mini project

GAN FOR HANDWRITING

박주현 윤지영 이종호

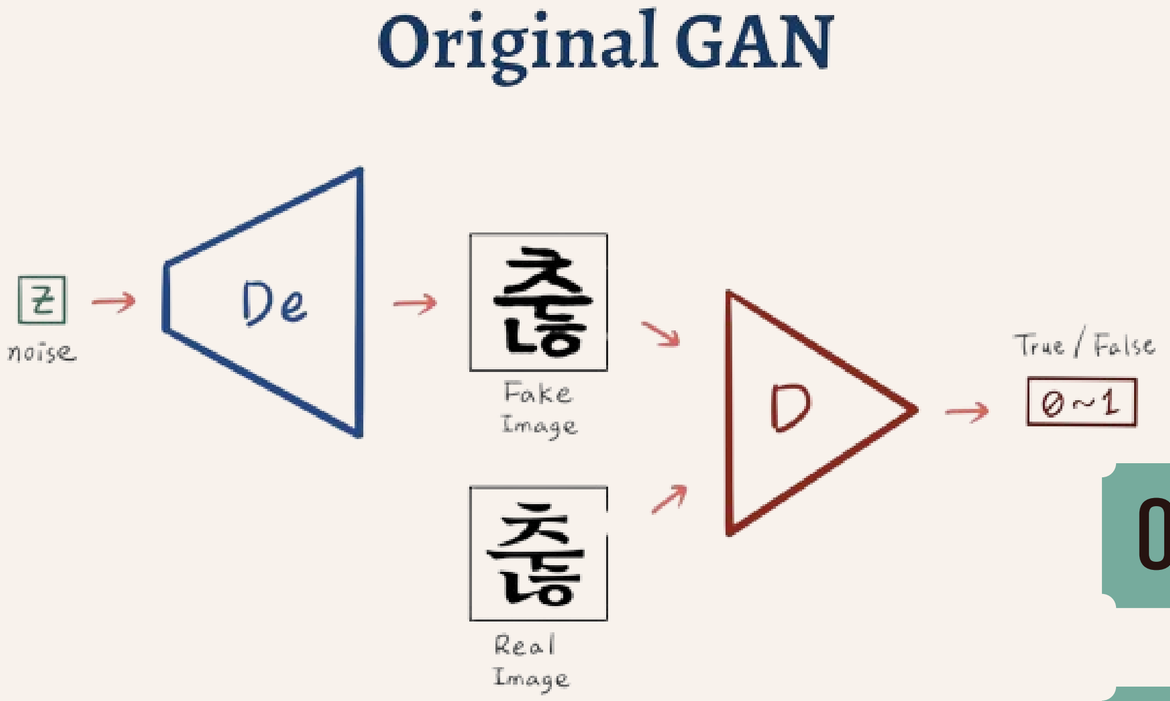
Project Overview

GAN 기반으로 된 모델을 활용해 사람의 손글씨를 학습하고 그 글씨체를 반영한 글자 이미지를 생성하는 프로젝트.
사람의 손글씨를 학습하기 전에, 먼저 약 75,000장 정도의 대량의 컴퓨터 폰트 글자 이미지로 사전 학습을 진행하고, 그 후 약 200장의 적은 양의 사람 손글씨 데이터로 Transfer Learning(전이학습)을 진행.

What is GAN?

GAN은 'Generative Adversarial Network'의 약자.
실제에 가까운 이미지나 사람이 쓴 것과 같은 글 등
여러 가짜 데이터들을 생성하는 모델

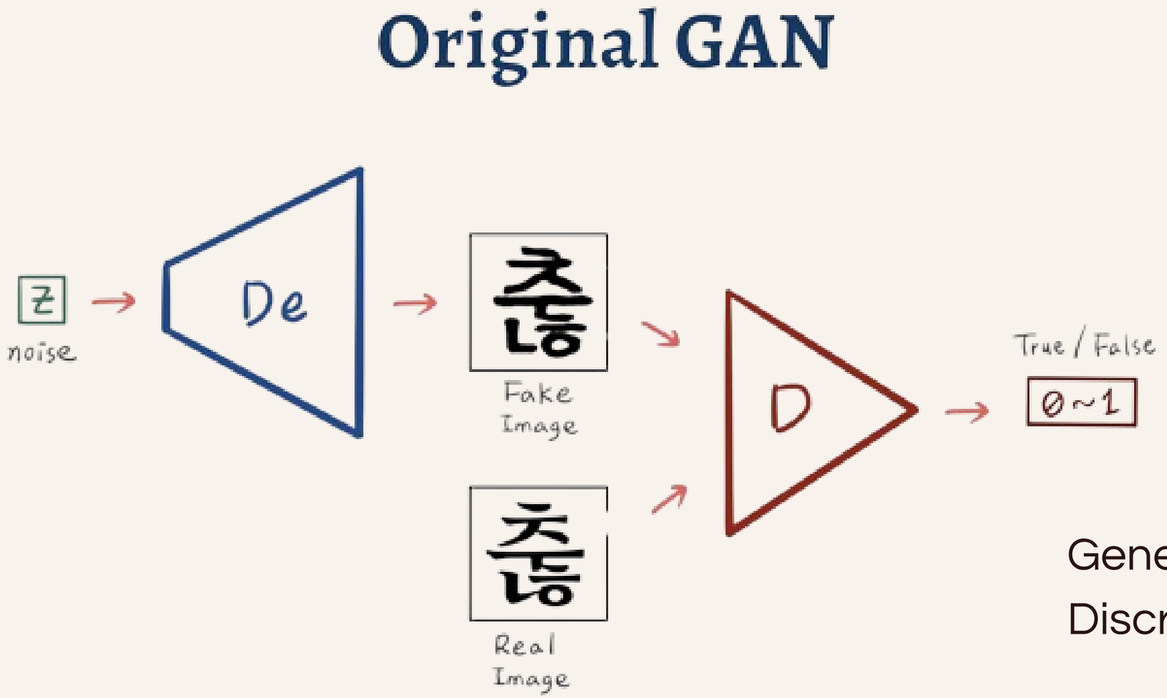
Model structure



01 Discriminator는 이미지를 입력받으면 진짜인지 가짜인지에 대한 예측을 0~1 사이의 값으로 확률값을 출력

02 처음에는 제대로 된 이미지를 생성하지 못하던 Generator는 Discriminator를 속일 수 있을만큼 점점 더 진짜같아 보이는 이미지 생성해내도록 학습

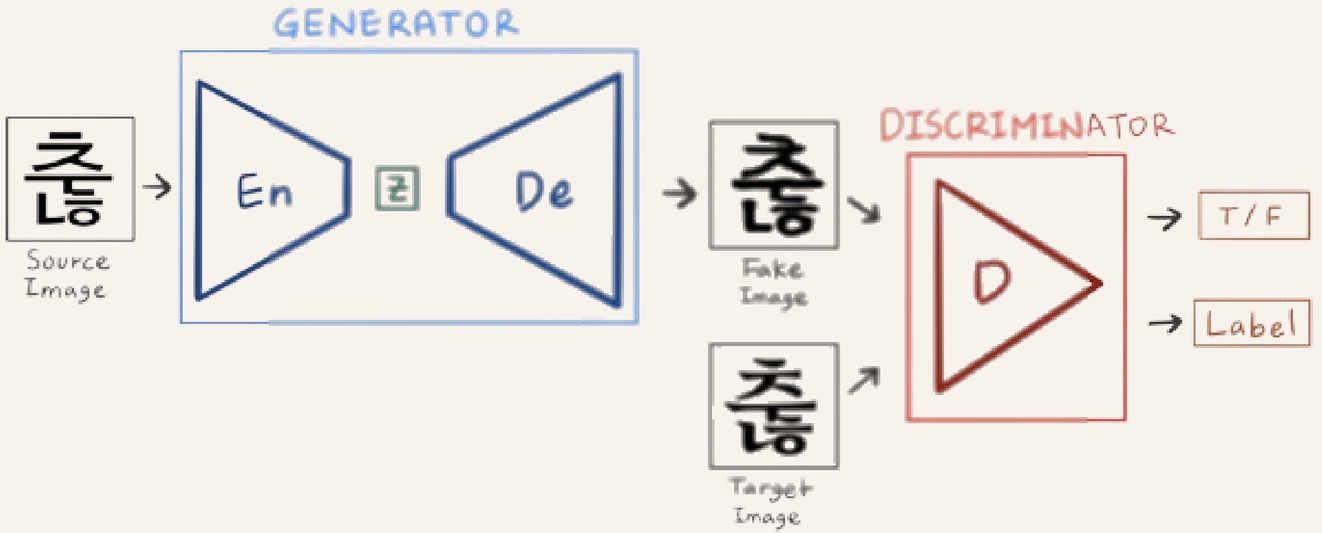
Model structure



Generator: Noise로부터 이미지를 생성
Discriminator: Fake Image인지 Real Image 인지 판별

Discriminator와 Generator가 서로 겨루며
학습하는 것이 GAN 학습의 기본 원리

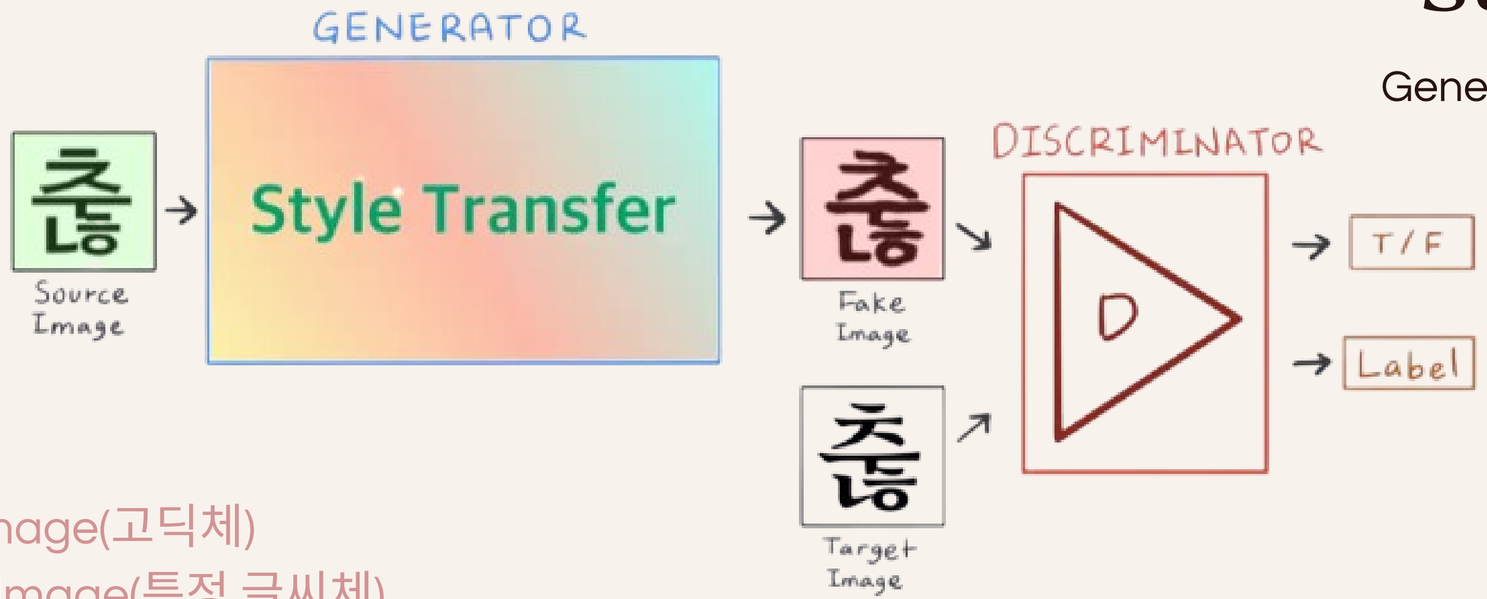
Model structure



Generator 부분은 이미지를 낮은 차원의 벡터로 Mapping 시키는 Encoder와
다시 이미지로 복원하는 Decoder, 두 가지로 구성

Model structure

Generator의 구조와 기능

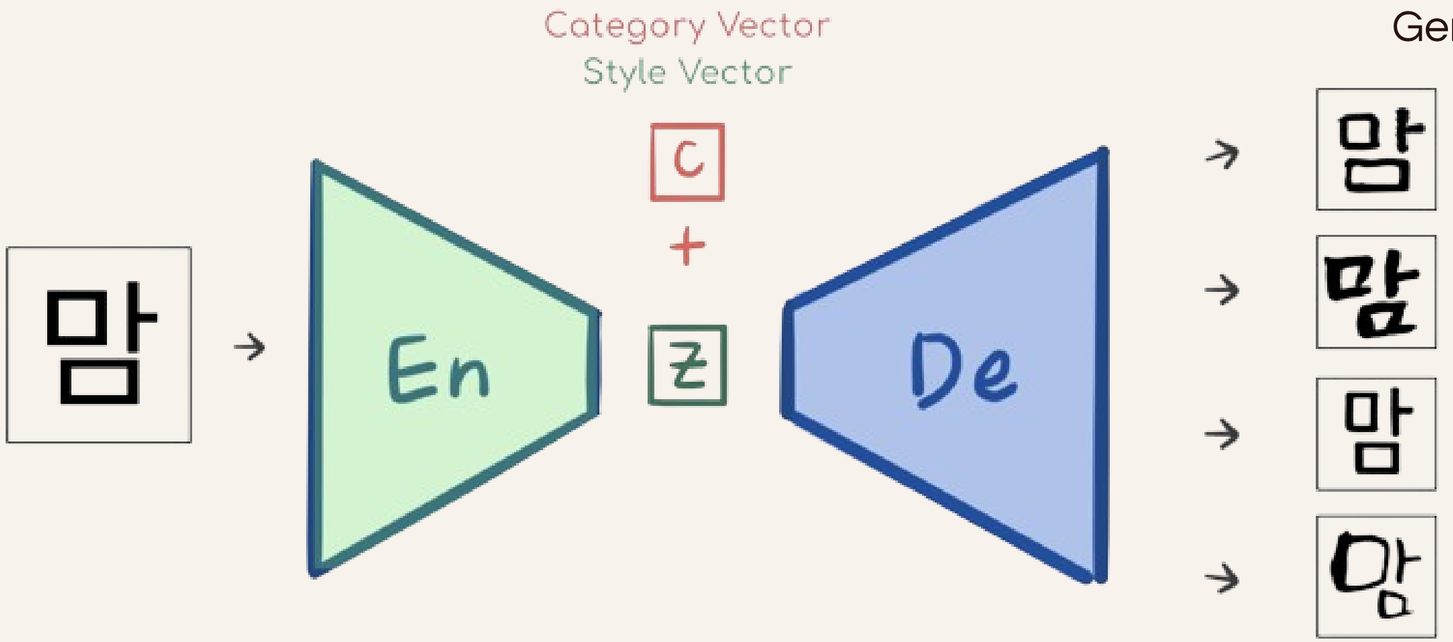


Input : Source Image(고딕체)
Output : Target Image(특정 글씨체)

Generator 부분에서는 고딕체를 새로운 글씨체로 바꾸는,
일명 "스타일 변환 (Style Transfer)" 이 이루어져야 한다.

Model structure

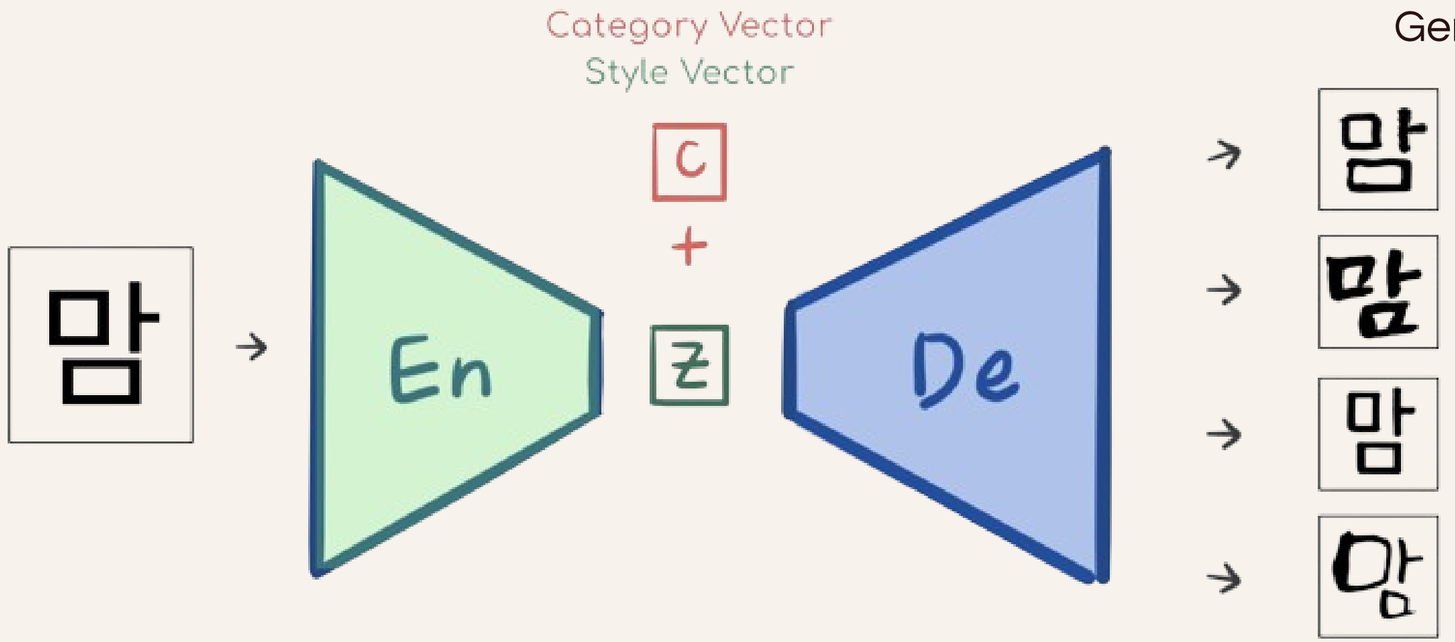
Generator의 구조와 기능



Encoder로 이미지 특징 추출이 끝난 후 Decoder 에 들어가기 전, z벡터와 함께 Decoder 에 입력

Model structure

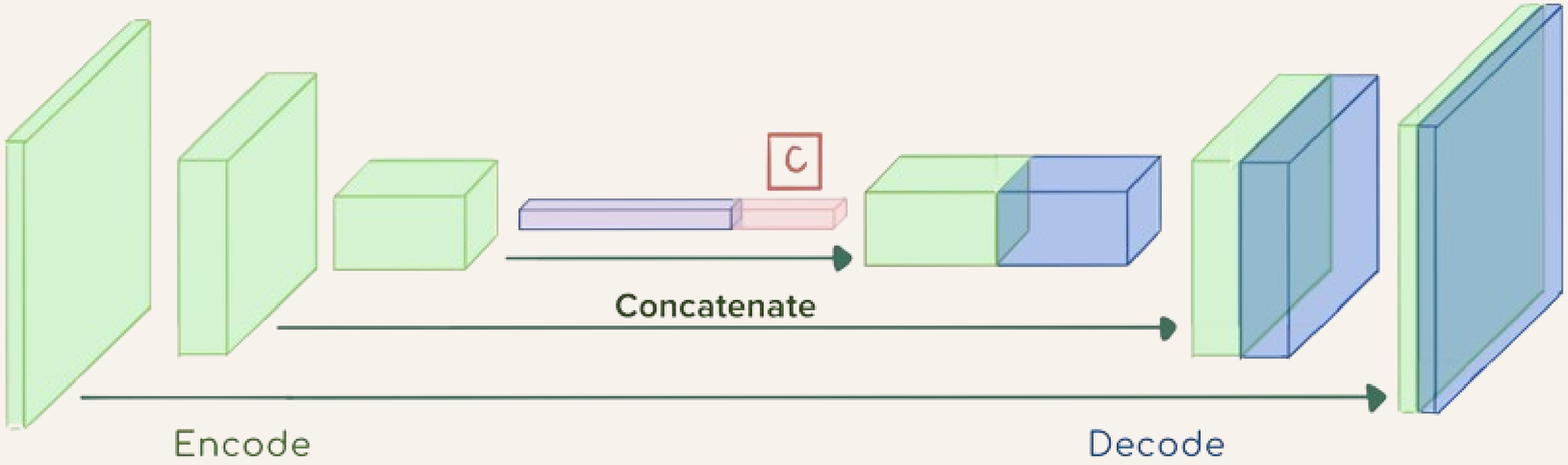
Generator의 구조와 기능



어떤 Style Vector를 입력해주느냐에 따라, 내가 원하는 폰트로 변환을 할 수 있게 되는 것

Model structure

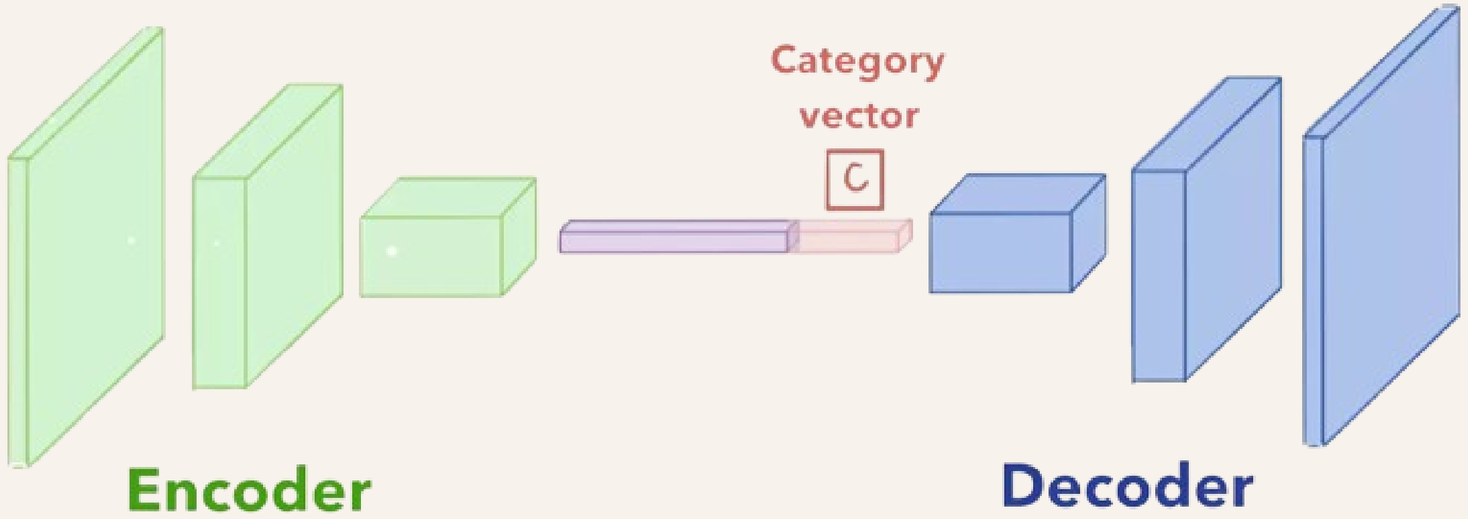
Generator의 구조와 기능



Encoder에서 단계적으로 축소되어가는 벡터들을 모두 Decoder에 단계적으로 입력하는, UNet 구조

Model structure

Generator의 구조와 기능

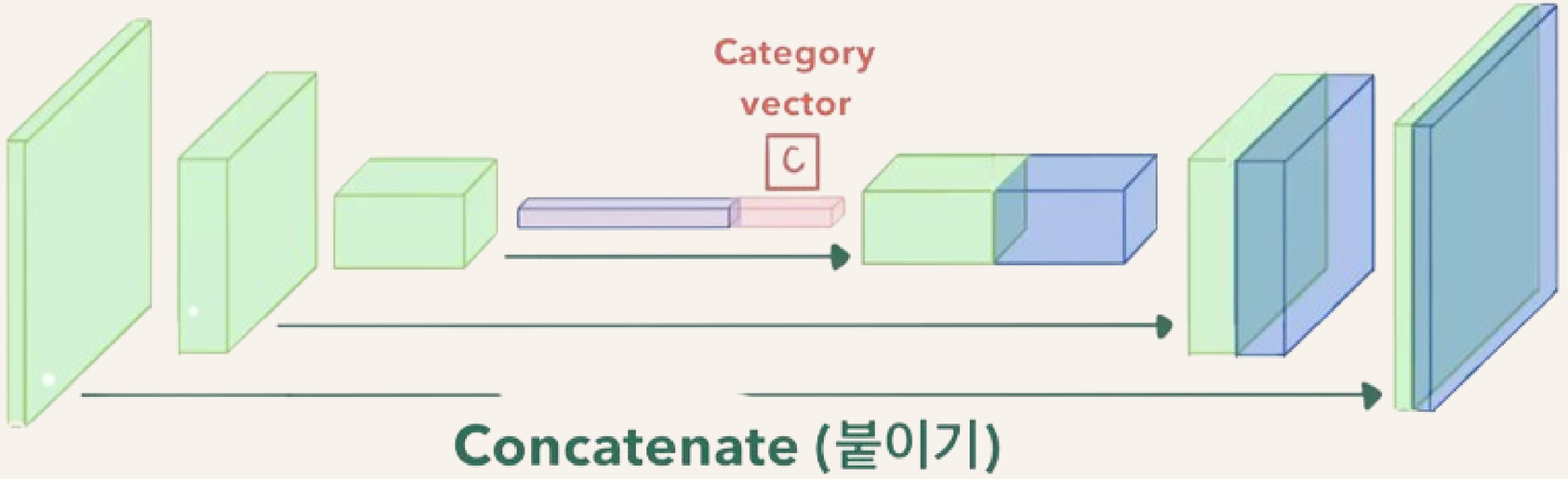


UNet?

이미지를 복원해야하는 Decoder에게 Encoder에서 추출되는 정보를
추가로 입력해줌으로써 약간의 도움을 주는 것

Model structure

Generator의 구조와 기능



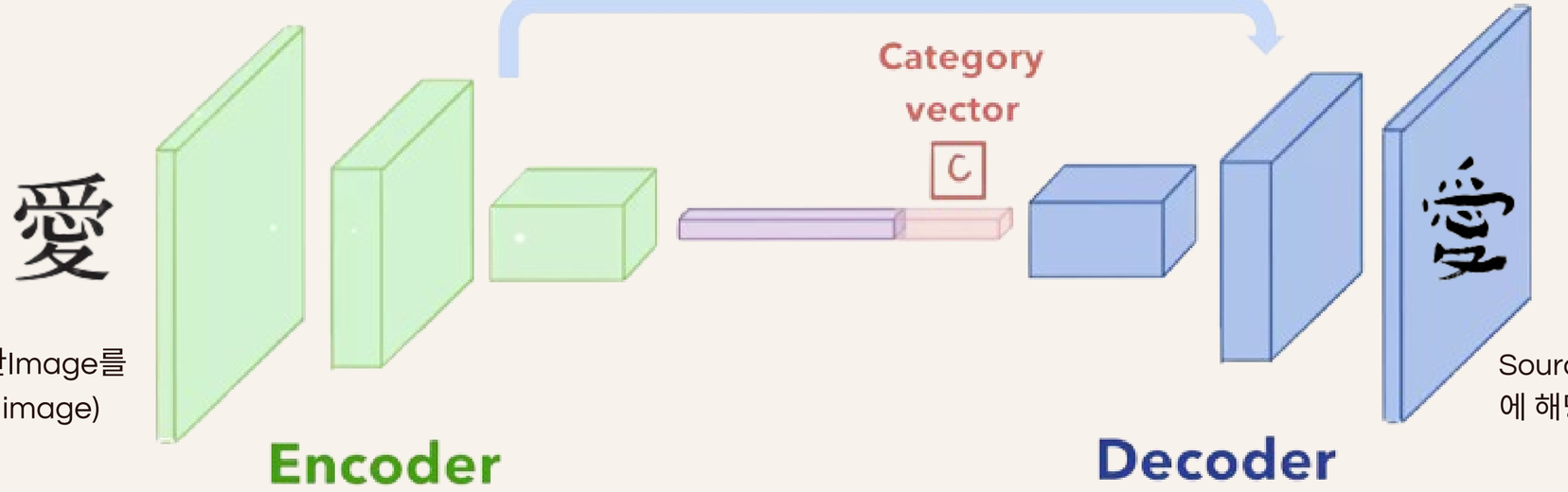
UNet?

이미지를 복원해야하는 Decoder에게 Encoder에서 추출되는 정보를
추가로 입력해줌으로써 약간의 도움을 주는 것

Structure

Generator의 encorder-decorder 구조와 UNet구조

Encoder의 각 레이어를 통과했던 output값들은 저장되어있다가
Decoder학습에 들어오는 input값과 합쳐서 foward된다.



기준이 되는 Basic한Image를
넣어준다.(source image)

Source로 넣은 글씨가 고정 벡터
에 해당하는 스타일로 나온다.

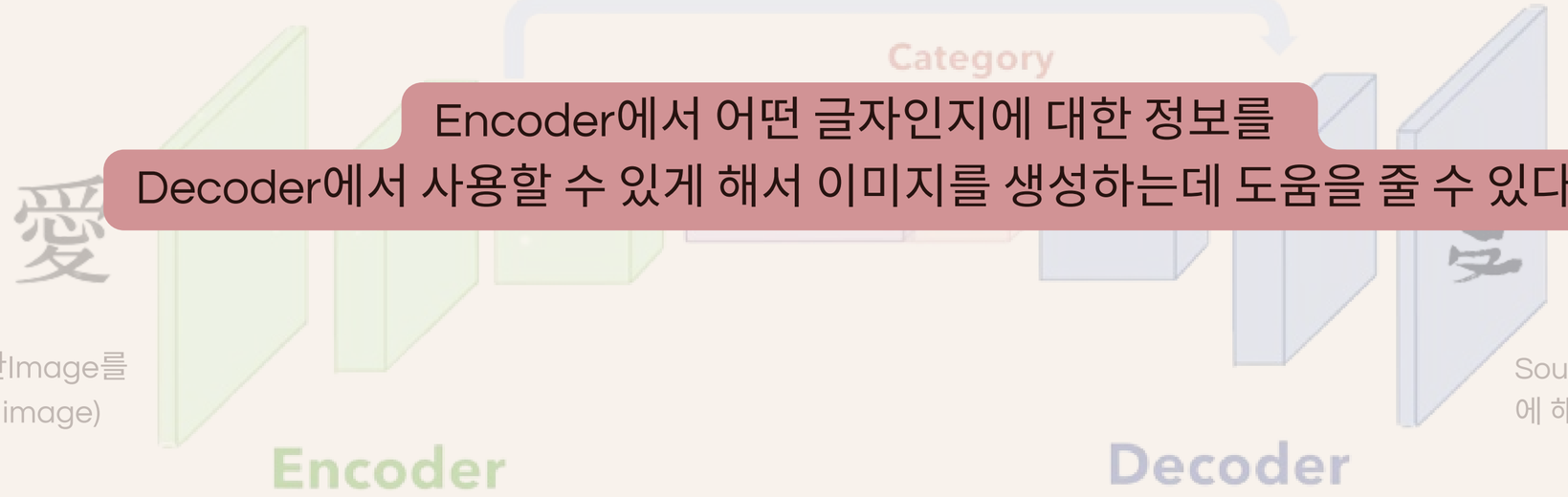
Structure

Generator의 encorder-decorder 구조와 UNet구조

Encoder의 각 레이어를 통과했던 output값들은 저장되어있다가
Decoder학습에 들어오는 input값과 합쳐서 foward된다.

Category

Encoder에서 어떤 글자인지에 대한 정보를
Decoder에서 사용할 수 있게 해서 이미지를 생성하는데 도움을 줄 수 있다.



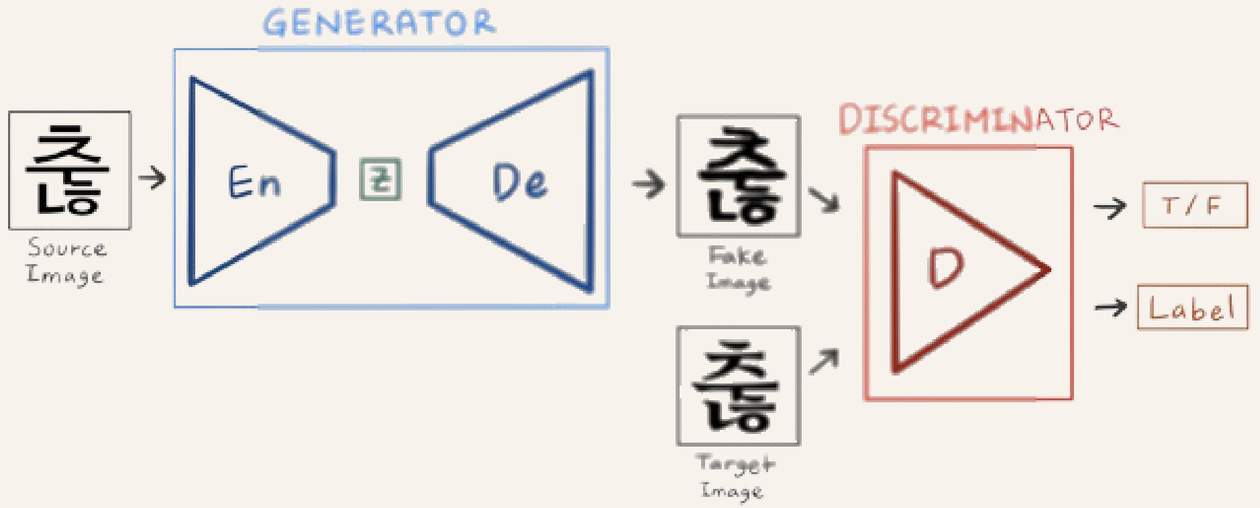
기준이 되는 Basic한Image를
넣어준다.(source image)

Source로 넣은 글씨가 고정 벡터
에 해당하는 스타일로 나온다.

Structure

Dataset

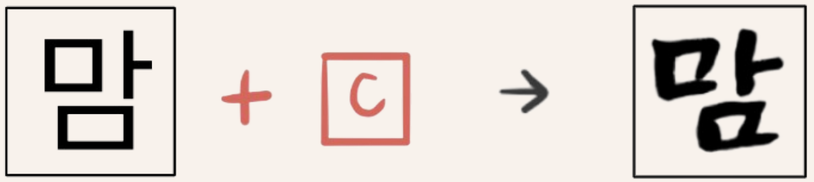
1. Source Image: Generator에 넣을 Source Image → 가장 무난한 고딕체 font 사용
2. Target Image: 모델이 만들어 낸 Fake Image와 비교하며 잘 만들었는지 아닌지 평가 기준이 될 Label 역할을 하는 여러 폰트 → 다양한 스타일을 모델에 학습시키기 위해서 25가지 font(.ttf)사용



First step

pre-Traininig 사전 학습모델 만들기

- Pre-Training 단계에서는 사람의 손글씨 대신, 대량의 데이터셋을 만들기 쉬운 컴퓨터 폰트 이미지로 학습
- 고딕체 글자 + 폰트 카테고리 정보(c) → 스타일이 변환 된 글자
- 추후 모델 학습시 생성하고, fake image가 실제와 동일한지 판별하는 과정을 계속 수행하게 되므로, 필요 시 둘을 분리해 쌍으로 바로 사용할 수 있도록 각 글자에 대해 고딕체 글자와 새로 만든 여러 스타일의 글자의 쌍으로 이미지를 저장

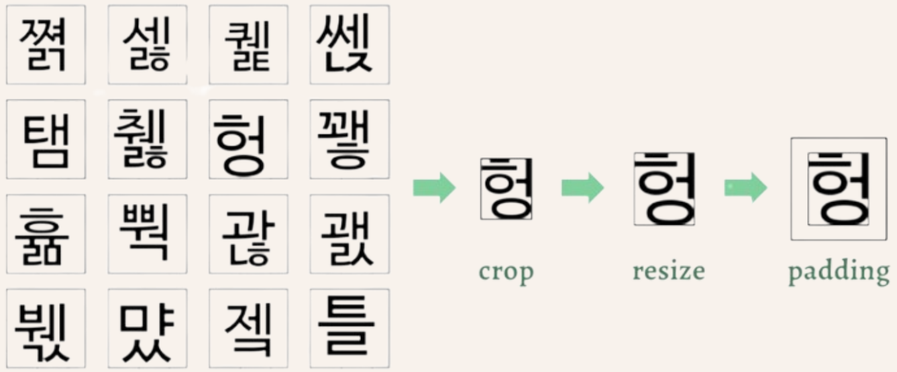


First step

pre-Traininig 사전 학습모델 만들기

```
start = 0xAC00
end = 0xD7A3

uni_hangul = [chr(code) for code in range(start, end + 1)]
uni_num = [ord(chr(code)) for code in range(start, end + 1)]
uni_num = list(map(str, uni_num))

def draw_single_char(ch, font, canvas_size):
    image = Image.new('L', (canvas_size, canvas_size), color=255)
    drawing = ImageDraw.Draw(image)
    w, h = drawing.textsize(ch, font=font)
    drawing.text(((canvas_size-w)/2, (canvas_size-h)/2), ch, fill=(0), font=font)
    flag = np.sum(np.array(image))
    return image

def draw_example(ch, src_font, dst_font, canvas_size):
    dst_img = draw_single_char(ch, dst_font, canvas_size)
    src_img = draw_single_char(ch, src_font, canvas_size)
    example_img = Image.new("RGB", (canvas_size * 2, canvas_size), (255, 255, 255)).convert('L')
    example_img.paste(dst_img, (0, 0))
    example_img.paste(src_img, (canvas_size, 0))
    return example_img

def create_images(src_font_path, dst_font_paths, output_folder, canvas_size):
    src_font = ImageFont.truetype(src_font_path, size=canvas_size)
    for dst_font_path in dst_font_paths:
        dst_font_name = dst_font_path.split('/')[-1][:2]
        dst_font = ImageFont.truetype(dst_font_path, size=canvas_size)

        for idx, char in enumerate(uni_hangul):
            example_img = draw_single_char(char, src_font, canvas_size)
              |....
```

- 25가지의 font(***.ttf 파일형태)로 각각 11720자의 fake image 생성
- Garbage In, Garbage Out → Image preprocessing
- image file → byte 형태의 .pkl 파일로 저장



First step

pre-Traininig 사전 학습모델 만들기

```
data_dir = "./get_data/fonts"
fixed_dir = "./get_data/fonts"
save_path = "./get_data/fonts/fake_samples_img"
from_model_path = "./get_data/save_model"
restore = ['20-1124-17:26-Encoder.pkl', '20-1124-17:26-Decoder.pkl', '20-1124-17:26-Discriminator.pkl']
to_model_path = "./get_data/final_model"

GPU = True
fonts_num = 10
batch_size = 8
img_size = 128

max_epoch = 20
schedule = 10
model_save_step = 2
```

```
[ ] trainer = Trainer(GPU, data_dir, fixed_dir, fonts_num, batch_size, img_size)
```

```
Unpickled total 60015 examples
Unpickled total 14985 examples
train examples -> 60015, val examples -> 14985
total batches: 7502
```

```
[ ] # re-train
l1_losses, const_losses, category_losses, d_losses, g_losses = trainer.train(
    max_epoch, schedule, save_path, to_model_path, lr=0.001, log_step=100, sample_step=2000,
    fine_tune=False, flip_labels=False, restore=restore, from_model_path=from_model_path, with_charid=False,
    freeze_encoder=False, save_nrow=10, model_save_step=model_save_step, resize_fix=90
)
```

```
20 epoch trained model has restored
23:57:59 Epoch [21/40], step [100/7502], l1_loss: 36.9881, d_loss: 3.7656, g_loss: 37.3925
23:58:08 Epoch [21/40], step [200/7502], l1_loss: 39.2969, d_loss: 3.9830, g_loss: 39.6758
23:58:18 Epoch [21/40], step [300/7502], l1_loss: 38.0843, d_loss: 4.0150, g_loss: 38.4162
23:58:27 Epoch [21/40], step [400/7502], l1_loss: 49.6430, d_loss: 4.0772, g_loss: 49.9776
23:58:36 Epoch [21/40], step [500/7502], l1_loss: 40.4632, d_loss: 4.0275, g_loss: 40.8179
23:58:46 Epoch [21/40], step [600/7502], l1_loss: 45.5235, d_loss: 3.9948, g_loss: 45.8757
23:58:56 Epoch [21/40], step [700/7502], l1_loss: 48.4901, d_loss: 4.3210, g_loss: 48.8275
23:59:06 Epoch [21/40], step [800/7502], l1_loss: 43.2269, d_loss: 4.1366, g_loss: 43.5380
```

[1단계] Training from Scratch
: 밑바닥부터 학습하는 단계 (30epoch)

[2단계] Further Training for Sharp Image
: 보다 뚜렷한 이미지로 개선하는 단계 (120epoch)

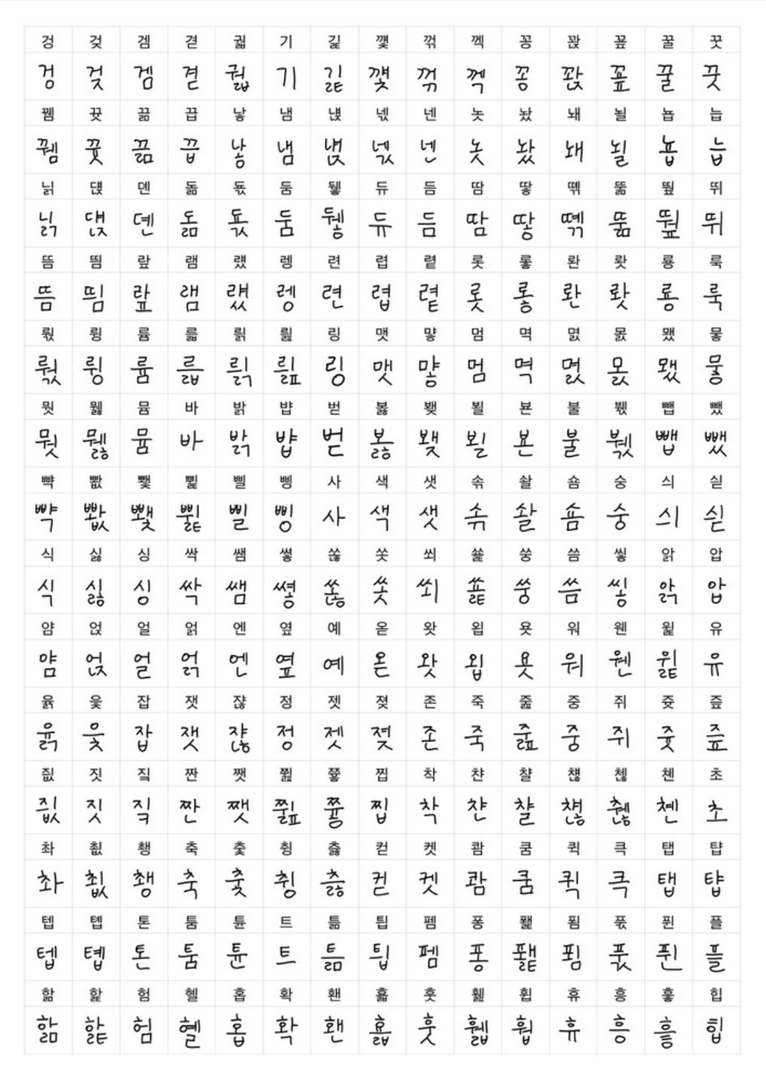
GPU 기준 약 35시간 소요

Second step

Transfer Learning 손글씨 학습하기

사람의 손글씨 모델이 생성한 글자

컴퓨터 폰트에 대해서는 거의 99% 완벽하게 베껴내는 사전학습 모델을 가지고,
보다 어려운 사람의 손글씨를 학습시키는 것



Second step

Transfer Learning 손글씨 학습하기

- 210자의 글자 템플릿 위에 직접 손글씨를 작성해서 전이학습에 사용할 데이터를 생성
- 210자는 모델이 학습하기 어려운 복잡한 글자와 많이 사용되는 비교적 쉬운 글자에 대해 균형잡힌 학습을 할 수 있도록 적절히 섞어서 구성되어 있음
- pre-training 모델에 사용한 데이터처럼 고딕체와 쌍을 이룬 데이터로 만들어서, 동일하게 crop → resize → padding 의 전처리 작업을 진행
- 글자의 특징을 추출하는 Encoder는 학습된 그대로 사용 & 생성하는 Decoder는 다시 학습

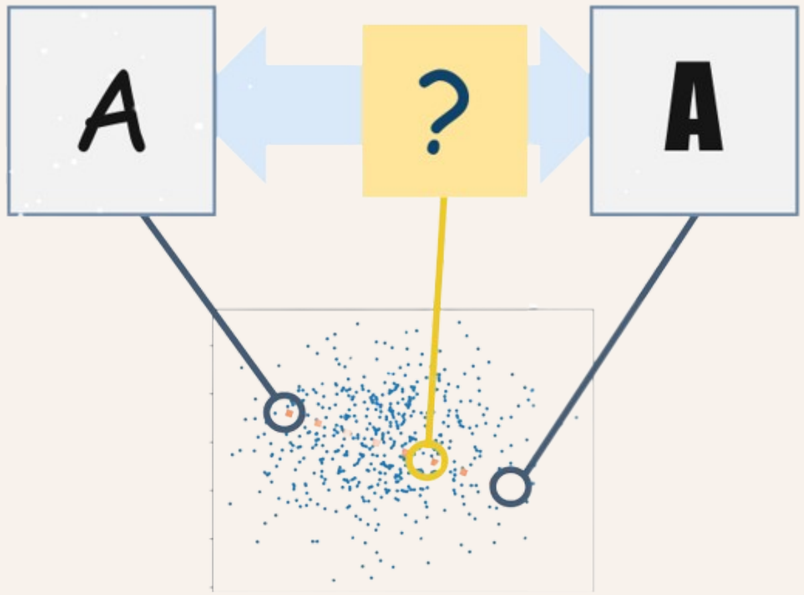
Limitation

한계

150 epoch을 돌렸지만 학습시키는데 시간이 많이 소모됨
다른 GAN 모델처럼 제대로 글씨가 생성되지 않음

Limitation

한계

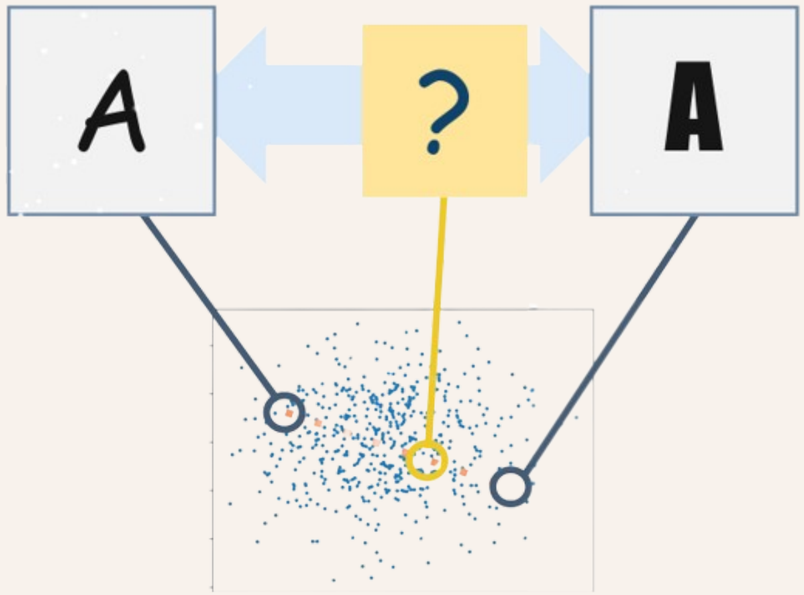


직접 폰트를 함축된 표현으로 매핑

➡ 잠재 공간(함축된 표현) 사이를 스크롤하는 방식으로 구현

Limitation

한계



유저가 두개의 폰트를 선택하면 폰트를 결합해 새로운 폰트를 만드는 것

Create Dataset

Font to Image

- Source Image: 고딕체
- Target Image: 5가지 종류의 font 사용
- crop-resize-padding

Image to pkl

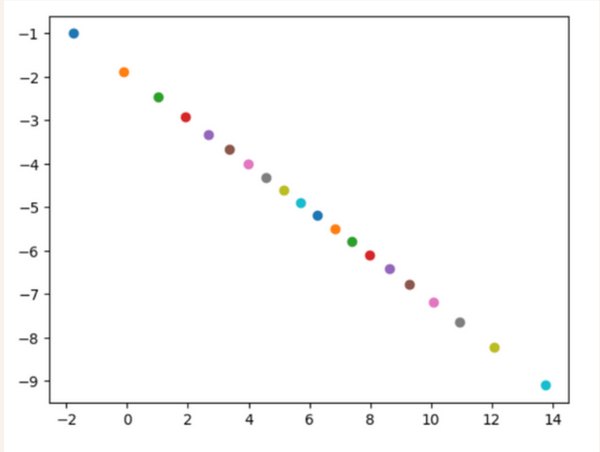
- .png 파일은 무겁고, 많은 이미지 파일이 학습에 사용되므로, .pkl 파일로 만들어서 보관하는 방법을 선택

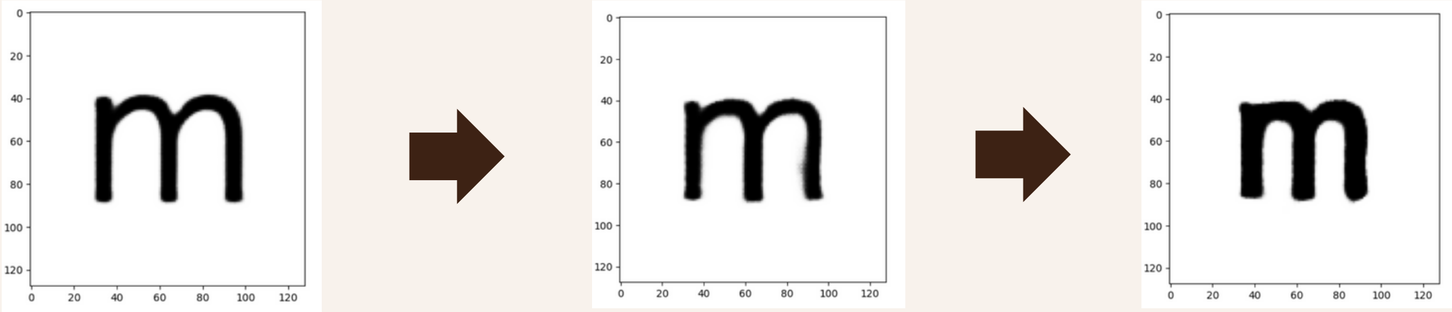
Normalization

- 모델의 표현력을 높여주기위해서 (-1 ~ 1)로 정규화를 시켜준다.

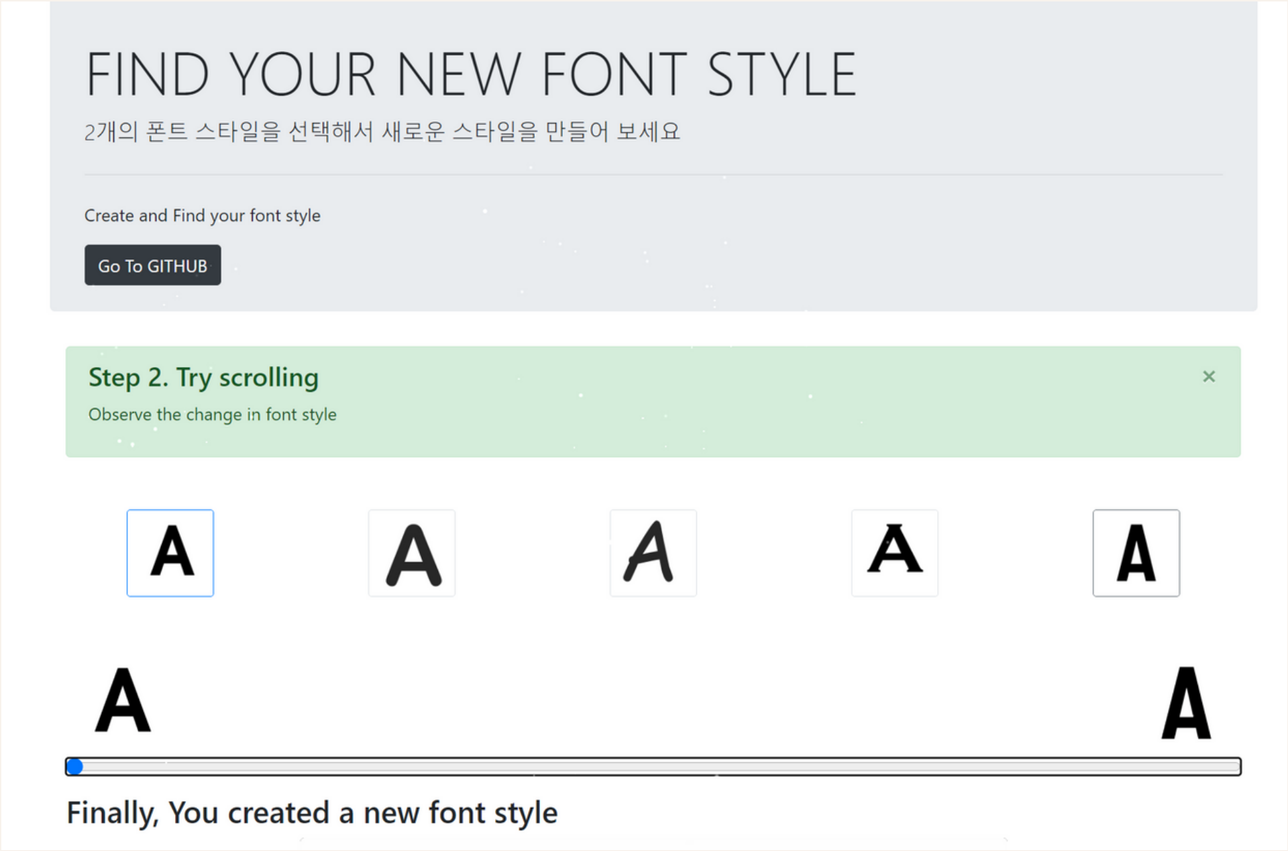
Model & Train

- 모델은 GAN을 기본 구조로 사용
- Generator는 Encoder와 Decoder로 나누었고 Encoder의 출력을 Decoder에서 사용하는 UNet 구조를 이용
- 이미지 사이즈는 128*128 사이즈로 Encoder에서 2배씩 늘어나고 Decoder에서는 2배씩 줄어든다.
- 총 Encoder는 7개의 레이어, Decoder는 8개의 레이어로 구성되어 있다.
- 이미지 스타일이 변해가는 과정을 확인할 수 있다.





Conclusions



file:///C:/Users/user/Downloads/index.html

Thank you!

박주현 윤지영 이종호