Image Generator

미니 프로젝트

김영민 김지수 이다인

1. 분석 주제

AI로 새로운 인쇄체를 만들어 낼 수 있을까?

목표

글자의 특징을 잘 학습해서 비슷한 스타일의 다른 글자들을 생성해낼 수 있도록 모델을 학습시키는 것

2. 데이터 설명

한국어 글자체 이미지 AI 데이터

데이터 설명

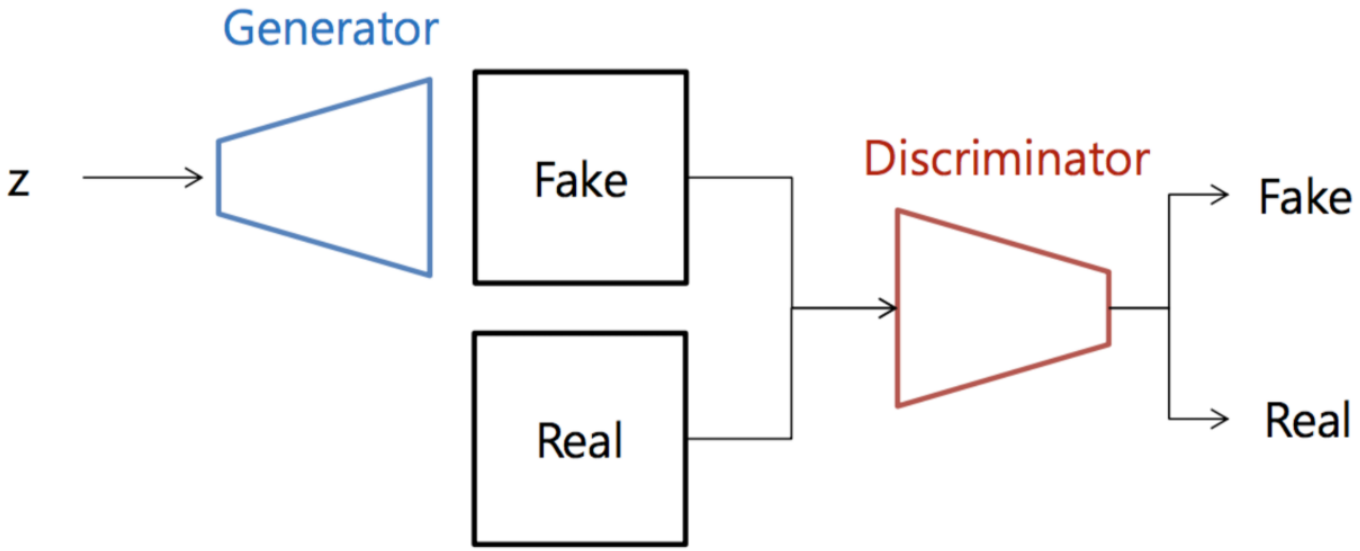
- 현대 한글 11172자(한글 자음+모음+받침 자음 조합 수)를 가장 많이 활용하는 폰트(글자서체) 50종을 선정하여 해당 글자체의 이미지로 인공지능 글자 인식을 위한 학습용 데이터 셋 (AIHub)

학습 데이터

- 32*32 사이즈
- 60839장의 이미지

3. 모델 구조

GAN



Generator(생성자)의 역할

- 진짜와 유사한 이미지를 생성해 판별자를 속인다

Discriminator(판별자)의 역할

- 진짜 이미지를 진짜라고 판별해야한다 (True: 1)
- 생성자가 만든 가짜 이미지를 보고 가짜라고 판별해야한다 (False: 0)

4. 진행 과정

Data preprocessing

데이터 선정

- 50만장의 데이터 중 약 6만장 랜덤 선택

이미지 사이즈 통일

- 32x32 size로 이미지 resize

이미지 픽셀 정규화

- 이미지의 픽셀 값을 [-1,1] 범위로 정규화

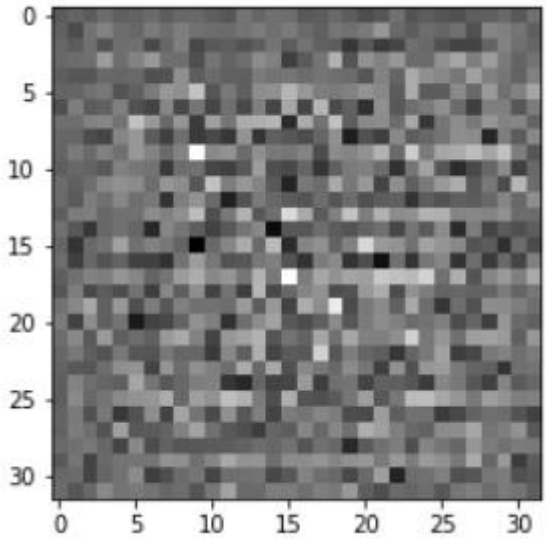
전처리 결과

4. 진행 과정

Modeling

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 1024)              102400
_________________________________________________________________
batch_normalization_4 (Batch (None, 1024)              4096
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 1024)              0
_________________________________________________________________
dense_3 (Dense)              (None, 8192)              8388608
_________________________________________________________________
batch_normalization_5 (Batch (None, 8192)              32768
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 8192)              0
_________________________________________________________________
reshape_1 (Reshape)          (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_transpose_3 (Conv2DTr (None, 8, 8, 128)         409600
_________________________________________________________________
batch_normalization_6 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_transpose_4 (Conv2DTr (None, 16, 16, 64)        204800
_________________________________________________________________
batch_normalization_7 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_transpose_5 (Conv2DTr (None, 32, 32, 3)         4803
=================================================================
Total params: 9,147,843
Trainable params: 9,129,027
Non-trainable params: 18,816
```

Generator Model



Discriminator Model

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 16, 16, 64)        4864
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 16, 16, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 8, 8, 128)         204928
_________________________________________________________________
leaky_re_lu_9 (LeakyReLU)    (None, 8, 8, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense_4 (Dense)              (None, 256)               2097408
_________________________________________________________________
leaky_re_lu_10 (LeakyReLU)   (None, 256)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 257
=================================================================
Total params: 2,307,457
Trainable params: 2,307,457
Non-trainable params: 0
_________________________________________________________________
```

```
predicted = discriminator(generated_image)
print(predicted)
```

```
tf.Tensor([[-0.01008528]], shape=(1, 1), dtype=float32)
```

4. 진행 과정

Optimizer + loss function

Optimizer

```
# 생성자용
generator_optimizer = tf.keras.optimizers.Adam(1e-4)
# 판별자용
discriminator_optimizer = tf.keras.optimizers.Adam(1e-4)
```

Loss function

$$\underset{G}{min}\underset{D}{Max}V(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[log(1-D(G(z))]$$

Generator Model

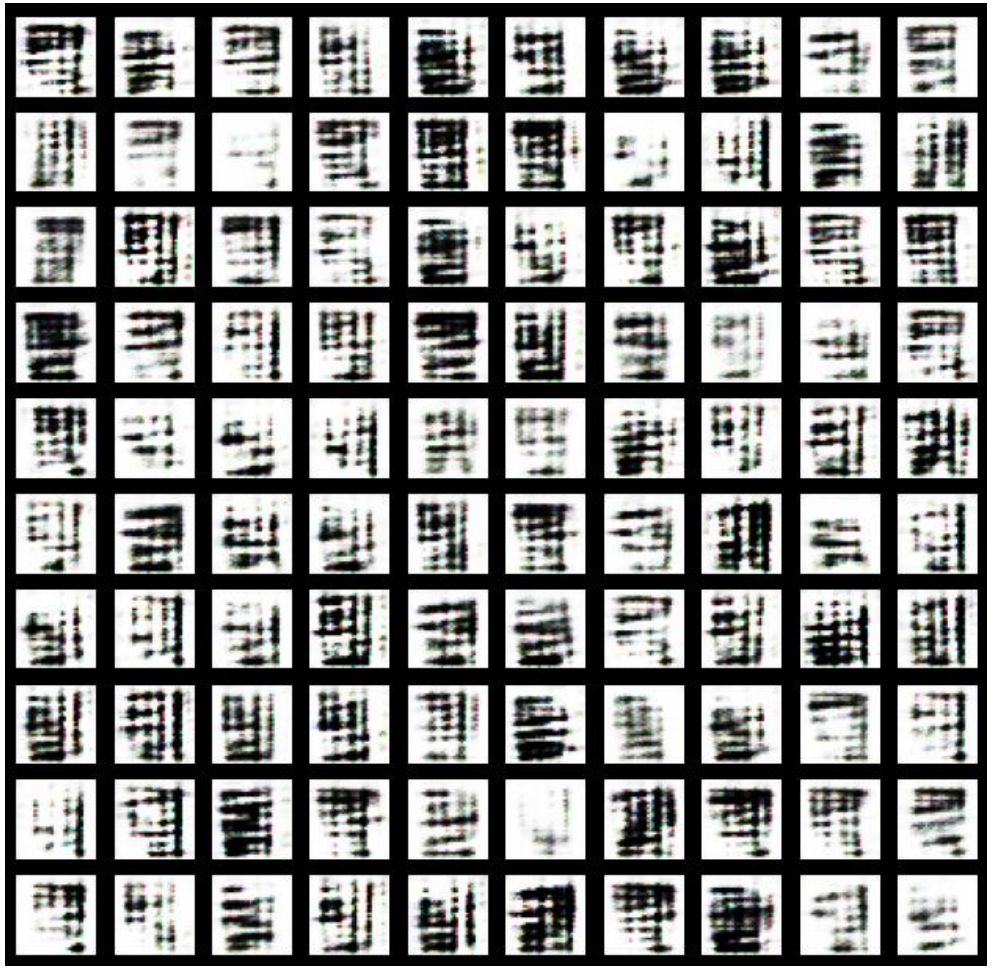
```
# 생성자 모델 목표 :
# 판별자 모델이 가짜 이미지를 판별했을 때 판별 값이 1에 가까워지도록
def generator_loss(fake_output):
    return cross_entropy(tf.ones_like(fake_output), fake_output) # 1과 가짜 이미지를 판별 값 비교
```

Discriminator Model

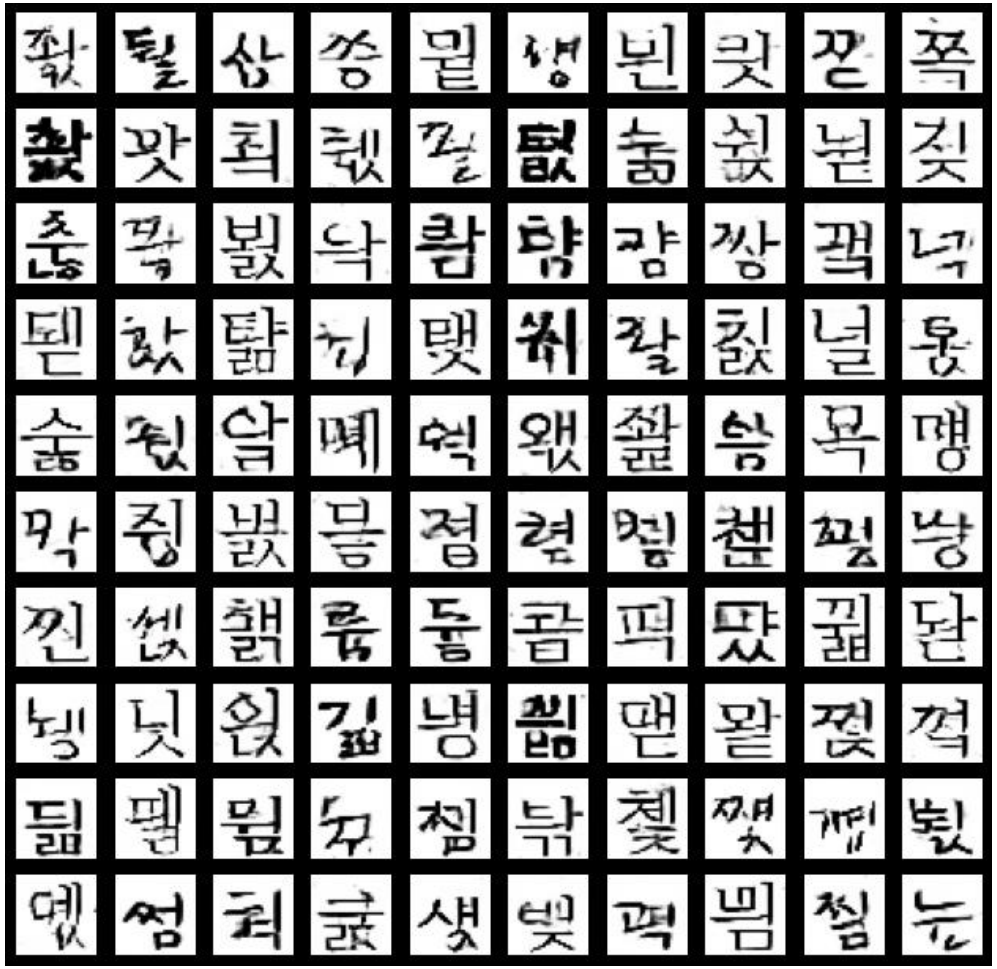
```
# 판별자 모델 목표 :
# 1. 진짜 이미지를 판별했을 때 판별 값이 1에 가까워지도록
# 2. 가짜 이미지를 판별했을 때 판별 값이 0에 가까워지도록
def discriminator_loss(real_output, fake_output):
    real_loss = cross_entropy(tf.ones_like(real_output), real_output) # 1. 1과 진짜 이미지 판별 값 비
    fake_loss = cross_entropy(tf.zeros_like(fake_output), fake_output) # 2. 0과 가짜 이미지 판별 값 비
    total_loss = real_loss + fake_loss
    return total_loss
```

5. 결과

Generated Images on EPOCH : 0



Generated Images on EPOCH : 300



1. 분석 주제

사진에서 그림으로 그림을 다른 스타일로

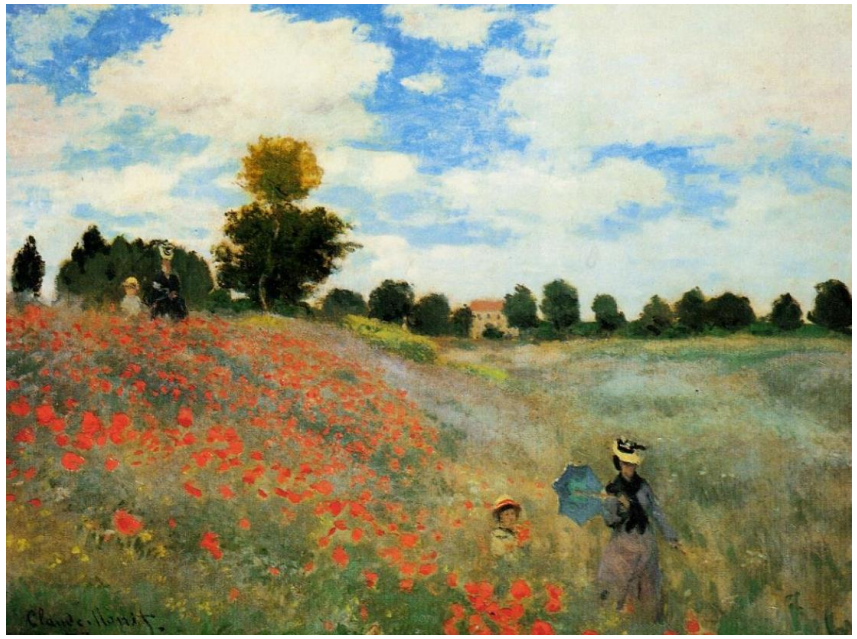
목표

실제 사진을 다른 스타일의 그림으로 바꾸는 것

2. 데이터 설명

빈센트 반 고흐

모네

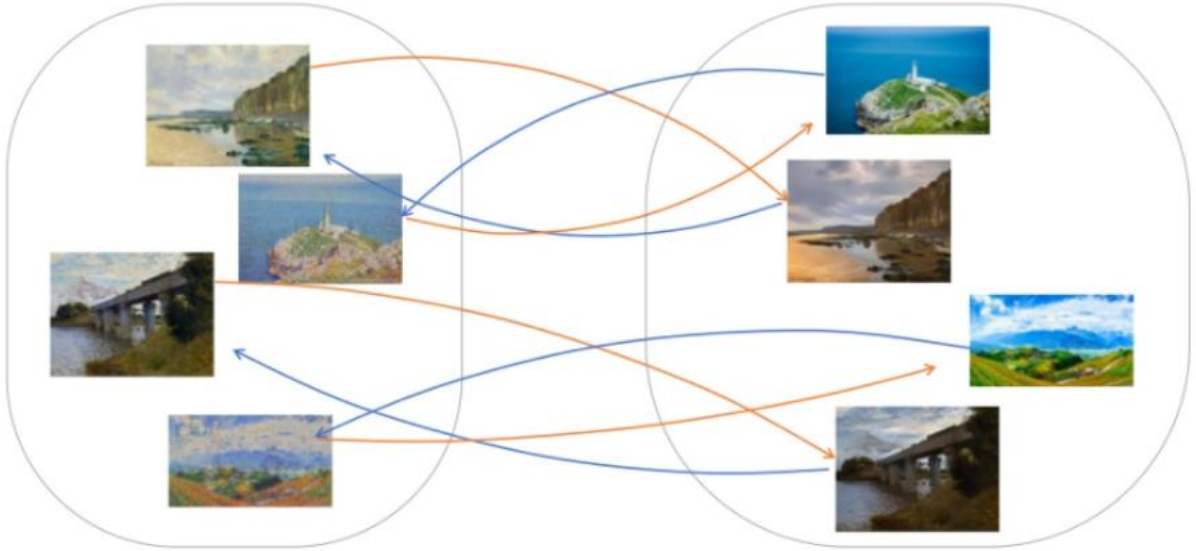


학습 데이터

- 256*256 사이즈

3. 모델 구조

CycleGAN



$$L_{GAN}(G(x),y)+\|F(G(x))-x\|_1 \quad + \quad L_{GAN}(F(y),x)+\|G(F(y))-y\|_1$$

Picture -> Real

- G(x)

Real -> picture

- F(y)

4. 결과

Real -> Monet

4. 결과

Real ->Van gogh