비 전 A I 와 비 즈 니 스 프 로 젝 트 최 종 발 표

Color Restoration of Damaged gray facial photo

AI빅데이터융합경영학과 20195262 장예진
20195263 주고은

INDEX

01 주제 설명

Post processing

We are gonna do this process ..(process to proceed)

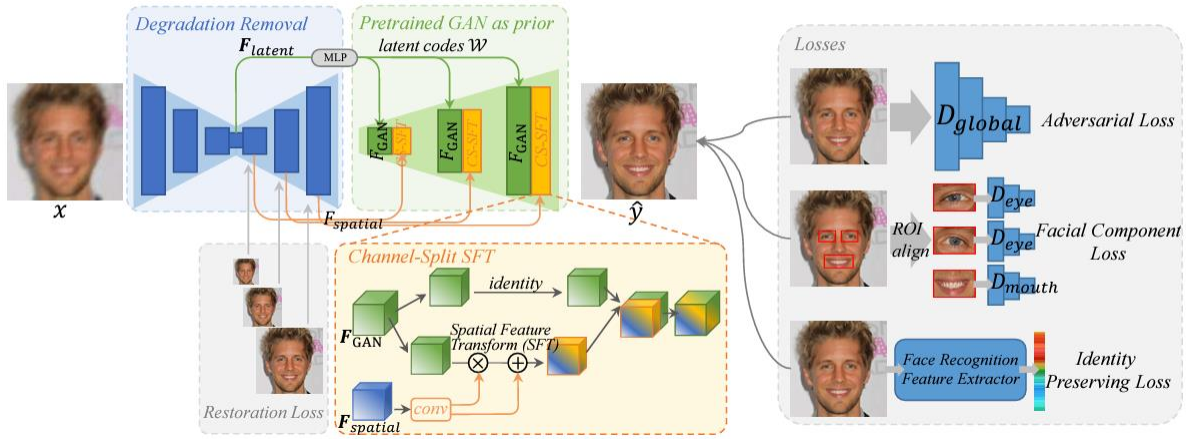
손상된 흑백 얼굴 이미지 >>> 컬러로 복원하는 작업 진행

기존에 진행하려 했던 GAN구조

- + 환경 문제에서 속도가 안 남
- + 커널 데드 현상
- + 런타임 중단 현상
- + 모델이 제대로 훈련되지 않는 문제..
  Too many..

CNN기반의 Autoencoder로 모델 변경 결정 !

02 Prior research-GFPGAN



Towards Real-World Blind Face Restoration with Generative Facial Prior, 2021

✓ 풍부하고 다양한 facial prior을 활용하여
Blind face restoration 수행하는 GFPGAN 제안

✓ facial prior:얼굴에 관한 정보나 특징을 포함하는 선험적인
정보나 모델

✓ 손상제거 모듈과 pretrained face GAN을 결합한 구조
잠재 코드 매핑과 여러 CS-SFT레이어에 의해 이어짐

✓ 결과 및 한계
어두운 얼굴 및 다양한 인구 그룹에서 잘수행되며
높은 복원을 보이나 입력 이미지가 그레이 스케일이면
얼굴 색상에 편향 발생.

Link: https://arxiv.org/abs/2101.04061

02 Prior research-autoencoder

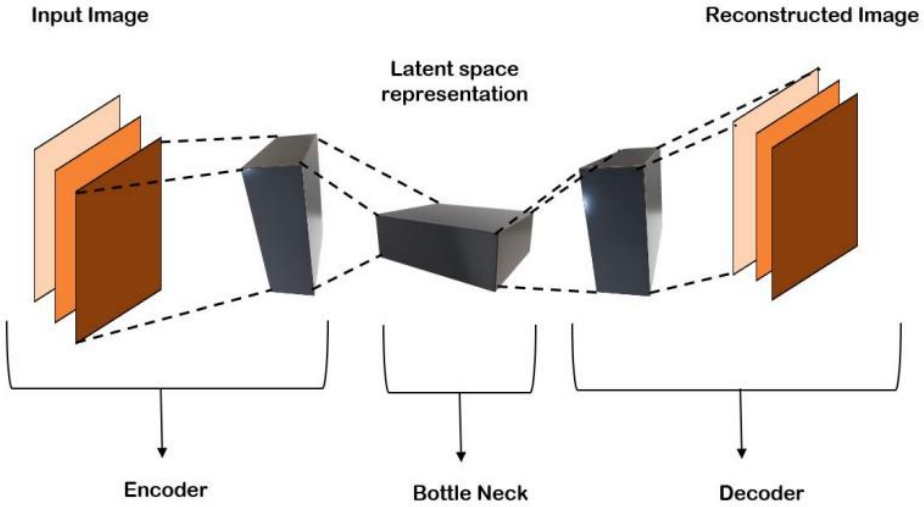


Fig. 1 Autoencoder Architecture

Image Colorization Using AutoEncoder, 2021

✓ 컨볼루션 신경망을 활용해 흑백 이미지를 현실적으로 컬러화하는 방법 제안

✓ 방법론: Autoencoder

입력과 출력이 동일한 feed forward신경망

입력을 낮은 차원의 코드로 압축하고 다시 복원함

결과적으로 효율적인 데이터 표현을 학습.

✓ 결과 및 한계

이미지 색상을 정확하게 판단할 수 있지만

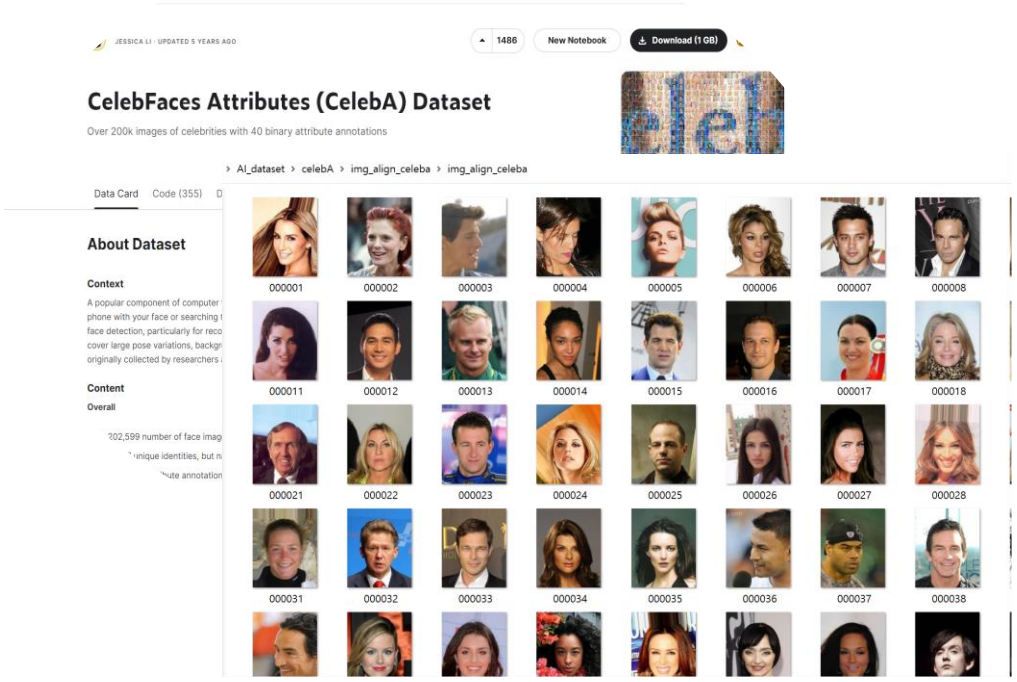
입력 정보 크기가 줄어들면서 특징 손실이 발생 가능

이로 인해 패치 형태의 색조나 블러 발생.

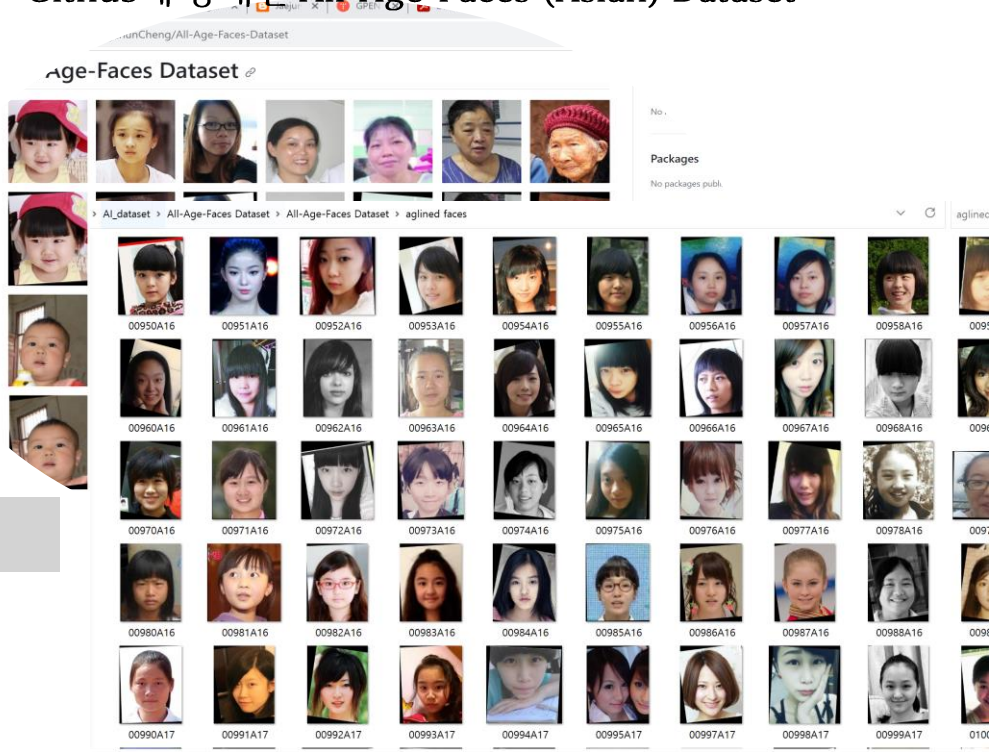
Link: https://www.researchgate.net/publication/354964311_Image_Colorization_Using_Autoencoders?enrichId=rgreq-7a270a1915e71a2ae3cd8cfe2a469e06-XXX&enrichSource=Y292ZXJQYWdlOzM1NDk2NDMxMTtBUzoxMTEzMjE5OTg5MDIwNjc0QDE2NDI0MjM3MzY1NTE%3D&el=1_x_2&_esc=publicationCoverPdf

03 Dataset

Kaggle의 Celeb A Dataset



Github에 공개된 All-Age-Faces (Asian) Dataset



Celeb A의 데이터 20,000개 가량 추출 + Asian 데이터(15,000개) 병합

>>새로운 face_data set생성 (43,322)

link(1): https://www.kaggle.com/datasets/jessicali9530/celeba-dataset
link(2): https://github.com/JingchunCheng/All-Age-Faces-Dataset

03 Preprocessing

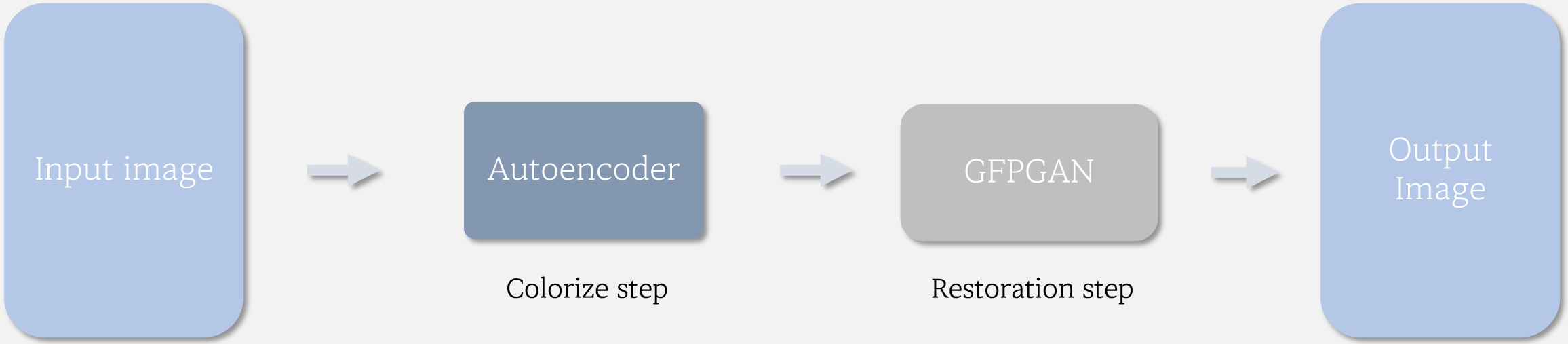
- ✓ get_input(): 흑백 이미지에 대한 입력 전 처리 수행
  BGR 색상 공간을 gray scale로 변환
  0-1 사이의 값으로 나타내기 위해 정규화 (/255)
  Resize (224*184)
  (IMG_HEIGHT, IMG_WIDTH, 1) 형태의 3D 배열로 재구성

- ✓ get_output(): 예측 컬러 이미지에 대한 출력 전 처리 수행
  0-1 사이의 값으로 나타내기 위해 정규화 (/255)
  Resize (224*184)\

- ✓ image_generator(): 모델 훈련 시 사용되는 데이터 배치 단위로 생성하여 모델에 입력으로 제공

- ✓ train & test set 8:2 분할

04 Model Architecture



05 Model(1) - Autoencoder(채색)

```
[ ]  model = Sequential()
     model.add(Conv2D(64, (3, 3), activation='relu', padding='same', strides=2,
                      input_shape=(IMG_HEIGHT, IMG_WIDTH, 1)))
     model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
     model.add(Conv2D(128, (3,3), activation='relu', padding='same', strides=2))
     model.add(Conv2D(256, (3,3), activation='relu', padding='same'))
     model.add(Conv2D(256, (3,3), activatio
     model.add(Conv2D(512, (3,3), activatio
     model.add(Conv2D(256, (3,3), activatio

     model.add(Conv2D(128, (3,3), activatio
     model.add(UpSampling2D((2, 2)))
     model.add(Conv2D(64, (3,3), activation=
     model.add(UpSampling2D((2, 2)))
     model.add(Conv2D(32, (3,3), activation=
     model.add(Conv2D(16, (3,3), activation=
     model.add(Conv2D(3, (3, 3), activation=
     model.add(UpSampling2D((2, 2)))
     model.compile(optimizer='adam', loss='
     model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 112, 92, 64)       640

 conv2d_1 (Conv2D)           (None, 112, 92, 128)      73856

 conv2d_2 (Conv2D)           (None, 56, 46, 128)       147584

 conv2d_3 (Conv2D)           (None, 56, 46, 256)       295168

 conv2d_4 (Conv2D)           (None, 28, 23, 256)       590080

 conv2d_5 (Conv2D)           (None, 28, 23, 512)       1180160

 conv2d_6 (Conv2D)           (None, 28, 23, 256)       1179904

 conv2d_7 (Conv2D)           (None, 28, 23, 128)       295040

 up_sampling2d (UpSampling2  (None, 56, 46, 128)       0
 D)

 conv2d_8 (Conv2D)           (None, 56, 46, 64)        73792

 up_sampling2d_1 (UpSamplin  (None, 112, 92, 64)       0
 g2D)

 conv2d_9 (Conv2D)           (None, 112, 92, 32)       18464

 conv2d_10 (Conv2D)          (None, 112, 92, 16)       4624

 conv2d_11 (Conv2D)          (None, 112, 92, 3)        435

 up_sampling2d_2 (UpSamplin  (None, 224, 184, 3)       0
 g2D)

=================================================================
Total params: 3859747 (14.72 MB)
Trainable params: 3859747 (14.72 MB)
Non-trainable params: 0 (0.00 Byte)
```

Model architecture

Encoder : down sampling & convolutional layer

Decoder: up sampling & convolutional layer

Activation function: Relu & tanh

Loss function: MSE

Optimizer: Adam

05 Model(1) - Autoencoder(채색)

```
[ ]  model = Sequential()
     model.add(Conv2D(64, (3, 3), activation='relu', padding='same', strides=2,
                      input_shape=(IMG_HEIGHT, IMG_WIDTH, 1)))
     model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
     model.add(Conv2D(128, (3,3), activation='relu', padding='same', strides=2))
     model.add(Conv2D(256, (3,3), activation='relu', padding='same'))
     model.add(Conv2D(256, (3,3), activatio
     model.add(Conv2D(512, (3,3), activatio
     model.add(Conv2D(256, (3,3), activatio

     model.add(Conv2D(128, (3,3), activatio
     model.add(UpSampling2D((2, 2)))
     model.add(Conv2D(64, (3,3), activation=
     model.add(UpSampling2D((2, 2)))
     model.add(Conv2D(32, (3,3), activation=
     model.add(Conv2D(16, (3,3), activation=
     model.add(Conv2D(3, (3, 3), activation=
     model.add(UpSampling2D((2, 2)))
     model.compile(optimizer='adam', loss='
     model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 112, 92, 64)       640

 conv2d_1 (Conv2D)           (None, 112, 92, 128)      73856

 conv2d_2 (Conv2D)           (None, 56, 46, 128)       147584

 conv2d_3 (Conv2D)           (None, 56, 46, 256)       295168

 conv2d_4 (Conv2D)           (None, 28, 23, 256)       590080

 conv2d_5 (Conv2D)           (None, 28, 23, 512)       1180160

 conv2d_6 (Conv2D)           (None, 28, 23, 256)       1179904

 conv2d_7 (Conv2D)           (None, 28, 23, 128)       295040

 up_sampling2d (UpSampling2  (None, 56, 46, 128)       0
 D)

 conv2d_8 (Conv2D)           (None, 56, 46, 64)        73792

 up_sampling2d_1 (UpSamplin  (None, 112, 92, 64)       0
 g2D)

 conv2d_9 (Conv2D)           (None, 112, 92, 32)       18464

 conv2d_10 (Conv2D)          (None, 112, 92, 16)       4624

 conv2d_11 (Conv2D)          (None, 112, 92, 3)        435

 up_sampling2d_2 (UpSamplin  (None, 224, 184, 3)       0
 g2D)

=================================================================
Total params: 3859747 (14.72 MB)
Trainable params: 3859747 (14.72 MB)
Non-trainable params: 0 (0.00 Byte)
```

Encoder

input image를 저차원으로 압축

CNN레이어들을 사용하여 공간 특징 추출

최종 레이어는 저차원의 잠재 표현을 생성

05 Model(1) - Autoencoder(채색)

```
[ ]  model = Sequential()
     model.add(Conv2D(64, (3, 3), activation='relu', padding='same', strides=2,
                      input_shape=(IMG_HEIGHT, IMG_WIDTH, 1)))
     model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
     model.add(Conv2D(128, (3,3), activation='relu', padding='same', strides=2))
     model.add(Conv2D(256, (3,3), activation='relu', padding='same'))
     model.add(Conv2D(256, (3,3), activatio
     model.add(Conv2D(512, (3,3), activatio
     model.add(Conv2D(256, (3,3), activatio

     model.add(Conv2D(128, (3,3), activatio
     model.add(UpSampling2D((2, 2)))
     model.add(Conv2D(64, (3,3), activation=
     model.add(UpSampling2D((2, 2)))
     model.add(Conv2D(32, (3,3), activation=
     model.add(Conv2D(16, (3,3), activation=
     model.add(Conv2D(3, (3, 3), activation=
     model.add(UpSampling2D((2, 2)))
     model.compile(optimizer='adam', loss='
     model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 112, 92, 64)       640

 conv2d_1 (Conv2D)           (None, 112, 92, 128)      73856

 conv2d_2 (Conv2D)           (None, 56, 46, 128)       147584

 conv2d_3 (Conv2D)           (None, 56, 46, 256)       295168

 conv2d_4 (Conv2D)           (None, 28, 23, 256)       590080

 conv2d_5 (Conv2D)           (None, 28, 23, 512)       1180160

 conv2d_6 (Conv2D)           (None, 28, 23, 256)       1179904

 conv2d_7 (Conv2D)           (None, 28, 23, 128)       295040

 up_sampling2d (UpSampling2  (None, 56, 46, 128)       0
 D)

 conv2d_8 (Conv2D)           (None, 56, 46, 64)        73792

 up_sampling2d_1 (UpSamplin  (None, 112, 92, 64)       0
 g2D)

 conv2d_9 (Conv2D)           (None, 112, 92, 32)       18464

 conv2d_10 (Conv2D)          (None, 112, 92, 16)       4624

 conv2d_11 (Conv2D)          (None, 112, 92, 3)        435

 up_sampling2d_2 (UpSamplin  (None, 224, 184, 3)       0
 g2D)

=================================================================
Total params: 3859747 (14.72 MB)
Trainable params: 3859747 (14.72 MB)
Non-trainable params: 0 (0.00 Byte)
```

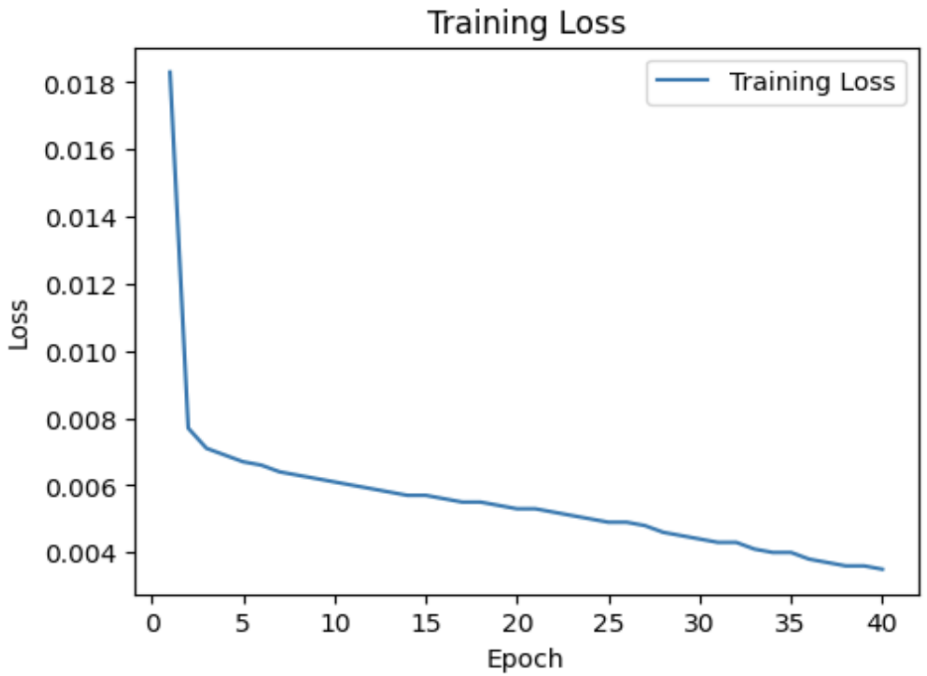
Decoder

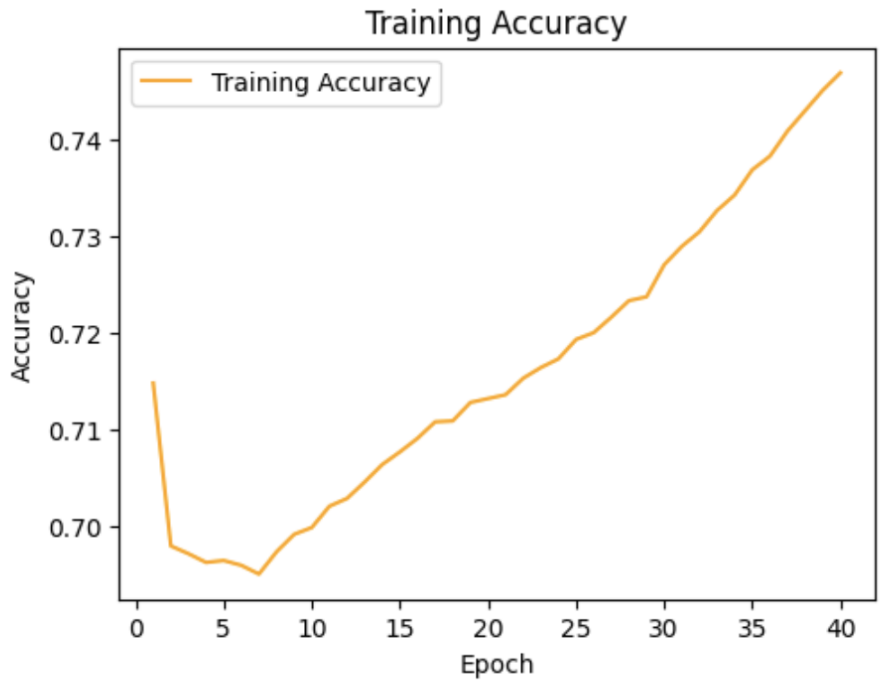
Encoder에서 얻은 저차원의 표현을 사용하여
원래 input data로 복원
CNN레이어들을 사용하여 공간 특징 복원
Decoder의 최종 레이어는 입력 이미지의 차원과
같은 크기의 출력 생성

>>>>>

Input image를 Encoder에 통과시켜 저 차원 표현을 얻은 후
Decoder를 사용하여 저 차원 표현을 컬러 이미지로 복원

05 Model(1) -Training loss & accuracy

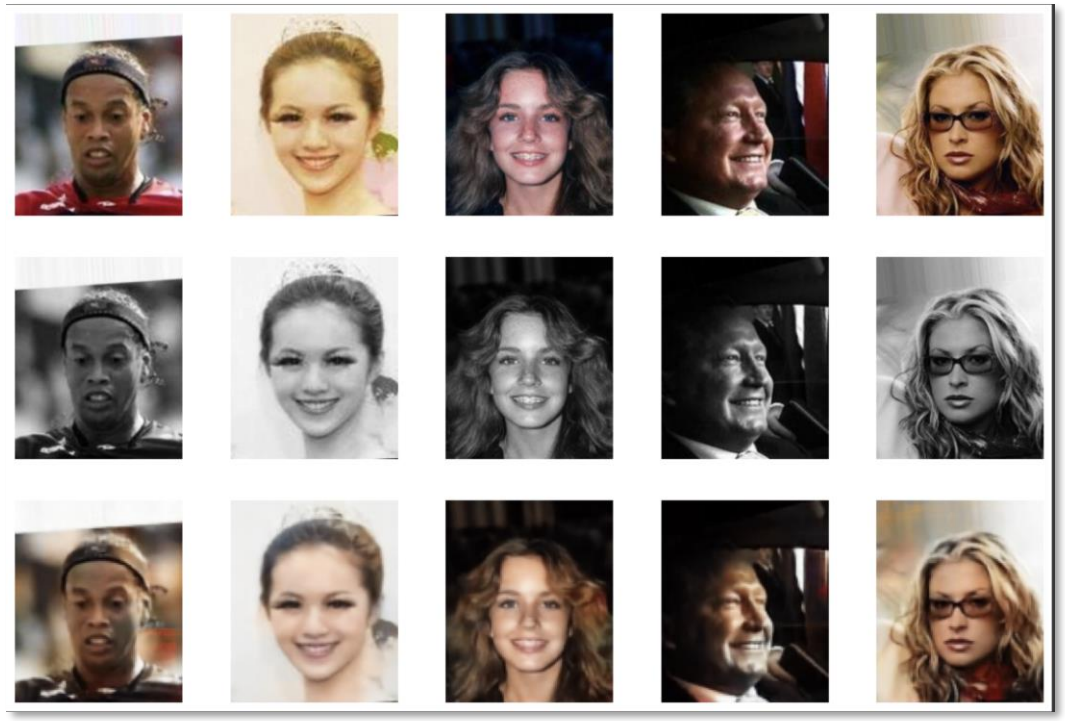




BATCH SIZE = 128
Epoch = 40
Loss = 0.0035
Accuracy = 0.7468

이후 epoch 50이상 수행할 때,
계속하여 loss 감소, accuracy 상승하는 경향 보였으나
colab GPU할당량 초과로 런타임 중단됨

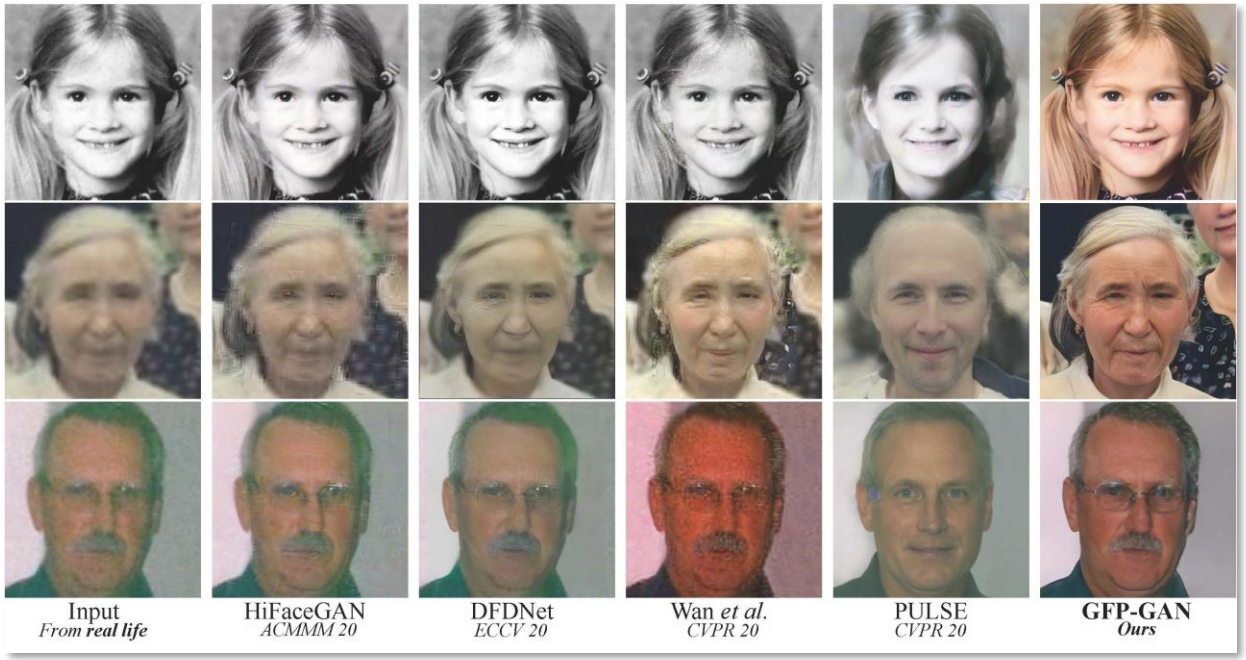
06 실험 및 결과 –Colorization 단독 수행



트루컬러이미지가 있는 테스트데이터셋

트루 컬러 이미지가 없는 오래된 흑백 이미지

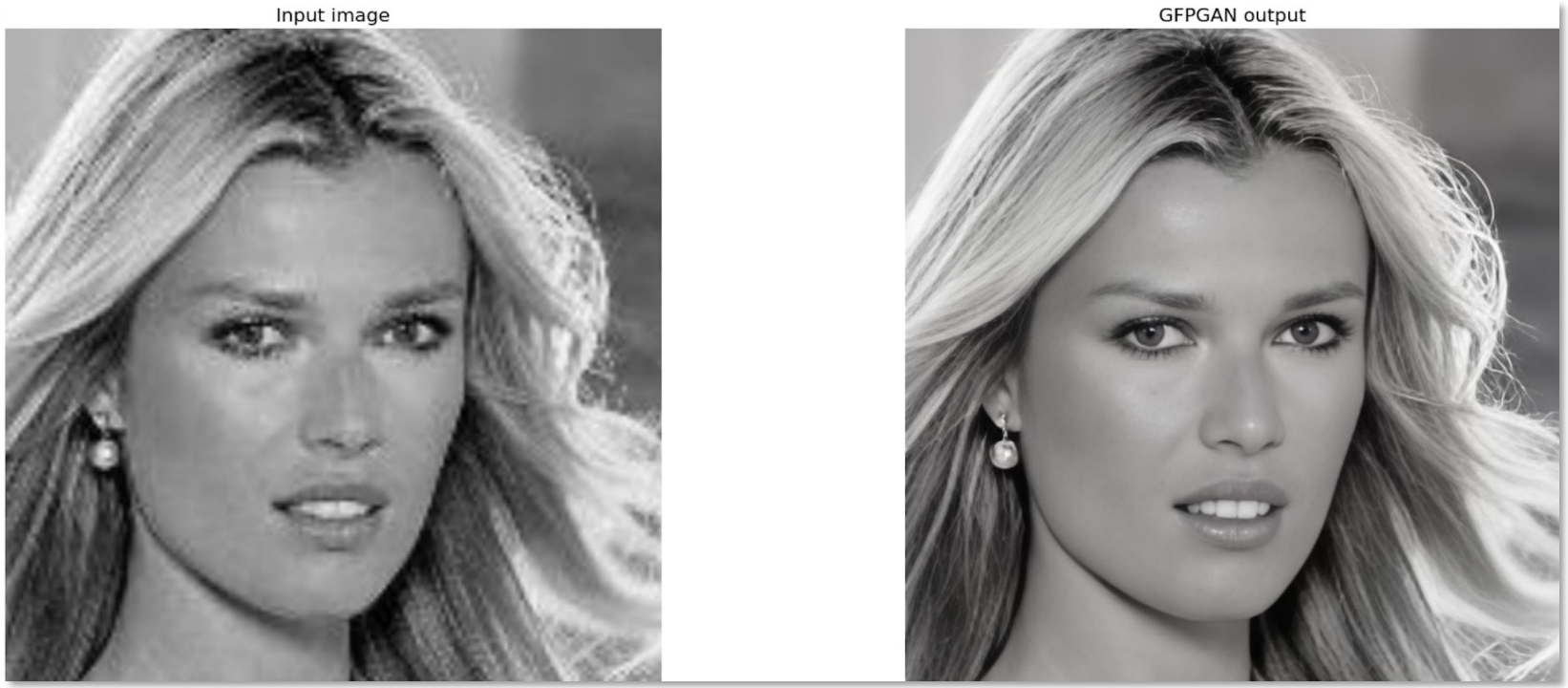
05 Model(2) – GFPGAN



컬러화 기능 없는 GFPGANv1.3 사용

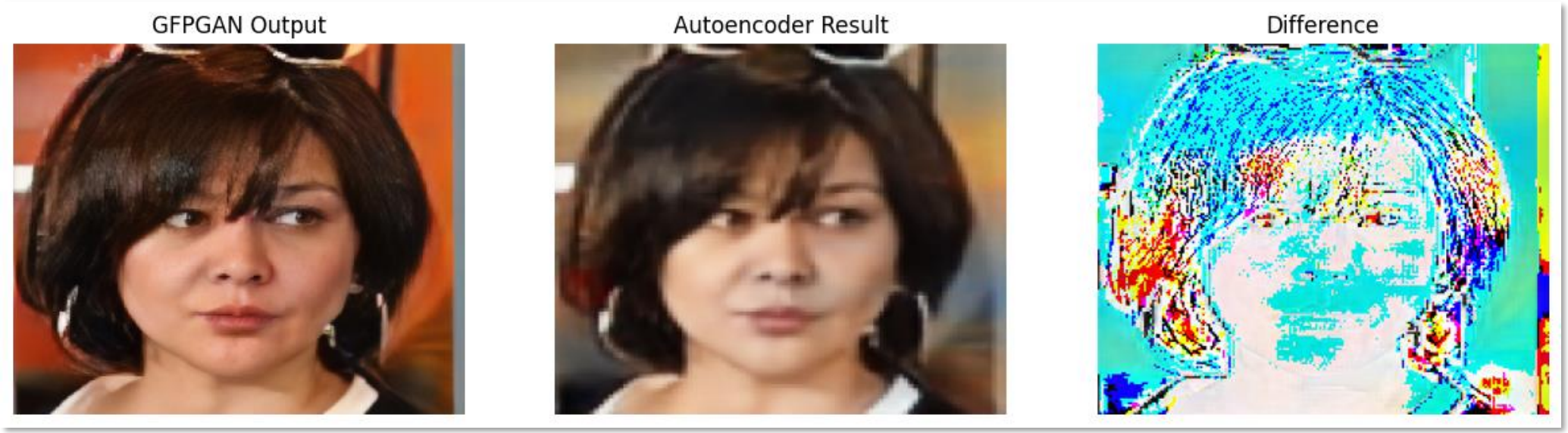
- ✓ 자연스러운 복원 결과
- ✓ 아주 저품질이거나 고품질인 데이터에도 높은 성능 !!
- ✓ 반복(2회) 복원 가능

06 실험 및 결과 -GFPGAN 단독수행



성능 대박 !!

06 실험 및 결과

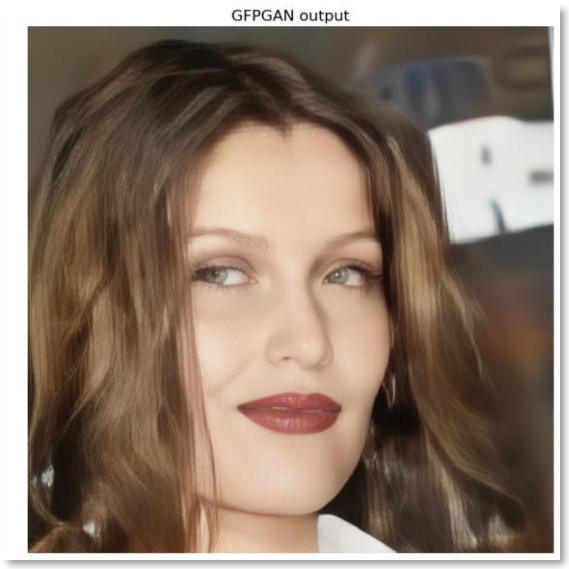


본래 파이프라인은 복원(GFPGAN) -> 컬러화(autoencoder)

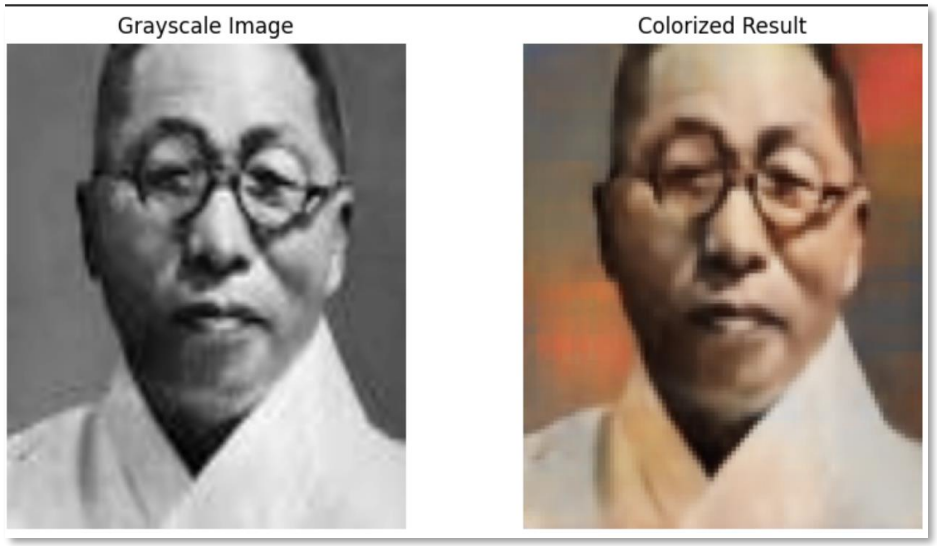
그런데 !

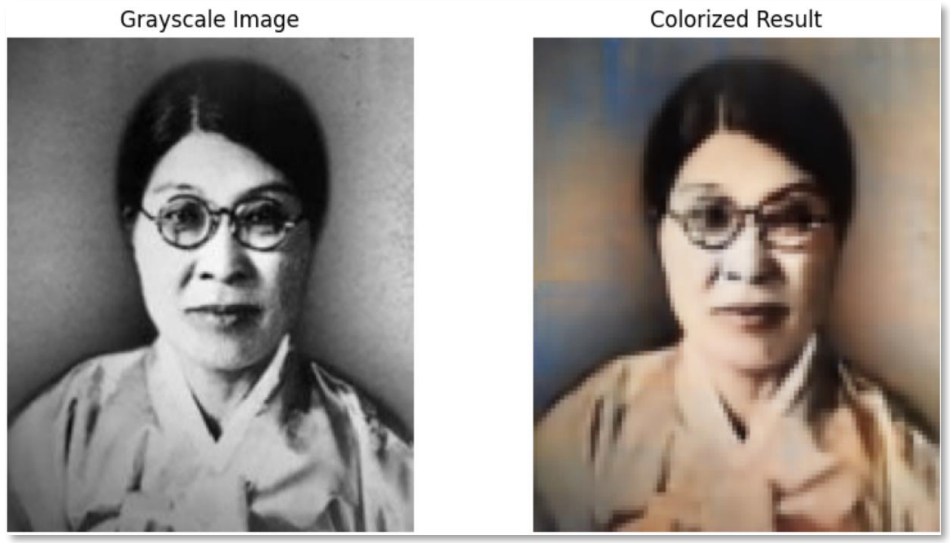
GFPGAN을 거친 고품질의 이미지가 autoencoder를 거치며 저품질화 되는 현상…

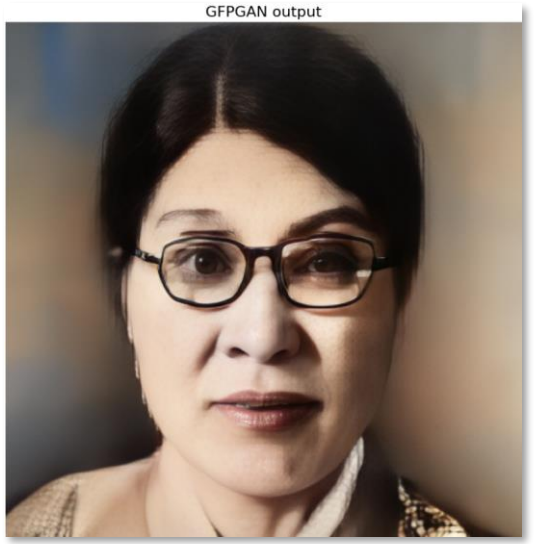
따라서 컬러화(autoencoder) -> 복원(GFPGAN)의 구조로 변경



06 실험 및 결과

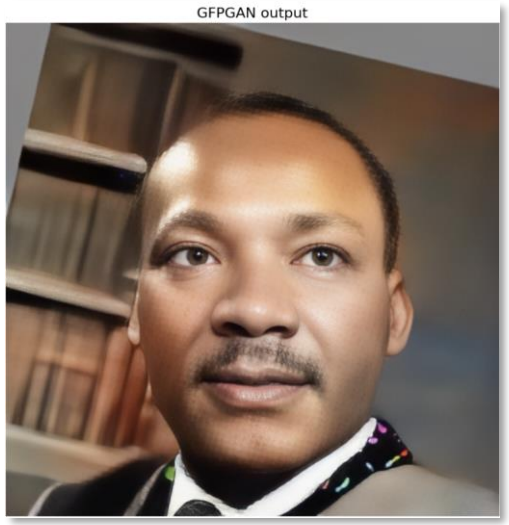

Grayscale Image
Colorized Result


Grayscale Image
Colorized Result


GFPGAN output

06 실험 및 결과


Grayscale Image
Colorized Result


GFPGAN output

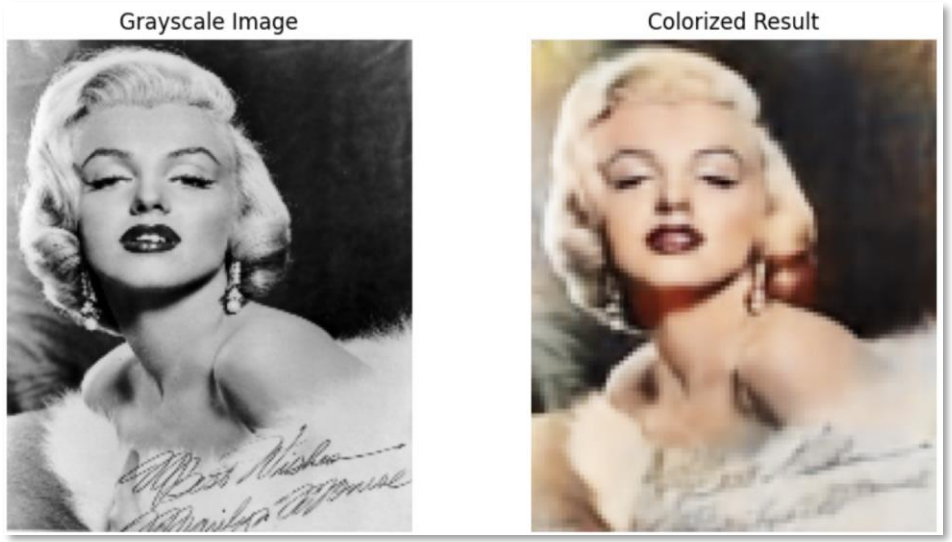

Grayscale Image
Colorized Result

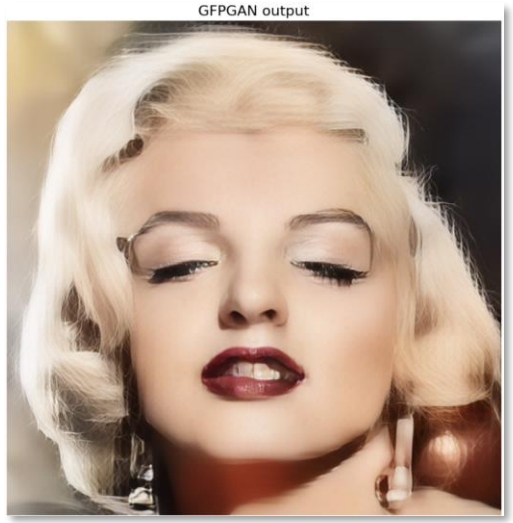

GFPGAN output

06 실험 및 결과


Grayscale Image
Colorized Result

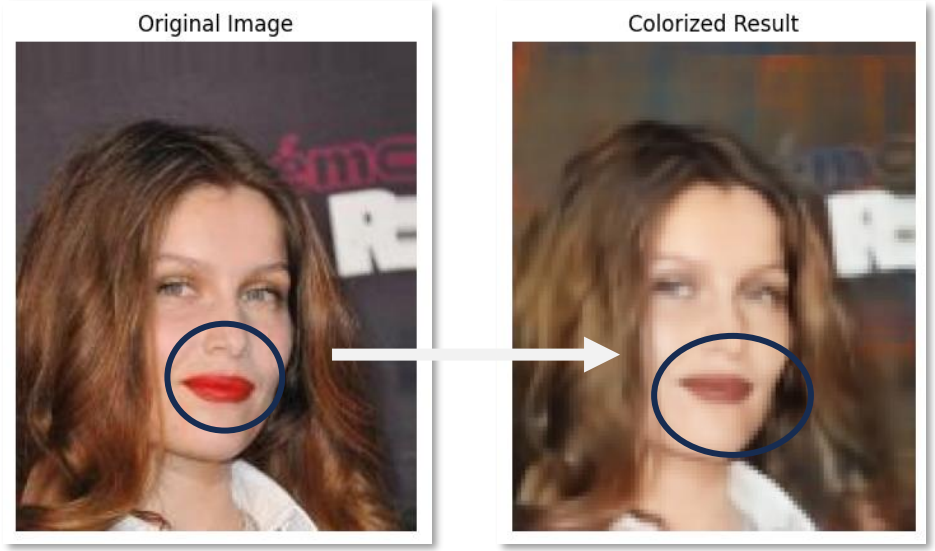

GFPGAN output


Grayscale Image
Colorized Result


GFPGAN output

04 Model - 한계 및 보완점

1. GPU 환경 문제로 인해 원하는 만큼의 학습 수행이 힘듦
2. 컬러화 모델의 이미지 품질 저하 현상
3. 컬러화 결과에서 대체로 쨍한 레드 컬러가 톤 다운 된 브릭레드로 복원되는 경향을 보임.
4. 전체적으로 세피아 느낌의 브라운 컬러화가 강함



비 전 A I 와 비 즈 니 스 프 로 젝 트 최 종 발 표

Thank you