

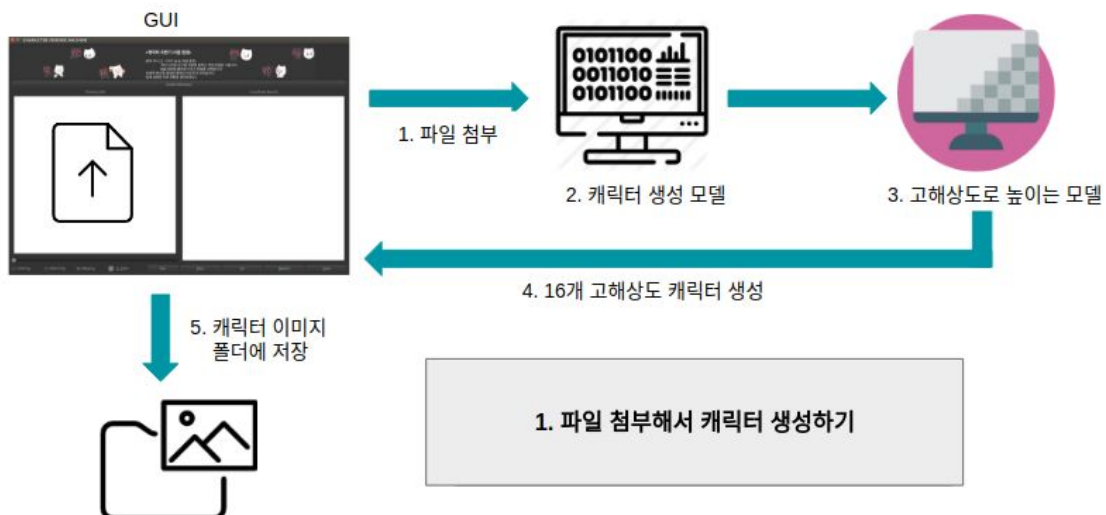
Case Study AI 응용 실습 프로젝트 결과보고서

프로젝트명	캐릭터 자판기		
프로젝트요약	사용자가 게임 캐릭터를 디자인할 때 드는 어려움을 덜어주고자 하는 프로젝트로, 사용자가 원하는 이미지의 특정 배경을 선택하거나 스케치하면 그러한 특징을 갖는 게임 캐릭터가 생성된다.		
팀명	GOO	팀원	고민주 오영진 오현지
대분류	영상	소분류	Image Generation
AI 모델	DNN(DCGAN)	학습방식	Unsupervised Learning

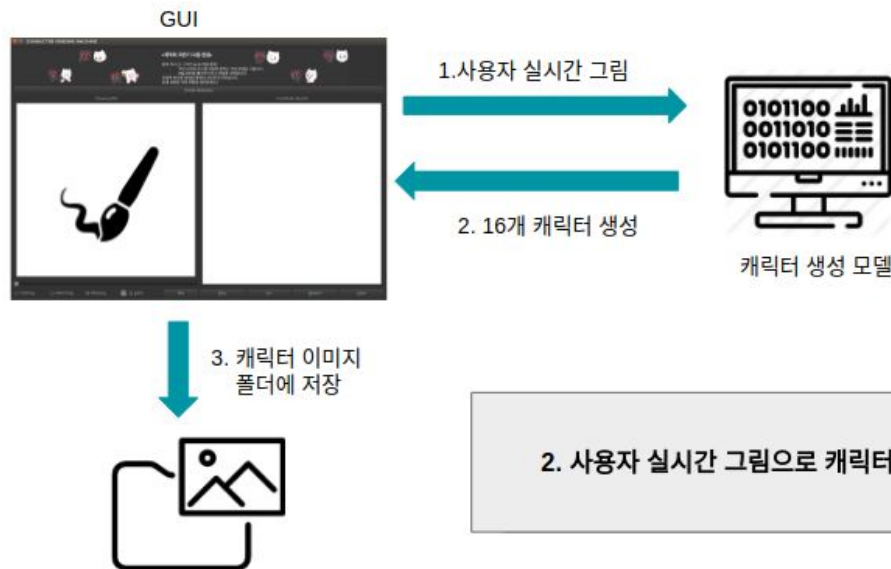
1. AI 프로젝트 개요

AI 개발환경	Docker 환경구성	Ubuntu 16.04, CUDA 8.0, cuDNN 8.0.61
	Docker Hub주소	(iGAN) https://hub.docker.com/r/ggmmjj1/mini4/ (srgan) https://hub.docker.com/r/oyj9097/srgan/ (두 개의 도커를 동시에 실행해서 사용함)
	프레임워크(버전)	Tensorflow 1.4
	언어(버전)	Python3
데모 스펙	기타(IDE 등)	spyder3, openCV3, theano, pyqt4, hdf5
	HW(장비포함)	운영체제 : Linux ktai03-Alienware-Aurora-R7 4.13.0-43-generic #48~16.04.1-Ubuntu SMP Thu May 17 12:56:46 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux 프로세서 : Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz 메모리 : 31GiB
	SW	GUI: pyqt4
	프로젝트 구조도	

기능1. 파일 첨부하여 컨셉에 맞는 캐릭터 생성



기능2. 그림 그려 실시간으로 캐릭터 생성



2. AI 프로젝트 결과

1. 관련연구 분석

년도	저자	모델	데이터셋	분석	논문정보 (citation 형식)
2016	Jun-Yan Zhu, Philipp Krahenbuhl, Eli Shechtman, and Alexei A. Efros	iGAN	https://people.eecs.berkeley.edu/~junyanz/projects/gvm/datasets/	기존 DCGAN 모델을 활용하여 단순한 이미지 생성뿐 아니라 interactively 이미지 형태 변환이 가능하다. 1) DCGAN (이미지생성) $z \rightarrow \text{Generator} \rightarrow X(\text{image})$ 2) AlexNet ~4th Conv. (형태변환, 스케치) $X(\text{image}) \rightarrow \text{predict_z} \rightarrow Z\text{-vector}$	Zhu, Jun-Yan, et al. "Generative visual manipulation on the natural image manifold." European Conference on Computer Vision. Springer, Cham, 2016.
2015	Radford, Alec, Luke Metz, and Soumith Chintala.	DC-GAN	LSUN, images containing human faces from random	기존 GAN에 존재했던 fully-connected 구조의 대부분을 CNN 구조로 대체한다. Discriminator에서는 모든 pooling layers를 strides	Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised

			web image queries of peoples names	convolutions으로 바꾸고, Generator에서는 pooling layers를 fractional-stride convolutions으로 바꾼다. Generator output layer, Discriminator input layer를 제외한 layer에 batch-normalization을 사용한다. Fully-connected hidden layers를 삭제한다. 이런 구조를 통해 대부분의 상황에서 언제나 안정적으로 학습이 가능하고 벡터 산술 연산이 가능하다.	sed representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).
2005	Dalal, Navneet, and Bill Triggs.	HOG	MIT pedestrian database	Object Tracking에 많이 사용되는 feature중 하나로 Image의 지역적인 Gradient를 해당 영상의 특징으로 사용하는 방법이다. Edge의 양과 방향을 구분하는 특성을 가지며, Overlap을 이용하여 계산하기 때문에 어느정도 Shift에도 적응할 수 있다. Image에서 Gradient를 계산하고 계산된 값을 이용하여 Local Histogram을 생성한다. 이렇게 생성된 Histogram을 이어붙여 1차원 vector를 생성한다. 성능 지표: FPPW (False Positives Per Window)	Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.
2016	Chao Dong, Chen Change Loy, Member, IEEE, Kaiming He, Member, IEEE, and Xiaoou Tang, Fellow, IEEE	SR-CNN	ILSVRC 2013 ImageNet	입력 저해상도 이미지는 Y, 복원한 출력 고해상도 이미지는 F(Y), ground truth 고해상도 이미지는 X로 표현할 때, 고해상도 이미지를 복원할때 저해상도 이미지 Y로부터 patch 추출하는 Patch extraction and representation, 다차원 patch 벡터를 다른 다차원 patch 벡터로 mapping하는 Non-linear mapping, 다차원 patch 벡터에서 최종 고해상도	DONG, Chao, et al. "Image super-resolution using deep convolutional networks." IEEE transactions on pattern analysis and

				이미지 생성하는 Reconstruction, 이 세가지 연산을 Super-Resolution Convolutional Neural Network (SRCNN)으로 처리한다.	machine intelligence, 2016, 38.2: 295-307.
2017	Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi	SR-GAN	ILSVRC 2013 ImageNet	DCGAN을 활용하여 저해상도 이미지를 고해상도 이미지로 변환을 시도한다. Discriminator와 ResNet을 포함한 Generator를 학습하고 이용한다. 기존의 Super Resolution 기법의 SRCNN 구조를 사용할 때와 ResNet을 포함한 SRGAN 구조를 사용할 때의 결과와 성능을 비교한다. 성능 지표: PSNR, SSIM, MOS	Ledig, Christian, et al. "Photo-realistic single image super-resolution using a generative adversarial network." arXiv preprint(2017).
2017	Tero Karras, Timo Aila, Samuli Laine, Jaakko Lehtinen	Progressive GAN	CELEBA, LSUN, CIFAR10 https://github.com/tkarras/progressive_growing_of_gans/blob/master/dataset.py	Generator와 Discriminator의 layer를 대칭적으로 하나씩 쌓아가며 학습하게 되면 large-scale structure를 먼저 파악한 뒤에 finer scale detail로 점점 학습의 주안을 옮겨간다. layer를 늘리는 시점에 갑작스러운 충격을 막기 위해 Highway network의 구조를 사용한다. 이 방법을 사용함으로 학습의 안정성 외에도 학습 속도를 획기적으로 올릴 수 있다.	KARRAS, Tero, et al. Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196, 2017.

II. 사용 데이터

i. 출처

- <http://yurudora.com/tkool/>

ii. 구성

- 1) 64 x 64 게임 캐릭터 이미지
- 2) 128 x 128 게임 캐릭터 이미지

iii. 개수

- 1) 1,023x64 개

- 2) 40,000 개 (1,034개를 변환함)
- iv. 다운경로
- 1) 64x64 게임 캐릭터 이미지
<https://drive.google.com/file/d/1Z-6c61dPqspxCrs7vQ4ZIDldqGa47i9-/view>
 - 2) 128x128 게임 캐릭터 이미지
https://drive.google.com/open?id=1a_YG1Ny8qQZOgaVlpkNwu6bg1-MNlxi3

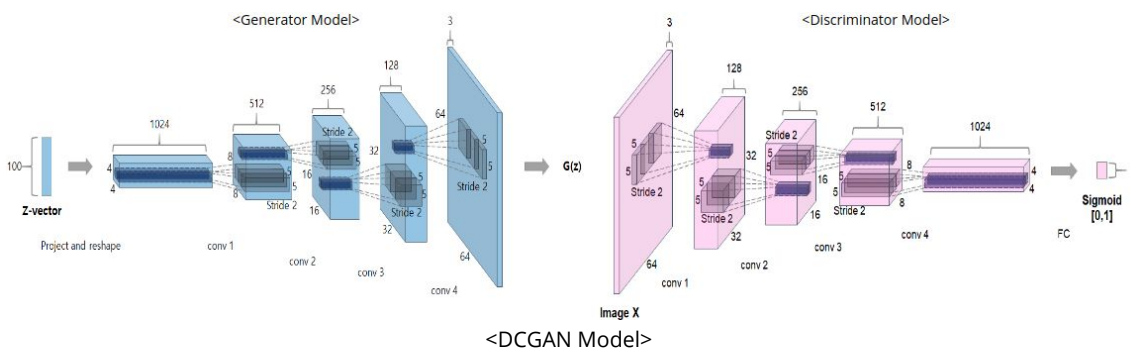
III. AI 모델

i. 역할

(* 프로젝트의 두 가지 기능인 파일 첨부와 그림 그리기로 나누어 AI 모델을 설명한 후, 고해상도로 변환하는 AI 모델을 설명한다.)

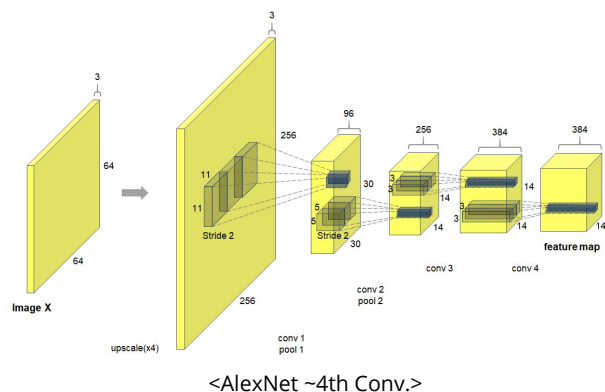
공통) 파일 첨부와 그림 그리기 기능에서 공통으로 이용하는 AI 모델 (DCGAN Model)

Discriminator와 Generator Model을 학습시키고, 이후 학습된 Generator Model은 Z-vector를 받아 캐릭터 Image를 생성하는 역할을 한다.



기능 1) 파일 첨부 기능에서 이용하는 두 가지 AI 모델 (AlexNet)

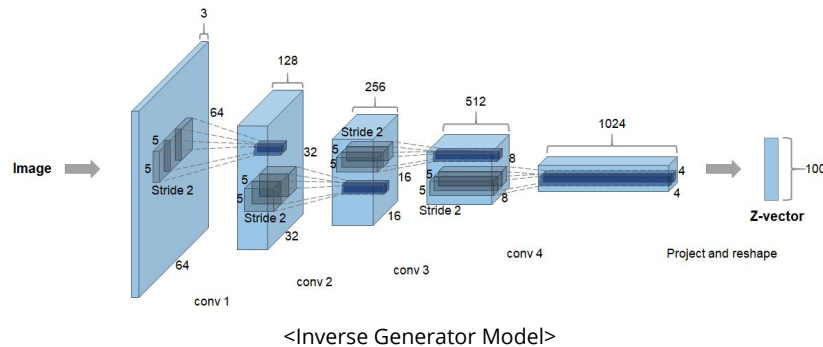
AlexNet의 4번째 Convolutional Network까지만 이용하며, real Image와 Generated Image를 각각 받아 feature map을 구하여 두 feature map의 차이인 cost를 계산하는데 이용된다.



(Inverse Generator)

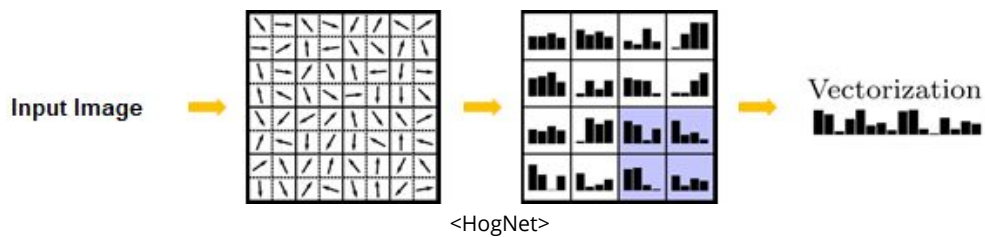
DCGAN의 Generator Model의 Inverse 형태로, Image를 받았을 때 Z-vector를

예측하기 위한 모델이다. 그리고 예측된 Z-vector는 캐릭터 Image 생성을 위해 학습된 Generator를 거치게 된다.



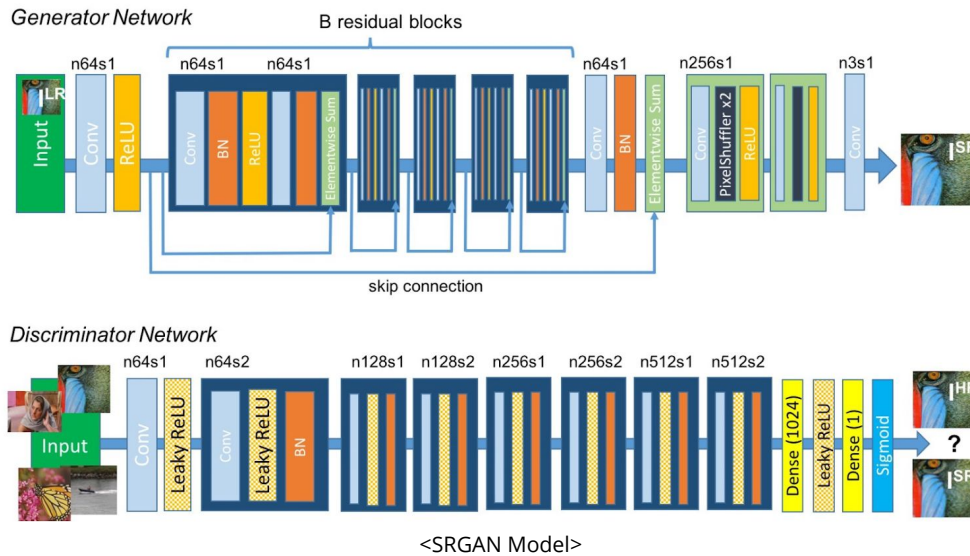
기능 2) 그림 그리기 기능에서 이용하는 AI 모델 (HogNet)

Interactively 형태 변환을 시행할 수 있게 이용한 네트워크로 Image를 받아 Z-vector를 생성한다. 이 네트워크는 Image를 받으면 픽셀 당 local gradient를 계산하고 계산된 값을 이용하여 local histogram을 생성한다. 그리고 생성된 histogram을 이어붙여 1차원 Z-vector를 생성한다. 이 때 생성된 Z-vector는 캐릭터 Image 생성을 위해 학습된 Generator를 거치게 된다.



고해상도 변환) 고해상도로 변환할 때 이용하는 AI 모델 (SRGAN)

SRGAN의 Generator와 Discriminator Model을 학습하여 저해상도 이미지를 고해상도 이미지로 변환하고자 하는 모델이다. Generator Model에는 ResNet 구조가 포함되어있고, 이미 학습된 Model을 고해상도 캐릭터 Image로 재학습하였다. 앞에서 Generator가 생성한 저해상도 캐릭터 Image가 SRGAN의 Generator Model을 거쳐 고해상도 캐릭터 Image로 변환된다.



ii. 입/출력

Model		Input	Output
DCGAN	Discriminator	64x64 Image	constant
	Generator	Z-vector	64x64 Image
AlexNet~4th Conv.		256x256 Image	14x14x384 feature map
Inverse Generator		64x64 Image	Z-vector
HogNet		64x64 Image	Z-vector
SRGAN		64x64 Image	128x128 Image

<Model의 Input과 Output>

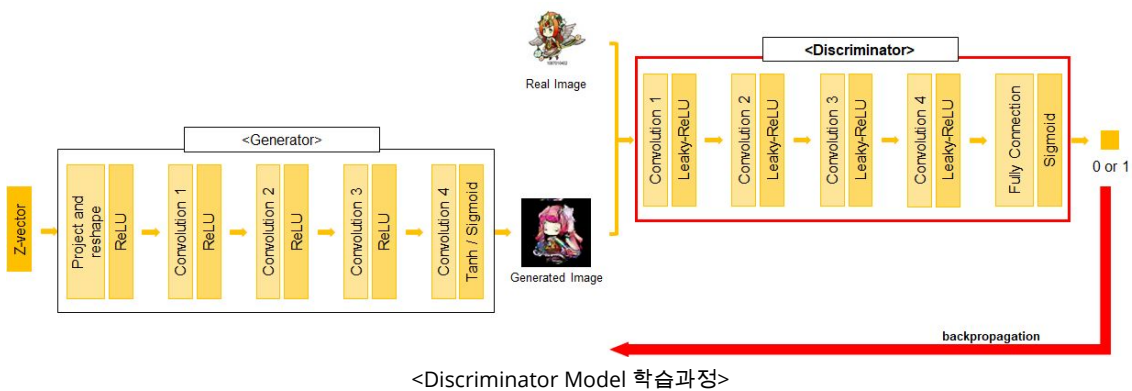
iii. 구조

두 가지 기능(파일 첨부와 그림 그리기)에서 캐릭터 생성을 위해 공통으로 필요한 DCGAN Model 학습 과정(1)을 먼저 설명하고, 두 가지 기능에 따라 사용된 Model의 학습과정과 적용 결과(2-1, 2-2)를 보여준 뒤, 고해상도로 변환하기 위한 Model의 학습과정과 적용결과(3)를 설명한다. 그리고 최종적으로 전체 구조도(4)를 볼 수 있다.

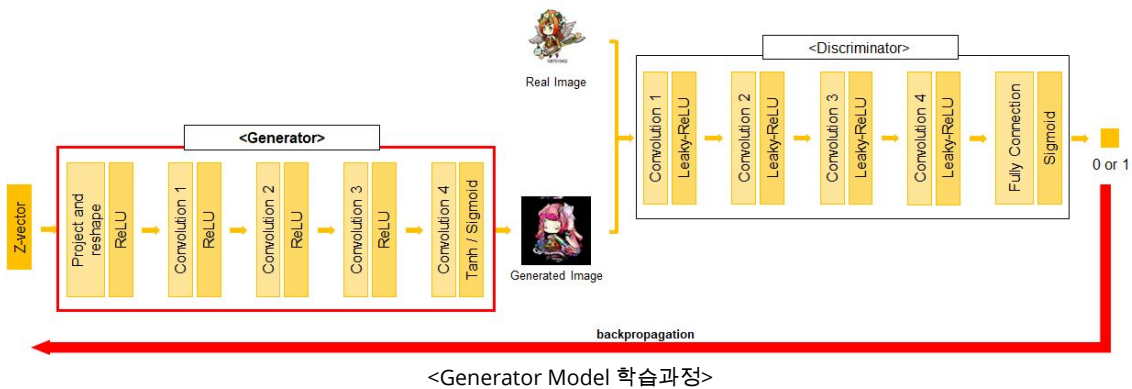
1. DCGAN Model 학습 과정

먼저 DCGAN Model의 Discriminator와 Generator Model을 학습시켜 캐릭터 생성을 위한 Generator Model을 준비한다. DCGAN의 학습과정과 학습 parameter는 아래와 같다.

- Discriminator 학습 : Discriminator는 Real Image를 real로(=1), Generated Image를 fake로(=0) 올바르게 판별하도록 학습한다.



- Generator 학습 : Generator의 목적은 Real Image와 비슷하게 Image를 생성하고자 하는 것이기 때문에 Discriminator가 Generated Image를 real(=1)로 잘못 판별하도록 학습한다.



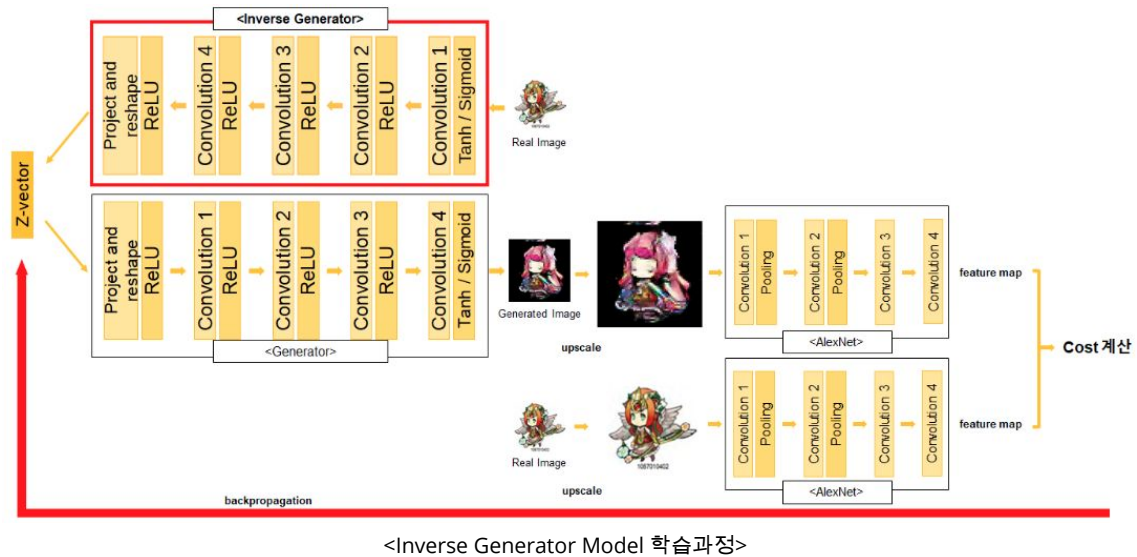
Model Constructing Input Parameter		
Parameter	값	의미
n_layers	3	# of convolution layers
n_f	128	# of (last) feature channels
npx	64	height (= width)
nc	3	# of image channels (RGB)
nz	100	# of dim for Z-vector
niter	25	# of iter at starting learning rate
niter_decay	25	# of iter to linearly decay learning rate to zero
init_sz	4	# of first feature map size(4*4)
fs	5	fractionally-strided convolution filter size
bs	32	batch_size
lr	0.0002	learning_rate
b1	0.5	Momentum

<DCGAN 학습 Parameter>

2. 기능에 따른 각각의 Model 이용

2-1-1. 파일 첨부 기능 - Inverse Generator Model 학습 과정

파일을 첨부하여 원하는 컨셉의 캐릭터를 생성하기 위하여 캐릭터 Image로 Inverse Generator Model을 학습한다. 학습과정과 parameter는 아래와 같다.



- ① Real Image가 Inverse Generator를 거쳐 Z-vector를 생성한다.
- ② 생성된 Z-vector는 이미 학습된 Generator를 거쳐 캐릭터 Image를 생성한다.
- ③ 생성된 Image와 Real Image를 각각 AlexNet의 4번째 Convolutional Network까지 통과시킨다.
- ④ AlexNet을 거쳐 나온 각각의 feature map을 비교하여 cost를 계산한다.
- ⑤ 이를 반복하며 cost가 작아지도록 Inverse Generator를 학습한다.

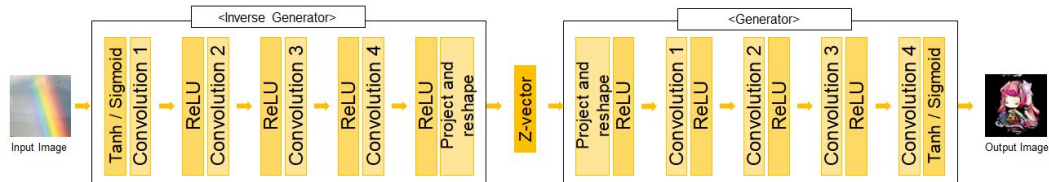
Model Constructing Input Parameter		
Parameter	값	의미
n_layers	3	# of convolution layers
n_f	128	# of (last) feature channels
npx	64	height (= width)
nc	3	# of image channels (RGB)
nz	100	# of dim for Z-vector
niter	25	# of iter at starting learning rate
niter_decay	25	# of iter to linearly decay learning rate to zero
init_sz	4	# of first feature map size(4*4)
fs	5	fractionally-strided convolution filter size
bs	32	batch_size
lr	0.0002	learning_rate
b1	0.5	Momentum

<Inverse Generator Model 학습 Parameter>

2-1-2. 파일 첨부 기능 - 학습된 Model을 이용한 이미지 생성 과정

학습된 Generator와 Inverse Generator Model을 이용하여 Input Image를 넣었을 때 Output

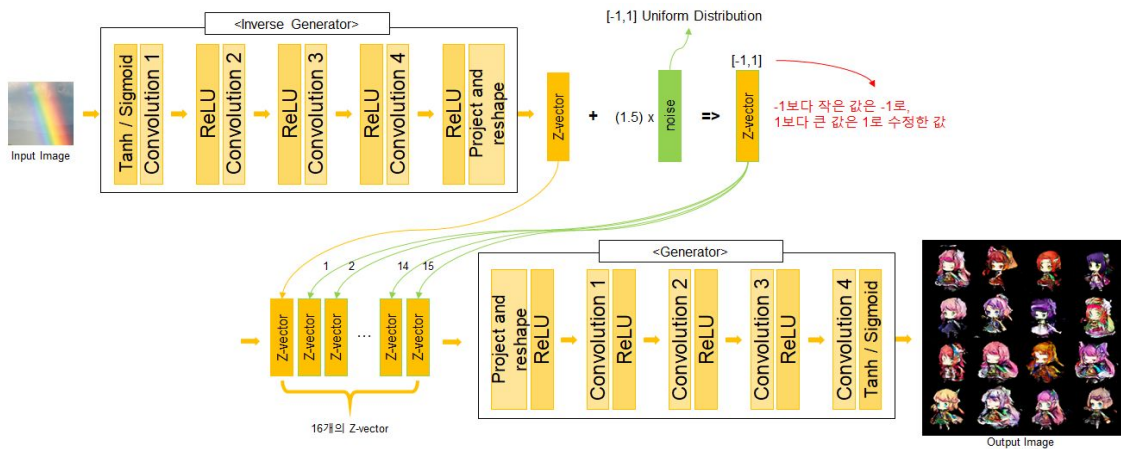
Image가 출력되는 과정이다.



<학습된 Inverse Generator를 통한 Z-vector 생성과 학습된 Generator를 통한 캐릭터 Image 생성 과정>

- ① 원하는 컨셉의 Image를 Inverse Generator가 받아 Z-vector를 예측한다.
- ② 예측된 Z-vector가 Generator를 거쳐 캐릭터 Image를 생성한다.

앞의 과정은 1개의 Input으로 1개의 Output을 출력하였다. 이제 총 16개의 Output Image를 출력하기 위하여 생성된 Z-vector에 noise vector를 더해 준다. 자세한 내용은 아래와 같다.

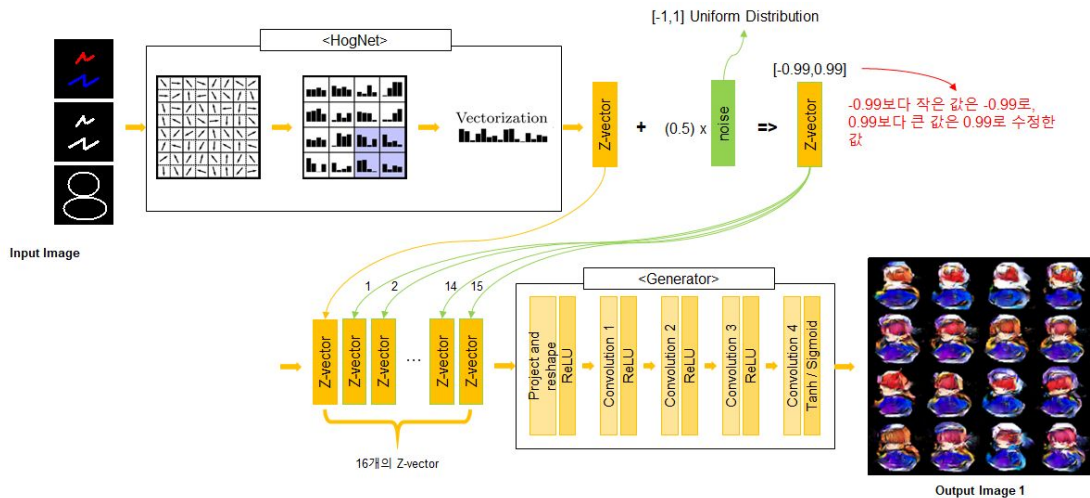


<학습된 Inverse Generator를 통한 Z-vector 생성과 16개의 캐릭터 Image 생성 과정>

- ③ [-1,1] 범위의 Uniform 분포를 따르는 길이 100의 random noise vector 15개를 생성한다.
- ④ 앞에서 예측된 Z-vector에 noise vector의 weight를 1.5로 하여 noise vector를 각각 더한 새로운 Z-vector 15개를 생성한다.
- ⑤ 새로운 Z-vector의 범위가 [-1,1]이 되도록 -1보다 작은 값은 -1로, 1보다 큰 값은 1로 바꾼다.
- ⑥ 총 16개의 Z-vector가 학습된 Generator를 거쳐 16개의 캐릭터 Image가 생성된다.

2-2-1. 그림 그리기 기능 - HogNet을 이용한 이미지 생성 과정

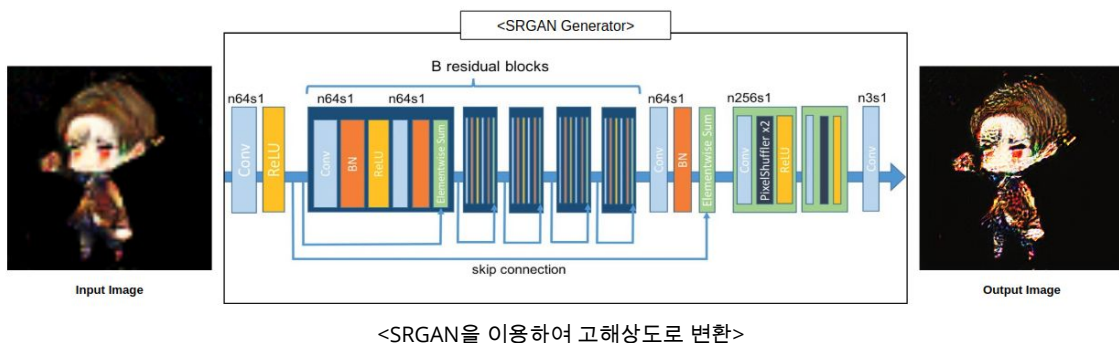
그림을 그렸을 때 HogNet을 이용하여 캐릭터 Image가 생성되는 과정과 16개의 Output Image를 생성하기 위한 과정은 아래와 같다.



- ① 그림을 그리면 color와 color mask(color를 입력할 영역), edge(sketch 형태) Image가 생성된다.
- ② 세 개 Image를 받아 픽셀 당 local gradient를 계산하고, 계산된 값을 이용하여 local histogram을 생성한다. 생성된 histogram을 이어붙여 1차원 Z-vector를 생성한다.
- ③ 앞의 기능과 마찬가지로 16개의 Output을 출력하기 위하여 생성된 Z-vector에 noise vector를 더해 준다. 단, 앞과 달리 noise vector의 weight 값은 0.5이며, noise vector를 더한 15개의 Z-vector의 범위는 [-0.99, 0.99]로 -0.99보다 작은 값은 -0.99로, -0.99보다 큰 값은 0.99로 바꾼다.
- ④ 16개의 Z-vector는 학습된 Generator를 거쳐 16개의 캐릭터 Image를 생성한다.

3. 저해상도에서 고해상도로 변환

두 가지 기능으로 출력한 16가지의 캐릭터 Image는 64x64 저해상도 Image이다. 이 Image를 128x128 고해상도 Image로 변환하기 위하여 SRGAN Model을 이용한다. ILSVRC 2013 ImageNet의 395,909개 이미지로 학습된 SRGAN Model을 이용하여 고해상도로 변환한 결과이다.



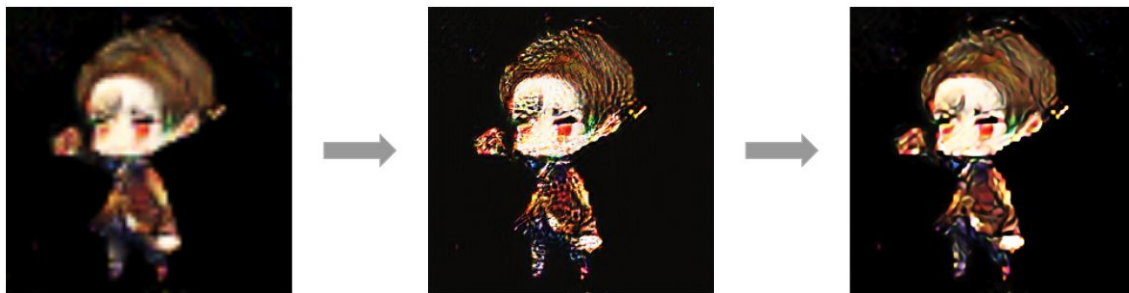
3-1. SRGAN Model 재학습

학습된 SRGAN을 40,000개의 128x128 고해상도 캐릭터 Image dataset을 이용하여 재학습한다.

이전 결과보다 재학습한 Generator Model을 이용하여 고해상도 Image로 변환하였을 때가 더 자연스럽게 캐릭터가 형성되었다. 따라서 고해상도 캐릭터로 재학습된 SRGAN Model을 이용하여 16개의 64x64 저해상도 캐릭터 Image를 128x128 고해상도 캐릭터로 변환한다. SRGAN의 Model 학습 parameter는 아래와 같다.

Model Constructing Input Parameter	
Parameter	값
batch_size	16
learning_rate	1e-4
momentum	0.9
epoch	100
decay_every	50
lr_decay	0.1

<SRGAN 학습 Parameter>



(a)Original Image

(b)이미 학습된 Model 적용 결과 Image

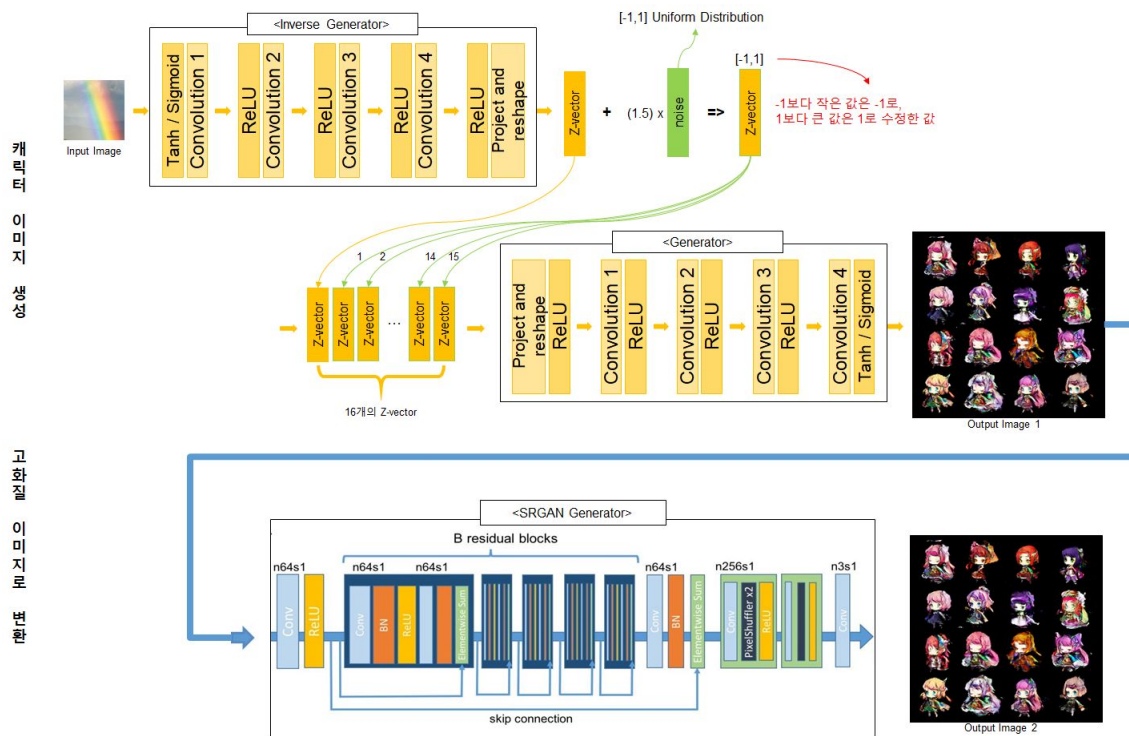
(c)재학습된 Model 적용 결과 Image

<이미 학습된 SRGAN과 재학습된 SRGAN의 Generator Model을 이용한 결과>

4. 전체 구조도

학습된 모델을 이용하여 캐릭터를 생성하는 과정은 아래와 같다.

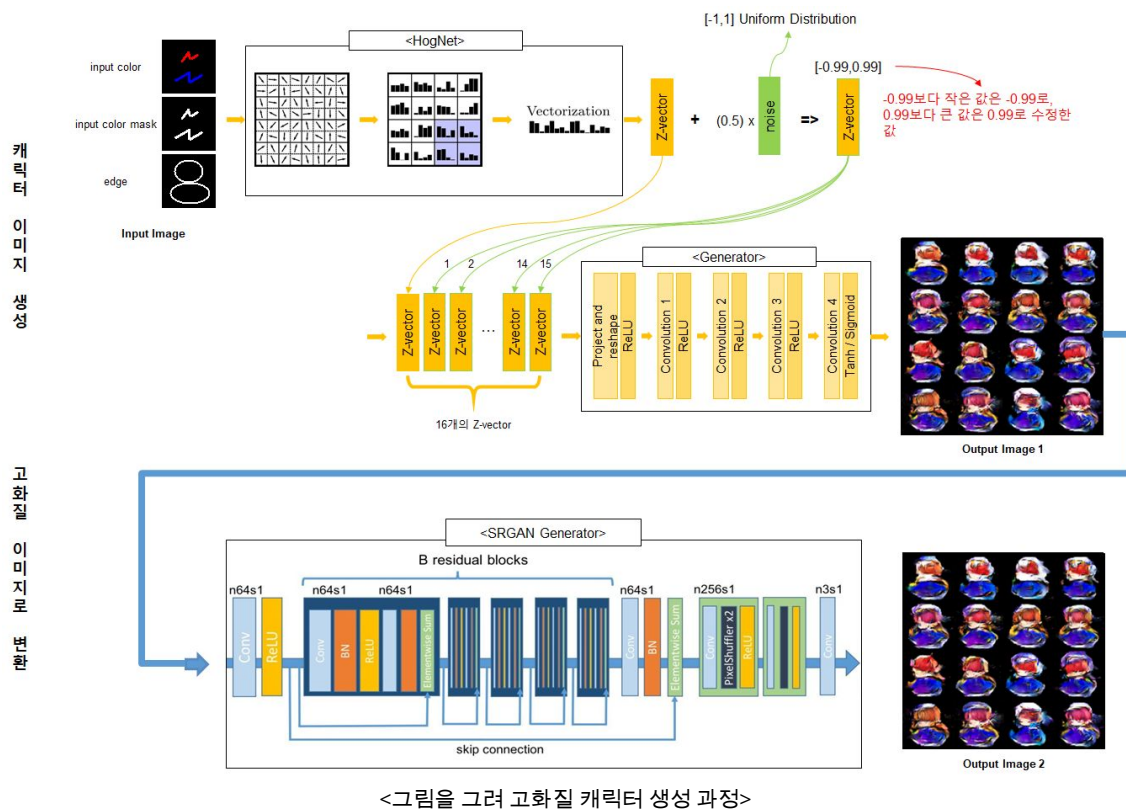
기능 1. 파일 첨부



<파일을 첨부하여 고화질 캐릭터 생성 과정>

- ① 원하는 컨셉의 이미지를 넣으면 Inverse Generator를 거쳐 1개의 Z-vector가 생성된다.
- ② 생성된 Z-vector에 noise-vector 15개를 넣어 총 16개의 Z-vector를 생성한다.
- ③ 16개의 Z-vector가 Generator를 거쳐 16개의 64x64 캐릭터 Image를 생성한다.
- ④ 64x64 캐릭터 Image를 SRGAN Generator를 통해 128x128 고화질 캐릭터 Image로 변환한다.

기능 2. 그림 그리기



- ① 그림을 그리면 세 가지 Input Image가 생성되고 HogNet을 거쳐 Z-vector가 생성된다.
- ② 생성된 Z-vector에 noise-vector 15개를 넣어 총 16개의 Z-vector를 생성한다.
- ③ 16개의 Z-vector가 Generator를 거쳐 16개의 64x64 캐릭터 Image를 생성한다.
- ④ 64x64 캐릭터 Image를 SRGAN Generator를 통해 128x128 고화질 캐릭터 Image로 변환한다.

iv. 코드설명 및 코드주소

- 코드 압축 파일

<https://drive.google.com/open?id=1nhzfn6gmrHa6hmn5lKyG0lgBYWUxpnix>

- Readme 파일

https://drive.google.com/open?id=1mxccCqh1S_gEPos94flj9YzepRW8Y5aysKcXocixyU

(Readme파일 구글 드라이브 주소)

** 수정 파일 -> 파란색 박스

<iGAN 코드 설명>

iGAN

iGAN_main.py
모델, 파라미터 세팅
GUI 화면 실행

iGAN_predict.py
파일에 대한
캐릭터 16개 생성

iGAN_script.py
UI없이
스크립트 실행

UI

gui_design.py
Drawing Tool Design
layout에 대한 코드

gui_draw.py
사용자 실시간으로
그리는 것에 대한 코드

gui_vis.py
그림을 그렸을때,
오른쪽 화면에 나오는
결과물들에 대한 코드

gui_loading.py
이미지 파일을 넣었을때
나오는 로딩 화면

save_result.py
save 버튼을 눌렀을때,
이미지를 저장하는 코드

ui_sketch.py
그림 그릴때,
스케치하는 것에 대한 코드

ui_color.py
그림 그릴때,
사용자가 선택하는
컬러에 대한 코드

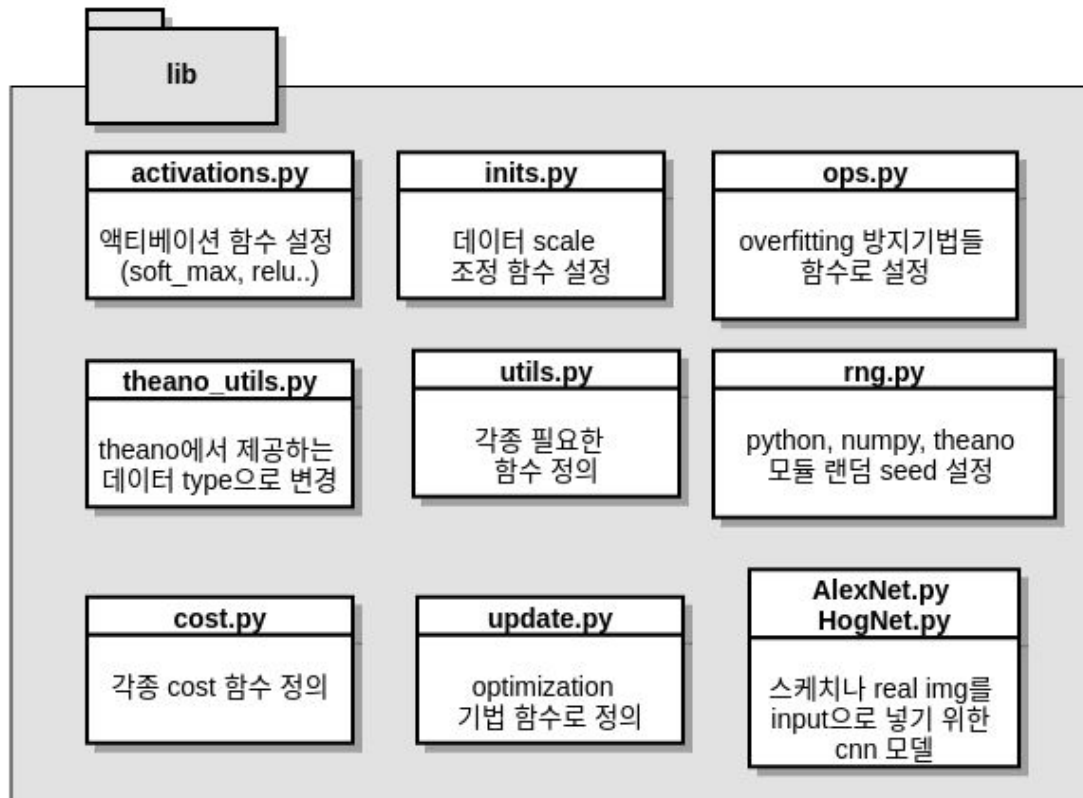
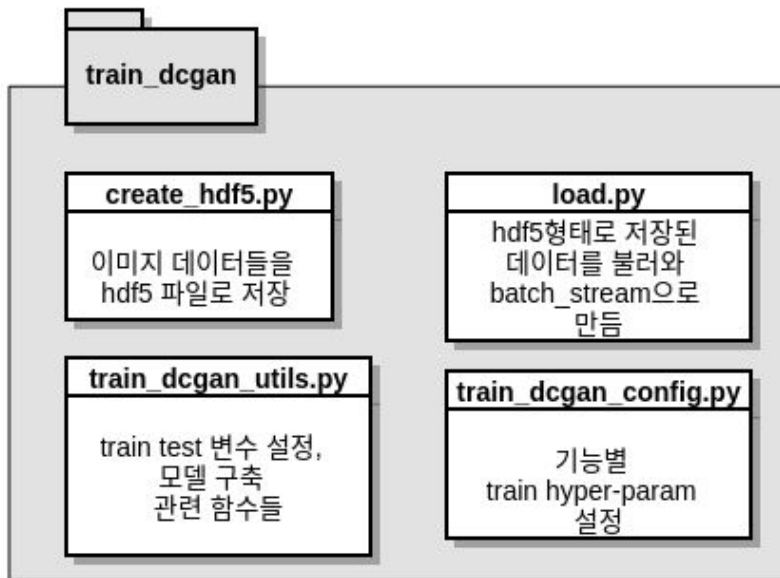
ui_wrap.py
그림 그릴때,
warping 하는 것에 대한 코드
(더 정교한 작업)

ui_recorder.py
그림 그리는 과정을 저장해서
play 버튼을 눌렀을때
과정을 보여주는 코드

model_def

dcgan_theano.py
main 실행했을때
학습된 모델을
불러오는 기능을 담은
함수들 정의

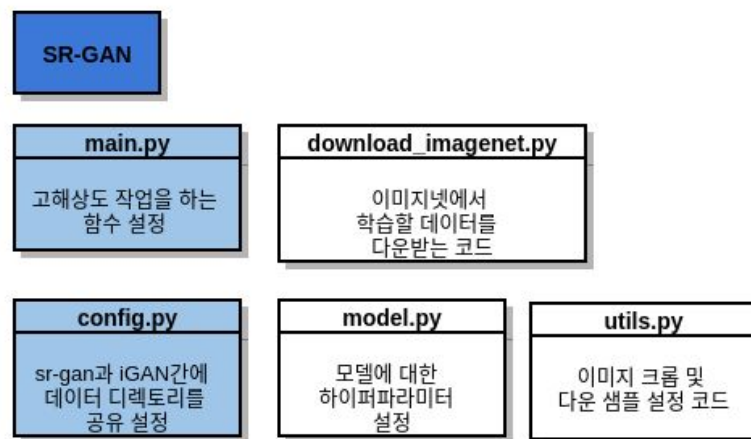
dcgan_theano_config.py
실행할 모델에 대한
하이퍼파라미터 정의



- **iGAN_main.py** : QSplashScreen 위젯을 통해 splash 화면이 4초 가량 나타나게 하였다.
- **iGAN_predict.py** : 이미지를 넣었을때 캐릭터 생성을 처리하는 부분을 구현한 코드이다. Z-vector에 15개의 noise vector 를 추가하여 16개의 캐릭터 생성 과정.
 - Inverse Generator Model에 의해 생성된 Z-vector에 15개의 랜덤 noise-vector추가 (noise-vector는 [-1,1] uniform distribution을 따르는 길이 100인 vector)

- 생성된 Z-vector + 1.5 x noise-vector. (1.5는 noise vector의 weight 값)
- Z-vector가 [-1,1] 범위가 되도록 -1 보다 작은 값은 -1로, 1 보다 큰 값은 1로 수정
- 생성된 16개 Z-vector는 생성된 1개의 Z-vector와 noise를 추가한 15개의 Z-vector
- 학습된 Generator를 이용하여 16개의 캐릭터 이미지 생성
- 16개 캐릭터 이미지 converted_pics 폴더에 저장. 파일 이름 형식은 HHMMSS(생성시간, 분, 초)_cnn_opt_a.png이며 16개의 이미지는 뒤에 a~p의 값으로 구분된다.
- **gui_design.py** : pyqt를 이용해 GUI를 구성하는데, UI 레이아웃에 대한 디자인을 구현한 코드이다. 기존 github에 올라온 코드에서 설명서와 이미지 파일 추가, 고해상도 이미지 처리에 대한 함수들을 추가 구현하였다. 레이아웃 상단에는 간단한 설명이 나오고, 아래 버튼을 클릭하면, 설명서에 대한 상세 이미지들이 gif형식으로 QDialog위젯 상에 나오게 된다. 이미지 파일이 추가 버튼을 클릭하게 되면, 왼쪽 화면에서 draw할 수 있는 위젯을 파일 이미지로 보여주기 위해 stackedWidget을 이용해서 drawWidget과 이미지 파일을 보여주는 label widget 간에 stack index를 switch해서 변경하는 방식으로 처리하였다. 이미지 파일을 넣게 되면, subprocess함수를 통해 main함수가 아닌 predict.py shell script를 호출해서 첨부한 이미지에 대한 캐릭터를 생성하도록 했다. predict를 실행하는 동안에는 gui_loading파일의 Overlay함수를 호출해서 로딩 화면이 나오게 했다. predict.py를 통해 생성된 16개의 이미지는 오른쪽 화면에서 GridLayout위젯에서 4x4형식으로 나타나게 하였다.
- **gui_loading.py** : 이미지 파일을 넣을때 로딩화면에 화면에 중첩되어 나타나게 구현한 코드이다. QTimer를 이용해 5초 가량 indicator들이 애니메이션 효과로 나타났다가 사라지게 된다.

<SRGAN 코드 설명>



- **main.py(srgan)** : evaluate함수에서 해상도를 높이고 이미지 저장하는 부분만 사용. iGAN에 의해서 캐릭터가 새로 생성되었을 때 바로 고해상도 작업을 실행시킬 조건문과 반복문을 설정.
- **config.py(srgan)** : 학습된 모델이 적용될 이미지 디렉토리를 iGAN과 공유하는 공유하는 폴더를 재설정. 학습 Parameter는 그대로 사용


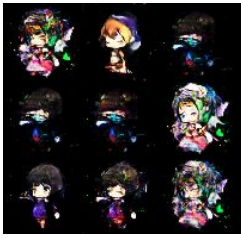

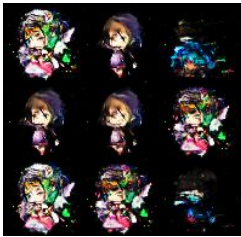
v. 성능지표


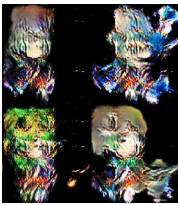
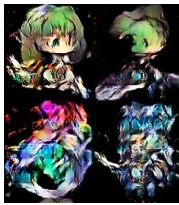
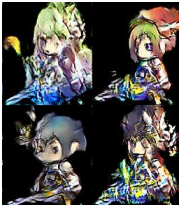
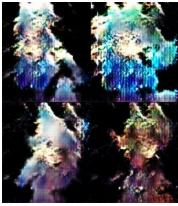

특별한 성능지표 없이 눈으로 보고 사용한 dataset의 캐릭터와 비교하여 가장 비슷하다고

판단되어지는 경우 성능이 좋다고 판단한다.

vi. 성능탐색

64x64, 128x128 캐릭터 Image에 대한 Model의 epoch과 Z-vector의 길이의 변화에 따른 결과표이다.

64x64모델	N_Z 100	N_Z 200
epoch 50		 mode collapse
epoch 100	 noise	 noise + mode collapse

128x128모델	N_Z 100	N_Z 200	N_Z 400
epoch 50			
epoch 100			

vii. 최고성능 및 성능 결과 분석

Original 캐릭터 dataset과 비교하여 64x64 모델 학습시 Z-vector의 길이가 100이고 50 epoch일 때 가장 성능이 좋았다.

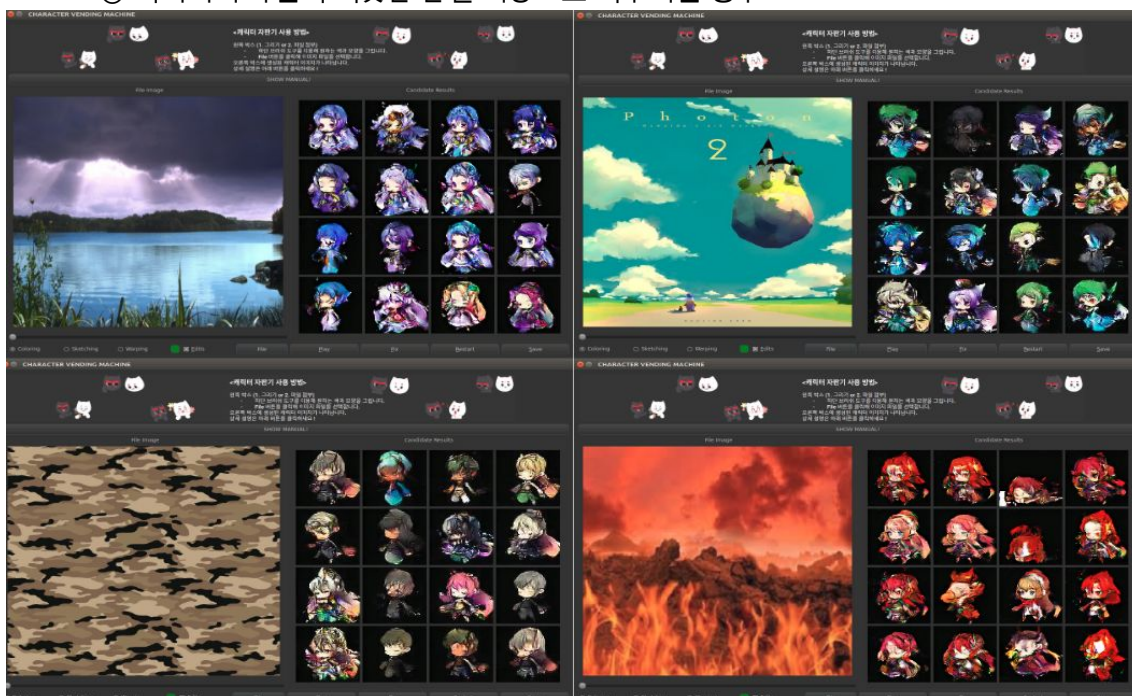
64x64모델	N_Z 100
epoch 50	

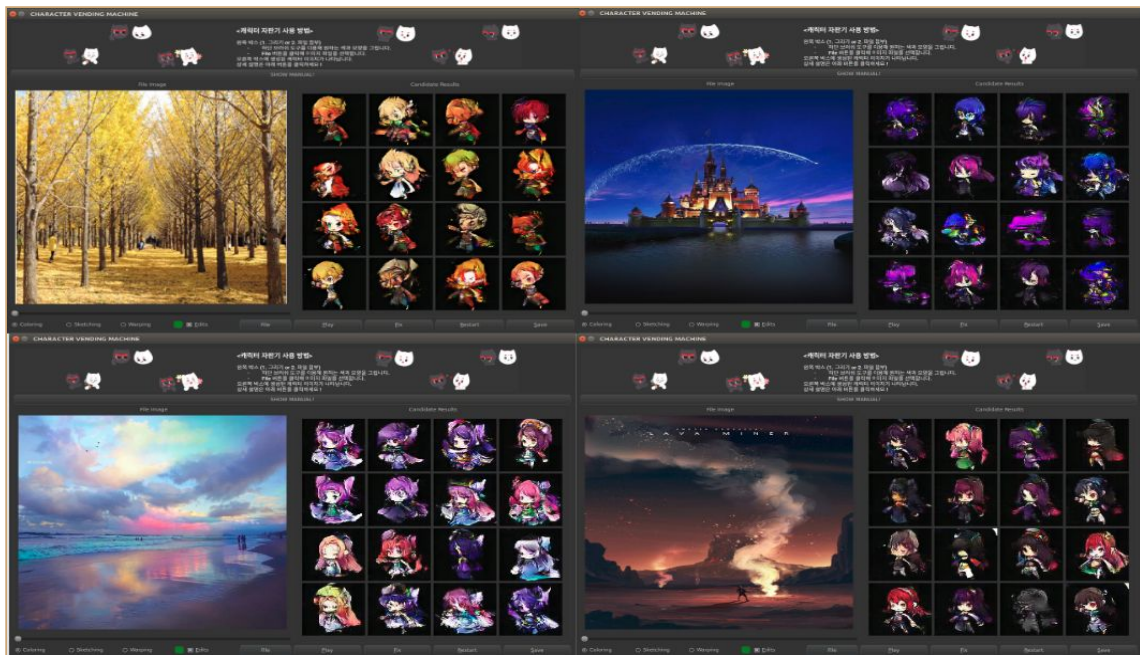
viii. 케이스 분석

파일을 첨부하거나 그림을 그릴 때, 경우에 따라 잘된 케이스와 잘 안된 케이스로 구분된다.

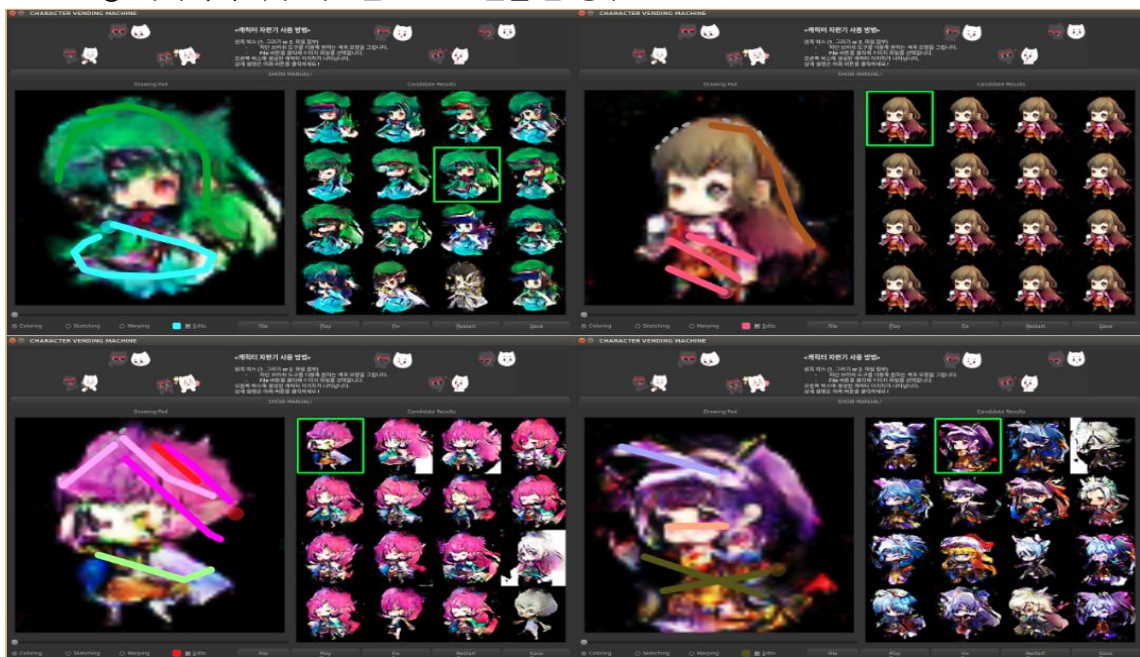
- 잘 된 케이스

① 이미지의 사진이 비슷한 단일 색상으로 이루어진 경우

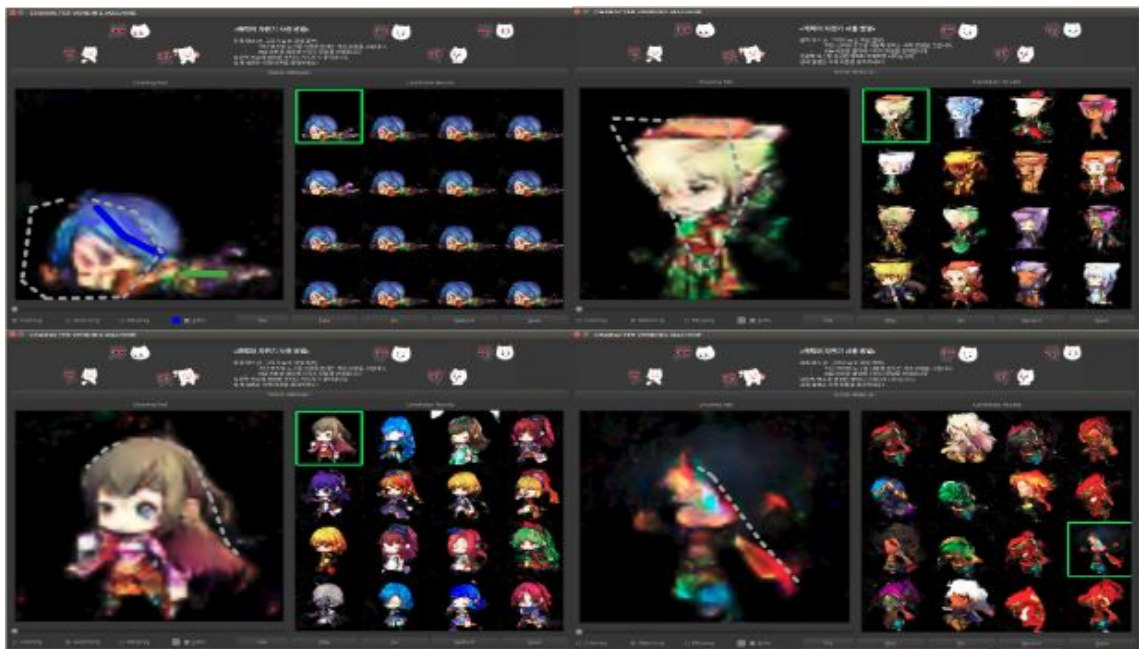




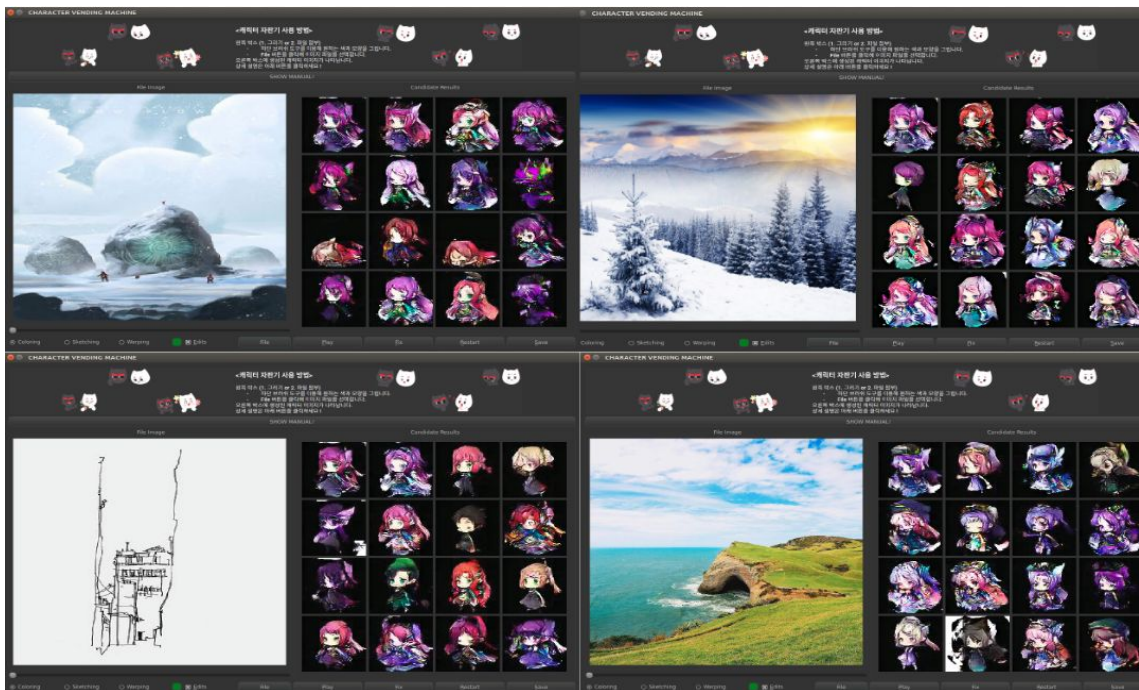
② 캐릭터에 자주 나오는 sketch 선을 딴 경우



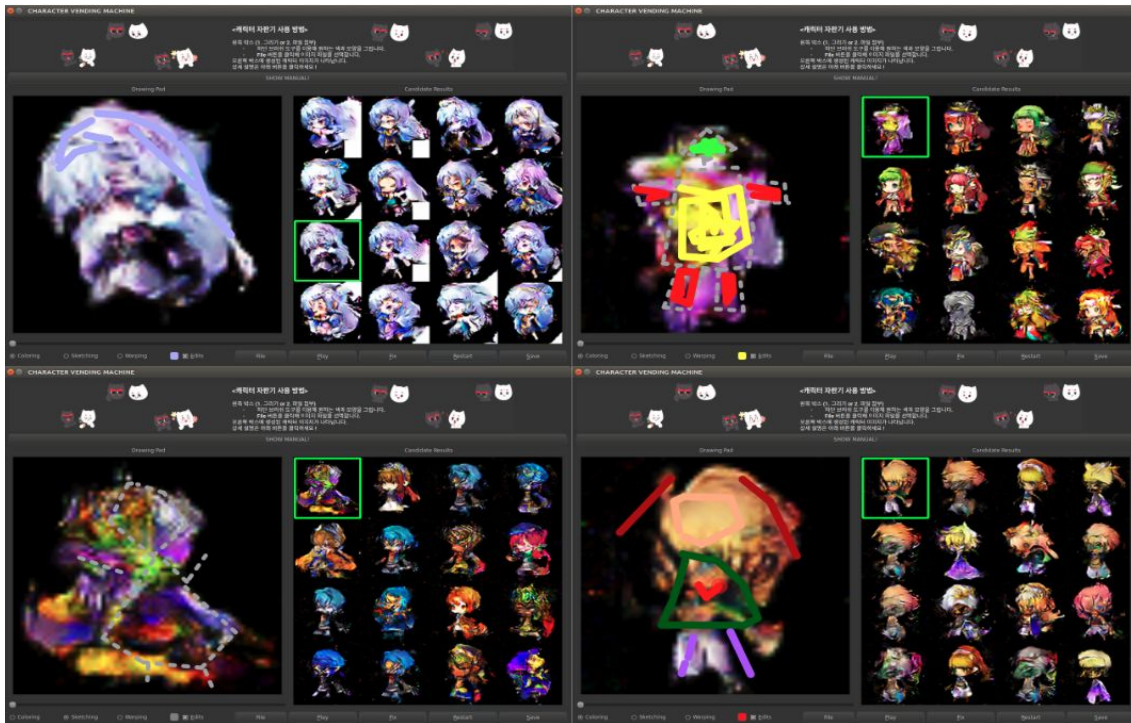
③ coloring 색상이 제한적인 경우



- 잘 안된 케이스
① 이미지에 하얀 색상이 많이 들어간 경우



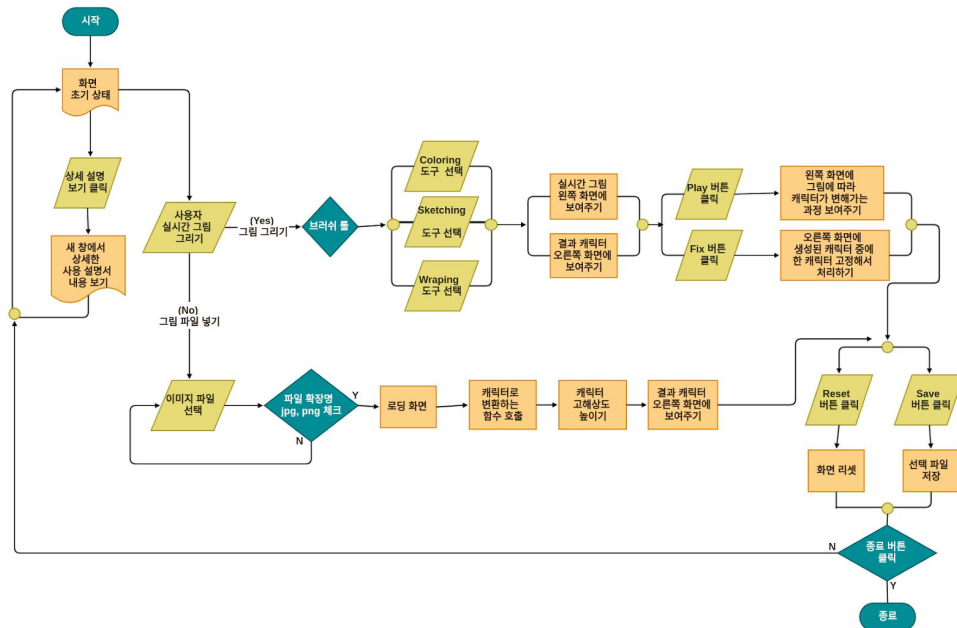
- ② coloring이 많은 경우, ③ sketching에서 캐릭터에 없는 모양을 그린 경우



3. 프로젝트 데모

데모
시나리
오

https://drive.google.com/open?id=1-8PXL9XI_h_PPxAbj0bulhsuef9j1H-
(데모 시나리오 이미지 구글 드라이브 주소)

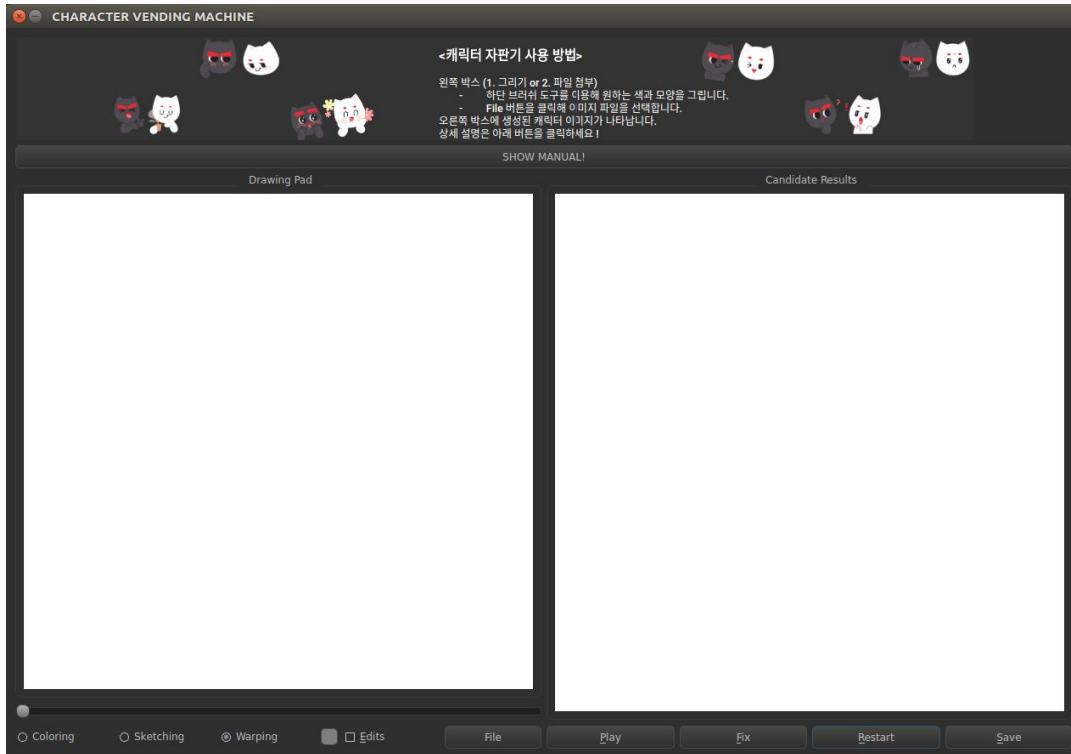


데모
동영상
주소

<https://drive.google.com/open?id=1a5JWn8s8nqNrFxZsQmrnCtUxhD7Gxmvd>

데모 시연

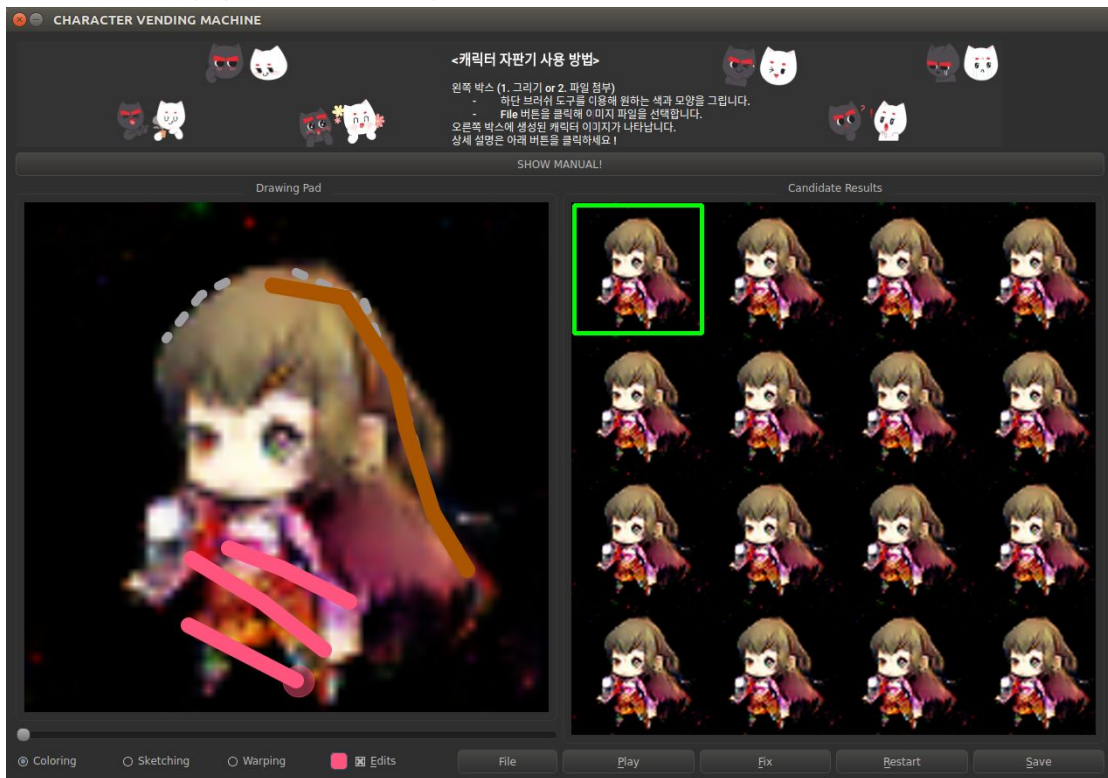
1. 초기화면



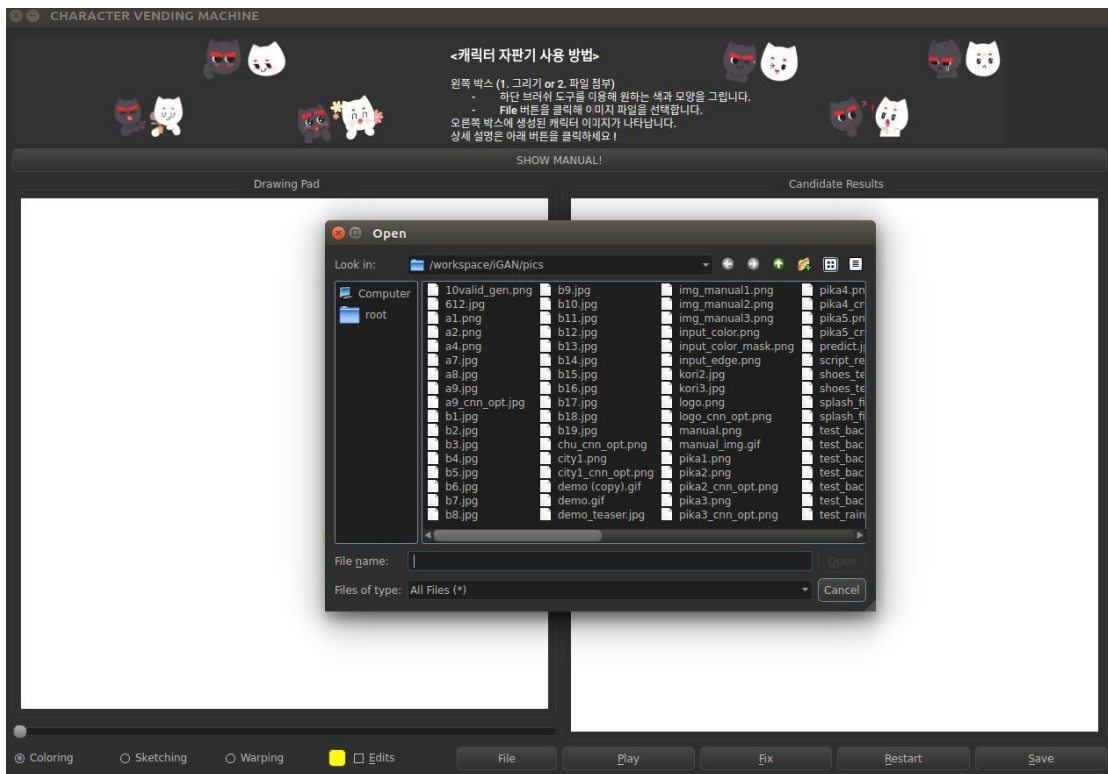
2. 사용 설명서 클릭시 새창에서 사용 설명서(.gif) 화면 나옴



3. 사용자 입력에 따른 16개의 캐릭터 생성



4. 이미지 파일 첨부시



5. 왼쪽 이미지 그림 첨부 시, 오른쪽 화면에 16개의 캐릭터 생성



4. 프로젝트 실제 추진일정

세부과제	1주차							2주차							3주차							비고
논문 분석	■	■	■																			
개발 환경 설정		■	■																			
모델 탐색		■	■	■	■																	
코드 및 모델 분석			■	■	■	■	■	■	■	■	■											
UI 개발					■	■	■	■	■	■	■	■										
학습 데이터 - UI 연동 개발											■	■	■	■	■	■	■					
모델 수정 및 성능개선												■	■	■	■	■	■	■				
해상도 높이기													■	■	■	■	■	■				
테스트 및 마무리 작업																				■		

5. 팀원 및 역할분담

구분	이름	담당분야	역할	개인성장
팀장	오영진	모델 학습	<ul style="list-style-type: none"> - 논문 분석 - 코드 분석 및 수정 - Data 전처리 - DCGAN모델 학습 - SRGAN모델 학습 	<p>데이터를 수집하고 전처리하는 단계부터 시작하여 시모델을 적용하고 서비스를 구현하기 까지 다양한 경험을 할 수 있었다. 같은 모델에서 동일한 크기의 이미지를 넣었을 때 이미지의 특성에 따라 적합한 파라미터값이 달랐다. AI분야는 기초적인 지식과 코딩 능력뿐만 아니라 다양한 시도를 통해 많은 실패와 성공을 겪으면서 쌓은 경험또한 중요한 역량이라고 생각했다. 또한, 특정 논문을 토대로 아이디어를 구상할 때는 관련논문을 충분히 탐구하고 분석하여야 최신의 기술을 활용하여 높은 수준의 성능을 낼 수 있다는 것을 직접적으로 경험할 수 있었음.</p>
팀원	고민주	UI 개발	<ul style="list-style-type: none"> - 논문 분석 - UI 코드 분석/수정 - UI와 학습 모델 연동 	<p>논문을 토대로 아이디어를 구상하고, 학습 모델을 탐색하고 학습하는 과정을</p>

- Test 및 디버깅
- 프로그램 Architecture
- 결과 케이스 분석

3주동안 하면서 시가 적용되는 과정들을 체감할 수 있었다. 그 전에 논문 세미나에서는 코드의 단편적인 부분만 보았는데, 이번 프로젝트에서는 많은 코드들을 분석하면서 전체적인 구조를 파악할 수 있었다. pyqt를 처음 사용해봤는데 버전에 따른 참고 예제 코드가 많이 없어서 자료를 찾기가 힘들었다. 테스트 및 결과 산출을 위해 대략 50장의 이미지를 넣어보고 여러번 스케치를 그렸을때, 생각한 것보다 잘 나온 캐릭터도 있었고 예상하지 못한 결과가 나온 경우도 있었는데 캐릭터 학습 데이터가 더 많았다면 더 좋은 결과를 얻을 수 있을것 같아 아쉬웠다. 논문/코드 분석을 주로 하여 AI 모델의 구조를 파악하고 학습 parameter와 과정을 자세하게 들여다 볼 수 있었다. 여러 논문을 읽으면서 학습과정에 대한 설명이나 찾고자하는 내용을 찾아가는 요령을 얻게 되었고, 처음엔 이해하기 어렵고 힘들었던 코드도 많은 시간동안 분석도 해보고 구현도 해보면서 이해도가 높아졌다. 그리고 코드를 구현할 때 생기는 오류들을 직접 해결할 때 검색도 하고 팀원들과 같이 해결해나가면서 문제 해결 능력을 키울 수 있었다. 또한, 원하는 기능과 기술을 추가하고자 직접 코드를 수정하고 구현하면서 코딩 능력을 키우는 경험이었다. 역할 분담이 잘 이루어졌지만, 모델과 코드 분석 부분이 아닌 UI 부분은 많이 못 다뤄봐서 아쉽다.

팀원	오현지	모델, 코드 분석	<ul style="list-style-type: none"> - 논문 분석 - iGAN 코드 분석/수정 - hyperparameter 정리 - 학습 모델 Architecture 구조 파악 및 정리
----	-----	-----------	--

6. 레퍼런스

항목	주소
iGAN (논문)	Jun-Yan et."Generative Visual Manipulation on the Natural Image Manifold", in European Conference on Computer Vision (ECCV). 2016. (https://arxiv.org/pdf/1609.03552.pdf)
iGAN (코드)	https://github.com/junyanz/iGAN
iGAN (관련참고 영상)	https://www.youtube.com/watch?v=9c4z6YsBGQ0
DC-GAN (논문)	RADFORD, Alec; METZ, Luke; CHINTALA, Soumith. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434,2015. (https://arxiv.org/abs/1511.06434)
DC-GAN (코드)	https://github.com/SeitaroShinagawa/DCGAN-chainer
HOG (논문)	DALAL, Navneet; TRIGGS, Bill. Histograms of oriented gradients for human detection. In: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. IEEE, 2005. p. 886-893. (https://lear.inrialpes.fr/pubs/2005/DT05/hog_cvpr2005.pdf)
SR-CNN (논문)	DONG, Chao, et al. "Image super-resolution using deep convolutional networks." IEEE transactions on pattern analysis and machine intelligence, 2016, 38.2: 295-307.
SR-CNN (코드)	https://github.com/tegg89/SRCNN-Tensorflow
SR-GAN (논문)	DONG, Chao, et al. Image super-resolution using deep convolutional networks. IEEE transactions on pattern analysis and machine intelligence, 2016, 38.2: 295-307. https://arxiv.org/abs/1501.00092v3
SR-GAN (코드)	https://github.com/tegg89/SRCNN-Tensorflow
Progressive GAN (논문)	KARRAS, Tero, et al. Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196, 2017. (https://arxiv.org/abs/1710.10196)
Progressive GAN (코드)	https://github.com/tkarras/progressive_growing_of_gans
게임 캐릭터 데이터	http://yurudora.com/tkool/
HDF5 file 만들고 사용하기	http://machinelearningguru.com/deep_learning/data_preparation/hdf5/hdf5.html
UI 로딩화면	https://wiki.python.org/moin/PyQt/A%20full%20widget%20waiting%20indicator

pyqt layout	https://www.tutorialspoint.com/pyqt/pyqt_layout_management.htm
pyqt stackedWidget	http://pyqt.sourceforge.net/Docs/PyQt4/qstackedwidget.html
pyqt4 예제코드	http://nullege.com/codes/search/PyQt4