KUBIG 장기프로젝트-CV

손글씨 따라쓰기 프로젝트

14기 구은아
15기 공도웅
15기 박지우
15기 신윤
15기 이승은

순서

프로젝트 주제
및 배경

프로젝트
시도들

Transformer
모델

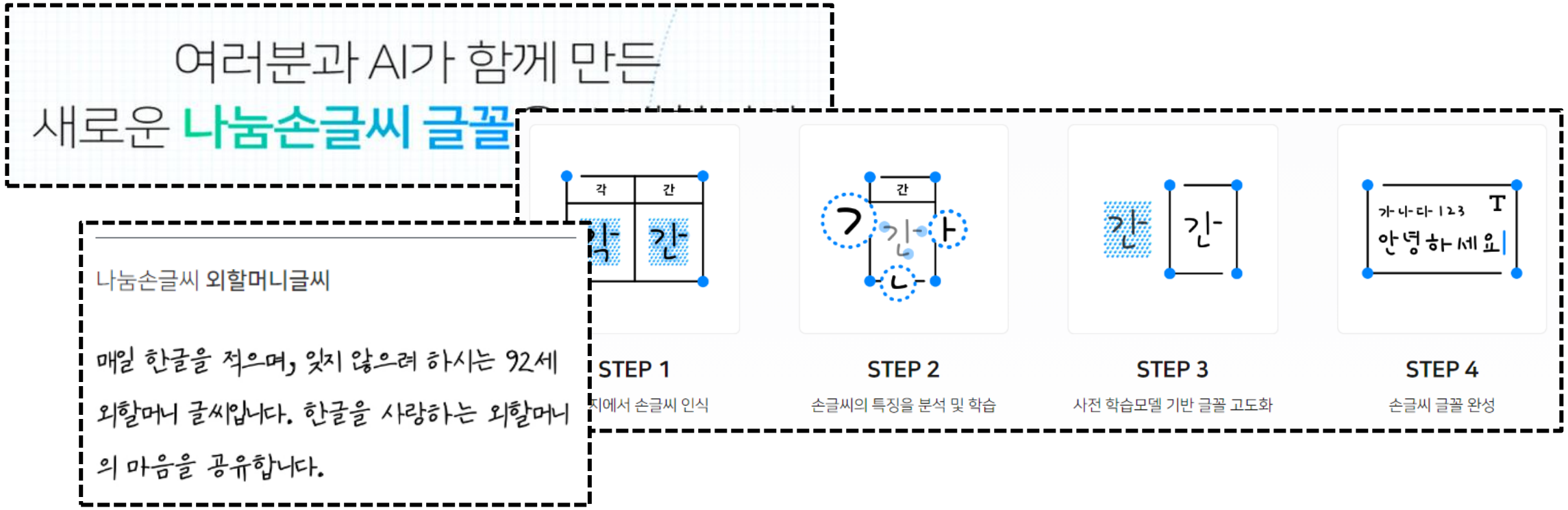
HWT 모델

프로젝트
결과

한계점
및 발전점

프로젝트 주제 및 배경

프로젝트 배경

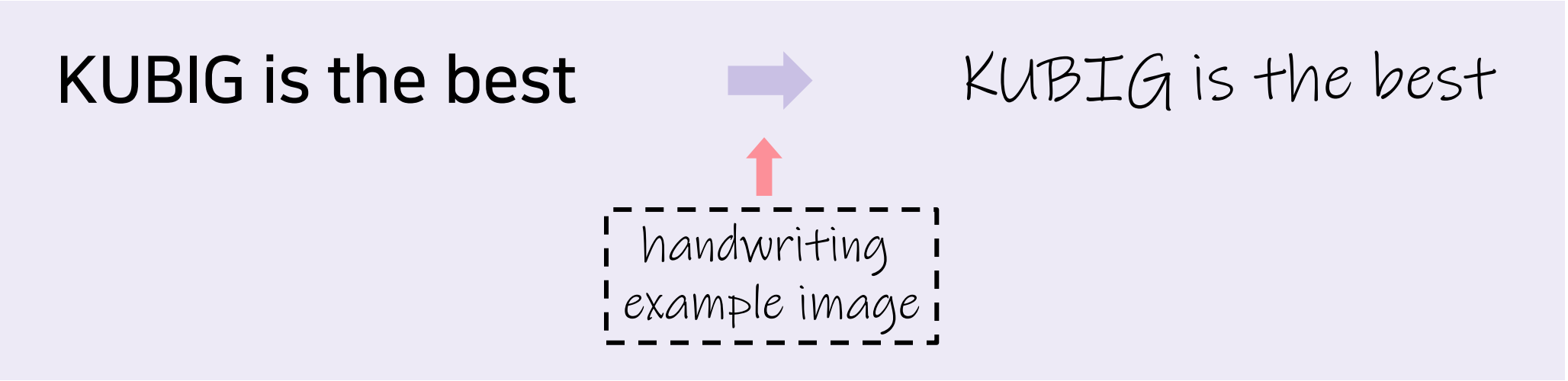


네이버 CLOVA에서 인공지능 기술을 활용하여

일반인의 손글씨를 한글 글꼴로 제작하여 배포한 프로젝트에서 착안

프로젝트 주제 및 배경

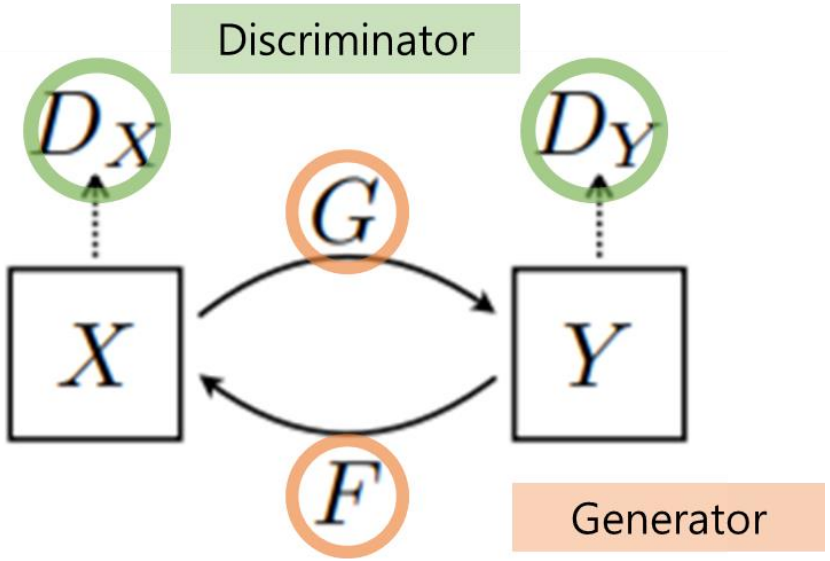
프로젝트 주제: 손글씨 따라쓰기



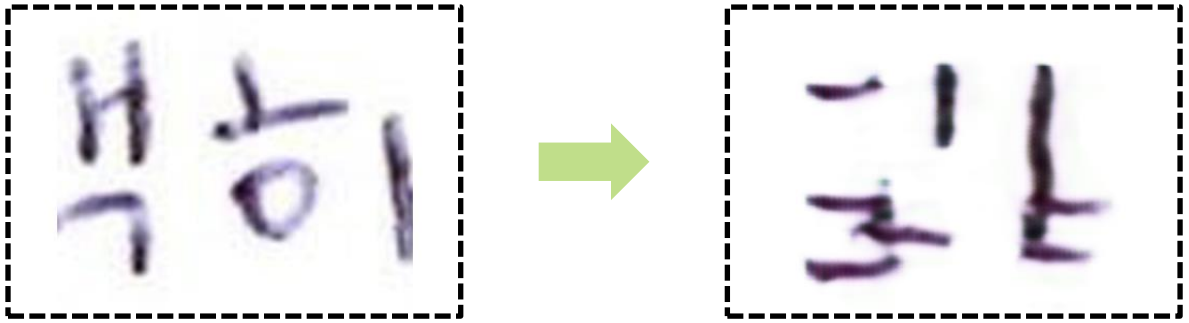
- 네이버의 프로젝트에서 착안하여 **컴퓨터 글꼴로 쓰인 문구를 주어진 손글씨로 따라쓰는** 프로젝트 주제 선정
- 폰트를 만들어 서비스화한 네이버의 프로젝트와 달리 본 프로젝트는 기존 문구를 손글씨로 변환하는 프로젝트

프로젝트 시도들

Cycle GAN



- 두 이미지가 상대 이미지의 스타일로 변환하는 style translation 모델
- Generator(G, F): 상대 이미지 스타일로 변환된 이미지 생성
- Discriminator(D_X, D_Y): 생성된 이미지와 실제 이미지를 구분



Cycle GAN으로 글씨체를 바꿔보고자 했으나 style image의 전체적인 스타일만 (ex. 펜, 종이 재질) 반영되고,
글씨 모양 자체는 변화하지 않음

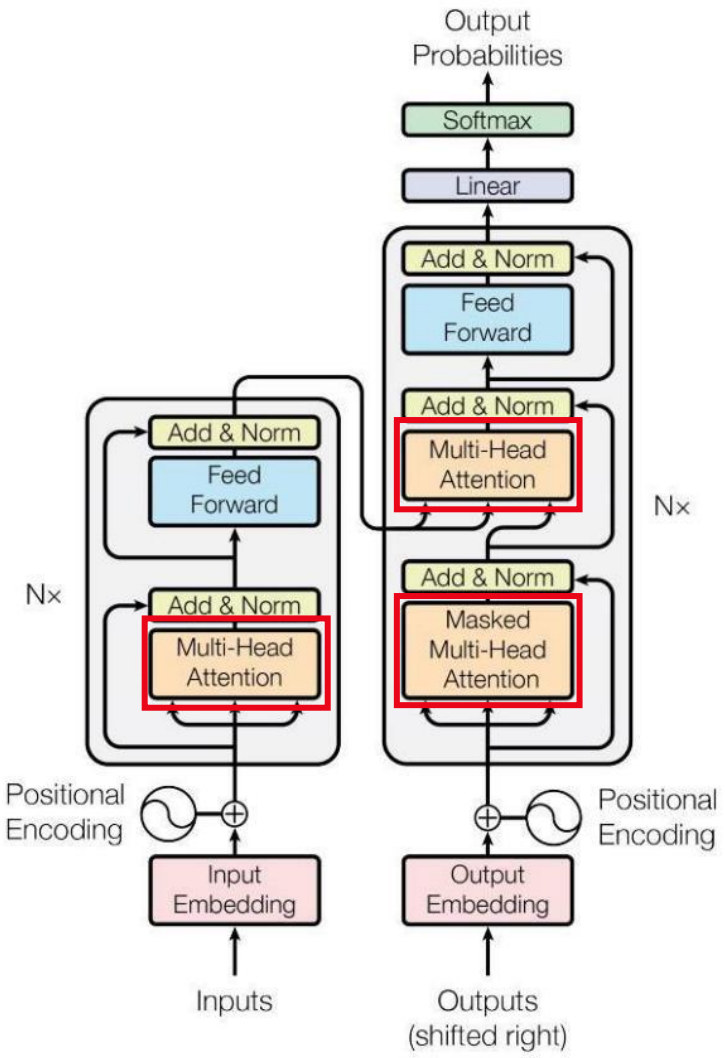
프로젝트 시도들

Selective Style Transfer for Text

- 이미지에서 변환하고자 하는 텍스트 부분만 스타일을 변환하는 style transfer 모델
- 이미지에서 변환하고자 하는 글자 위치를 인식 후, 해당 부분만 crop
- source content 이미지에 스타일을 적용

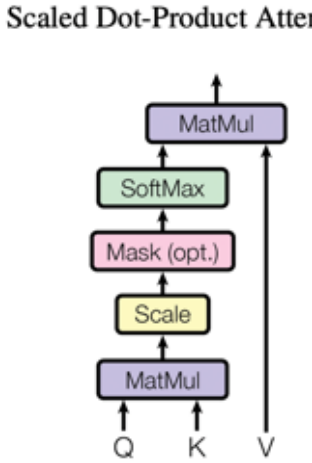
Cycle GAN과 동일하게 style image의 전체적인 스타일만 반영되고,
글씨 모양 자체는 변화하지 않는 문제 발생

Transformer 모델



Transformer 모델

- encoder-decoder 구조로 이루어져 있으며, 모두 attention만을 사용하여 만든 모델
- scaled dot-product attention 기법으로 input token 간 연관도를 수치로 계산

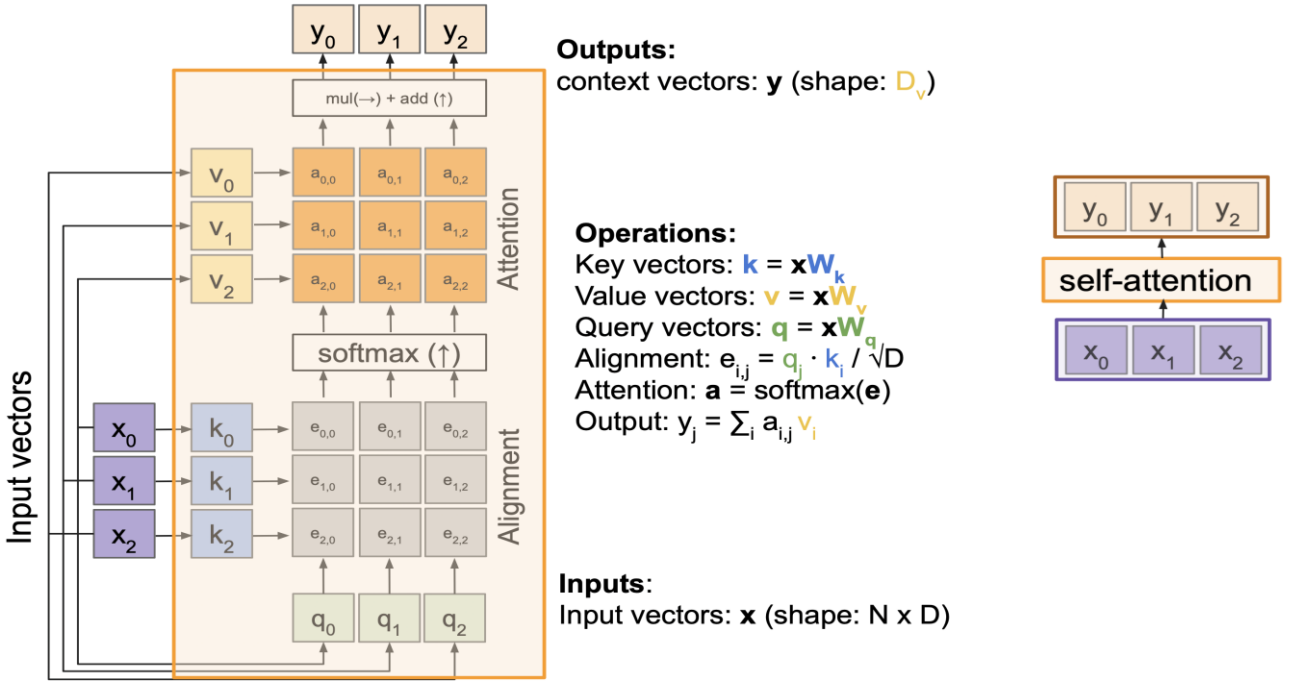


$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

- 이후 사용할 HWT 모델은 Transformer 모델을 기반으로 만들어짐

Transformer 모델

Self attention



나는	자연어	자연어	자연어	처리를	자연어	즐겨한다	자연어
Dot product		Dot product		Dot product		Dot product	
0.3		0.3		0.25		0.15	

- 원래 attention은 query와 key vector간 연관도를 계산하여 value vector에 적용하는 기법
- self-attention은 query, key, value vector가 모두 같음
- 즉, input을 구성하는 token 간 연관도를 계산하여 같은 vector 내부에서 중요한 부분을 파악

HWT 모델

Motivation

1. Style-Content Entanglement: 이전 GAN 기반 모델은 style과 content를 따로 처리하고 나중에 연결했기 때문에 두 feature의 연결이 약했는데, 이 문제를 해결하여 style-content entanglement를 강화.
2. Global and Local Style Imitation: 글씨체에서 global한 특성(예: 두께 등), local한 특성(예: 글씨체 등) 2가지 style 모두를 잘 학습해야 함.

Generator

주어진 style을 기반으로 손글씨 이미지 생성. encoder와 decoder로 이루어짐.

Discriminator

Generator에서 생성된 손글씨 이미지가 생성된 것인지 실제 이미지인지 구분.

Recognizer

Generator에서 생성된 손글씨 이미지에 text content가 제대로 쓰여있는지 평가.

Style Classifier

Generator에서 생성된 손글씨 이미지에 글씨체 style이 제대로 반영되었는지 평가.

HWT 모델

Generator 구조

encoder

$style\ example\ \mathrm{X}_i^s \rightarrow feature\ vector\ Z$

1) style example을 CNN에 통과시켜 style 정보를 담은 sequence Z 생성
2) Z를 multi-headed self-attention 레이어에 통과시켜 final Z 생성

decoder

$(Z, text\ \mathcal{A}) \rightarrow handwritten\ image\ \tilde{X}_i^t$

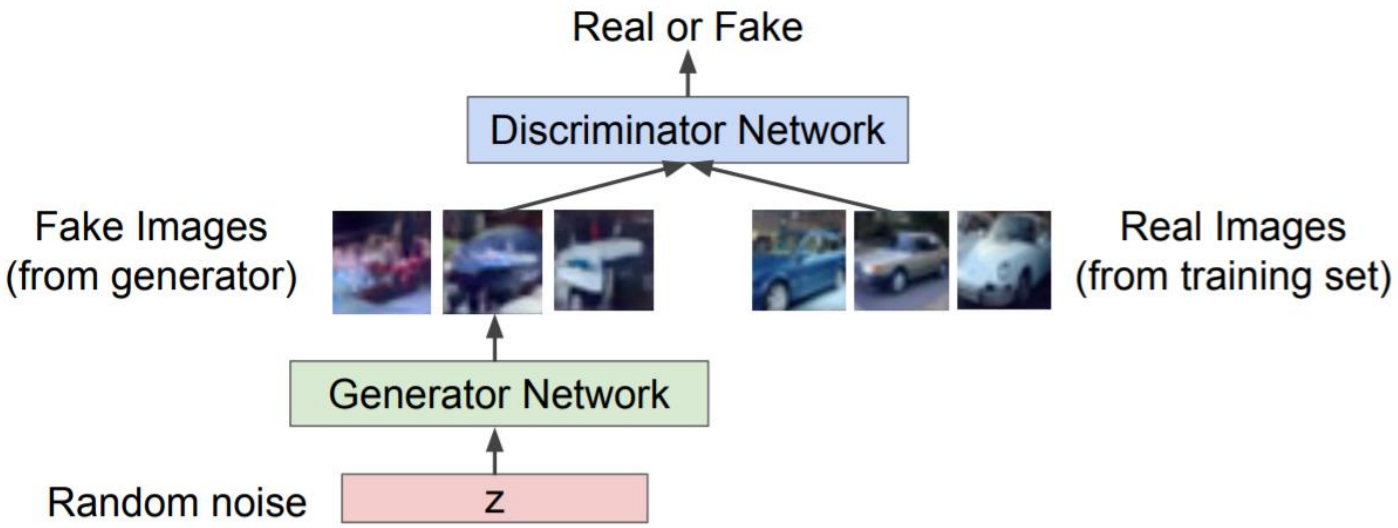
1) 초반: multi-headed attention과 encoder-decoder attention
2) 후반: 4개의 residual 레이어와 tanh 레이어
3) 위 레이어로 이루어진 decoder를 통과하여 손글씨 이미지 $\tilde{X}_i^t$ 생성

encoder와 decoder 모두 Transformer과 CNN 둘 다 기반으로 하는 하이브리드 구조로 설계함으로써 두 모델의 장점을 모두 가진 Generator를 만듦.

HWT 모델

Generator-Discriminator 학습

GAN과 같은 생성기-판별기 대결 구조로 학습



Generator

Discriminator를 속일 수 있을 정도로 진짜 같은 이미지를 만듦

↕

Discriminator

진짜 이미지와 Generator가 만든 가짜 이미지를 구별

HWT 모델

Generator-Discriminator 학습

GAN과 같은 생성기-판별기 대결 구조로 학습

loss function

$$L_{adv} = \mathbb{E}\left[max\left(1 - D(\mathrm{X}_i^s, 0)\right)\right] + \mathbb{E}\left[max\left(1 + D\left(G(\mathrm{X}_i^s, \mathcal{A})\right), 0\right)\right]$$

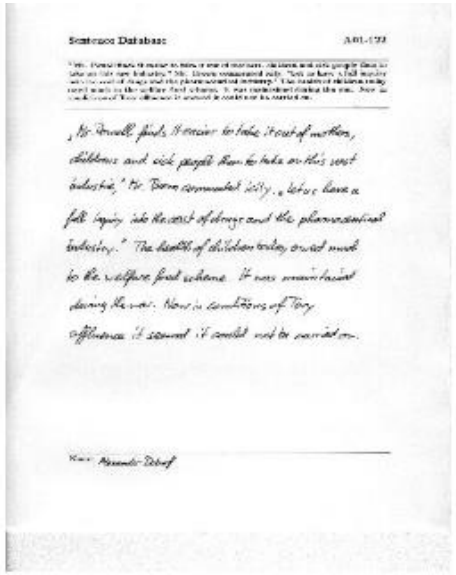
Discriminator가 **진짜 손글씨 이미지**를 판별할 때 발생하는 loss

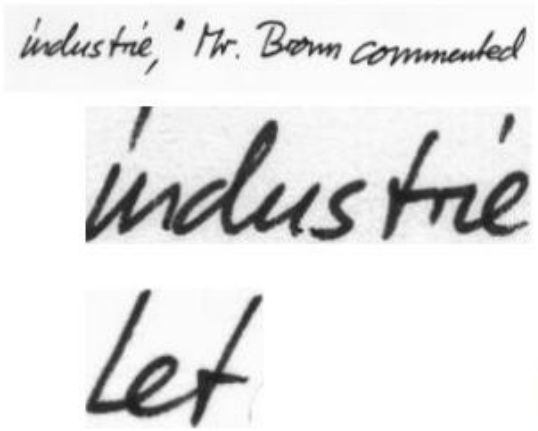
Discriminator가 **가짜 손글씨 이미지**를 판별할 때 발생하는 loss

원래 GAN 모델처럼 하나의 loss function에 Discriminator와 Generator의 optimize 조건을 결합하여 학습

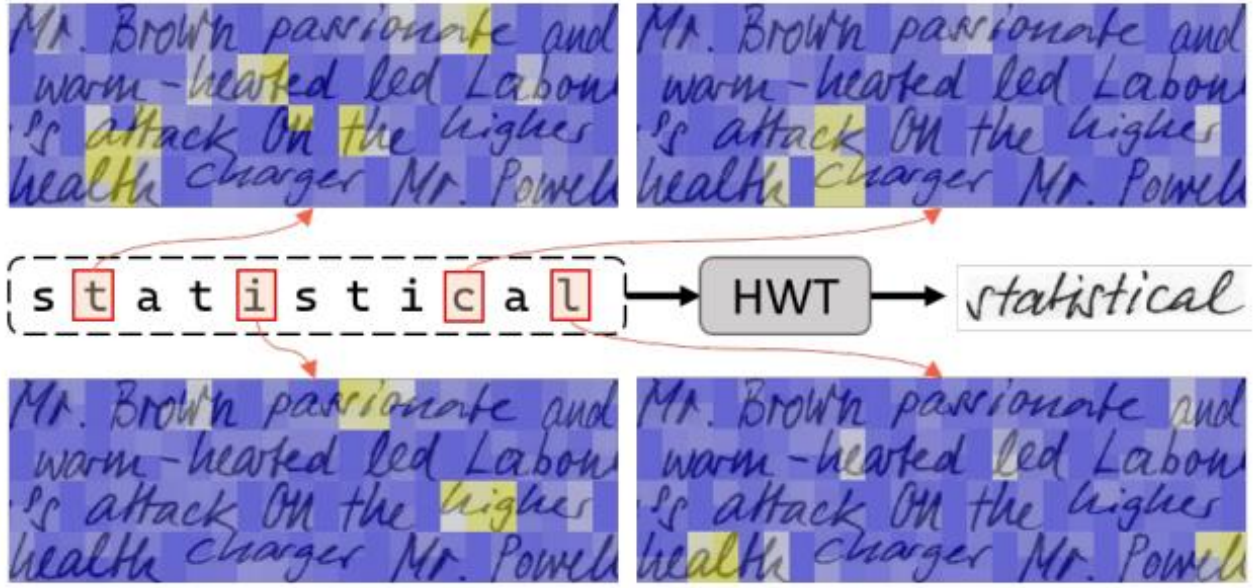
HWT 모델

학습 데이터 : IAM dataset





- 영어 손글씨 이미지 dataset
- 500명의 사람이 작성한 9862개의 문장 이미지가 포함
- 340명의 데이터는 training에, 160명의 데이터는 testing에 사용



HWT 모델

코드 실행

```
[ ]  # !git clone https://github.com/ankanbhunia/Handwriting-Transformers
     # %cd Handwriting-Transformers
     # !pip install --upgrade --no-cache-dir gdown
     # !gdown --id 16g9zgysQnWk7-353_tMig92KsZsrcM6k && unzip files.zip && rm files.zip
```

```
Cloning into 'Handwriting-Transformers'...
remote: Enumerating objects: 497, done.
remote: Counting objects: 100% (112/112), done.
remote: Compressing objects: 100% (71/71), done.
remote: Total 497 (delta 54), reused 72 (delta 41), pack-reused 385
Receiving objects: 100% (497/497), 67.85 MiB | 19.33 MiB/s, done.
Resolving deltas: 100% (197/197), done.
/content/drive/MyDrive/Handwriting-Transformers
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: gdown in /usr/local/lib/python3.8/dist-packages (4.4.0)
Collecting gdown
  Downloading gdown-4.6.0-py3-none-any.whl (14 kB)
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.8/dist-packages (from gdown) (2.23.0)
Requirement already satisfied: six in /usr/local/lib/python3.8/dist-packages (from gdown) (1.15.0)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.8/dist-packages (from gdown) (4.6.3)
```

1

HWT 모델 git clone

HWT 모델

코드 실행

```
[43] text = "Whenever you feel like criticizing anyone, just remember that all the people in this world haven't had the advantages that you've had"
     output_path = 'results'
```

2

text: 생성하고자 하는 문구 입력

```
     datasetval = torch.utils.data.DataLoader(
                  TextDatasetObjval,
                  batch_size=batch_size,
                  shuffle=True,
                  num_workers=0,
                  pin_memory=True, drop_last=True,
                  collate_fn=TextDatasetObjval.collate_fn)

     print ('(2) Loading model...')

     model = TRGAN()
     model.netG.load_state_dict(torch.load(model_path))
     print (model_path+' : Model loaded Successfully')
```

3

model: IAM dataset으로
pre-trained된 HWT 모델

HWT 모델

코드 실행

```
[53]  img = torch.stack((imgs_pad[0], imgs_pad[1], imgs_pad[2], imgs_pad[3], imgs_pad[4], imgs_pad[5], imgs_pad[6], imgs_pad[7]), 0)
```

```
[54]  img2 = torch.stack((imgs_wids[0], imgs_wids[1], imgs_wids[2], imgs_wids[3], imgs_wids[4], imgs_wids[5], imgs_wids[6], imgs_wids[7]), 0)
```

```
[56]  print(batch_size, img.shape, img2.shape)
```

```
      8 torch.Size([8, 15, 32, 768]) torch.Size([8, 15])
```

```
[55]  page_val = model._generate_page(img, img2, eval_text_encode,e
      cv2.imwrite(output_path+'/image' + '20221230-exp5' + '.png',
```

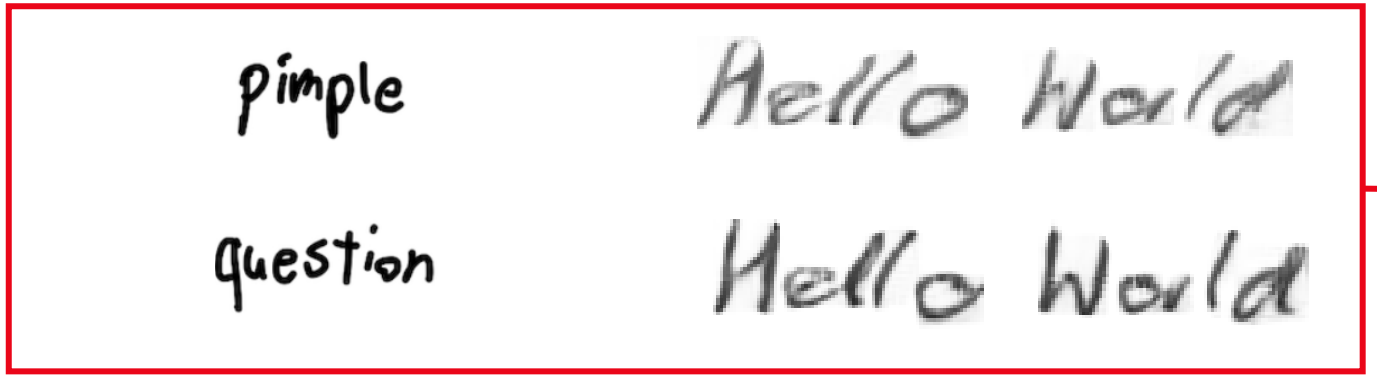
```
      True
```

4

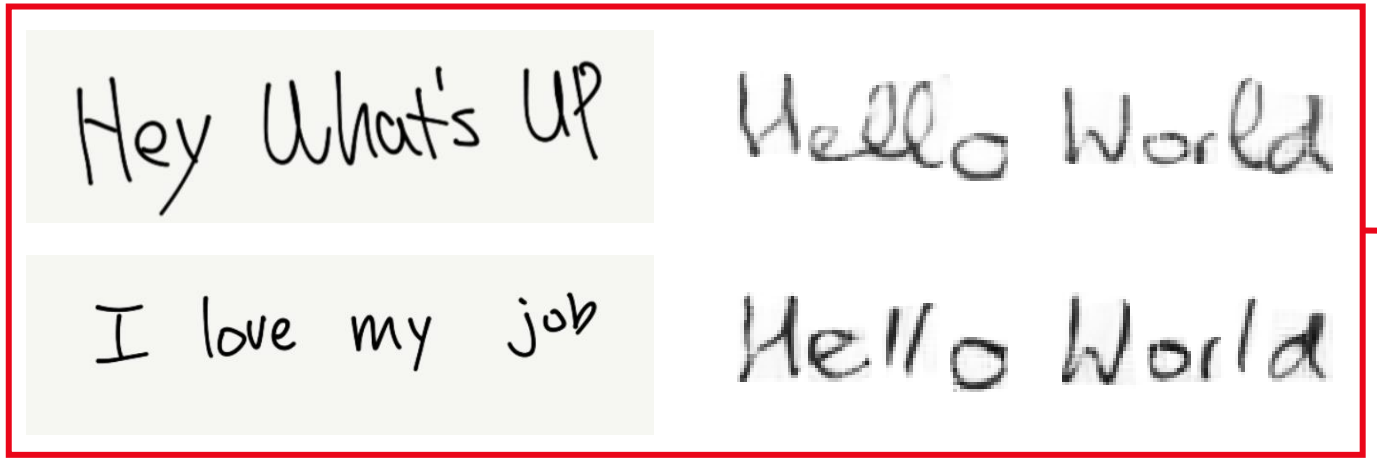
batch size 8
input size (8, 15, 32, 768)
로 모델 실행

프로젝트 결과

Example 1: Hello World



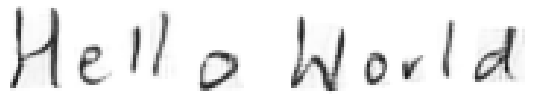
같은 글씨체 이미지를 사용하면 비슷한 결과가 나오는 것을 관찰할 수 있음

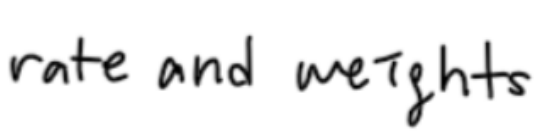


다른 글씨체 이미지를 사용하면 다른 결과가 나오는 것을 관찰할 수 있음

프로젝트 결과

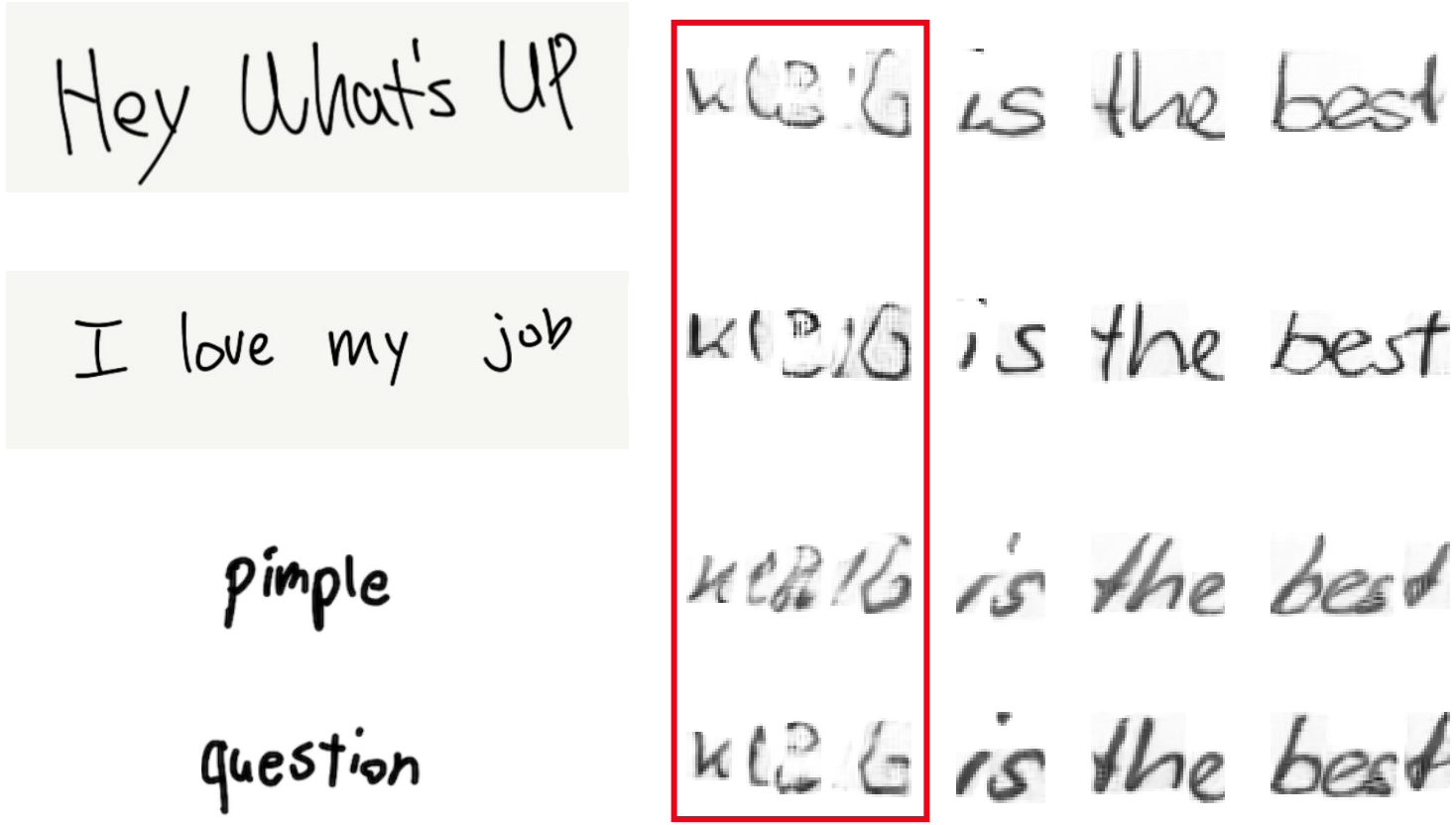
Example 1: Hello World





프로젝트 결과

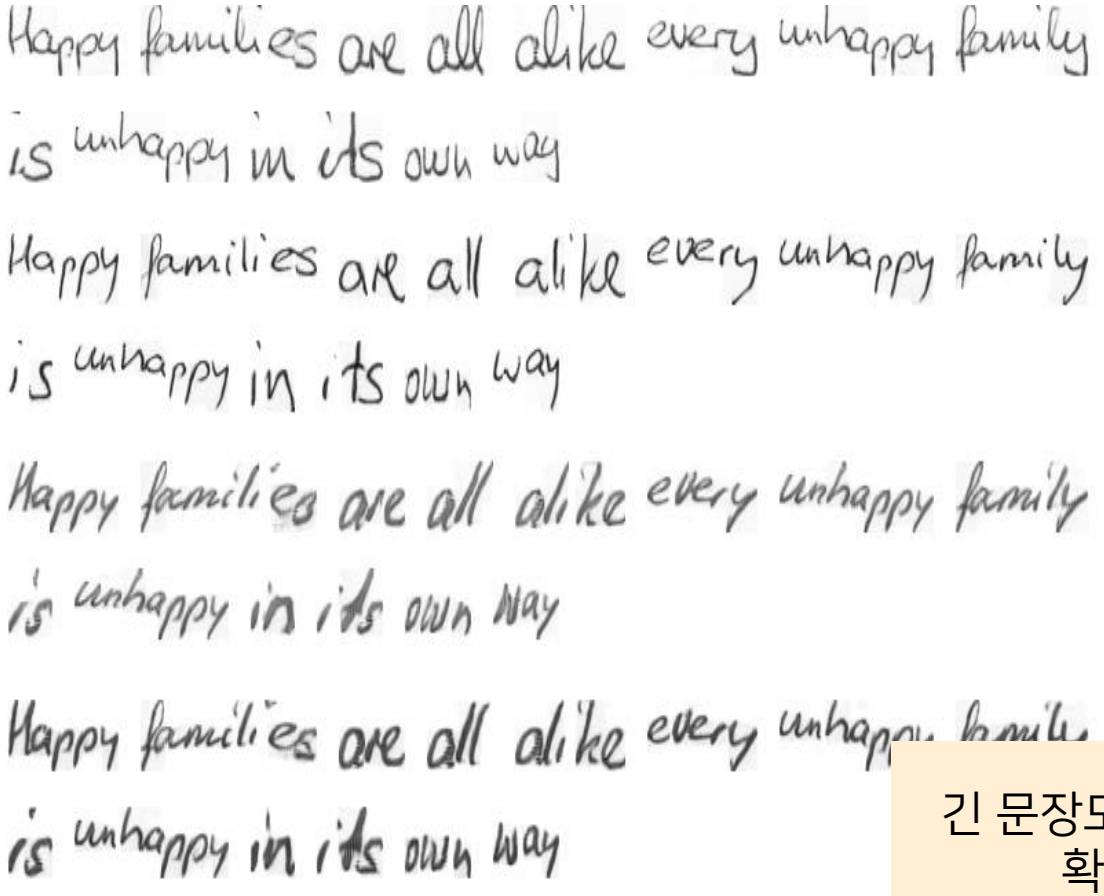
Example 2: KUBIG is the best



다만 일부 알파벳의 경우는
제대로 생성되지 않는 것을
확인할 수 있음

프로젝트 결과

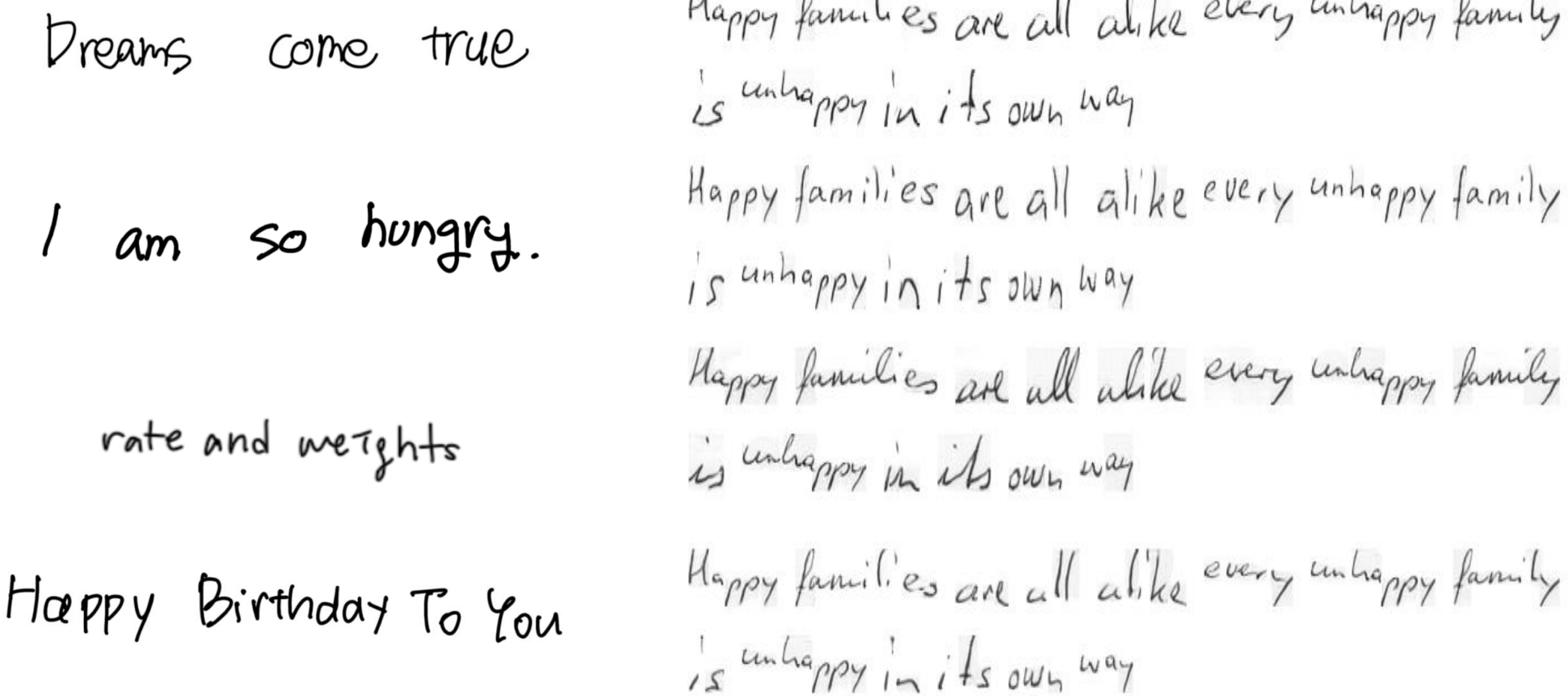
Example 3: Happy families are all alike every unhappy family is unhappy in its own way



긴 문장도 잘 생성하는 것을
확인할 수 있음

프로젝트 결과

Example 3: Happy families are all alike every unhappy family is unhappy in its own way



한계점 및 발전점

1. 한글 적용

- HWT 모델은 영어 OCR을 사용하여 이미지에서 영어 알파벳을 파악함.
- 한글 OCR을 적용할 시간이 부족하여 한글 적용까지는 성공하지 못했음.
- 한글 OCR을 적용하여 한글 손글씨 이미지에서 글자를 파악하도록 훈련시킨다면 한글 손글씨 이미지를 사용한 변환도 가능할 것으로 봄.

2. 새로운 스타일 적용

- 팀원의 글씨체 이미지만 가져와 변환해봄. 더 다양한 custom 글씨체 변환도 도전할 수 있었는데 그러지 못함.
- 많은 custom 글씨체 이미지를 수집하거나, HWT 모델에 이미지를 input하는 과정을 단순화하여 누구나 자신의 글씨체로 손글씨 이미지를 생성할 수 있도록 만들 수 있을 것으로 봄.

감사합니다.