2022 YBIGTA 신입기수 프로젝트 최종발표

다양한 GAN 기반 이미지 생성모델

6조 김동현 | 이주훈 | 이학민 | 임현정

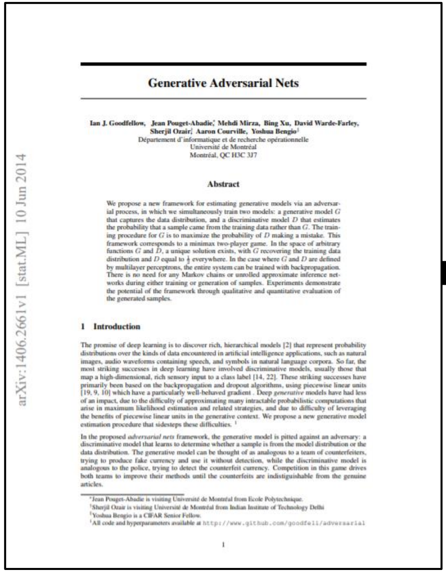
INDEX

1. Introduction

1. Introduction

Topic : 다양한 GAN 모델 기반으로 원하는 이미지를 생성하는 프로젝트

1. 함께 기본적인 Vanilla GAN 모델 논문 리딩 및 개념 스터디
2. 각자 원하는 Application을 위한 GAN 모델 논문 리딩 후 발제 진행
3. 각 모델에 대한 pretrained model weight 이용하여 test 진행



arXiv:1406.2661v1 [stat.ML] 10 Jun 2014

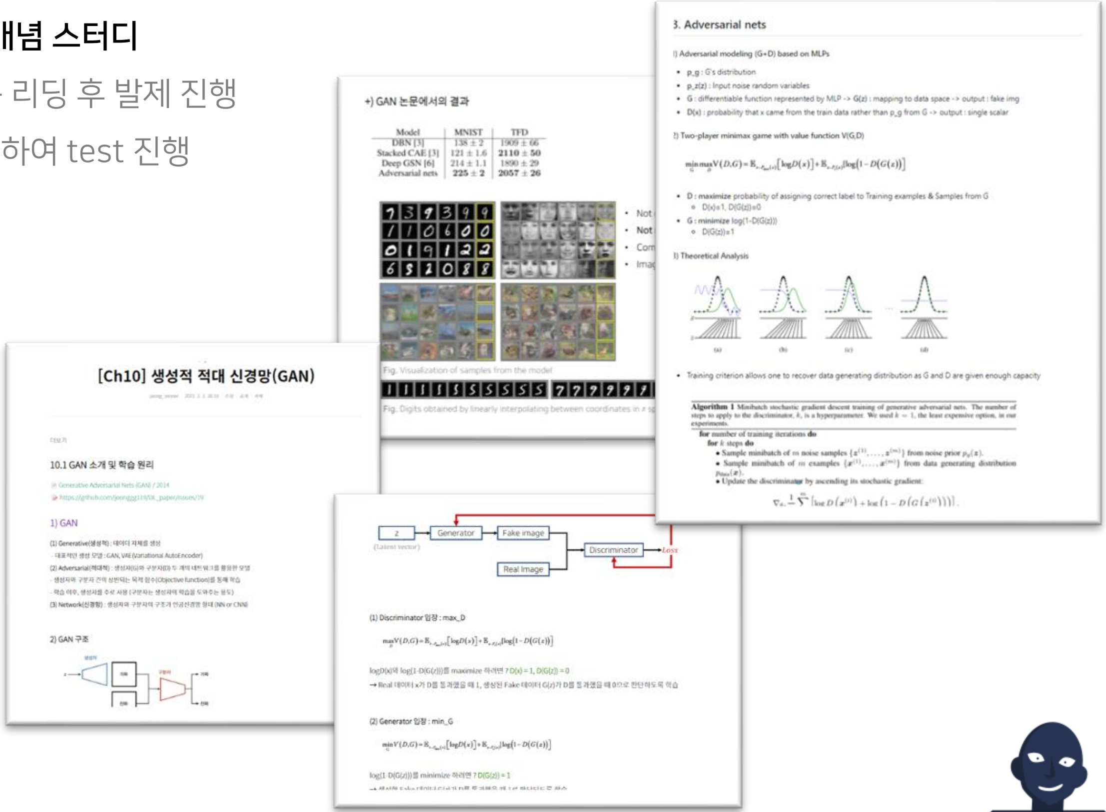
Generative Adversarial Nets

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

Abstract

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

1 Introduction

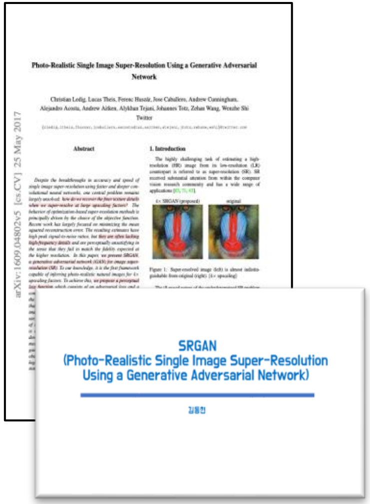


1. Introduction

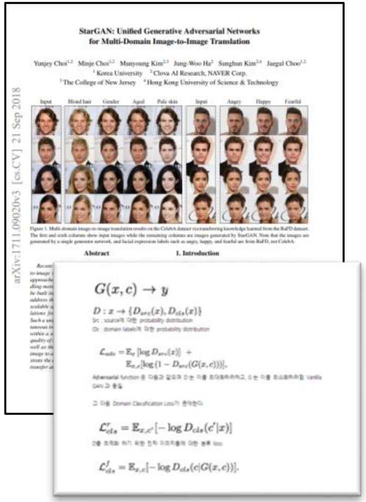
Topic : 다양한 GAN 모델 기반으로 원하는 이미지를 생성하는 프로젝트

1. 함께 기본적인 Vanilla GAN 모델 논문 리딩 및 개념 스터디
2. 각자 원하는 Application을 위한 GAN 모델 논문 리딩 후 발제 진행
3. 각 모델에 대한 pretrained model weight 이용하여 test 진행

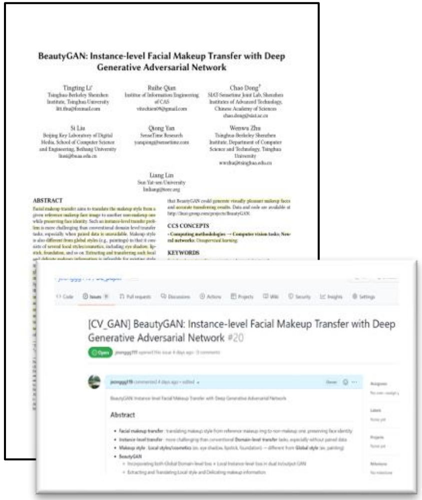
SRGAN



StarGAN



BeautyGAN



StyleGAN

2. GAN models

1) Vanilla GAN
2) SRGAN
3) StarGAN
4) BeautyGAN
5) StyleGAN

Vanilla GAN

[2014 NIPS] Generative Adversarial Nets

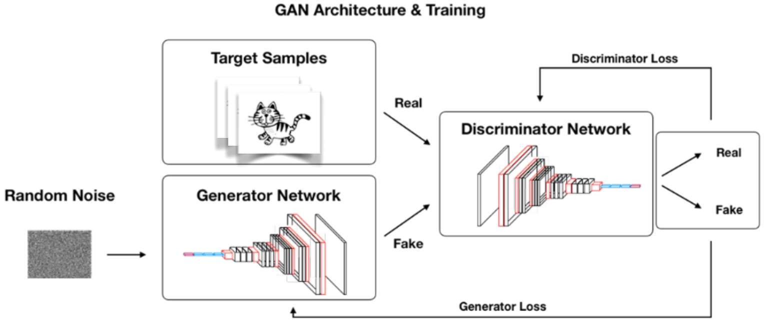
$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_{data}(x)}\left[\log D(x)\right] + \mathbb{E}_{z\sim P_z(z)}\left[\log\left(1 - D(G(z))\right)\right]$$

Discriminator

learns to distinguish real from fake

Generator

learns to make fakes that look real



Model	MNIST	TFD
DBN [3]	138 ± 2	1909 ± 66
Stacked CAE [3]	121 ± 1.6	**2110 ± 50**
Deep GSN [6]	214 ± 1.1	1890 ± 29
Adversarial nets	**225 ± 2**	**2057 ± 26**

Discriminator (real=1, fake=0)

High value for real data
= max $\log D(x)$ where $x \sim p_{data}$

∴ maximize expected value
over all data.
$E_{x\sim p_{data}(x)}[\log D(x)]$

Low value for fake data
= max $\log(1 - D(G(z)))$ where $z \sim p_z$

∴ maximize expected value
over all fake images.
$E_{z\sim p_z(z)}[\log(1 - D(G(z)))]$

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

Generator (real=1, fake=0)

High value for fake data
= min $\log(1 - D(G(z)))$ where $z \sim p_z$

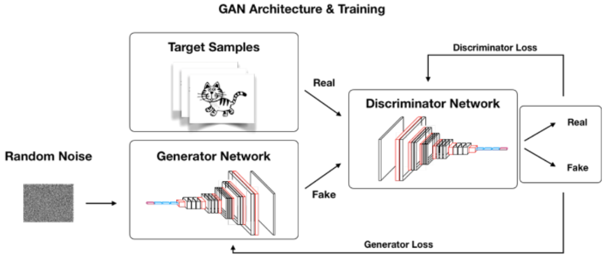
∴ maximize expected value
over all fake images.
$E_{z\sim p_z(z)}[\log(1 - D(G(z)))]$

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

Vanilla GAN

Discriminator
learns to distinguish real from fake

Generator
learns to make fakes that look real



$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim P_{data}(x)}\left[\log D(x)\right] + \mathbb{E}_{z \sim P_z(z)}\left[\log\left(1 - D(G(z))\right)\right]$$

```python
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator,self).__init__()
        self.layer1 = nn.Sequential(OrderedDict([
                        ('fc1',nn.Linear(784,middle_size)),
                        ('bn1',nn.BatchNorm1d(middle_size)),
                        ('act1',nn.LeakyReLU()),

        ]))
        self.layer2 = nn.Sequential(OrderedDict([
                        ('fc2', nn.Linear(middle_size,1)),
                        ('bn2', nn.BatchNorm1d(1)),
                        ('act2', nn.Sigmoid()),
        ]))

    def forward(self,x):
        out = x.view(batch_size, -1)
        out = self.layer1(out)
        out = self.layer2(out)
        return out
```

```python
class Generator(nn.Module):
    def __init__(self):
        super(Generator,self).__init__()
        self.layer1 = nn.Sequential(OrderedDict([
                        ('fc1',nn.Linear(z_size,middle_size)),
                        ('bn1',nn.BatchNorm1d(middle_size)),
                        ('act1',nn.ReLU()),
        ]))
        self.layer2 = nn.Sequential(OrderedDict([
                        ('fc2', nn.Linear(middle_size,784)),
                        ('bn2', nn.BatchNorm1d(784)),
                        ('tanh', nn.Tanh()),
        ]))

    def forward(self,z):
        out = self.layer1(z)
        out = self.layer2(out)
        out = out.view(batch_size,1,28,28)
        return out
```

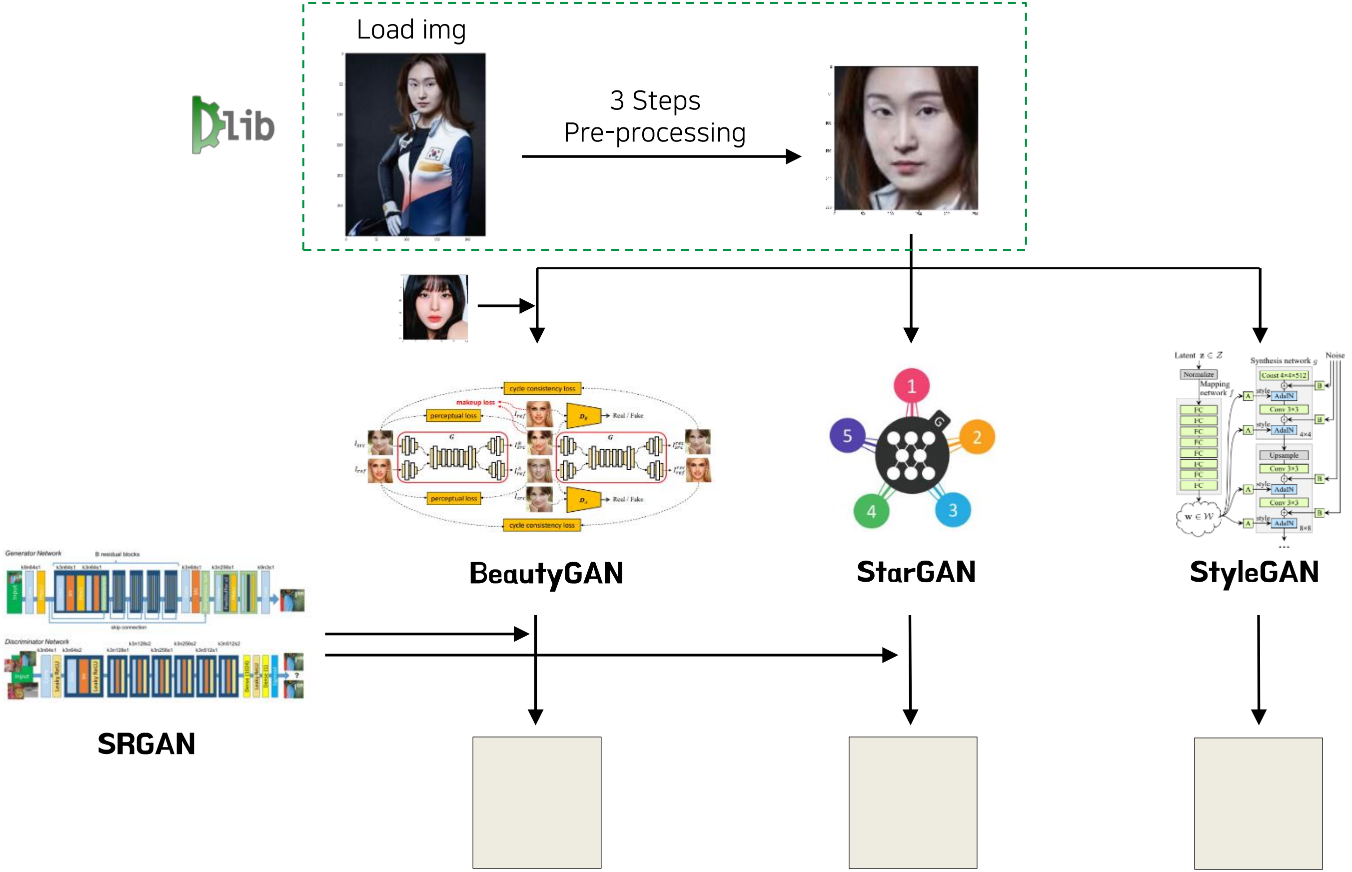
```python
for i in range(epoch):
    for j,(image,label) in enumerate(train_loader):
        image = image.to(device)

        # Discriminator 학습
        dis_optim.zero_grad()

        # Fake Data
        # 랜덤 노이즈 z를 Sampling
        z = init.normal_(torch.Tensor(batch_size,z_size),mean=0,std=0.1).to(device)
        gen_fake = generator.forward(z)
        dis_fake = discriminator.forward(gen_fake)

        # Real Data
        dis_real = discriminator.forward(image)

        # 두 Loss 더해 최종 Loss에 대해 기울기 계산
        dis_loss = torch.sum(loss_func(dis_fake,zeros_label)) + torch.sum(loss_func(dis_real,ones_label))
        dis_loss.backward(retain_graph=True)
        dis_optim.step()

        # Generator 학습
        gen_optim.zero_grad()

        # Fake Data
        # 랜덤 노이즈 z를 Sampling
        z = init.normal_(torch.Tensor(batch_size,z_size),mean=0,std=0.1).to(device)
        gen_fake = generator.forward(z)
        dis_fake = discriminator.forward(gen_fake)

        # 최종 Loss에 대해 기울기 계산
        gen_loss = torch.sum(loss_func(dis_fake,ones_label)) # fake classified as real
        gen_loss.backward()
        gen_optim.step()
```

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

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

3. Framework

3. Framework



Load img
3 Steps
Pre-processing
BeautyGAN
StarGAN
StyleGAN
SRGAN

4. Demonstration

1) Pre-processing

2) GAN Applications by 4 models

3) Results

4. Demonstration

What we want : 다양한 스타일의 올림픽 선수들 캐릭터 생성

최민정 황대헌 유영 차준환

4 GAN Applications

- BeautyGAN : Makeup style 선택
- StarGAN : Hair color, Gender, Age 변경
- StyleGAN : Webtoon 화풍 선택
- SRGAN : 고화질화 (마무리)

4. Demonstration

1) Pre-processing

다양한 Input에 대하여 모두 적용 가능하도록 dlib 라이브러리와 pretrained detector 모델을 이용하여 Face Alignment 전처리 진행

(1) Face Detection

```python
img_result = img.copy()

detector = dlib.get_frontal_face_detector() # 얼굴 영역 인식 모델 로드
dets = detector(img, 1) # 이미지에서 얼굴 영역 찾기

# 얼굴 영역의 개수가 0일 경우
if len(dets) == 0:
    print('cannot find faces!')

fig, ax = plt.subplots(1, figsize=(16, 10))

# 얼굴 찾으면 -> det : rectangle object
for det in dets:
    x, y, w, h = det.left(), det.top(), det.width(), det.height()
    rect = patches.Rectangle((x, y), w, h, linewidth=2, edgecolor='r', facecolor='none')
    ax.add_patch(rect) # 그래프 영역에 패치 추가

ax.imshow(img_result)
```

(2) Landmark Detection

```python
# 5점 랜드마크 찾는 모델 로드
sp = dlib.shape_predictor('/content/drive/MyDrive/models/shape_predictor_5_face_landmarks.dat')
```

```python
fig, ax = plt.subplots(1, figsize=(16, 10))

objs = dlib.full_object_detections() # 얼굴 수평 맞춰주기 위해 사용할 인스턴스

for detection in dets:
    s = sp(img, detection) # 위에서 로드했던, 얼굴의 랜드마크 찾는 모델에 img와 rectangle 위치 넣어줌
    objs.append(s)

    for point in s.parts(): # 5점이니까 for문 5번 반복
        circle = patches.Circle((point.x, point.y), radius=3, edgecolor='r', facecolor='r')
        ax.add_patch(circle)

ax.imshow(img_result)
```

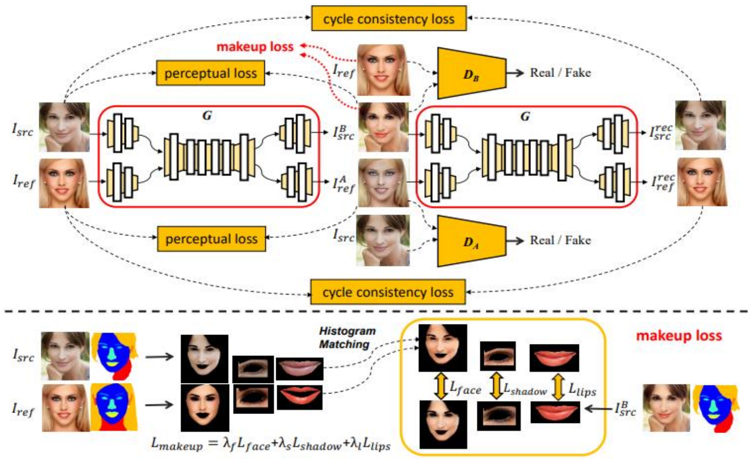
(3) Face Alignment

```python
# 원본 이미지 입력하면 crop 및 align 된 얼굴 이미지 반환

def align_faces(img):

    dets = detector(img, 1) # dectector 이용해서 이미지에서 얼굴 영역 찾기

    objs = dlib.full_object_detections() # 얼굴 수평 맞추기 위해 사용할 모델 인스턴스

    for detection in dets:
        s = sp(img, detection) # 얼굴의 랜드마크 찾는 모델에 img와 rectangle 위치 넣어줌
        objs.append(s)

    # 얼굴을 수평으로 회전해 얼굴 부분만 자른 이미지 반환, padding 작게 주면 타이트하게 자름
    faces = dlib.get_face_chips(img, objs, size=256, padding=0.35)

    return faces
```

```python
# test
test_faces = align_faces(img)

fig, axes = plt.subplots(1, len(test_faces)+1, figsize=(20, 16))
axes[0].imshow(img)

for i, face in enumerate(test_faces):
    axes[i+1].imshow(face)
```

BeautyGAN

[2018 ACM] BeautyGAN: Instance-level Facial Makeup Transfer with Deep Generative Adversarial Network



Perceptual loss $\mathcal{L}_{per} = \frac{1}{C_l \times H_l \times W_l} \sum_{ijk} E_l$

$$E_l = [F_l(I_{src}) - F_l(I_{src}^B)]_{ijk}^2 + [F_l(I_{ref}) - F_l(I_{ref}^A)]_{ijk}^2,$$

Cycle consistency loss $\mathcal{L}_{cyc} = \mathbb{E}_{I_{src}, I_{ref}}[dist(I_{src}^{rec}, I_{src}) + dist(I_{ref}^{rec}, I_{ref})], \quad (8)$

where $(I_{src}^{rec}, I_{ref}^{rec}) = G(G(I_{src}, I_{ref}))$. The distance function $dist(\cdot)$ could be chosen as $L1$ norm, $L2$ norm or other metrics.

Makeup loss $\mathcal{L}_{makeup} = \lambda_l \mathcal{L}_{lips} + \lambda_s \mathcal{L}_{shadow} + \lambda_f \mathcal{L}_{face},$

$$\mathcal{L}_{D_B} = \mathbb{E}_{I_{ref}}[\log D_A(I_{ref})] + \mathbb{E}_{I_{src}, I_{ref}}[\log(1 - D_A(I_{src}^B))].$$

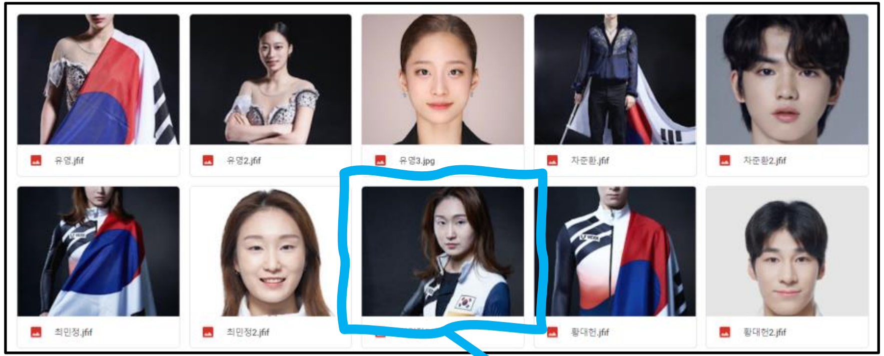
$$\mathcal{L}_{D_A} = \mathbb{E}_{I_{src}}[\log D_A(I_{src})] + \mathbb{E}_{I_{src}, I_{ref}}[\log(1 - D_A(I_{ref}^A))].$$

$$\mathcal{L}_G = \alpha \mathcal{L}_{adv} + \beta \mathcal{L}_{cyc} + \gamma \mathcal{L}_{per} + \mathcal{L}_{makeup},$$
$$\mathcal{L}_{adv} = \mathcal{L}_{D_A} + \mathcal{L}_{D_B}.$$

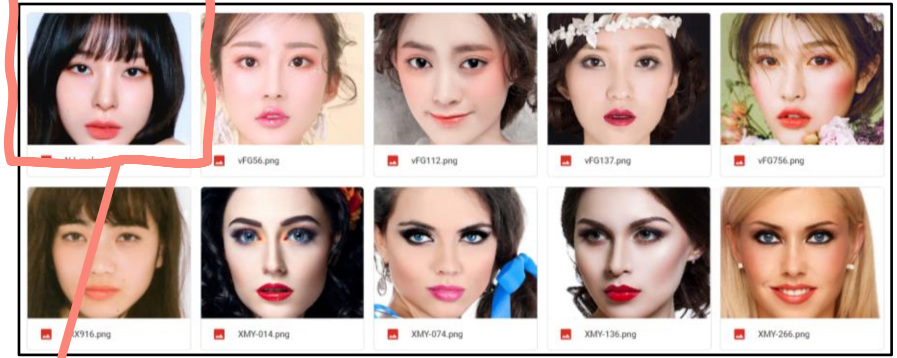
4. Demonstration

2) GAN Applications - BeautyGAN

Reference images



Source images (Makeup)



?

4. Demonstration

2) GAN Applications - BeautyGAN

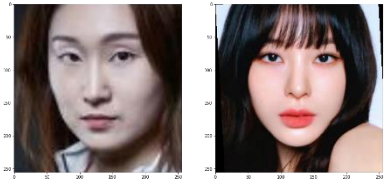
(1) Load Pretrained model

```
#To make tf 2.0 compatible with tf1.0 code, we disable the tf2.0 functionalities
tf.compat.v1.disable_eager_execution()
```

```
with tf.compat.v1.Session() as sess:
    sess.run(tf.compat.v1.global_variables_initializer())

    sess = tf.compat.v1.Session()
    saver = tf.compat.v1.train.import_meta_graph('/content/drive/MyDrive/BeautyGAN-master/models/model.meta')
    saver.restore(sess, tf.compat.v1.train.latest_checkpoint('/content/drive/MyDrive/BeautyGAN-master/models'))
    graph = tf.compat.v1.get_default_graph()

    X = graph.get_tensor_by_name('X:0') # source (no makeup img)
    Y = graph.get_tensor_by_name('Y:0') # reference (makeup img)
    Xs = graph.get_tensor_by_name('generator/xs:0') # output (generator가 만든 img)
```

(2) Load Images & align_faces

```
img1 = dlib.load_rgb_image('/content/drive/MyDrive/BeautyGAN-master/CMJ2.jfif') # source
img1_faces = align_faces(img1)

img2 = dlib.load_rgb_image('/content/drive/MyDrive/BeautyGAN-master/imgs/makeup/NJ_makeup.jpg') # reference
img2_faces = align_faces(img2)

fig, axes = plt.subplots(1, 2, figsize=(16, 10))
axes[0].imshow(img1_faces[0])
axes[1].imshow(img2_faces[0])
```

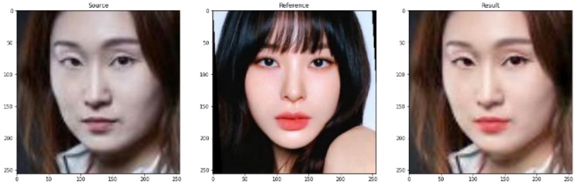


(3) Apply BeautyGAN

```
src_img = img1_faces[0]
ref_img = img2_faces[0]

X_img = preprocess(src_img)                  # array(256,256,3)
X_img = np.expand_dims(X_img, axis=0)        # array(1,256,256,3)

Y_img = preprocess(ref_img)
Y_img = np.expand_dims(Y_img, axis=0)

output = sess.run(Xs, feed_dict={
    X: X_img,
    Y: Y_img
})

output_img = postprocess(output[0])

fig, axes = plt.subplots(1, 3, figsize=(20, 10))
axes[0].set_title('Source')
axes[0].imshow(src_img)
axes[1].set_title('Reference')
axes[1].imshow(ref_img)
axes[2].set_title('Result')
axes[2].imshow(output_img)
```

Additional processing for display

```
# uint8(0~255) -> float32(-1~1)
def preprocess(img):
    return img.astype(np.float32) / 127.5 - 1.

# float32(-1~1) -> uint8(0~255)
def postprocess(img):
    return ((img + 1.) * 127.5).astype(np.uint8)
```

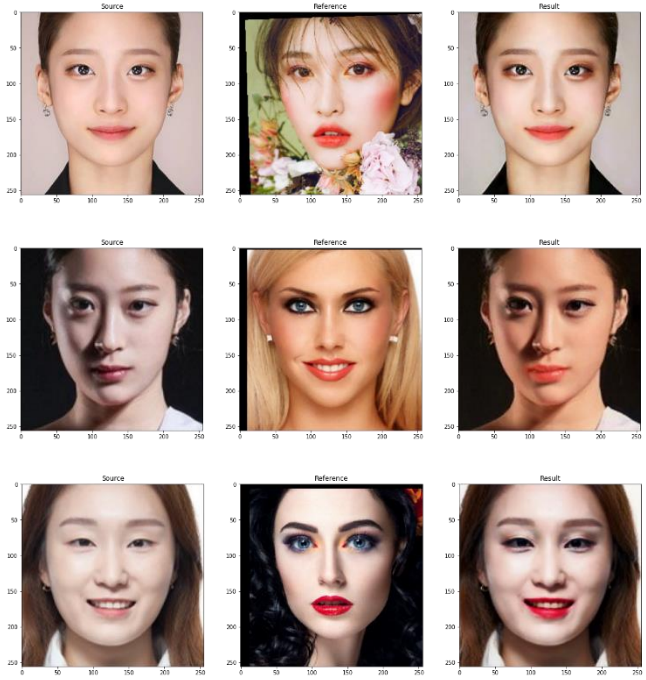


Save output_img for SRGAN

```
from PIL import Image
im = Image.fromarray(output_img)
im.save('/content/BeautyGAN_result.png','png')
```

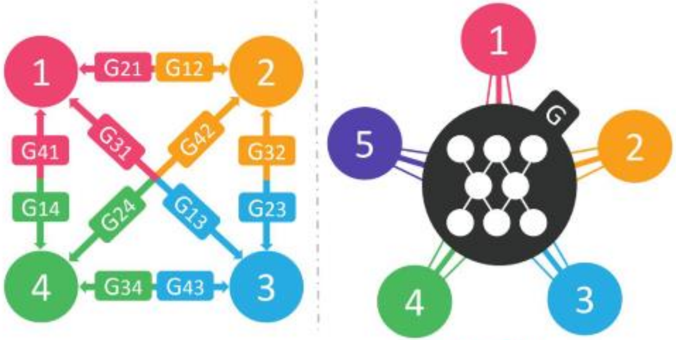
4. Demonstration

2) GAN Applications - BeautyGAN



StarGAN

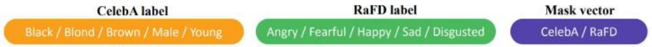
[2018 CVPR] StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation



Cross-domain models

StarGAN

$$\tilde{c} = [c_1, ..., c_n, m]$$



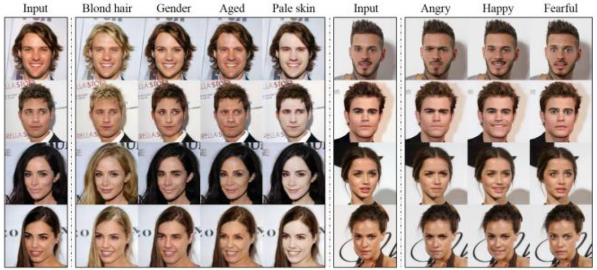
StarGAN: ① Adversarial loss + ② Domain classification loss + ③ Reconstruction loss

Adversarial $$\mathcal{L}_{adv} = \mathbb{E}_x[\log D_{src}(x)] + \mathbb{E}_{x,c}[\log(1 - D_{src}(G(x,c)))]$$

Domain classification
$$\mathcal{L}^f_{cls} = \mathbb{E}_{x,c}[-\log D_{cls}(c|G(x,c))]$$
$$\mathcal{L}^r_{cls} = \mathbb{E}_{x,c'}[-\log D_{cls}(c'|x)]$$

Reconstruction $$\mathcal{L}_{rec} = \mathbb{E}_{x,c,c'}[||x - G(G(x,c),c')||_1]$$

최종 목적 함수
$$\mathcal{L}_D = -\mathcal{L}_{adv} + \lambda_{cls}\,\mathcal{L}^r_{cls}$$
$$\mathcal{L}_G = \mathcal{L}_{adv} + \lambda_{cls}\,\mathcal{L}^f_{cls} + \lambda_{rec}\,\mathcal{L}_{rec}$$



4. Demonstration

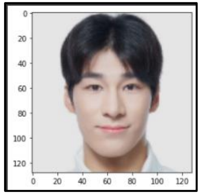
2) GAN Applications - StarGAN

(1) Load Pretrained model

```
# get StarGAN pretrained weights.
!wget https://nnabla.org/pretrained-models/nnabla-examples/GANs/stargan/pretrained_params_on_celebA.h5

# get StarGAN config file.
!wget https://nnabla.org/pretrained-models/nnabla-examples/GANs/stargan/pretrained_conf_on_celebA.json
```

(2) Load Images & align_faces

```
center = [detected_faces[2] - (detected_faces[2] - detected_faces[0]) / 2.0,
          detected_faces[3] - (detected_faces[3] - detected_faces[1]) / 2.0]
#center[1] = center[1] - (detected_faces[3] - detected_faces[1]) * 0.12
scale = (detected_faces[2] - detected_faces[0] + detected_faces[3] - detected_faces[1]) / 195
inp = crop(image, center, scale, resolution=128)
plt.imshow(inp)
```



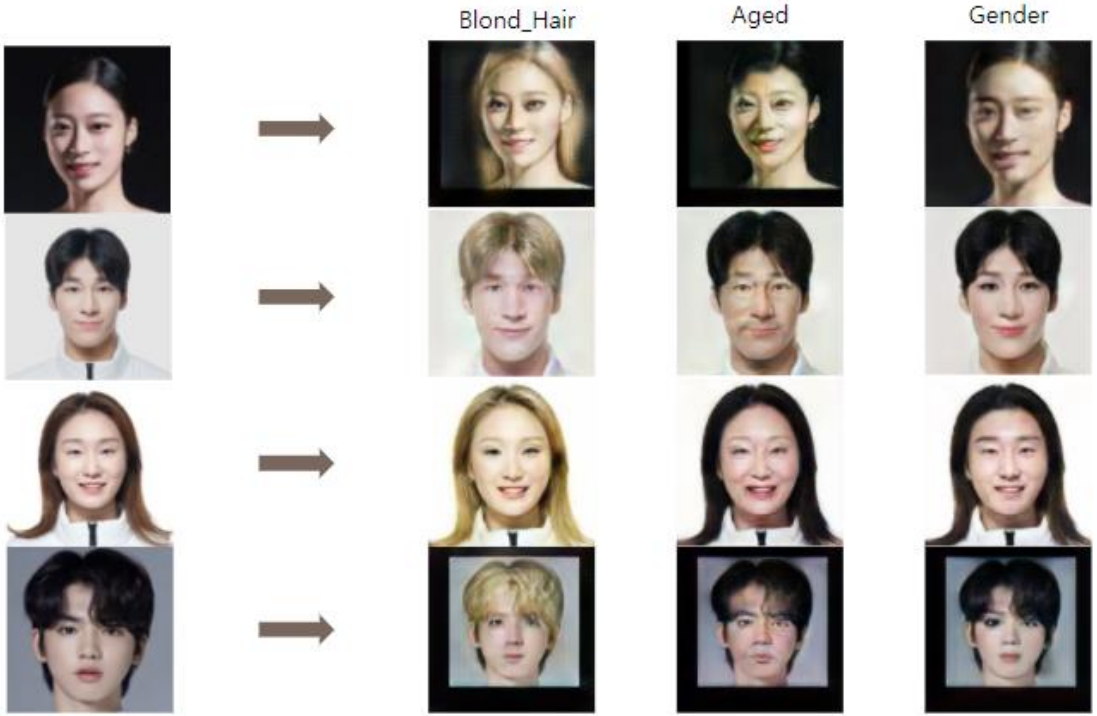
(3) Apply StarGAN

```
!python generate.py --pretrained-params pretrained_params_on_celebA.h5 --config pretrained_conf_on_celebA.json --test-image-path source_img

2022-02-25 03:23:38,373 [nnabla][INFO]: Initializing CPU extension...
2022-02-25 03:23:38,755 [nnabla][INFO]: Initializing CUDA extension...
2022-02-25 03:23:38,761 [nnabla][INFO]: Initializing cuDNN extension...
Learned attributes choice: ['Black_Hair', 'Blond_Hair', 'Brown_Hair', 'Male', 'Young']
Source image: input_image.png
Use 'Black_Hair'?
type yes or no: no
Use 'Blond_Hair'?
type yes or no: yes
Use 'Brown_Hair'?
type yes or no: no
Use 'Male'?
type yes or no: yes
Use 'Young'?
type yes or no: yes
Saved tmp.results/generated_0_Blond_Hair_Male_Young.png.
```

- Hair color (Black / Blond / Brown)
- Gender (Male / Female)
- Age 설정 가능

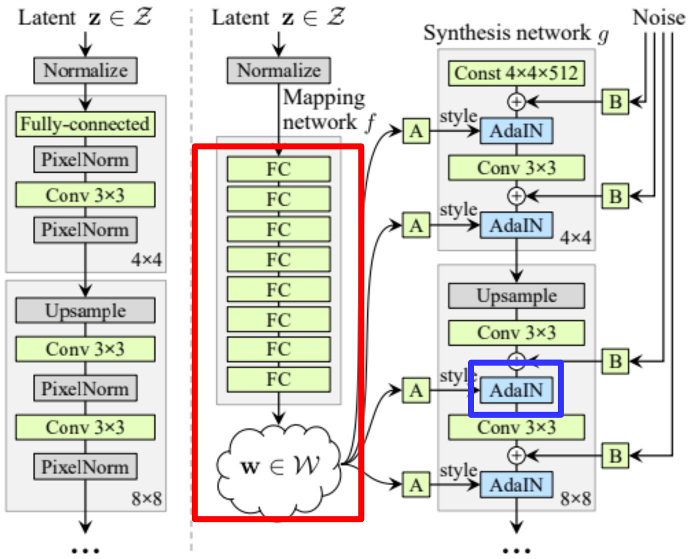
4. Demonstration

2) GAN Applications - StarGAN



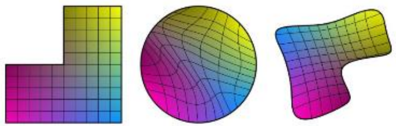
StyleGAN

[2019 CVPR] A Style-Based Generator Architecture for Generative Adversarial Networks



(a) Traditional (b) Style-based generator

① Mapping Network (f : z → w)



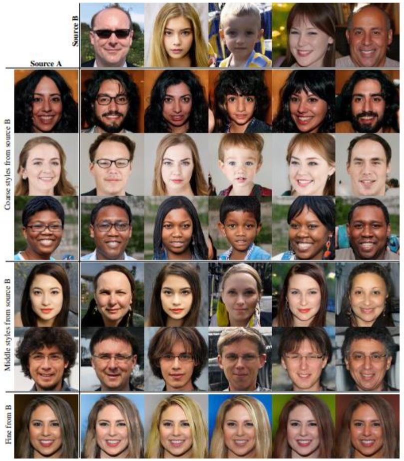
(a) Distribution of features in training set (b) Mapping from $\mathcal{Z}$ to features (c) Mapping from $\mathcal{W}$ to features

② AdaIN (Adaptive Instance Normalization)

$$\mathrm{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i},$$

Better Disentanglement (=More Linear)

→ 다양한 Style features 분리 가능



4. Demonstration

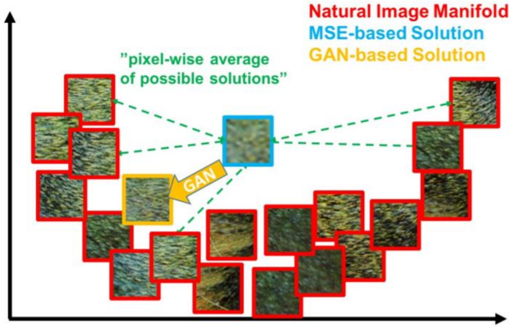
2) GAN Applications - StyleGAN

4. Demonstration

2) GAN Applications - StyleGAN

SRGAN

[2017 ICCV] Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network

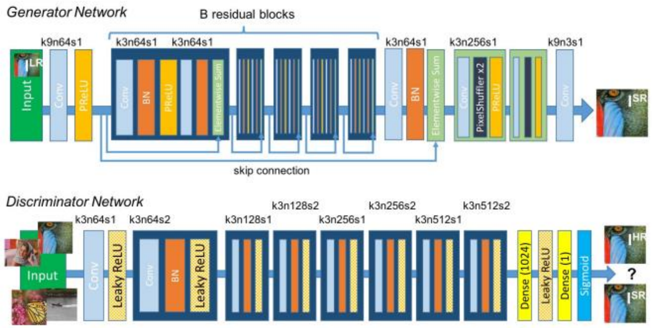


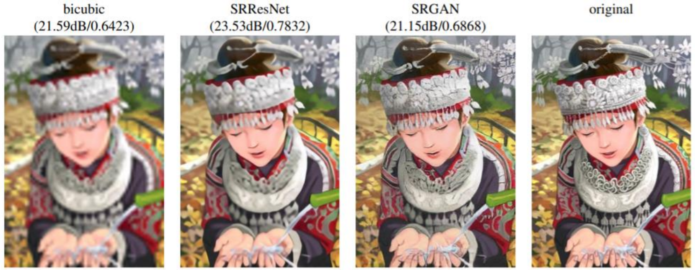
High-Frequency (Finer Texture) Detail 살리기 위해

Perceptual Loss Function 사용

$$l^{SR} = \underbrace{\underbrace{l^{SR}_{X}}_{\text{content loss}} + \underbrace{10^{-3} l^{SR}_{Gen}}_{\text{adversarial loss}}}_{\text{perceptual loss (for VGG based content losses)}}$$

= {MSE Loss + VGG Loss} + Adversarial Loss





4. Demonstration

2) GAN Applications - SRGAN

(1) Image Preprocessing

```python
import cv2
from matplotlib import pyplot as plt

img_path = "/Users/kimdonghyun/Downloads/img/황대헌2.jfif"
img = cv2.imread(img_path)
img = cv2.resize(img, (150, 150)) #사이즈를 작게!


img_without_alpha = img[:,:,:3]

def concat_tile(im_list_2d):
    return cv2.vconcat([cv2.hconcat(im_list_h) for im_list_h in im_list_2d])

final_img = concat_tile([[img, img_without_alpha]])

cv2.imwrite('/Users/kimdonghyun/Downloads/황대헌2_without_alpha.png', img_without_alpha

cv2.imshow('', final_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

(2) Apply Pretrained SRGAN

```python
[6] !python ./eval.py -i configs/srgan.json resources/pretrained/srgan.pth "/content/drive/MyDrive(

[7] import cv2

[8] def concat_tile(im_list_2d):
        return cv2.vconcat([cv2.hconcat(im_list_h) for im_list_h in im_list_2d])

[9] original_img = cv2.imread('/content/drive/MyDrive/Colab Notebooks/YBIGTA_실습용/신입기수 프로젝트/img/
    srgan_super_resolved_img = cv2.imread('/content/drive/MyDrive/Colab Notebooks/YBIGTA_실습용/신입기수

[10] original_img = cv2.resize(original_img, (srgan_super_resolved_img.shape[0], srgan_super_resolvec
     final_img = concat_tile([[or                                  g]])
                              Click to show 3 definitions.
                              ndarray: original_img
                              ndarray with shape (600, 600, 3)

from matplotlib import pyplo
final_img = cv2.cvtColor(fin
original_img = cv2.cvtColor(original_img, cv2.COLOR_BGR2RGB)
srgan_super_resolved_img = cv2.cvtColor(srgan_super_resolved_img, cv2.COLOR_BGR2RGB)
plt.figure(figsize = (20, 20))
plt.imshow(final_img)
plt.show()
```

4. Demonstration

2) GAN Applications - SRGAN

Before

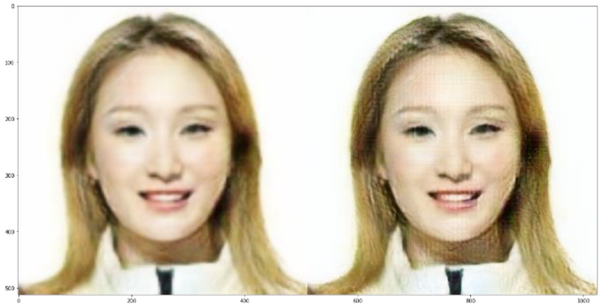
After

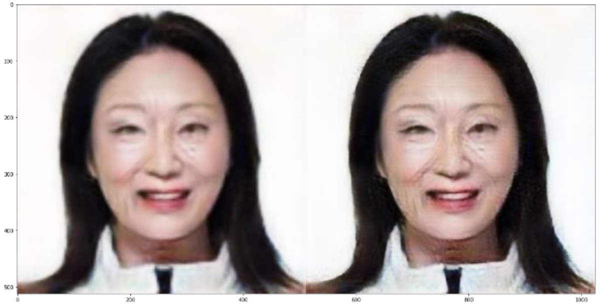
4. Demonstration

2) GAN Applications - SRGAN

StarGAN Output to SRGAN

Before After



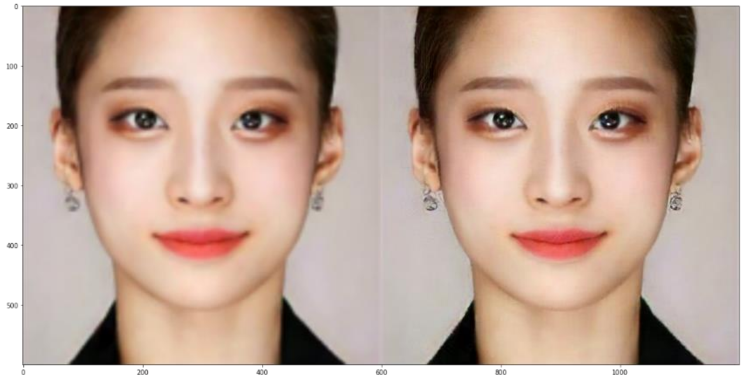


4. Demonstration

2) GAN Applications - SRGAN

BeautyGAN Output to SRGAN

Before After



Before After

5. Conclusion

5. Conclusion

Conclusion

- GAN의 기본 개념 및 알고리즘 + 코드와의 연결
- 여러 Application GAN 모델들에 대한 이론적 지식 습득
- 다양한 스타일의 원하는 이미지 생성 프로젝트 성공적 수행
- 초반 모델들이기에 성능적인 한계에 대한 아쉬움 → Advanced GAN

Future work

- 시도해보고 싶은 자신만의 캐릭터 제작 or 이상형 이미지 생성 등 활용 기대
- 유저 친화성을 위해 웹 또는 앱으로 구현 (End to end)
- Pretrained model 사용하지 않고 직접 GAN 학습
- 다른 Advanced GAN 또는 생성모델의 최근 경향 파악

Reference

[Paper]

https://arxiv.org/pdf/1406.2661.pdf

https://arxiv.org/pdf/1609.04802.pdf

https://arxiv.org/pdf/1711.09020.pdf

https://arxiv.org/abs/1812.04948

https://arxiv.org/abs/1912.04958

http://colalab.org/media/paper/BeautyGAN-camera-ready.pdf

[Code]

kairess/BeautyGAN: transfer the makeup style of a reference face image to a non-makeup face (github.com)

thaoshibe/awesome-makeup-transfer: A curated list of Awesome Makeup Transfer resources (github.com)

https://colab.research.google.com/github/sony/nnabla-examples/blob/master/interactive-demos/stargan.ipynb

https://github.com/bryandlee/naver-webtoon-faces

https://github.com/mseitzer/srgan

Thank you !