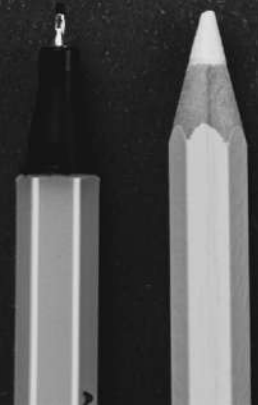
인공지능 폰트 스타일러

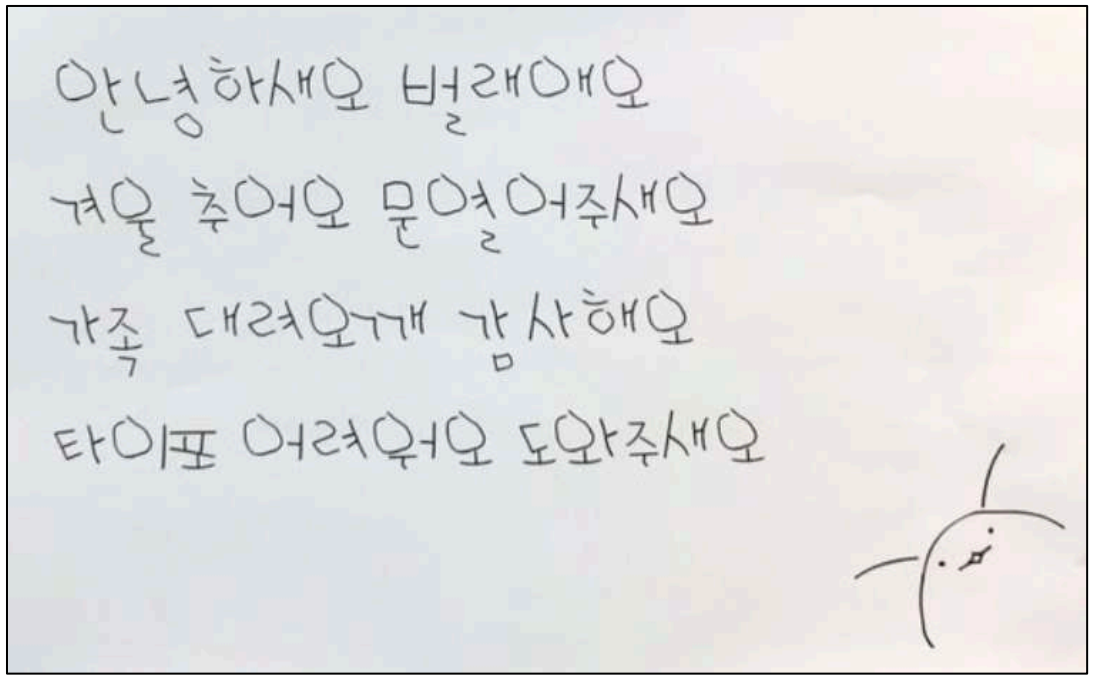
팀 7 to 12

7기 이수빈
10기 신훈철
11기 이영전
12기 김탁영

목차



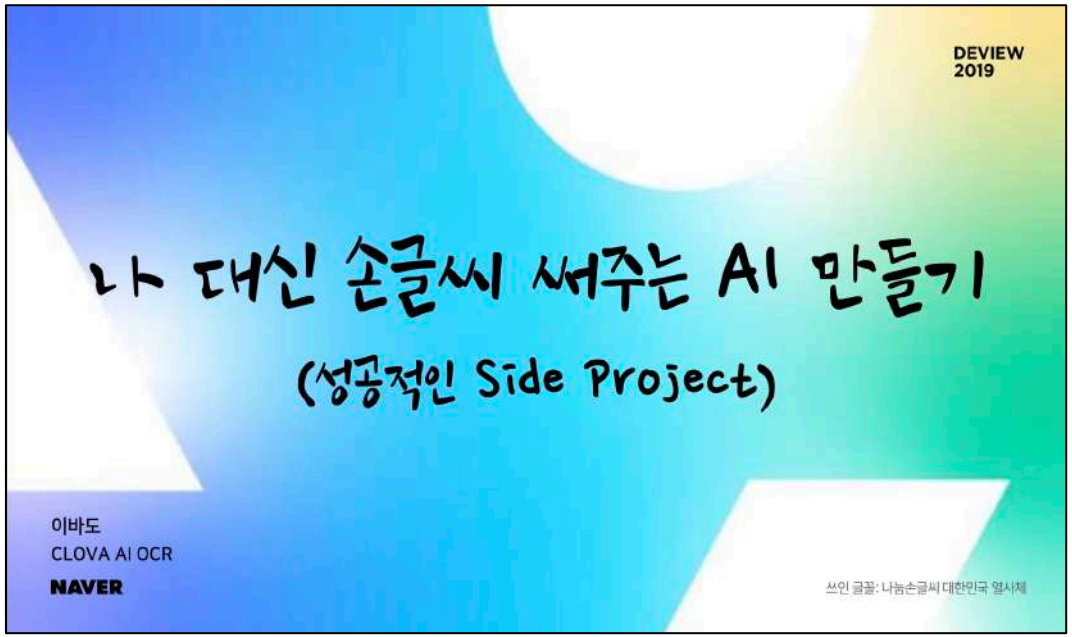
폰트의 힘



벌레가 귀여워 보이는 현상

관련 프로젝트

DEVIEW 2019



NAVER Clova 이바도님

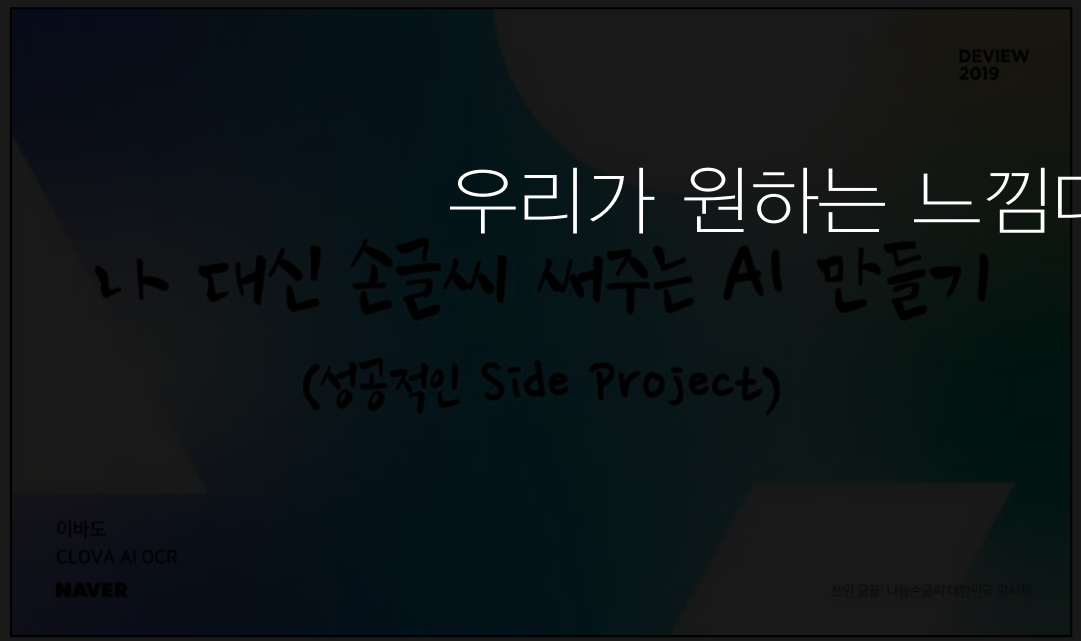
SINCE 2016
데이터야 놀자
SAT, OCT-19, 2019 10:00
datayanolja.github.io

정민정님

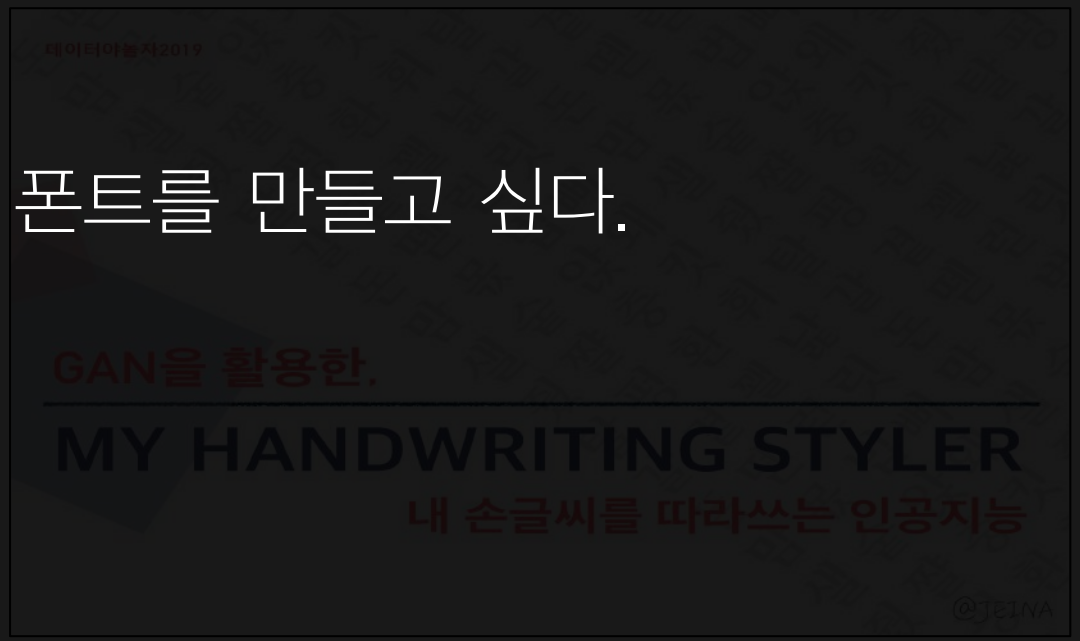
관련 프로젝트

DEVIEW 2019

SINCE 2016
데이터야 놀자
SAT, OCT-19, 2019



NAVER Clova 이바도님



정민정님

우리가 원하는 느낌대로 폰트를 만들고 싶다.

프로젝트 구상

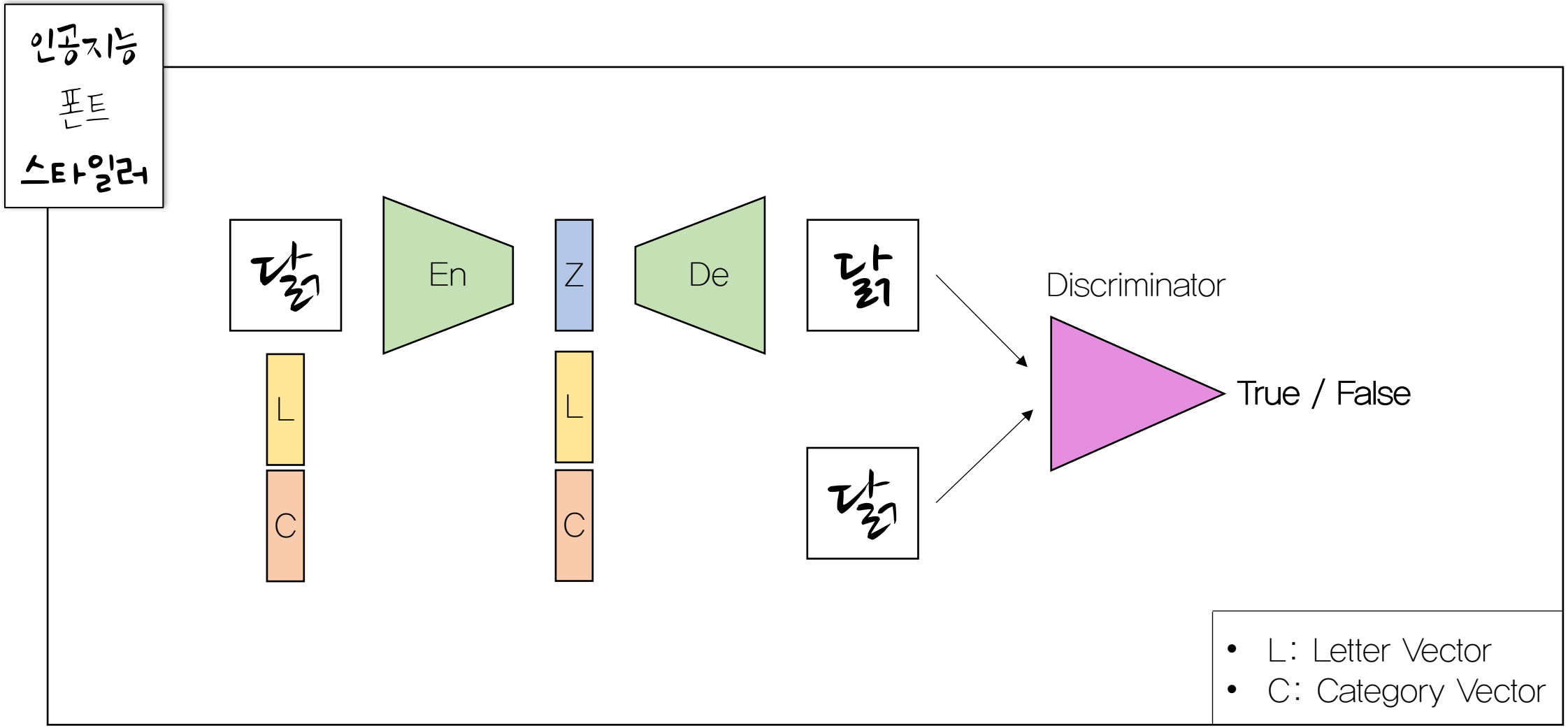
원하는 느낌을 입력하면
느낌에 맞는 폰트를 출력

인공지능
폰트
스타일러

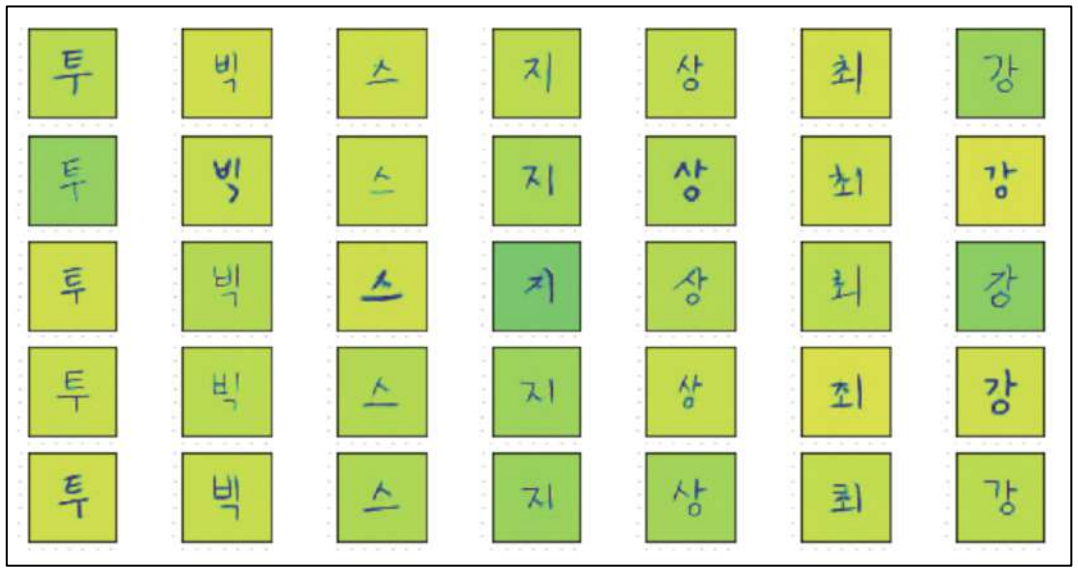
그냥 죽여줘..

원하는 느낌의 정도를 조절

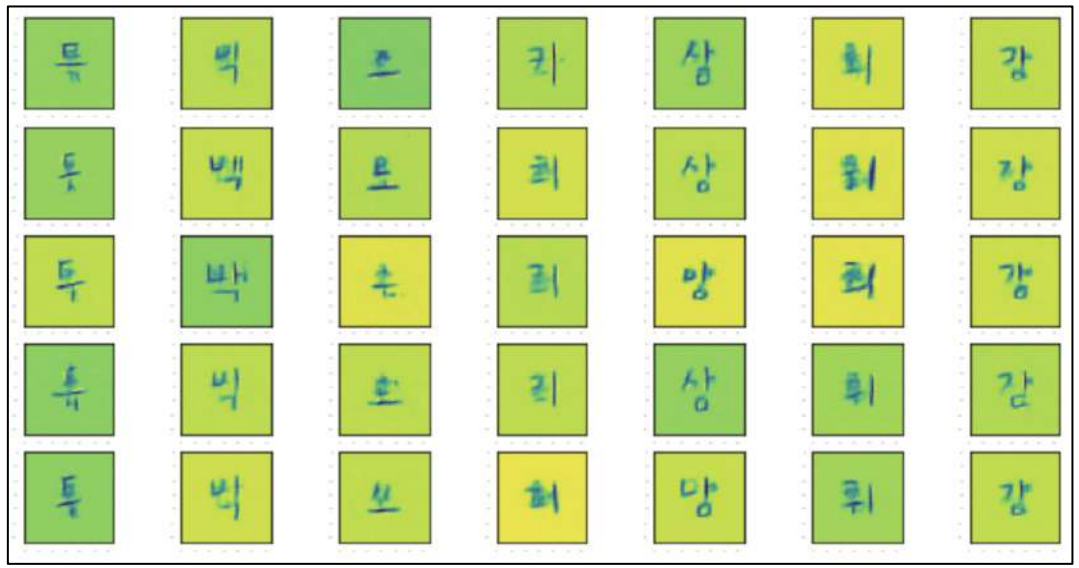
프로젝트 개요


인공지능
폰트
스타일러
달
En
Z
De
달
L
C
L
C
Discriminator
True / False
달
• L: Letter Vector
• C: Category Vector

프로젝트 개요



이상적인 결과



실제 결과

데이터

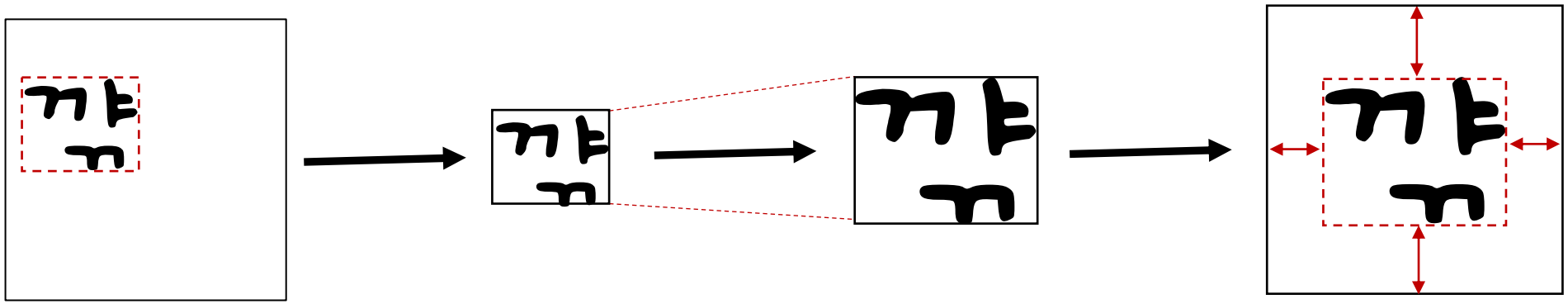
사용 데이터 – 폰트

네이버 한글날 손글씨 공모전
공개된 109개의 폰트 중 107개

사용 데이터 – 글자

가각간갇갈갉갊감갑값갓갔강갖갗같갚갛개객갠갤갬갭갯갰갱갸갹갼걀걋걍걔걘걜거걱건걷걸걺검겁것겄
겅겆겉겊겋게겐겔겜겝겟겠겡겨격겪견겯결겸겹겻겼경곁계곈곌곕곗고곡곤곧골곪곬곯곰곱곳공곶과곽관
괄괆괌괍괏광괘괜괠괩괬괭괴괵괸괼괌괩괫괭괴괵괸괼괌괩괫괭괴괵괸괼괌괩괫괭괴괵괸괼괌괩괫괭괴괵
귀귁귄귈귐귑귓규균귤그극근귿글긁금급긋긍긔기긱긴긷길긺김깁깃깅깆깊까깍깎깐깔깖깜깝깟깠깡깥깨
깩깬깰깸깹깻깼깽꺄꺅꺌꺼꺽꺾껀껄껌껍껏껐껑께껙껜껨껫껭껴껵껼껫껐껼꼐꼬꼭꼰꼲꼴꼼꼽꼿꽁꽃꽅꽈
꽉꽐꽜꽝꽤꽥꽹꾀꾄꾈꾐꾑꾕꾜꾸꾹꾼꿀꿇꿈꿉꿋꿍꿎꿔꿜꿨꿩꿰꿱꿴꿸뀀뀁뀄뀌뀐뀔뀜뀝뀨끄끅끈끊끌
끎끓끔끕끗끙끝끼끽낀낄낌낍낏깅나낙낚난낟날낡낢남납낫났낭낮낯낱낳내낵낸낼냄냅냇냈냉냐냑냔냘냠
냥너넉넋넌널넓넓넘넙넛넜넝넣네넥넨넬넴넵넷넸넹녀녁년녈념녑녔녕녘녜녠노녹논놀놂놈놉놋농높놓놔
놘놜놨뇌뇐뇔뇜뇝뇟뇨뇩뇬뇰뇹뇻뇽누눅눈눋눌눔눕눗눙눠눴눼뉘뉜뉠뉨뉩뉫뉴뉵뉼뉨뉩뉭느늑는늘늙늚늠
늡늣능늦늪늬늰늴니닉닌닐닒님닙닛닝닢다닥닦단닫달닭닮닯닳담답닷닸당닺닻닿대댁댄댈댐댑댓댔댕댜
더덕덖던덛덜덟덣덤덥덧덩덫덮데덱덴델뎀뎁뎃뎄뎅뎌뎐뎔뎠뎡뎨뎬도독돈돋돌돎돔돕돗동돛돝돠돤돨
돼됐되된될됨됩됫됴두둑둔둘둠둡둣둥둬뒀뒈뒝뒤뒨뒬뒵뒷뒹듀듄듈듐듑드득든듣들듦듬듭듯등듸디딕딘
딛딜딤딥딧딨딩딪따딱딴딸땀땁땃땄땅땋때땍땐땔땜땝땟땠땡떠떡떤떨떪떫떰떱떳떴떵떻떼떽떤떨뗌뗍뗏
뗐뗑뗘뗬또똑똔똘똥똬똴뙈뙤뙨뚜뚝뚠뚤뚫뚬뚱뛔뛰뛴뛸뜀뜁뜅뜌뜩뜬뜯뜰뜸뜹뜻띄띈띌띔띕띠띤띨띰띱
띳띵라락란랄람랍랏랐랑랒랖랗래랙랜랠램랩랫랬랭랴략랸럇량러럭런럴럼럽럿렀렁렇레렉렌렐렘렙렛렝
려력련렬렴렵렷렸령례롄롑롓로록론롤롬롭롯롱롸뢴뢍뢸룀룁룃룅료룐룔룝룟룡루룩룬룰룸룹룻룽
뤄뤘뤠뤼뤽뢴뢸뢺뤗뤙류륙륜률륨룁륫륭르륵른를름릅릇릉릊릍릎리릭린릴림립릿링마막만많맏말맑맒맘
맙맛망맞맡맣매맥맨맬맴맵맷맸맹맺먀먁먈먕머먹먼멀멂멈멉멋멍멎멓메멕멘멜멤멥멧멨멩며멱면멸몃몄
명몇몌모목몫몬몰몲몸몹못몽뫄뫈뫘뫙뫼묀묄묍묏묑묘묜묠묩묫무묵묶문묻물묽묾물뭅뭇뭉뭍뭏뭐뭔뭘뭡
뭣뭬뮈뮌뮐뮤뮨뮬뮴뮷므믄믈믐믓미믹민믿밀밂밈밉밋밌밍및밑바박밖밗반받발밝밟밤밥밧방밭배백밴
밸뱀뱁뱃뱄뱅뱉뱌뱍뱐뱝버벅번벋벌벎범법벗벙벚베벡벤벧벨벰벱벳벴벵벼벽변별볍볏볐병볕볘볜보복볶
본볼봄봅봇봉봐봔봤봬뵀뵈뵉뵌뵐뵘뵙뵤뵨부북분붇불붉붊붐붑붓붕붙붚붜붤붰붸뷔뷕뷘뷜뷩뷰뷴뷸뷺븃
븅브븍븐블븜븝븟비빅빈빌빎빔빕빗빙빚빛빠빡빤빨빪빰빱빳빴빵빻빼빽뺀뺄뺌뺍뺏뺐뺑뺘뺙뺨뺴뻐뻑뻔뻗
뻘뻠뻣뻤뻥뻬뼁뼈뼉뼘뼙뼛뼜뼝뽀뽁뽄뽈뽐뽑뽕뾔뾰뿅뿌뿍뿐뿔뿜뿟뿡쀼쀽쁘쁜쁠쁨쁩삐삑삔삘삠삡삣삥

데이터 전처리



폰트의 종류와 상관없이 균일한 형태의 데이터 확보

임베딩

임베딩 네트워크: 여러 가지 시도

Doc2Vec

- ✓ 네이버 손글씨 폰트 소개말을 20차원의 벡터로 변환
- ✓ 각 소개말을 임베딩하여 폰트의 설명값으로 사용
- ✓ 소개말이 시적인 분위기를 내기 때문에 서사적인 노래 가사 3만 곡의 데이터를 추가하여 학습

Latent Vector

- ✓ Convolutional AutoEncoder로 학습한 128차원의 벡터 사용
- ✓ 임베딩 벡터를 만들기 위한 별도의 Conv AE 사용
- ✓ ‘닭’, ‘카테고리나뉘어랏’ 등의 단어로 실험하였지만 유의미한 결과를 보이지 않음

Triplet Network

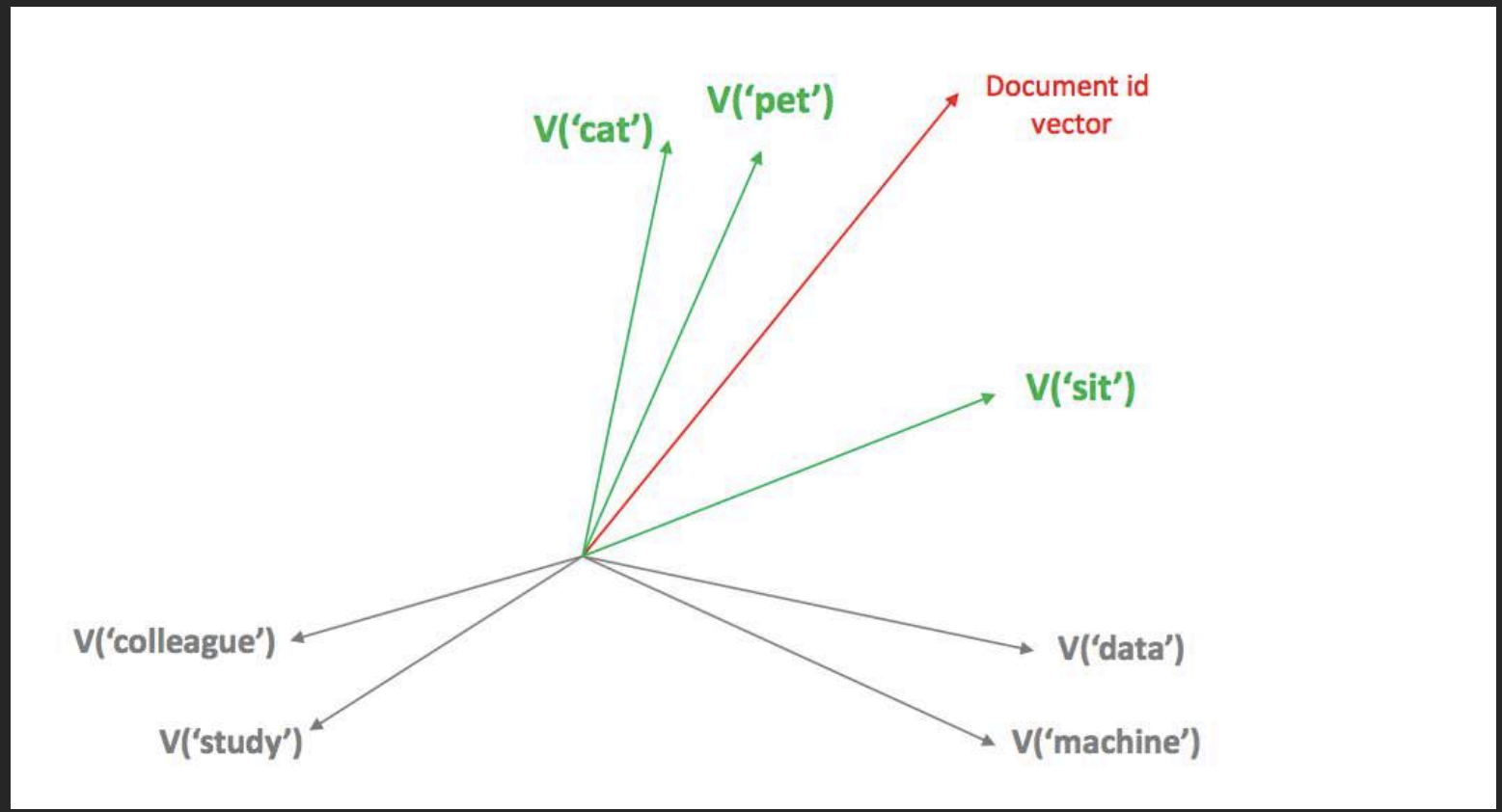
- ✓ 1) 같은 폰트끼리 뭉치도록
  2) 같은 글자끼리 뭉치도록
- ✓ 같은 기준끼리 뭉치도록 학습된 결과를 t-SNE로 시각화
- ✓ 두 경우 모두 잘 뭉치는 현상을 확인하였음

임베딩 네트워크 – Doc2Vec

Doc2Vec

- ✓ 네이버 손글씨 폰트 소개말을 20차원의 벡터로 변환
- ✓ 각 소개말을 임베딩하여 폰트의 설명값으로 사용
- ✓ 소개말이 시적인 분위기를 내기 때문에 서사적인 노래 가사 3만 곡의 데이터를 추가하여 학습

https://lovit.github.io/nlp/representation/2018/03/26/word_doc_embedding/

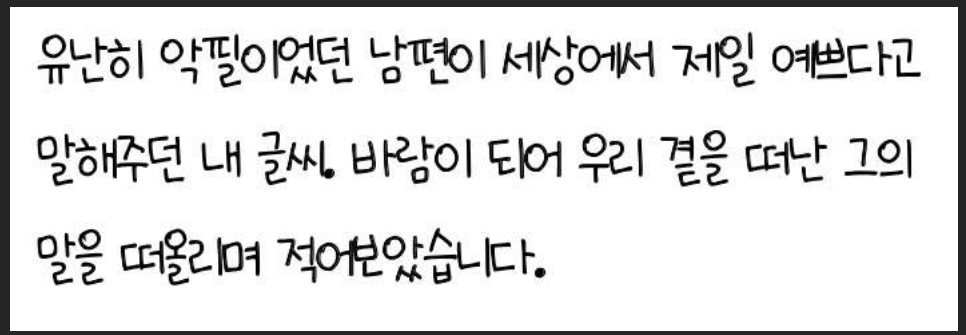


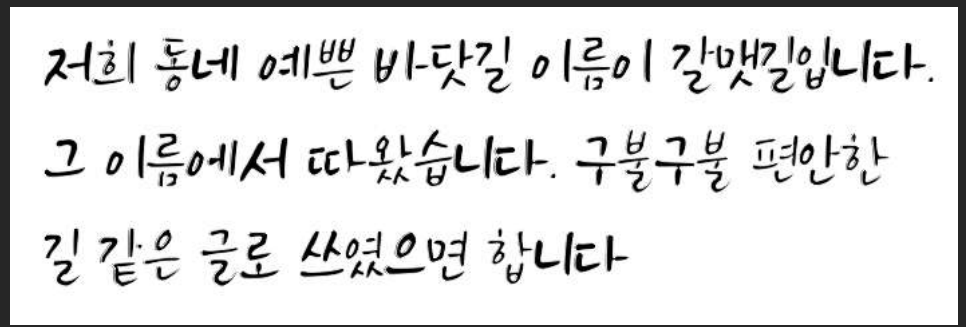
Document ID와 각 문서(혹은 문장)에 등장하였던 단어 좌표의 평균으로 document vector를 구함

임베딩 네트워크 – Doc2Vec

Doc2Vec

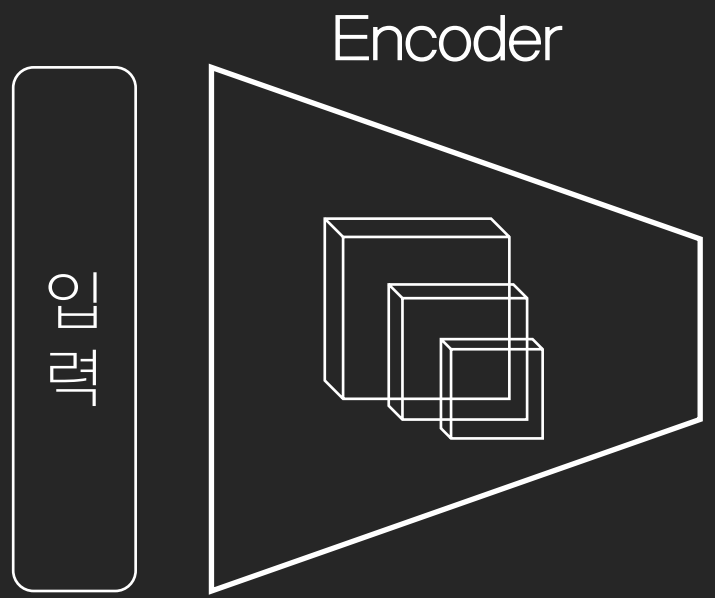
- ✓ 네이버 손글씨 폰트 소개말을 20차원의 벡터로 변환
- ✓ 각 소개말을 임베딩하여 폰트의 설명값으로 사용
- ✓ 소개말이 시적인 분위기를 내기 때문에 서사적인 노래 가사 3만 곡의 데이터를 추가하여 학습





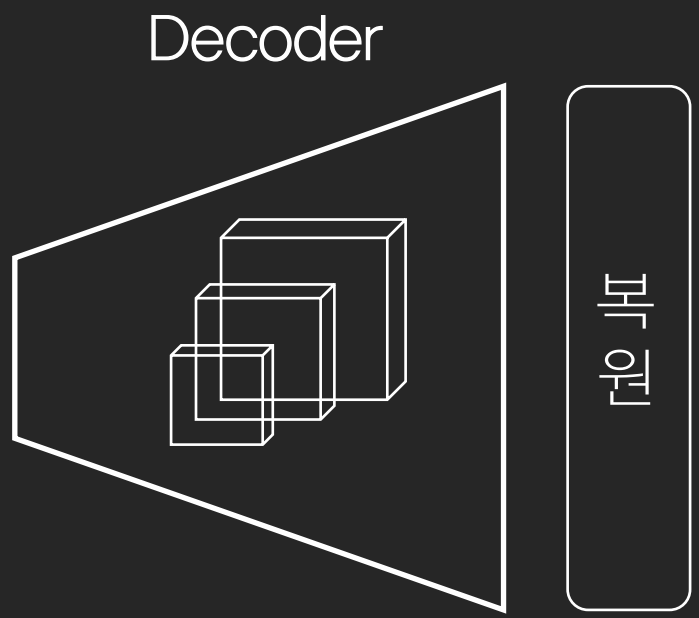
+ 시적인 노래 가사 3만 곡

임베딩 네트워크 – Latent Vector

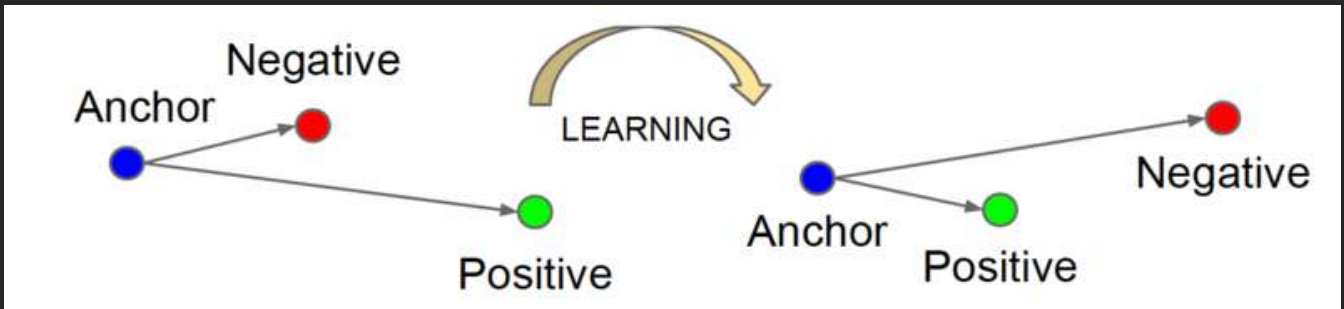


Latent Vector

- ✓ Convolutional AutoEncoder로 학습한 128차원의 벡터 사용
- ✓ 임베딩 벡터를 만들기 위한 별도의 Conv AE 사용
- ✓ ‘닭’, ‘카테고리나뉘어랏’ 등의 단어로 실험하였지만 유의미한 결과를 보이지 않음



임베딩 네트워크 – Triplet Network



3개의 데이터 묶음

1. Anchor : 기준 데이터
2. Positive : Anchor와 같은 클래스
3. Negative : Anchor와 다른 클래스

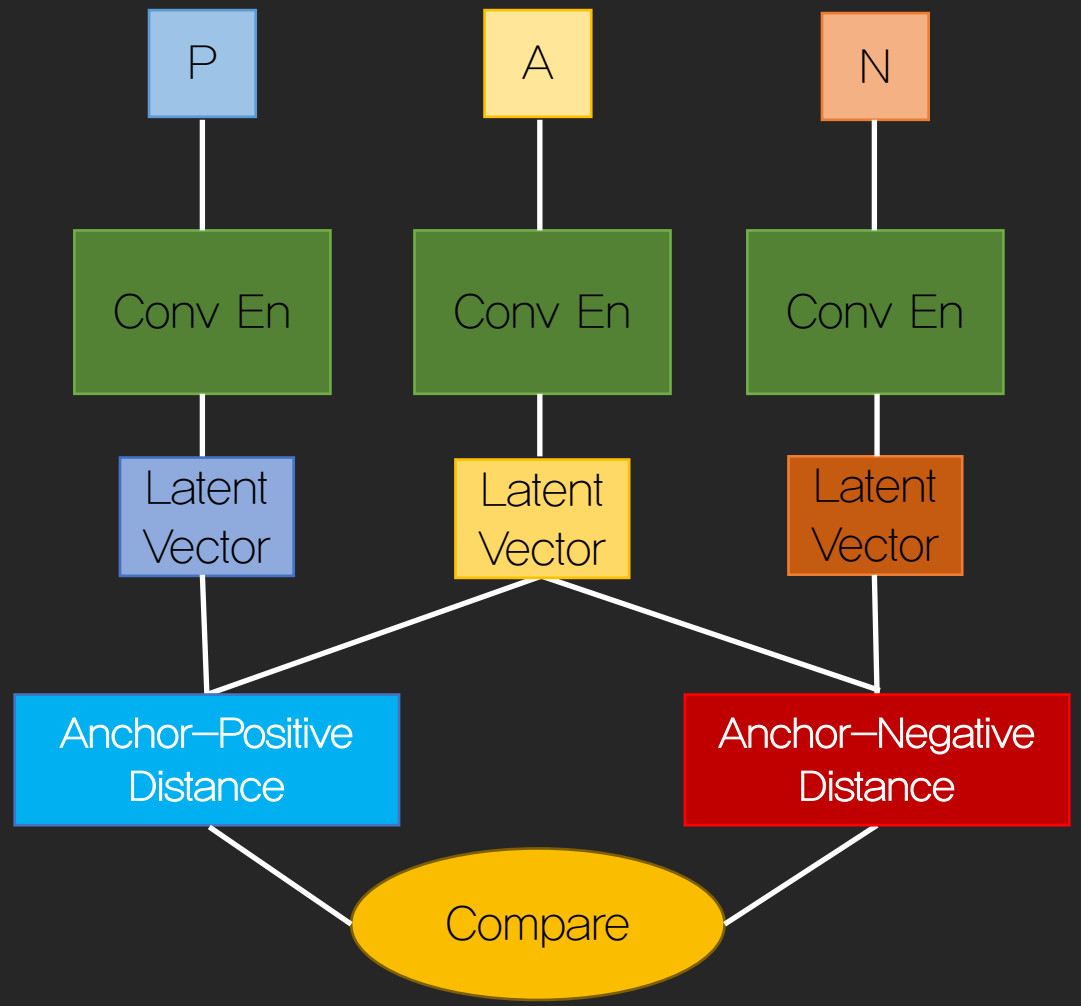
Positive 관계의 데이터는 가깝게,
Negative 관계의 데이터는 멀게 학습

Euclidean Distance + L2 Norm & Margin

Triplet Network

- ✓ 1) 같은 폰트끼리 뭉치도록
  2) 같은 글자끼리 뭉치도록
- ✓ 같은 기준끼리 뭉치도록 학습된 결과를 t-SNE로 시각화
- ✓ 두 경우 모두 잘 뭉치는 현상을 확인하였음

임베딩 네트워크 – Triplet Network

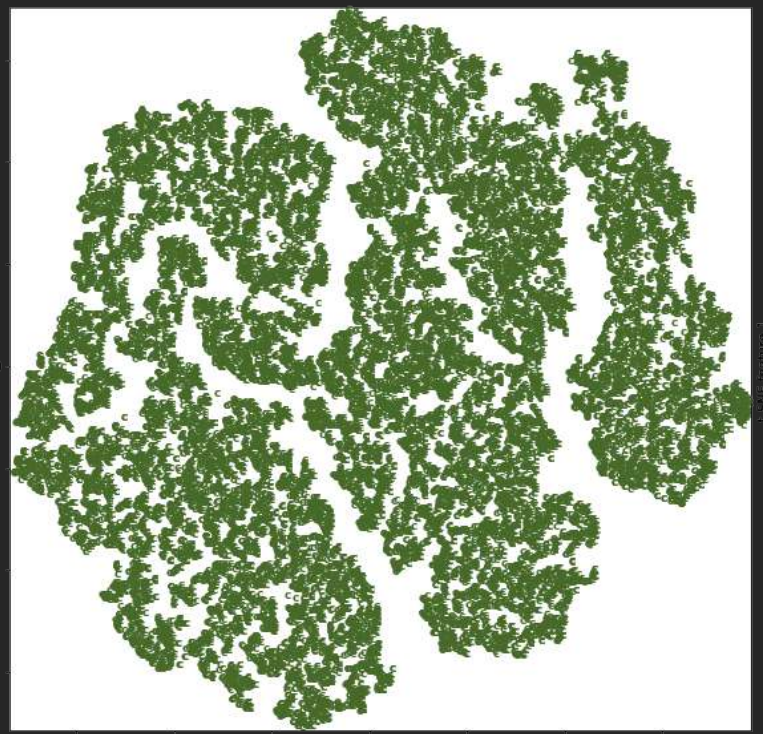


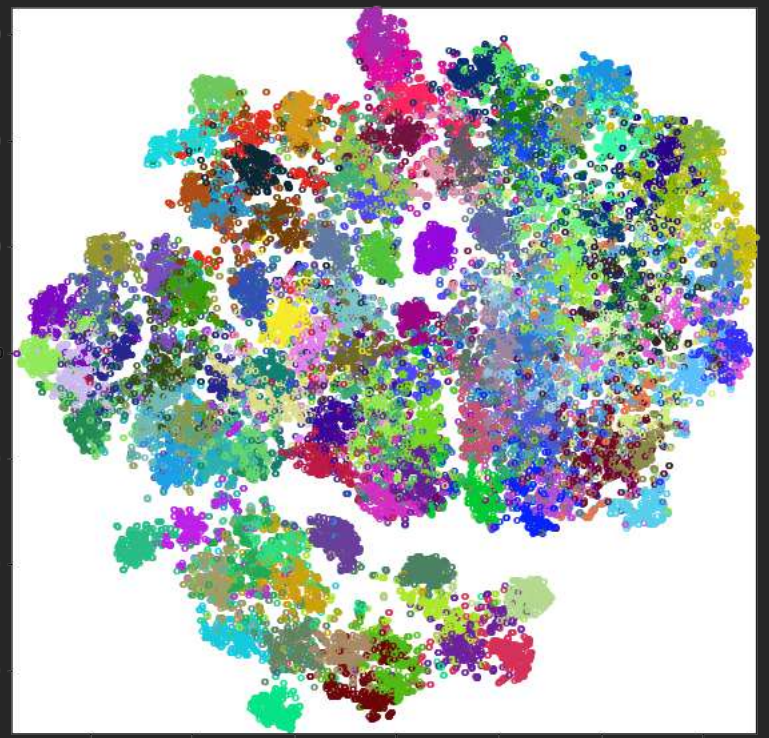
Triplet Network

- ✓ 1) 같은 폰트끼리 뭉치도록
  2) 같은 글자끼리 뭉치도록
- ✓ 같은 기준끼리 뭉치도록 학습된 결과를 t–SNE로 시각화
- ✓ 두 경우 모두 잘 뭉치는 현상을 확인하였음

임베딩 네트워크 – Triplet Network

[CASE 1] 같은 폰트끼리 뭉치는가? – 시각화





어느 정도 군집이 형성되고, 같은 폰트 클래스끼리 잘 뭉치는 것을 확인

Triplet Network

- ✓ 1) 같은 폰트끼리 뭉치도록
  2) 같은 글자끼리 뭉치도록
- ✓ 같은 기준끼리 뭉치도록 학습된 결과를 t-SNE로 시각화
- ✓ 두 경우 모두 잘 뭉치는 현상을 확인하였음

임베딩 네트워크 – Triplet Network

[CASE 1] 같은 폰트끼리 뭉치는가? – 시각화

0: 시크한 날씬이

1: 삐진 코흘리개

2: 정성스러운 범생이

Triplet Network

- ✓ 1) 같은 폰트끼리 뭉치도록
  2) 같은 글자끼리 뭉치도록
- ✓ 같은 기준끼리 뭉치도록 학습된 결과를 t-SNE로 시각화
- ✓ 두 경우 모두 잘 뭉치는 현상을 확인하였음

임베딩 네트워크 – Triplet Network

[CASE 1] 같은 폰트끼리 뭉치는가? – 시각화

3: 과감한 완벽주의자

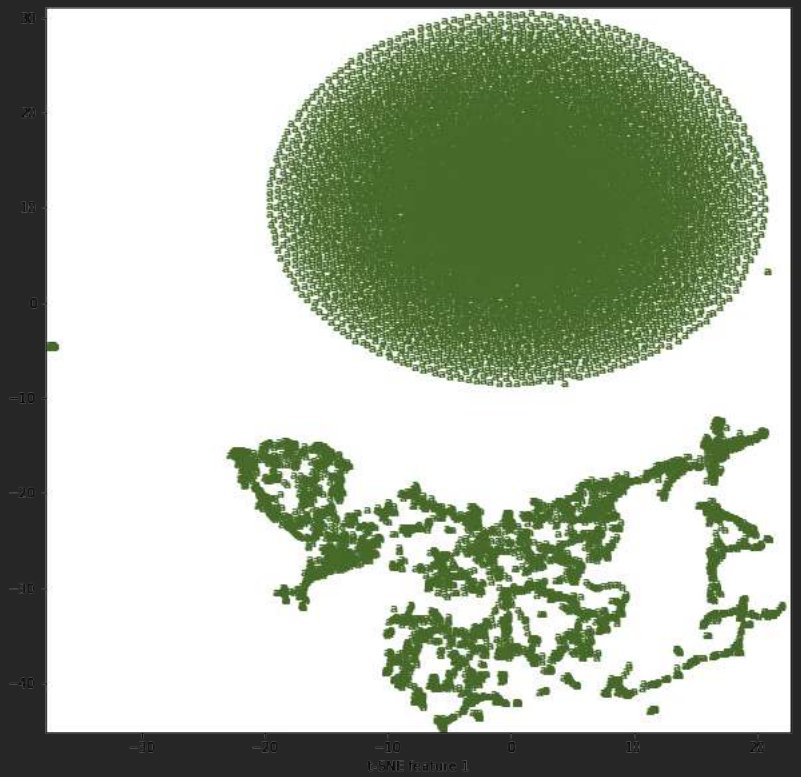
4: 여유로운 베짱이

Triplet Network

- ✓ 1) 같은 폰트끼리 뭉치도록
  2) 같은 글자끼리 뭉치도록
- ✓ 같은 기준끼리 뭉치도록 학습된 결과를 t-SNE로 시각화
- ✓ 두 경우 모두 잘 뭉치는 현상을 확인하였음

임베딩 네트워크 – Triplet Network

[CASE 2] 같은 글자끼리 뭉치는가? – 시각화



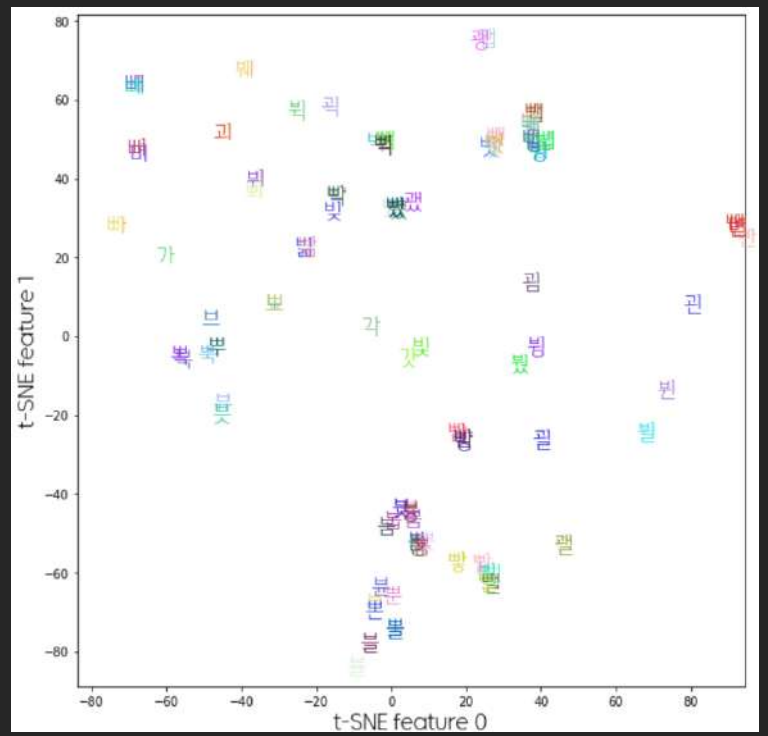
========== 쉬어가는 시간입니다 ==========

Triplet Network

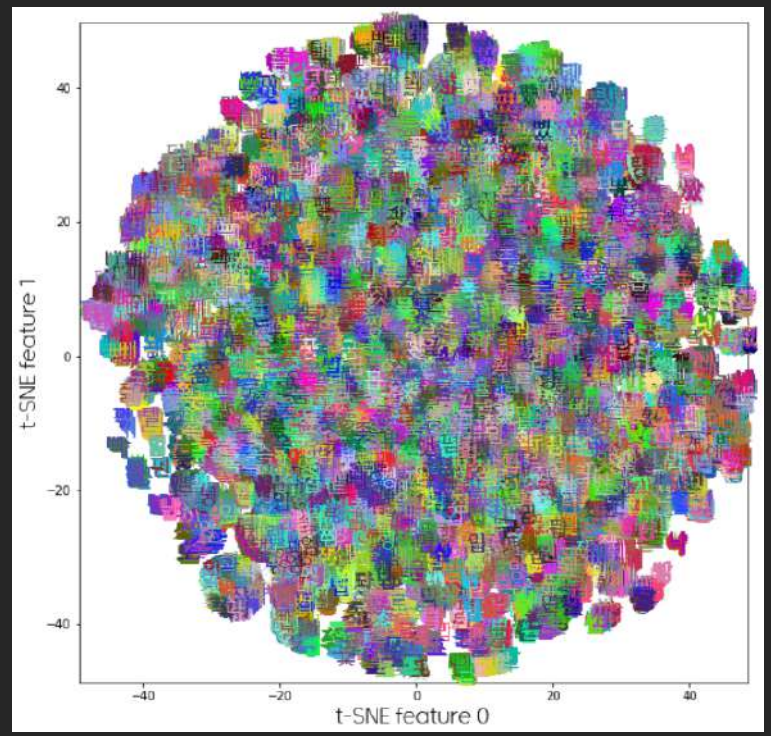
- ✓ 1) 같은 폰트끼리 뭉치도록
  2) 같은 글자끼리 뭉치도록
- ✓ 같은 기준끼리 뭉치도록 학습된 결과를 t-SNE로 시각화
- ✓ 두 경우 모두 잘 뭉치는 현상을 확인하였음

임베딩 네트워크 – Triplet Network

[CASE 2] 같은 글자끼리 뭉치는가? – 시각화



100자
괵–뷕 / 뷴–쀤 / 각–갓



2350자
같은 글자 = 같은 색

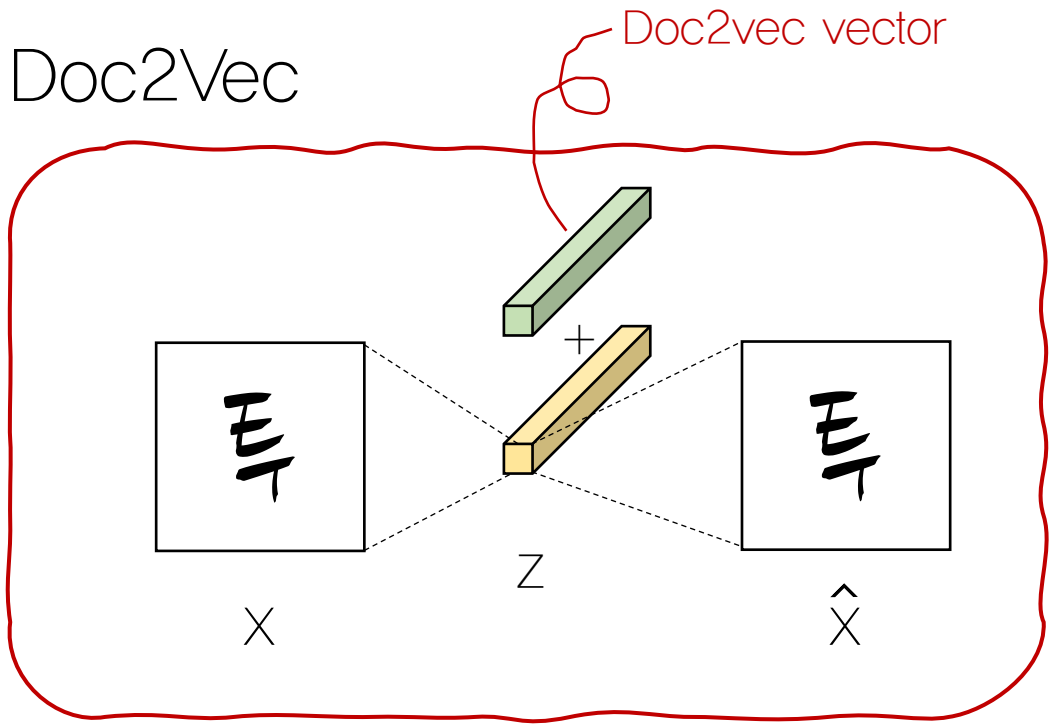
Triplet Network

- ✓ 1) 같은 폰트끼리 뭉치도록
  2) 같은 글자끼리 뭉치도록
- ✓ 같은 기준끼리 뭉치도록 학습된 결과를 t–SNE로 시각화
- ✓ 두 경우 모두 잘 뭉치는 현상을 확인하였음

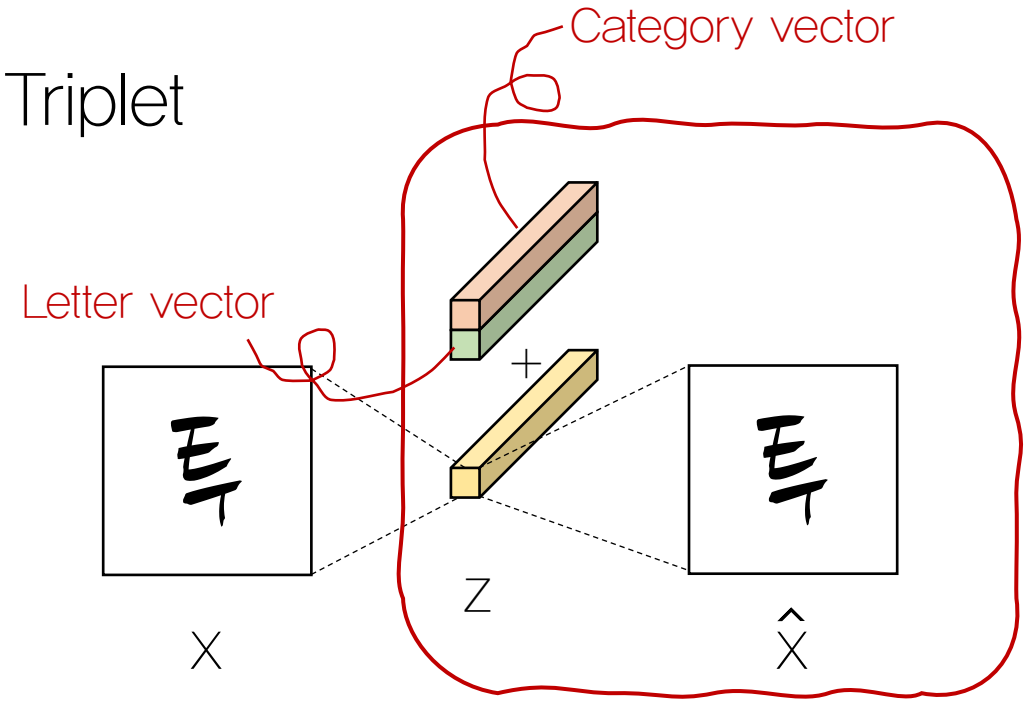
모델 구조

모델 아키텍쳐

Doc2Vec



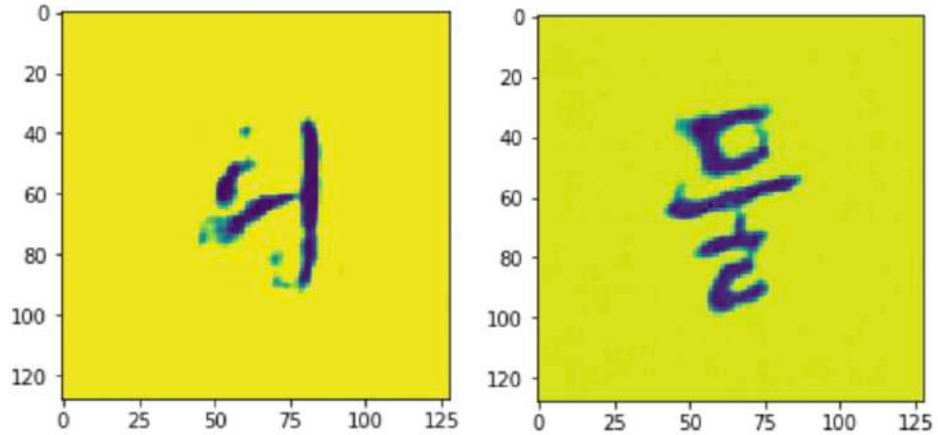
Triplet



결론 및 의의

생성 결과물

글자 생성 결과물 – Doc2Vec

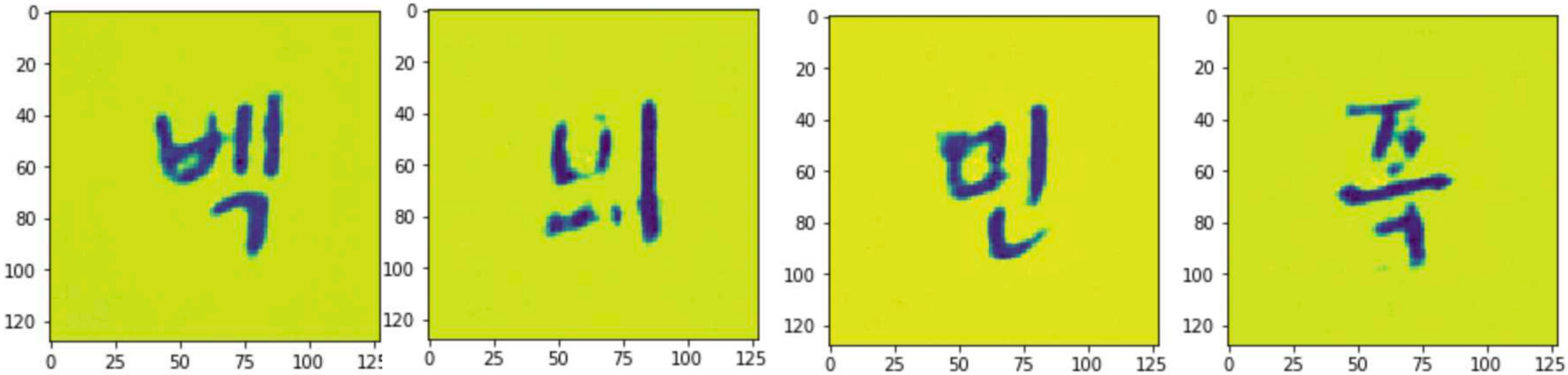


‘서울’ 이라는 글자에 ‘함께하는 복지도시’ 라는 느낌을 담은 output

Word Feel

생성 결과물

글자 생성 결과물 – Doc2Vec

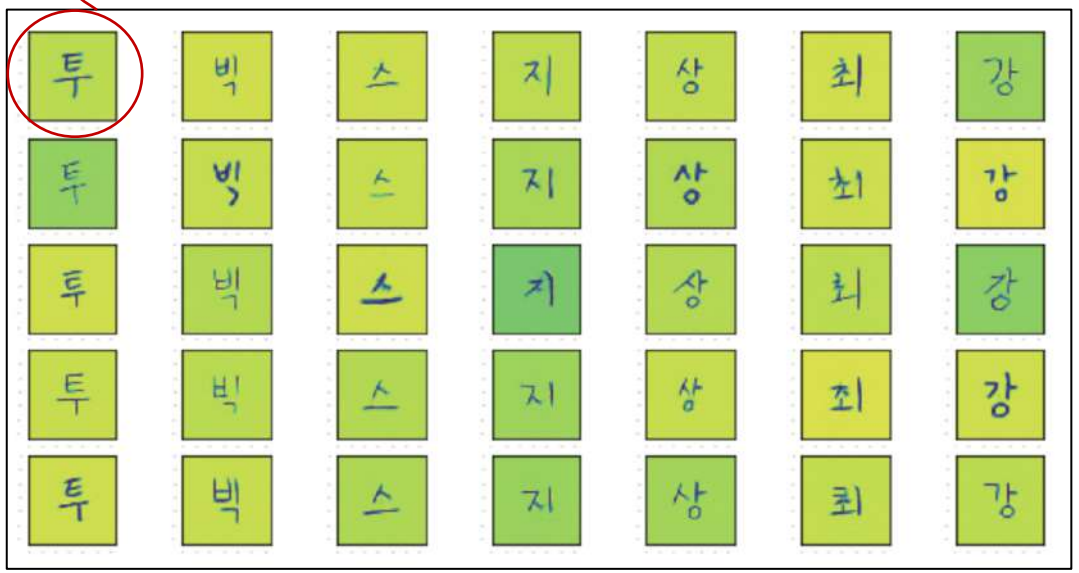


Word: ‘백의민족’
Feel: '우리가 어떤 민족입니까’

생성 결과물

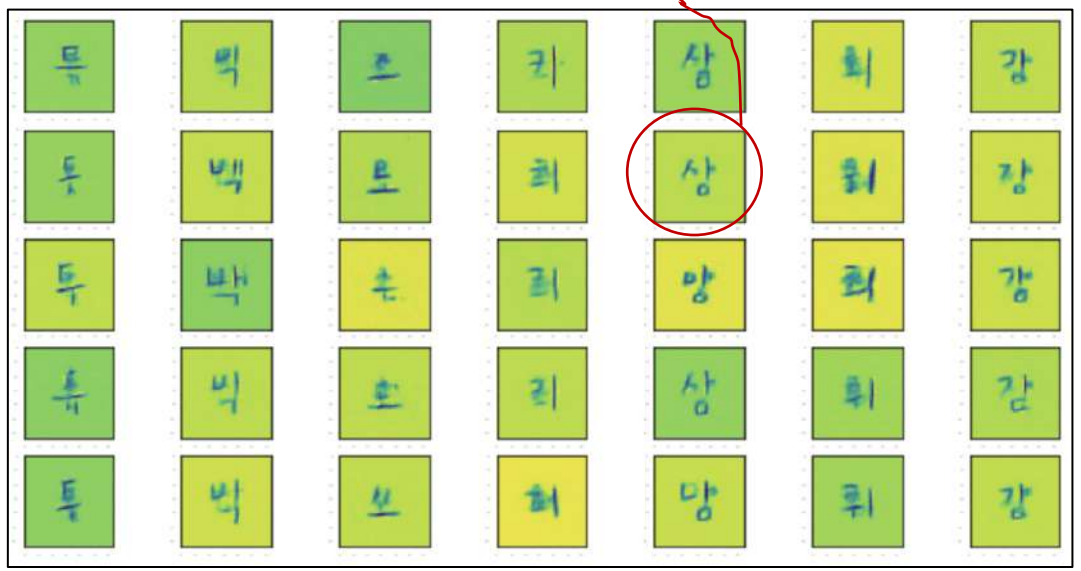
인코더를 통과한 z벡터 + letter vector + category vector

글자 생성 결과물 – Triplet



이상적인 결과

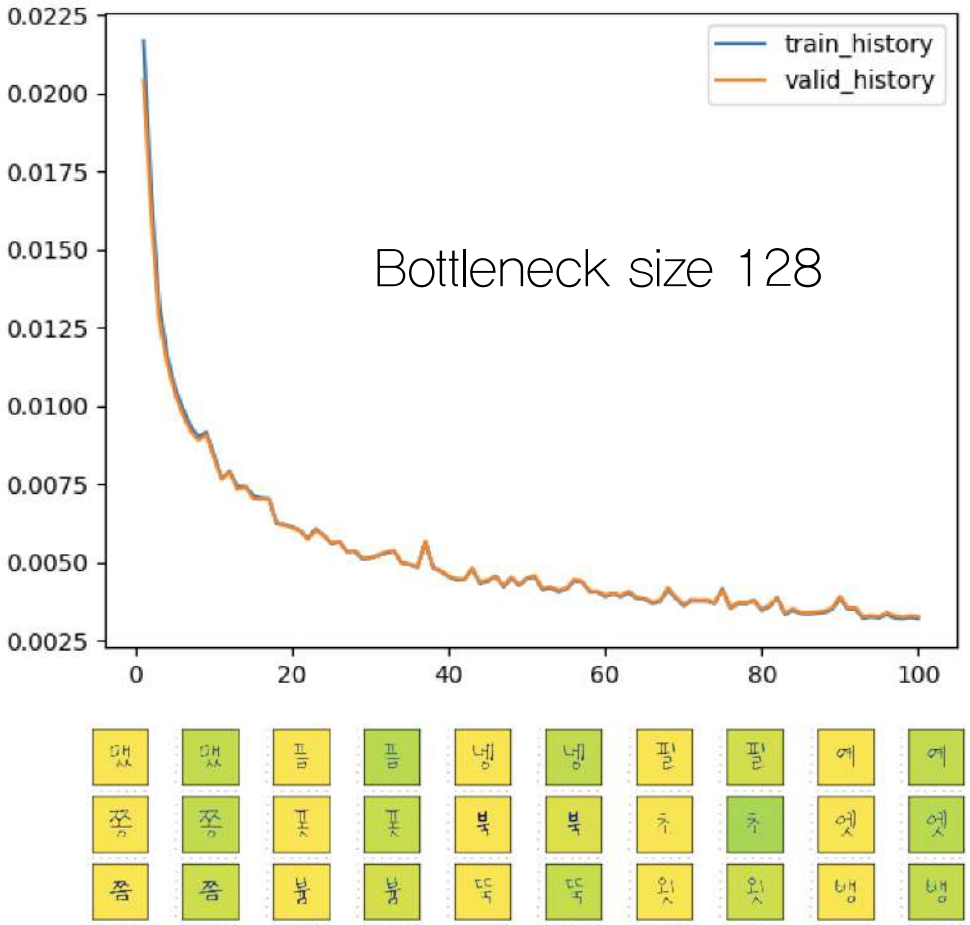
Gaussian Random Vector + letter vector + category vector

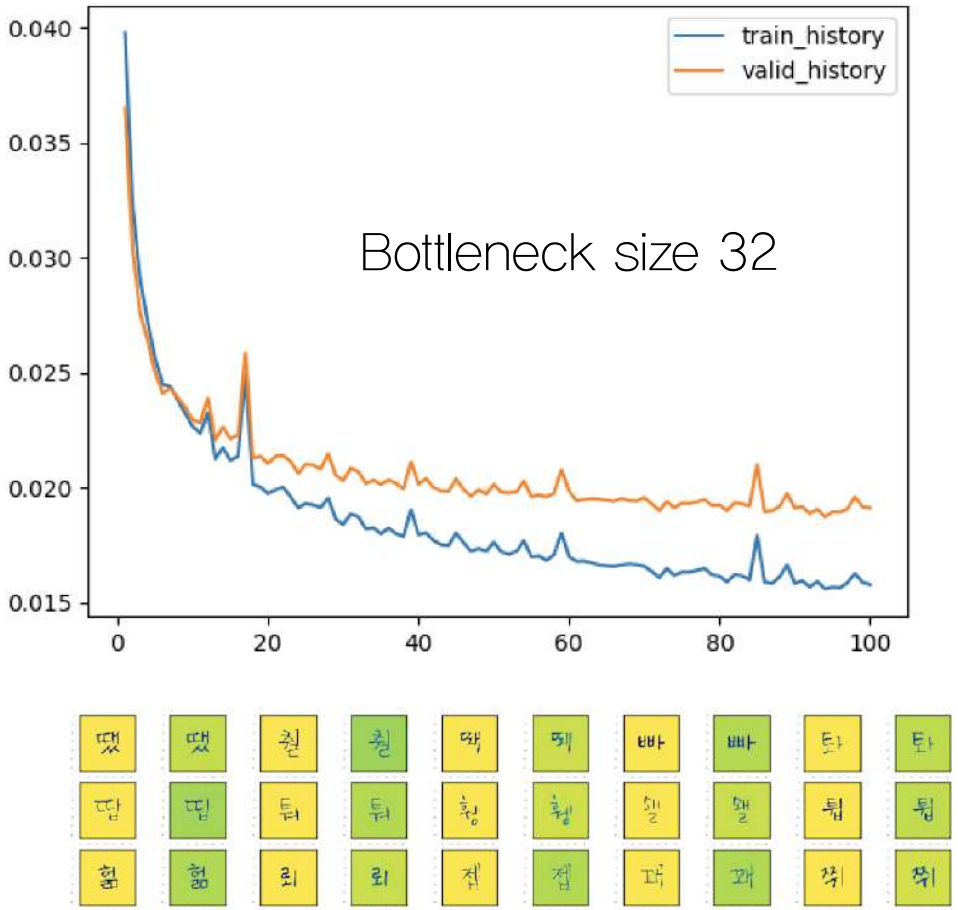


실제 결과

이상적인 결과물이 나오지 않은 이유?

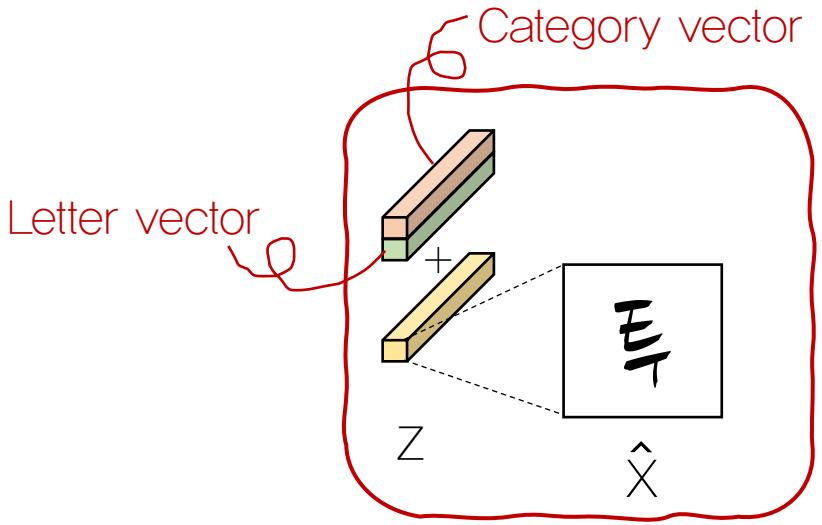
0. Baseline은 좋은 성능을 보임 (MSE ~0.02)

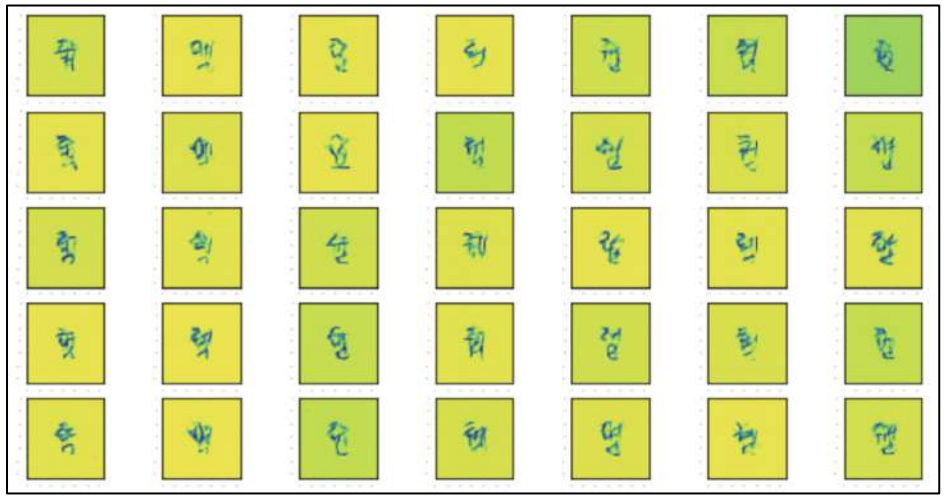




이상적인 결과물이 나오지 않은 이유?

1. Decoder만 사용하여 이미지를 생성하였을 때 매우 낮은 성능

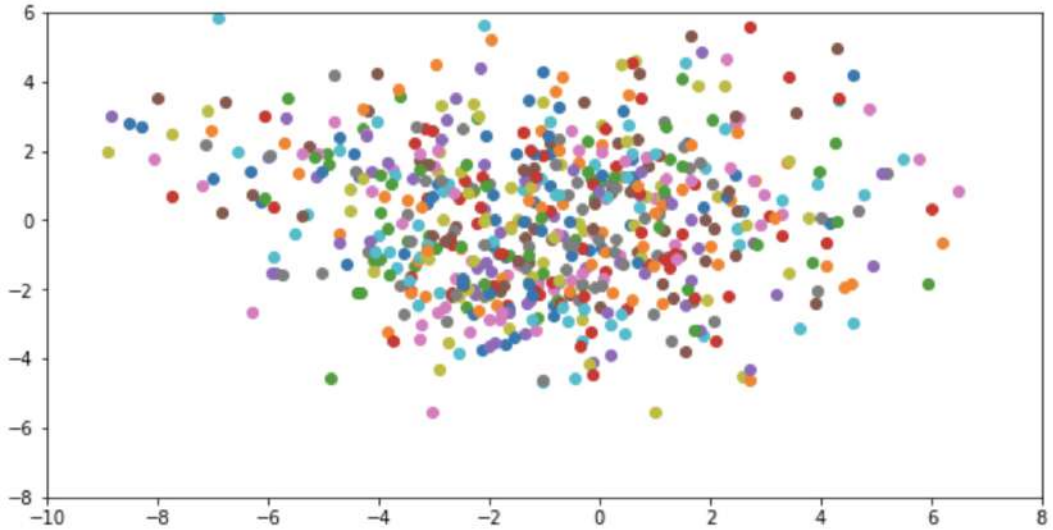




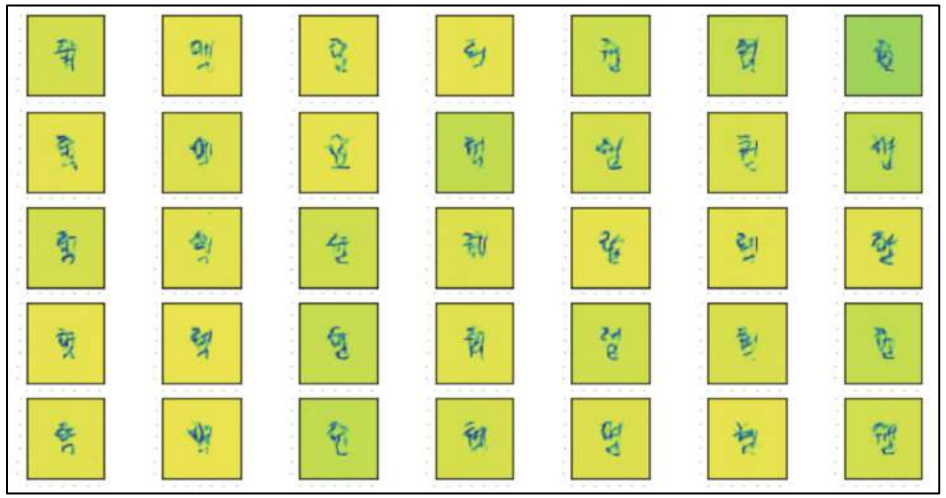
Bottleneck size=32
(with gaussian random vector for z)

이상적인 결과물이 나오지 않은 이유?

1. Decoder만 사용하여 이미지를 생성하였을 때 매우 낮은 성능



* btl = 32의 z값 분포



Bottleneck size=32
(with gaussian random vector for z)

Z값이 꽤 큰 범위에서 퍼져 있다.
Embedding Vector에 비해 값이 훨씬 크다.

⇒ Z의 값(영향)이 너무 크다.

이상적인 결과물이 나오지 않은 이유?

2. Z의 영향을 줄이기 위해 KLD Loss 도입

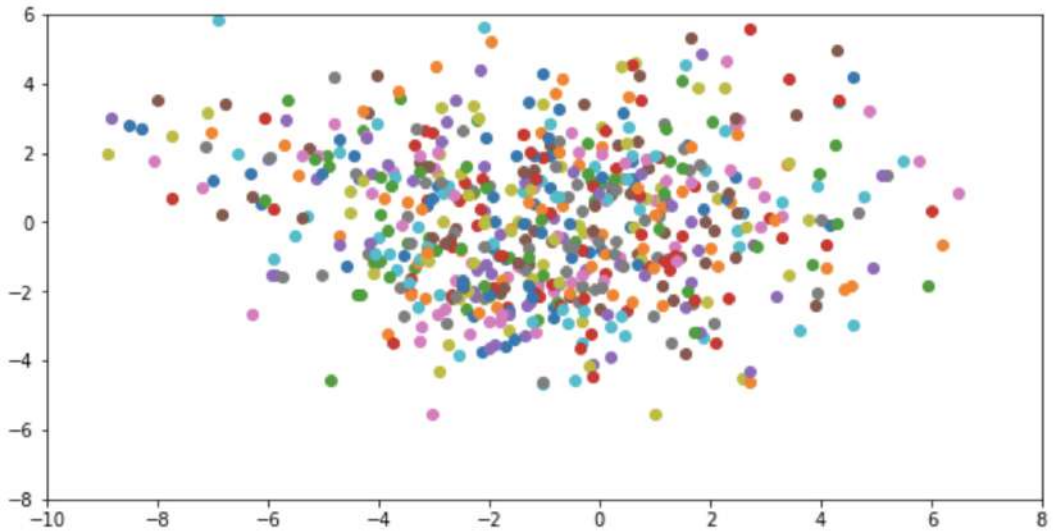
```
ITERATION - loss: 0.07942 mse: 0.07941 kld: 0.00001:  25%| | 3160/12632 [00:44<0
ITERATION - loss: 0.08620 mse: 0.08619 kld: 0.00001:  25%| | 3170/12632 [00:45<0
ITERATION - loss: 0.06758 mse: 0.06757 kld: 0.00001:  25%| | 3180/12632 [00:45<0
ITERATION - loss: 0.07880 mse: 0.07879 kld: 0.00001:  25%| | 3190/12632 [00:45<0
ITERATION - loss: 0.07573 mse: 0.07572 kld: 0.00001:  25%| | 3200/12632 [00:45<0
ITERATION - loss: 0.07974 mse: 0.07973 kld: 0.00001:  25%| | 3210/12632 [00:45<0
ITERATION - loss: 0.07468 mse: 0.07467 kld: 0.00001:  25%| | 3220/12632 [00:45<0
ITERATION - loss: 0.08012 mse: 0.08011 kld: 0.00001:  26%| | 3230/12632 [00:45<0
ITERATION - loss: 0.07342 mse: 0.07342 kld: 0.00001:  26%| | 3240/12632 [00:46<0
ITERATION - loss: 0.08204 mse: 0.08204 kld: 0.00000:  26%| | 3250/12632 [00:46<0
ITERATION - loss: 0.08726 mse: 0.08725 kld: 0.00001:  26%| | 3260/12632 [00:46<0
ITERATION - loss: 0.06150 mse: 0.06149 kld: 0.00000:  26%| | 3270/12632 [00:46<0
ITERATION - loss: 0.08704 mse: 0.08704 kld: 0.00000:  26%| | 3280/12632 [00:46<0
ITERATION - loss: 0.09302 mse: 0.09302 kld: 0.00000:  26%| | 3290/12632 [00:46<0
```

Z 공간은 바로 수렴
그러나 MSE가 줄어들지 않음

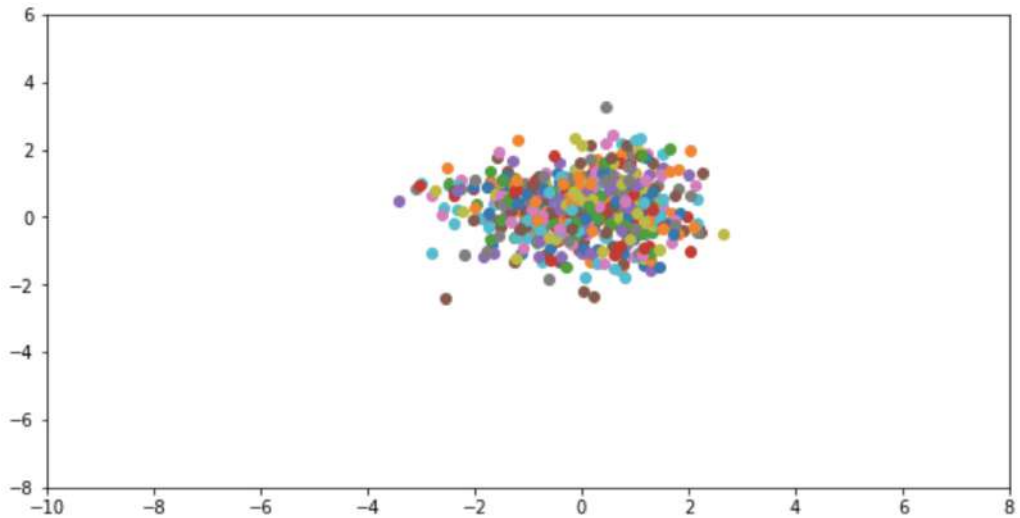
KLD term의 역할을 줄이기 위해
Beta term (penalty)을 도입
(10e–5)

이상적인 결과물이 나오지 않은 이유?

2. Z의 영향을 줄이기 위해 KLD Loss 도입



* btl = 32의 z값 분포

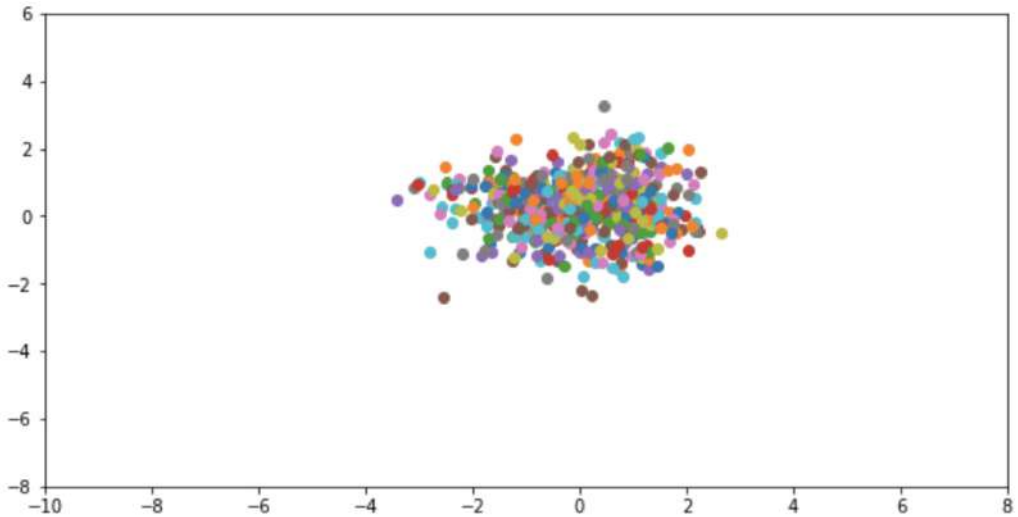


* KLD Term 추가 후 z값 분포

이전과 비교해 Z값이 훨씬 뭉쳐져 있는 것을 확인 가능

이상적인 결과물이 나오지 않은 이유?

2. Z의 영향을 줄이기 위해 KLD Loss 도입

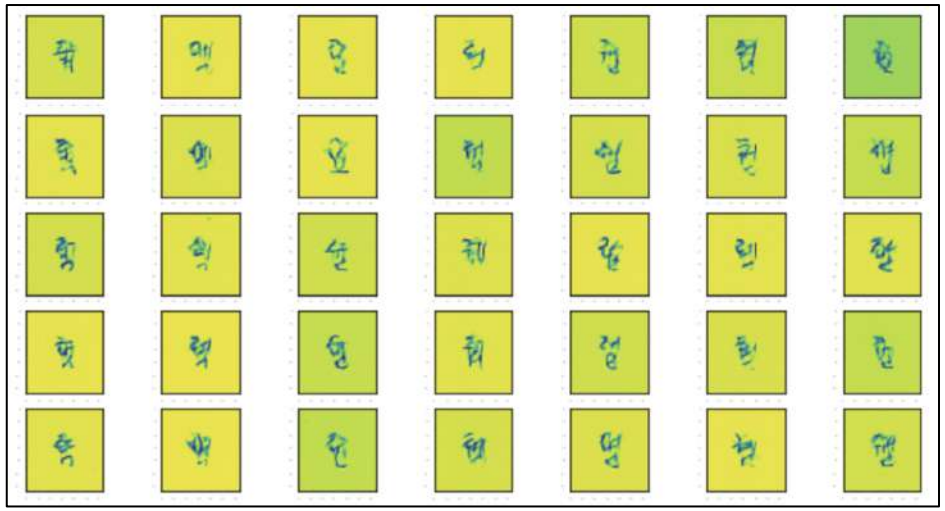


* KLD Term 추가 후 z값 분포

Bottleneck size=32, KLD
(with gaussian random vector for z)

이상적인 결과물이 나오지 않은 이유?

2. Z의 영향을 줄이기 위해 KLD Loss 도입



Bottleneck size=32,
(with gaussian random vector for z)

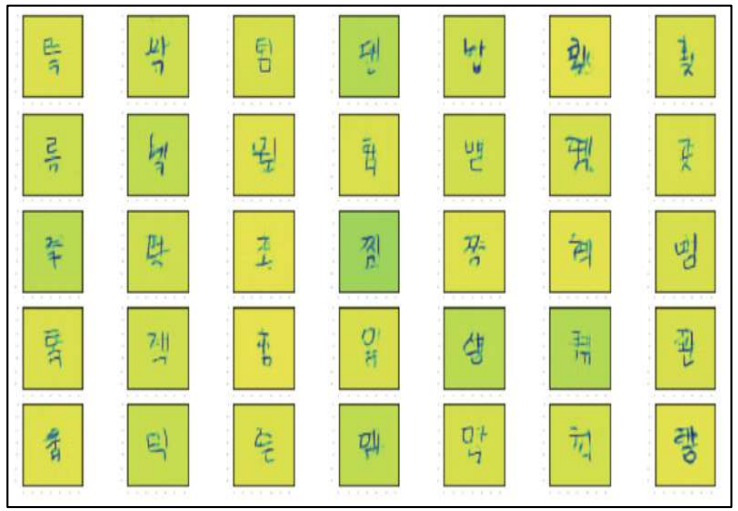
Bottleneck size=32, KLD
(with gaussian random vector for z)

KLD 추가 전보다 조금 더 나아졌지만 실제 글자와는 거리가 멀어 보임

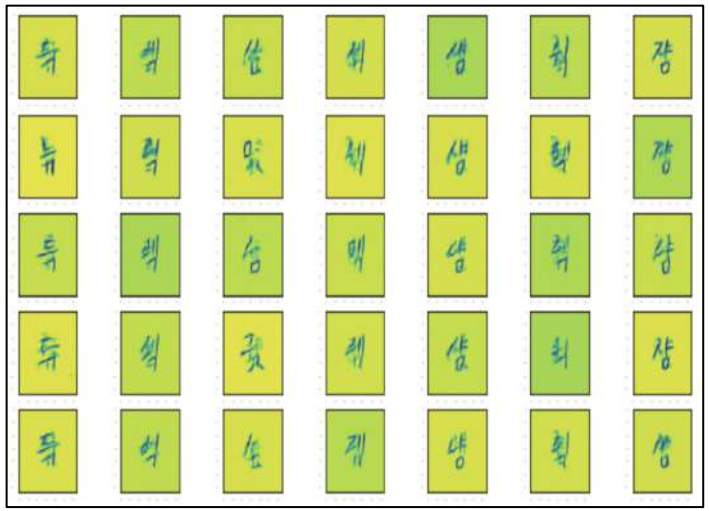
→ 여전히 Z의 영향력이 매우 크다는 가설 설정

이상적인 결과물이 나오지 않은 이유?

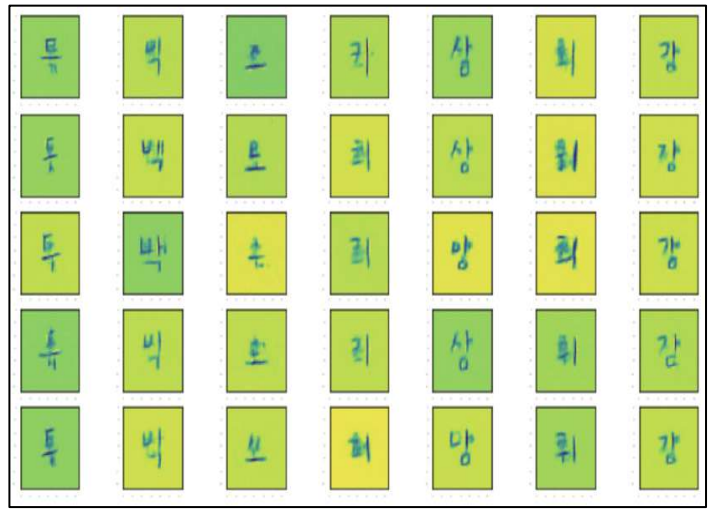
3. Bottleneck Size 조정



Bottleneck size=32, KLD
(with gaussian random vector for z)



Bottleneck size=16, KLD
(with gaussian random vector for z)



Bottleneck size=8, KLD
(with gaussian random vector for z)

Bottleneck size가 줄어들수록 점점 '투빅스지상최강' 의 모습과 유사해지는 것을 확인할 수 있음

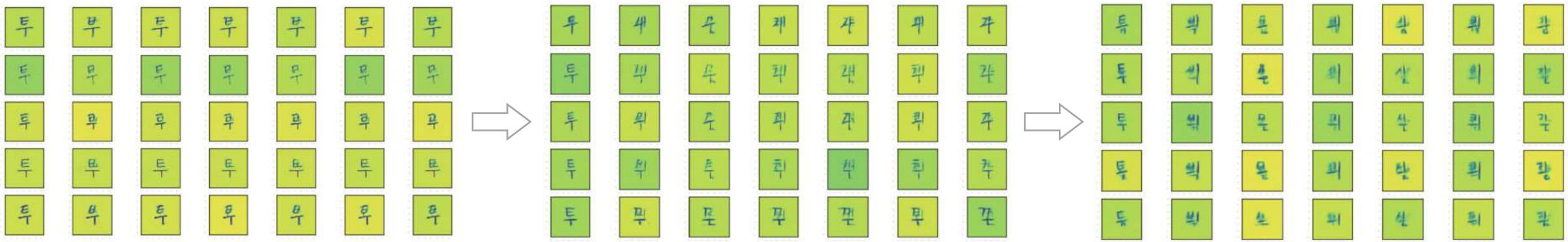
(Z의 영향을 줄일수록 상대적으로 embedding vector의 역할이 커짐)

→ *우리가 원하는 스타일의 글자를 만들 수 있다!*

이상적인 결과물이 나오지 않은 이유?

4. Letter Vector의 영향에 대한 확인

'투' 단어로 얻은 z (고정) + category vector + letter vector

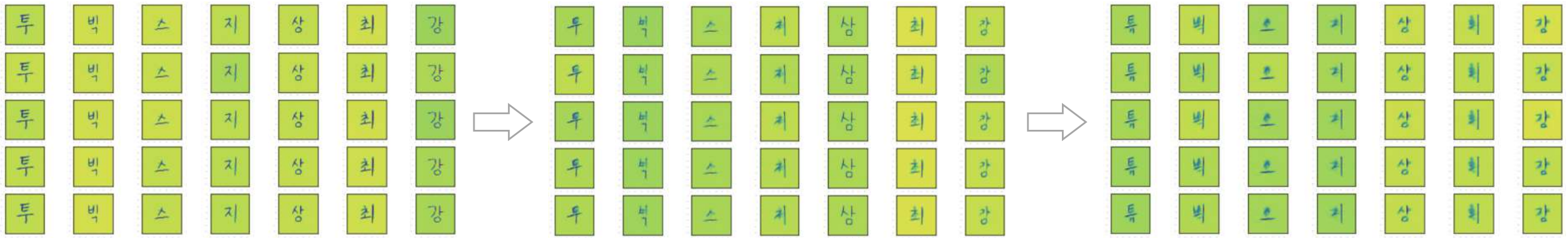


Letter vector로 인해 폰트가 변하는 것을 볼 수 있음

이상적인 결과물이 나오지 않은 이유?

5. Category Vector의 영향에 대한 확인

0 category로 얻은 z + category vector + letter vector



Category vector의 차이로 결과에서 큰 의미를 찾기는 어려움.
아무래도 같은 글자의 폰트 차이는 크지 않기 때문에
그 미묘한 차이를 폰트 카테고리로 구분하는 것은 상대적으로 어려워 보임.
(Category vector도 업데이트할 여지가 있어 보임)

의의 및 한계점

정리

폰트를 복원하는 것이 condition 보다는 폰트 이미지 자체의 latent vector 값에 매우 dependent했음.

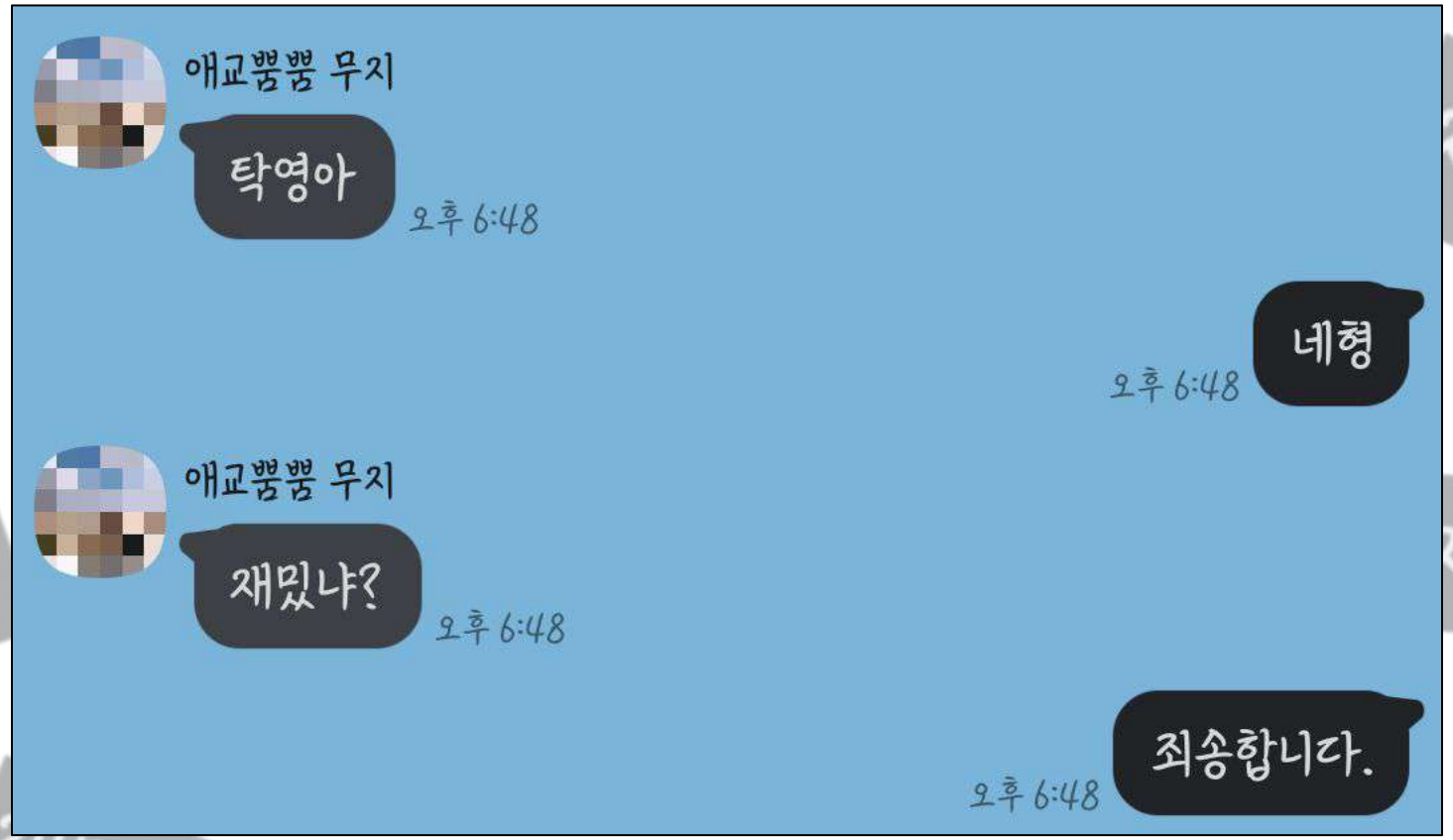
따라서 폰트를 생성하는 과정에서 condition의 영향을 키우는 동시에
z의 영향을 줄이는 데에 많은 시간과 고민을 할애하였음

Z의 dimension을 줄이고, condition에 가중치를 주는 등의 시도를 하였지만,
Condition에 dependent한 결과가 나올 만큼의 개선을 보이지는 못했다는 점에서 아쉬움이 있음.

Future Work

- Condition으로 추가된 embedding vector에 대해 아키텍쳐 내부에서 update를 진행
- Embedding vector의 차원 및 scale에 대한 변화
- Condition의 영향을 키워주기 위해 discriminator 이용 및 그 오차를 네트워크에 전파

마무리



쉽지 않았습니다.

마무리

(수면부족)

(퇴근하고)

프로젝트를 통해 얻은 점

- 아이디어에 맞춘 새로운 네트워크를 직접 디자인 & 구현
- Z 값을 실제로 plotting해 보고, 그에 맞는 weight도 직접 출력해 본 경험
- Git, Notion 등의 협업 툴에 익숙해지는 경험

(분노)

(절교)

멤버 소개

7기 이수빈

할까 말까 고민하다가 했는데 후호.. | ...가 아니고 너무나 유익했고, 한층 성장했음을 느낍니다. 근데 왜 때문에 loss 안줄어 ㅎ?

10기 신훈철

전처리 후처리 훈처리~!

11기 이영전

7to12 팀원분들께 감사를 전합니다.
컨퍼런스 그 날까지 밤새워 노력한 인공지능 폰트 스타일러 화이팅!

12기 김탁영

다음 컨퍼런스 때에는 저도 후배에게 재밌냐?고 물어볼 수 있는 든든한 선배가 되겠습니다.


thanks!
Tombow ABT
Acid Free
N15