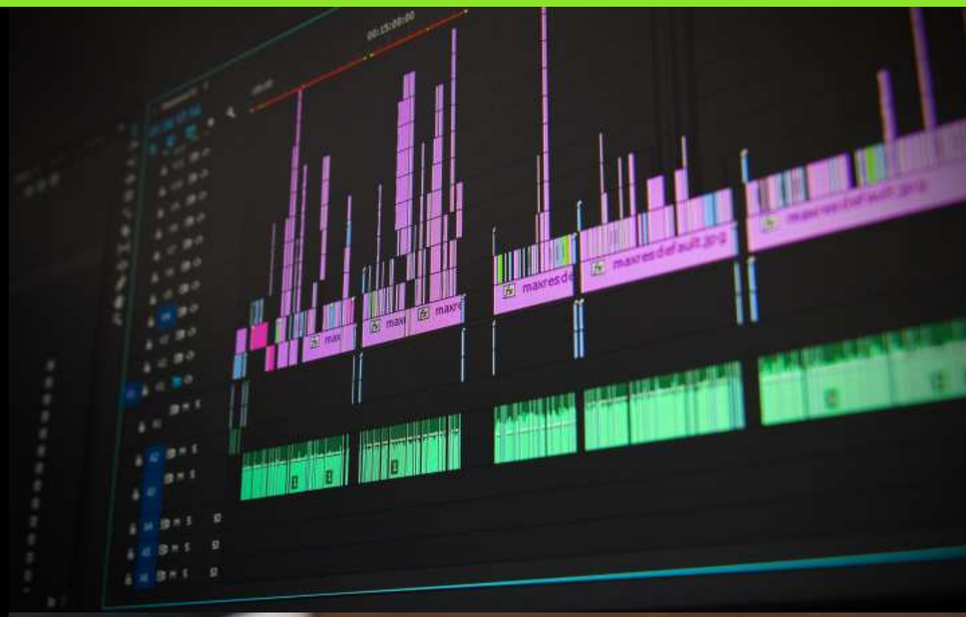
>> 학습 손실이 마이너스를 일으키고 있음.

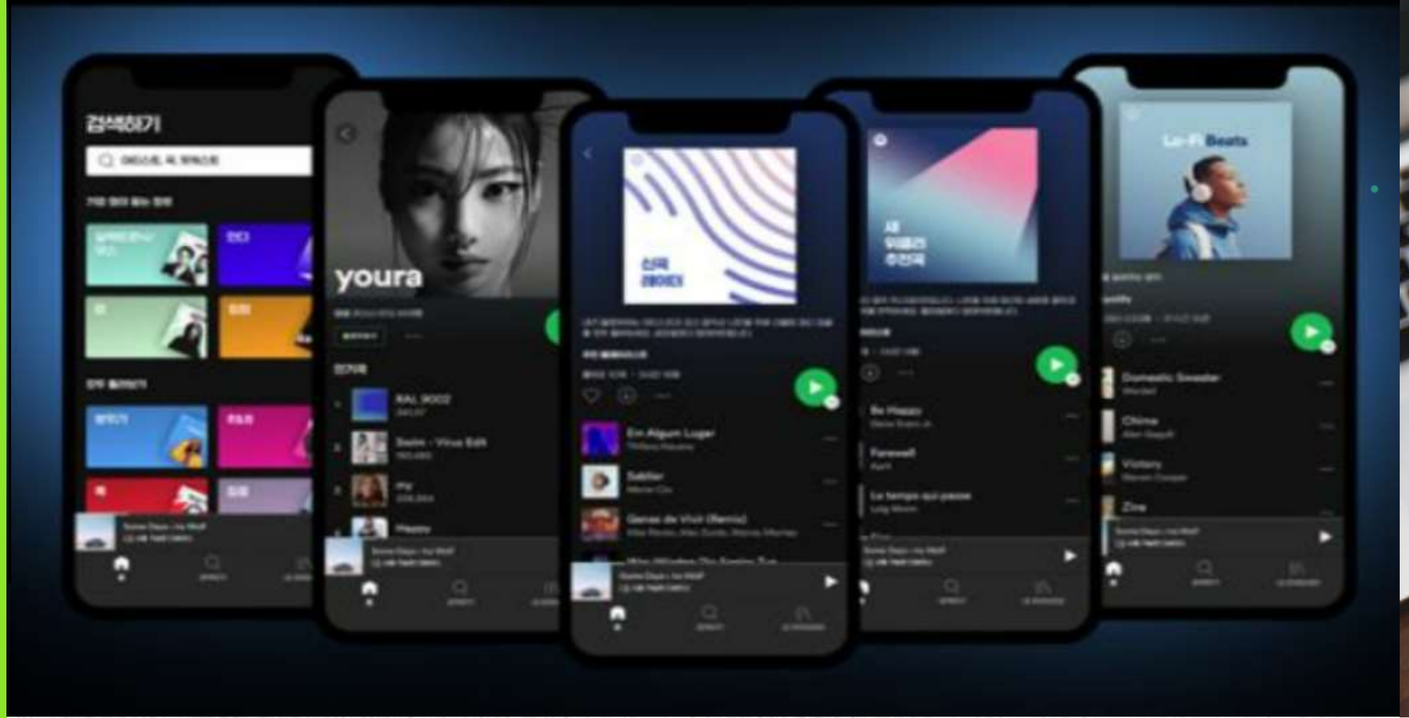
>> 손실은 증가하나 정확는 변화가 없음

>> Testset_loss의 오차는 변화가 적음

>> Trainset_loss의 오차는 일정하게 커침

멜론 파이





MELONPY

01

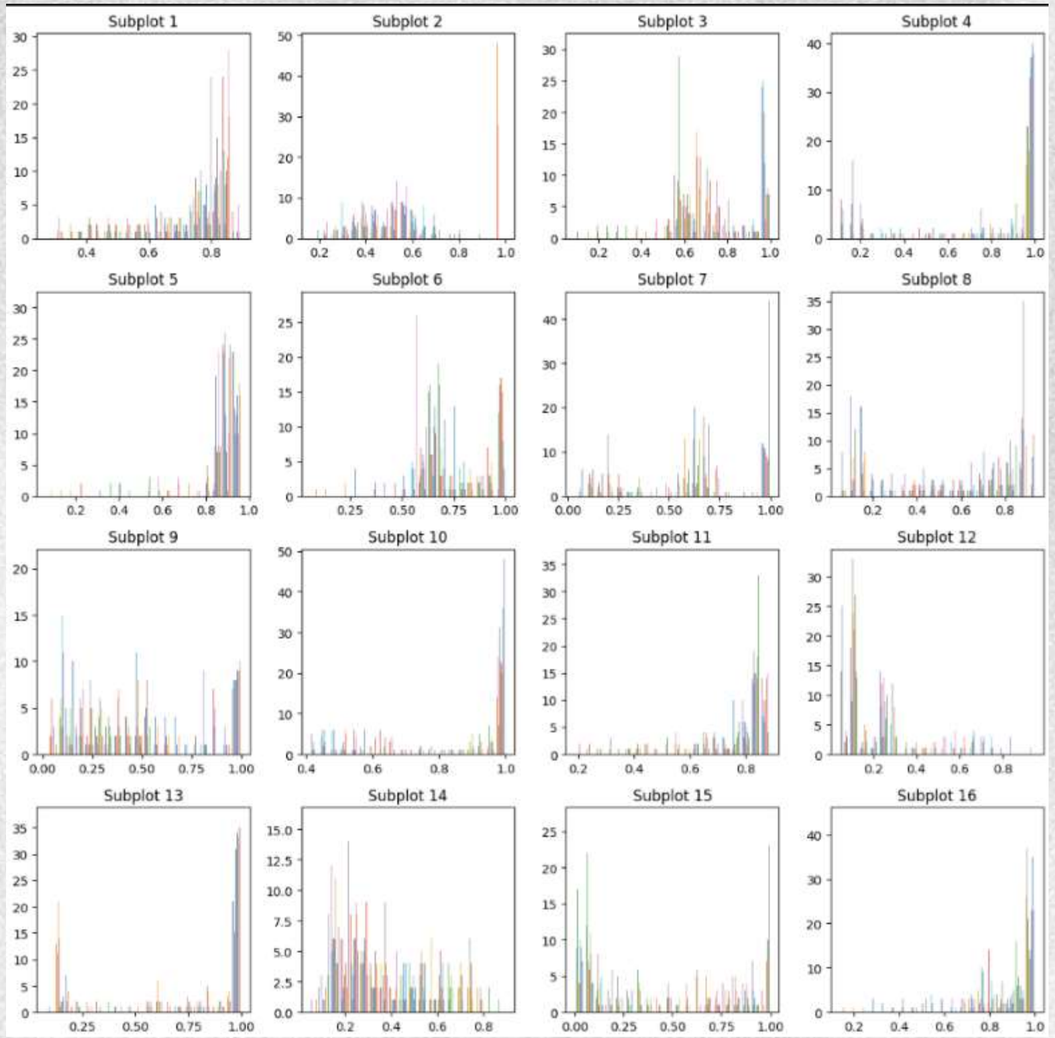
RGB 분석

MELONPY

RGB 픽셀별 분포 분석

(48,48,3)의 이미지들의 각 동일한 위치의 픽셀값을 가져와서 분석

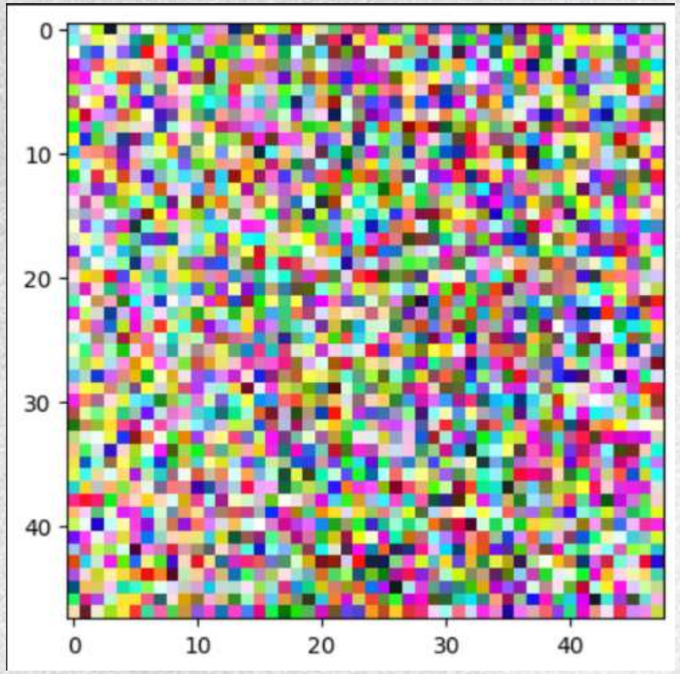
픽셀마다 분포가 다름
특징을 찾기 어려움



RGB 분석

RGB 픽셀별 분포 분석

랜덤으로 뽑아서 이미지를 만들어서 특징이 있는지 확인

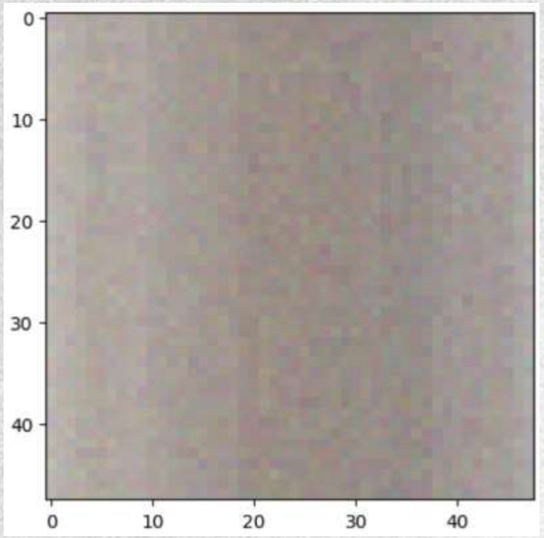


특징을 찾기 어려움

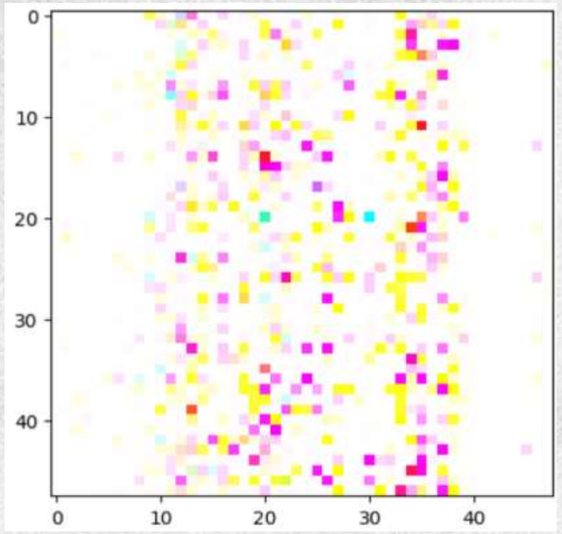
표본을 더 뽑아서 분석할 필요

RGB 픽셀별 분포 분석

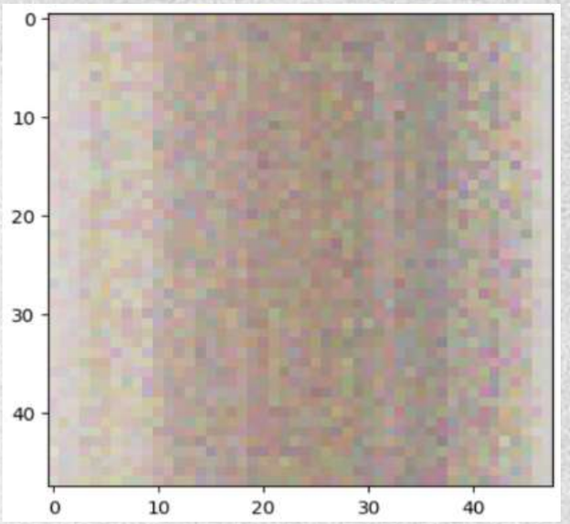
300개 이미지 생성해서 시각화



평균



최빈값



중앙값

전이학습으로 vgg16을 받아오고 은닉층 추가 후 분석

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 1, 1, 512)	14714688
flatten_11 (Flatten)	(None, 512)	0
dense_9 (Dense)	(None, 100)	51300
flatten_12 (Flatten)	(None, 100)	0
dense_10 (Dense)	(None, 100)	10100
output (Dense)	(None, 3)	303

Total params: 14,776,391
Trainable params: 61,703
Non-trainable params: 14,714,688

```
Epoch 1/1000
48/48 [==============================] - 22s 471ms/step - loss: 0.0102 - accuracy: 0.9987 - val_loss: 6.1921 - val_accuracy: 0.1850 - lr: 0.0010
Epoch 2/1000
48/48 [==============================] - 26s 554ms/step - loss: 0.0075 - accuracy: 0.9992 - val_loss: 5.8160 - val_accuracy: 0.2150 - lr: 0.0010
Epoch 3/1000
48/48 [==============================] - 30s 625ms/step - loss: 0.0065 - accuracy: 0.9992 - val_loss: 6.6350 - val_accuracy: 0.1733 - lr: 0.0010
Epoch 4/1000
48/48 [==============================] - 24s 494ms/step - loss: 0.0044 - accuracy: 1.0000 - val_loss: 6.3254 - val_accuracy: 0.1933 - lr: 0.0010
Epoch 5/1000
43/48 [=========================>....] - ETA: 1s - loss: 0.0037 - accuracy: 1.0000
```

과소대합이므로 하이퍼파라미터를 조정

MELONPY

03

모델 최적화

HyperOpt로 최적 파라미터 탐색 결과

런타임 에러

과대적합 해결 실패

MELONPY

Neural Style Transfer (NST) - 스타일 전이

참고 사이트:

https://bkshin.tistory.com/entry/%EC%BB%B4%ED%93%A8%ED%84%B0-%EB%B9%84%EC%A0%84-14-%EC%8A%A4%ED%83%80%EC%9D%BC-%EC%A0%84%EC%9D%B4Style-Transfer

이미지 ≫ 새로운 이미지 합성 ≪ 스타일