Style transfer using cycleGAN

Hyukhun koh, Junbeom Kwon

Objective : Image Translation

Style A : without glasses

Style B : with glasses

Why we need this Application?

- Recommend the best glasses for the user
- Applied to AI applications (included in camera application)
- Used to another domain that needs style transfer

Datasets

1. Kaggle Datasets (unpaired & unclassified datasets)
   - baseline model
   - glass(2144)/ no-glasses(2144)
   - resolution : 1024 * 1024
   - filtered out confusing data
   - Train(2139 each)/Test(5 each)

2. AI Hub

3. Google Image Search (by crawling)

4. celebA

Kaggle




AI Hub

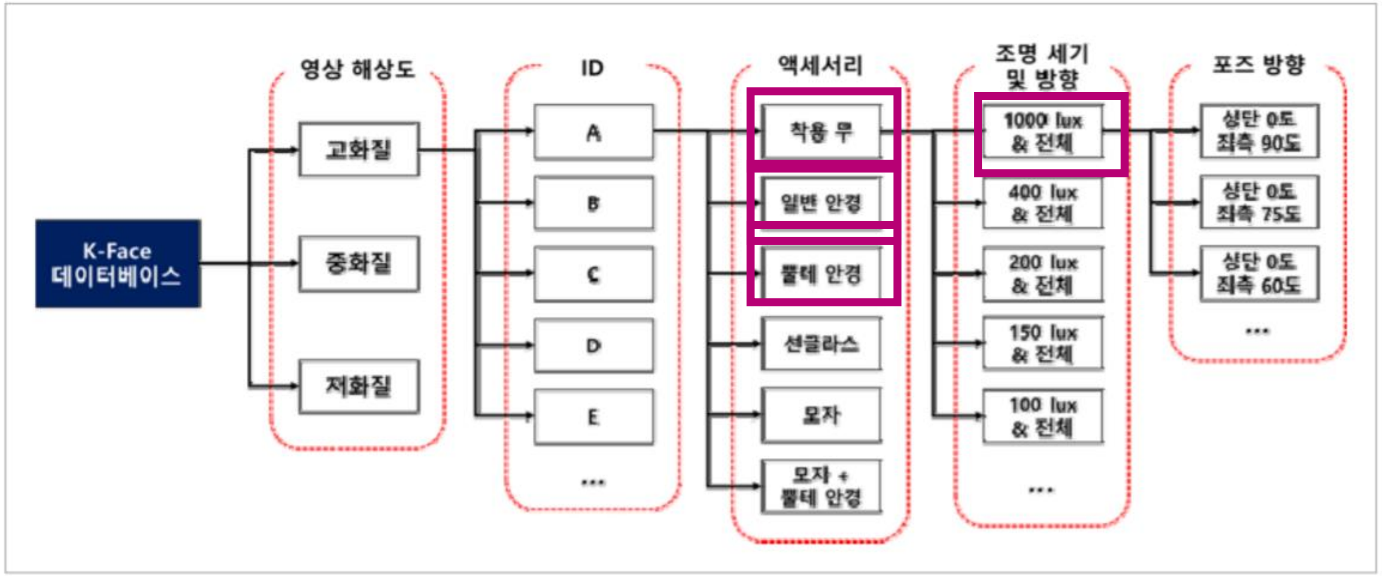


그림6 | 한국인 안면 이미지 데이터셋 구조

Google Image Search

```python
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
import urllib.request

driver = webdriver.Chrome()
driver.get("https://www.google.co.kr/imghp?hl=ko&ogbl")
elem = driver.find_element_by_name("q")
elem.send_keys("연예인 안경테")
elem.send_keys(Keys.RETURN)

SCROLL_PAUSE_TIME = 1

last_height = driver.execute_script("return document.body.scrollHeight")

while True:
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")

    time.sleep(SCROLL_PAUSE_TIME)

    new_height = driver.execute_script("return document.body.scrollHeight")
    if new_height == last_height:
        try:
            driver.find_element_by_css_selector(".mye4qd").click()
        except:
            break
    last_height = new_height
    print(new_height)
images = driver.find_elements_by_css_selector(".rg_i.Q4LuWd")
```

CelebA




Model

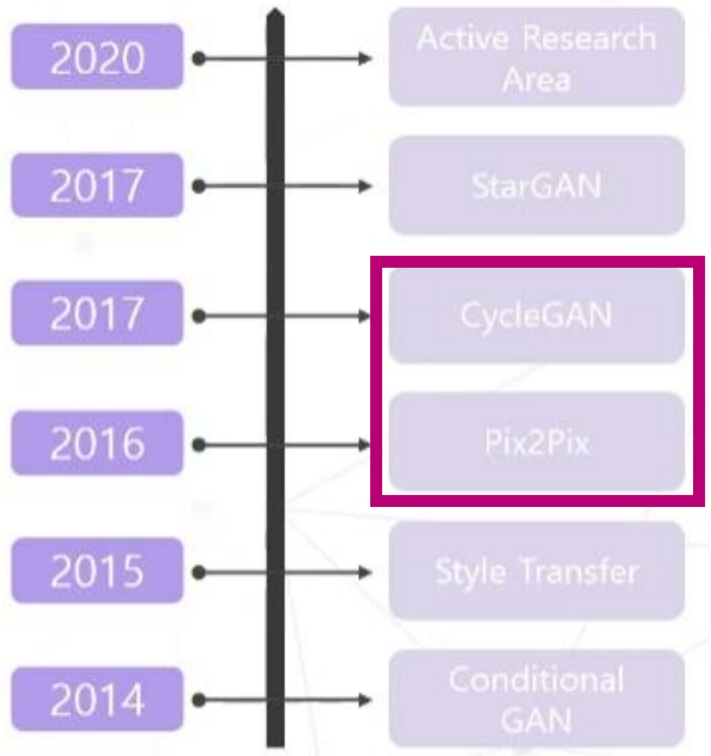


Image-To-Image Translation TimeLine

Useful for domain > 2

Applicable to paired datasets

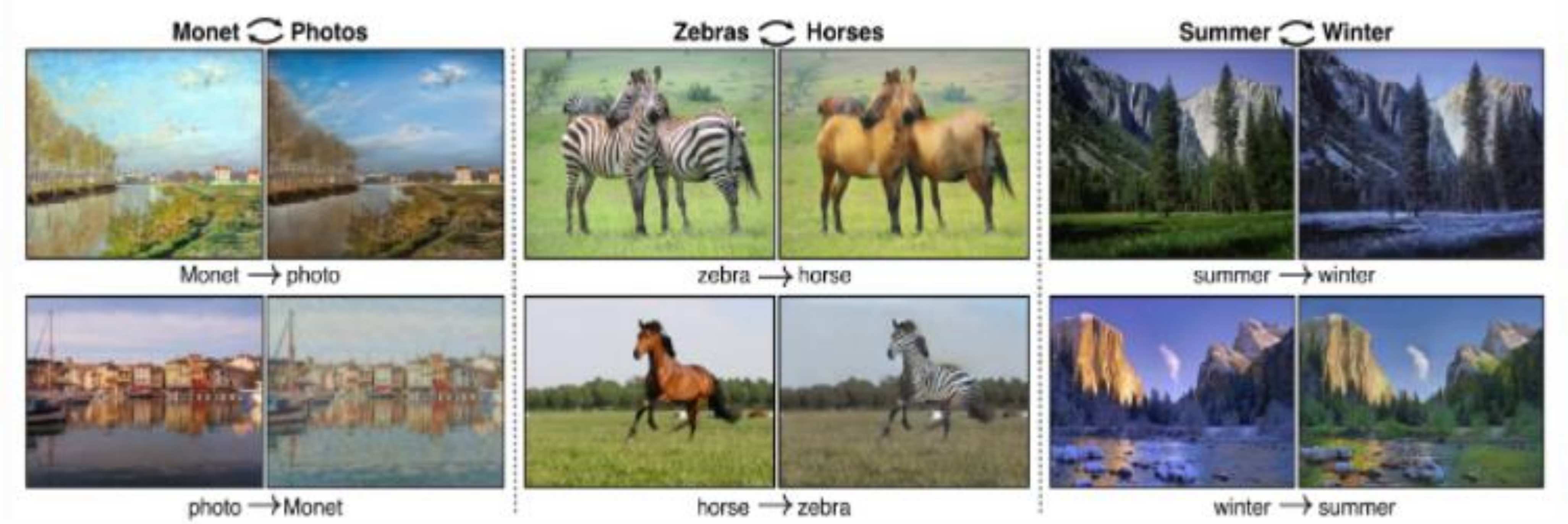
Only Applicable to paired datasets

Model

Pix2Pix CycleGAN

→ harder to collect paired datasets than unpaired datasets

CycleGAN


Monet Photos
Zebras Horses
Summer Winter
Monet → photo
zebra → horse
summer → winter
photo → Monet
horse → zebra
winter → summer

Architecture

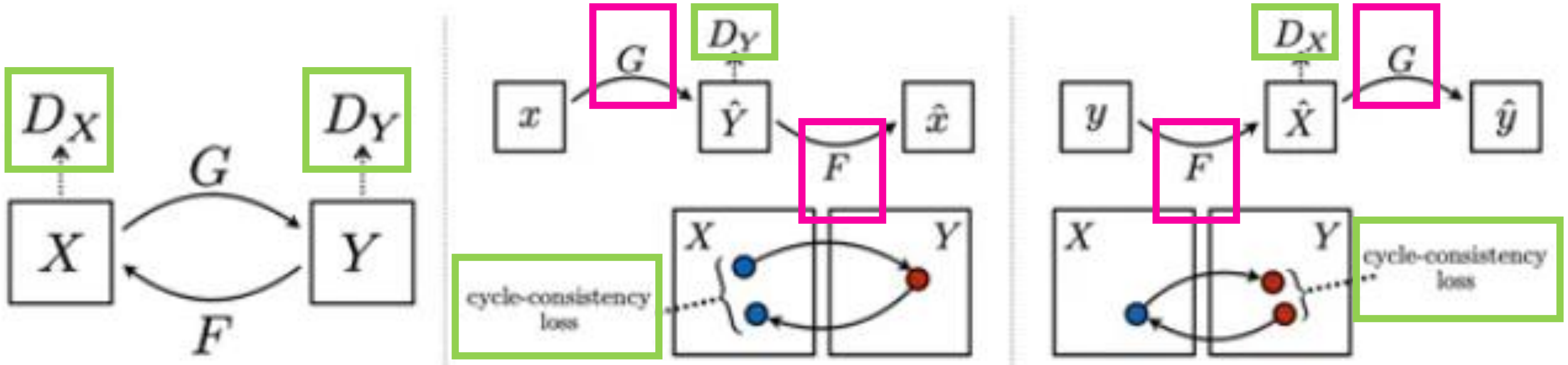
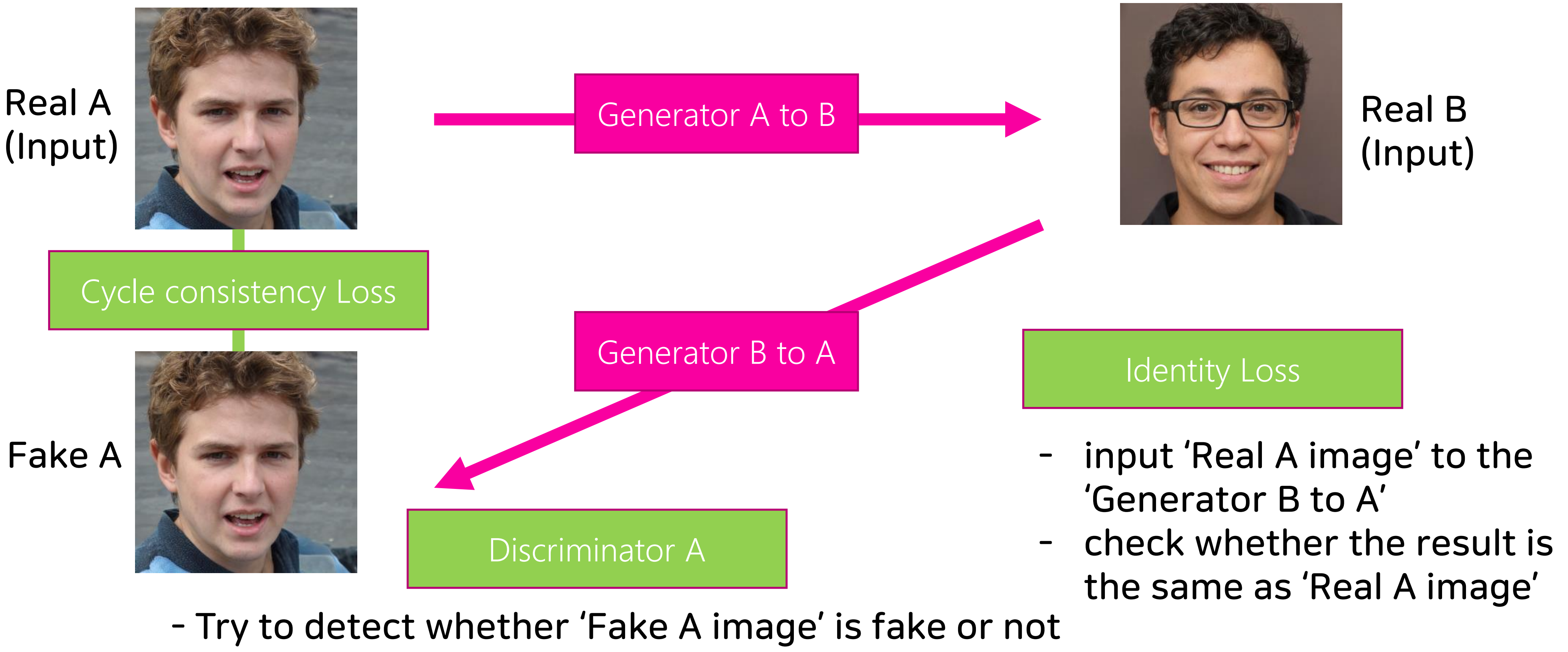


Fig5: The training procedure for CycleGAN.

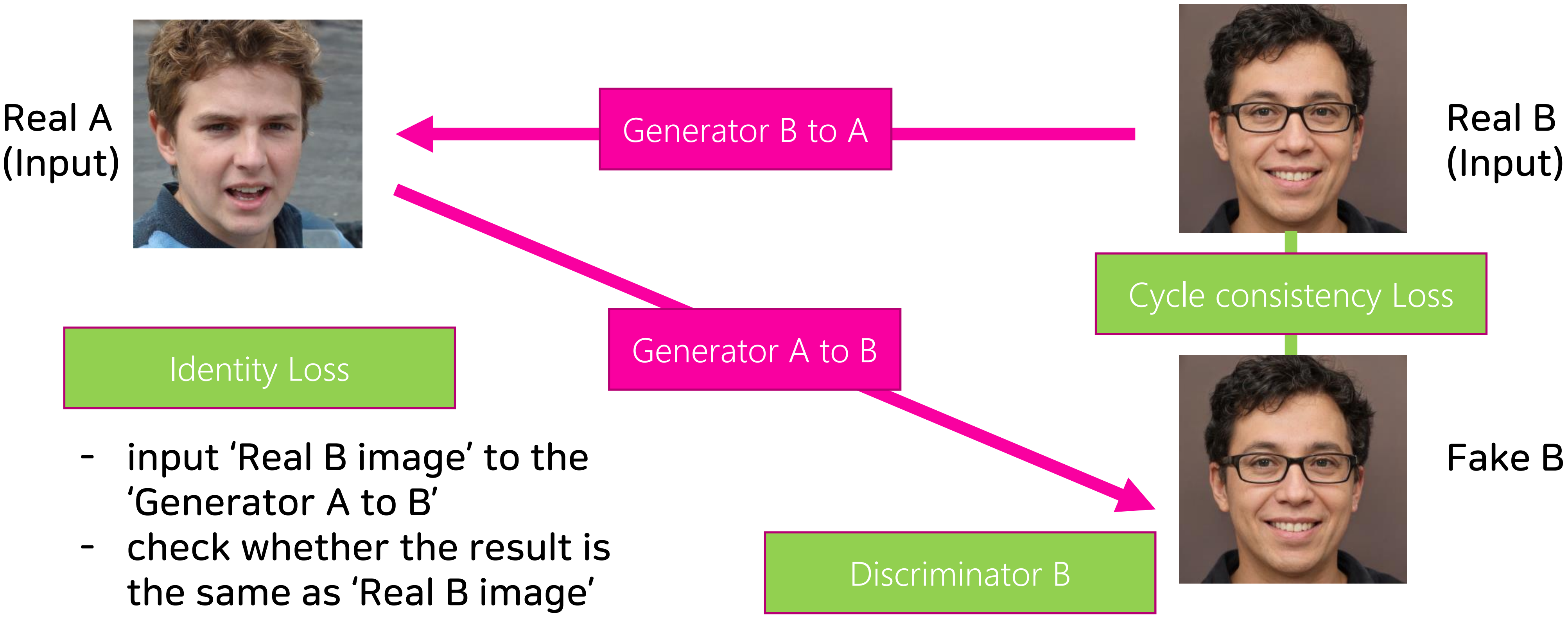
Model

- For training with unpaired dataset(from original A to fake A)



Model

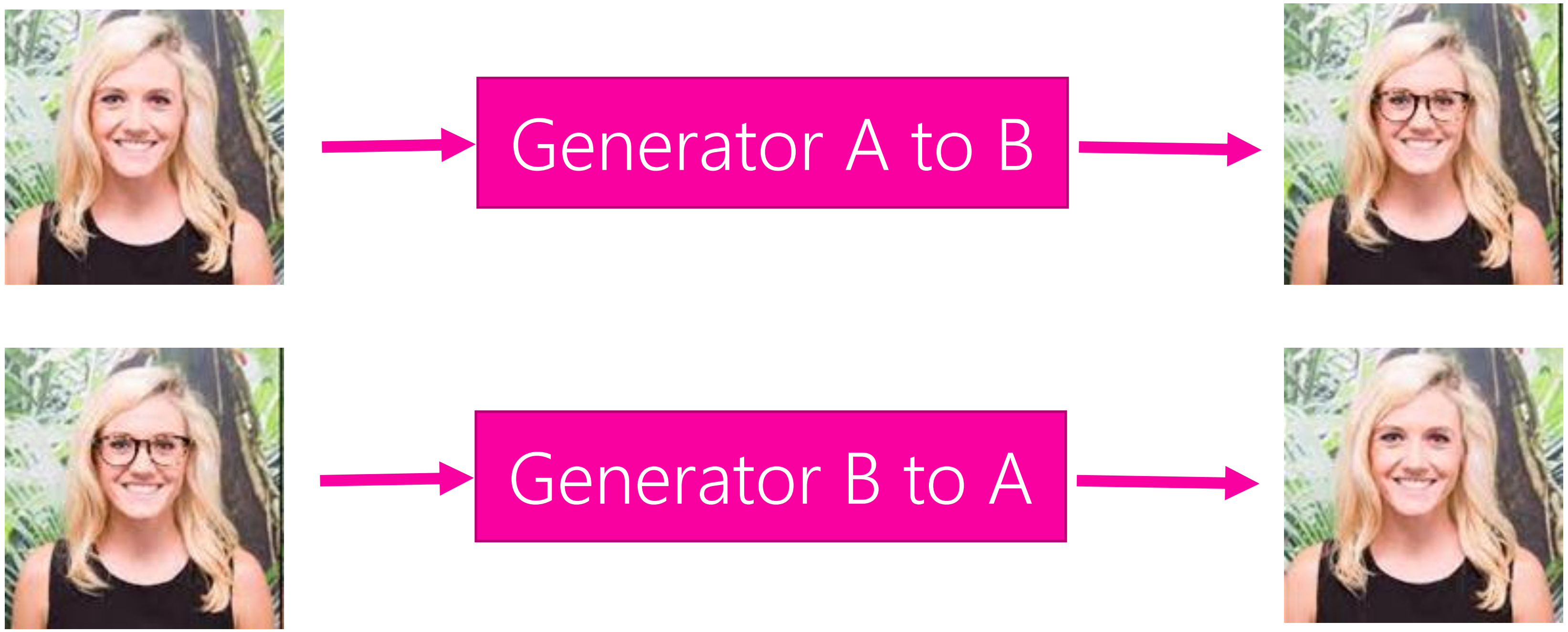
- For training with unpaired dataset(from original B to fake B)



- Try to detect whether 'Fake B image' is fake or not

Model

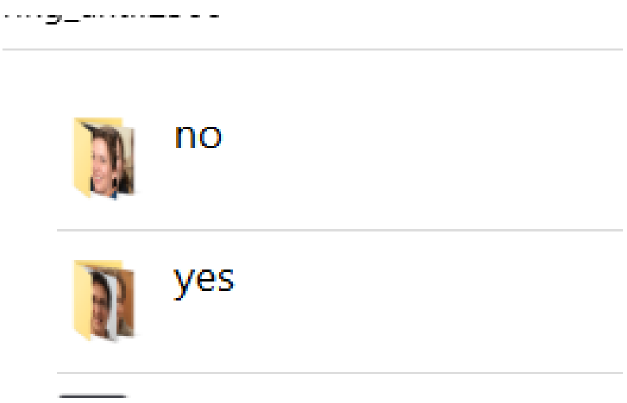
- For test



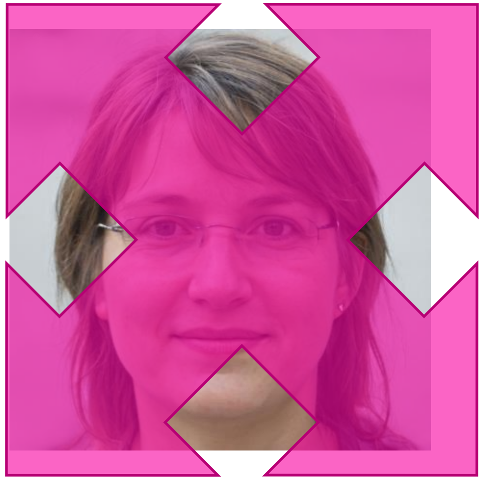
Preprocessing

Data Filtering

- Having full shape of glasses and two eyes
- One person per picture
- Face accounts for one third of the picture
- No mask / Included sunglasses



Preprocessing



Code – Model

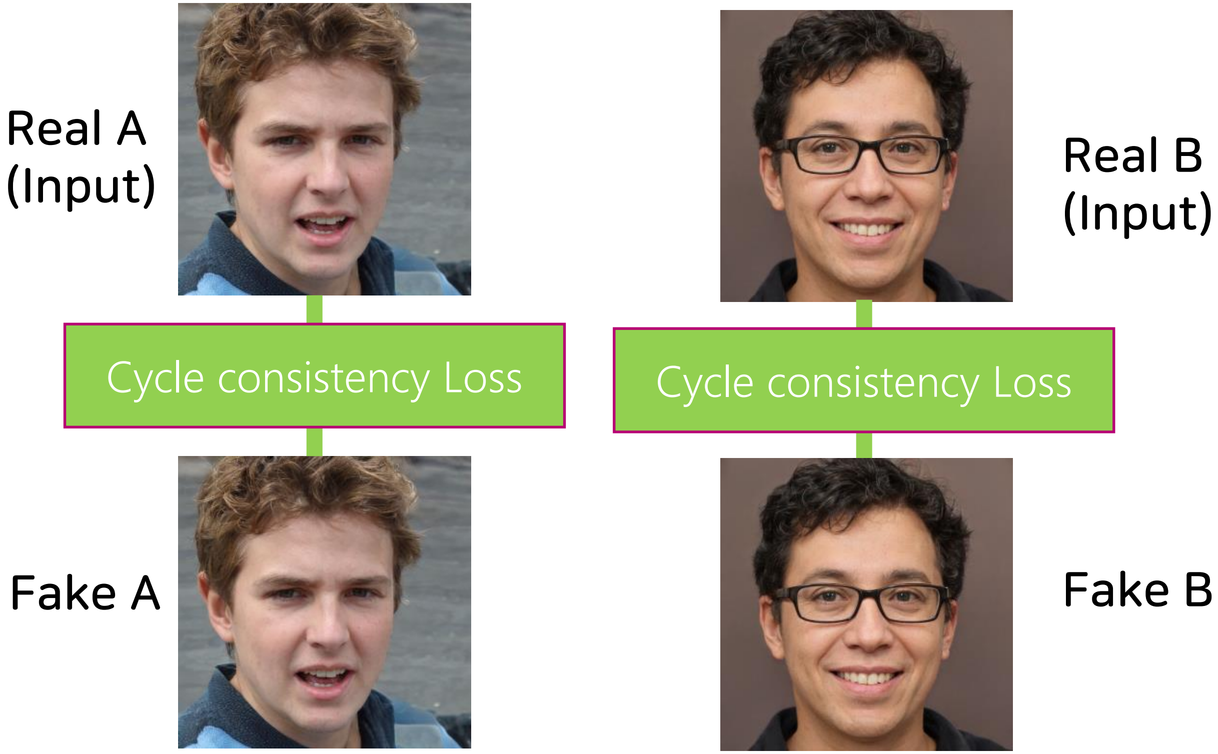
Generator

```
netG_a2b = CycleGAN(in_channels=nch, out_channels=nch, nker=nker, norm=norm, nblk=9).to(device)
netG_b2a = CycleGAN(in_channels=nch, out_channels=nch, nker=nker, norm=norm, nblk=9).to(device)

init_weights(netG_a2b, init_type='normal', init_gain=0.02)
init_weights(netG_b2a, init_type='normal', init_gain=0.02)
```

Discriminator

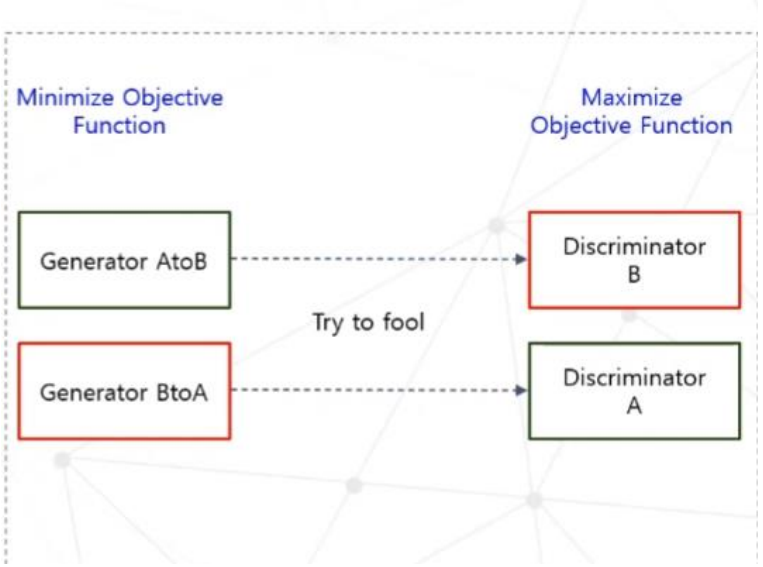
```
netD_a = Discriminator(in_channels=nch, out_channels=1, nker=nker, norm=norm).to(device)
netD_b = Discriminator(in_channels=nch, out_channels=1, nker=nker, norm=norm).to(device)

init_weights(netD_a, init_type='normal', init_gain=0.02)
init_weights(netD_b, init_type='normal', init_gain=0.02)
```

code – Cycle loss



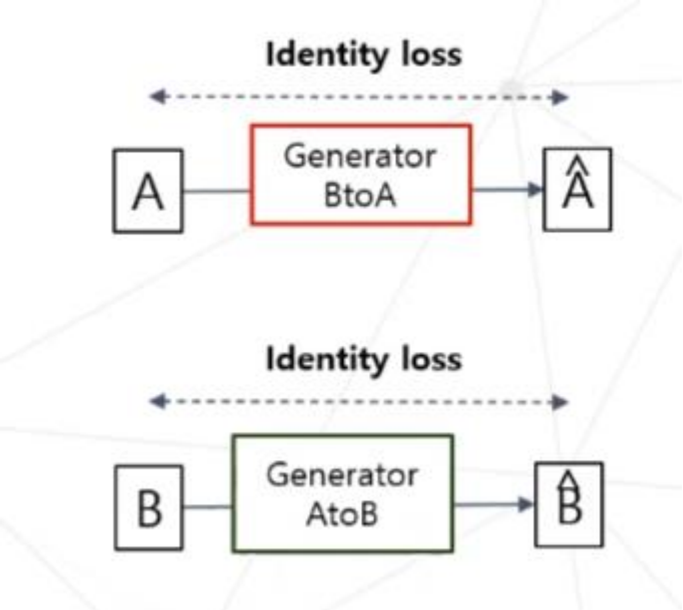

```
fn_cycle = nn.L1Loss().to(device)

loss_cycle_a = fn_cycle(input_a, recon_a)
loss_cycle_b = fn_cycle(input_b, recon_b)
```

code - Gan loss



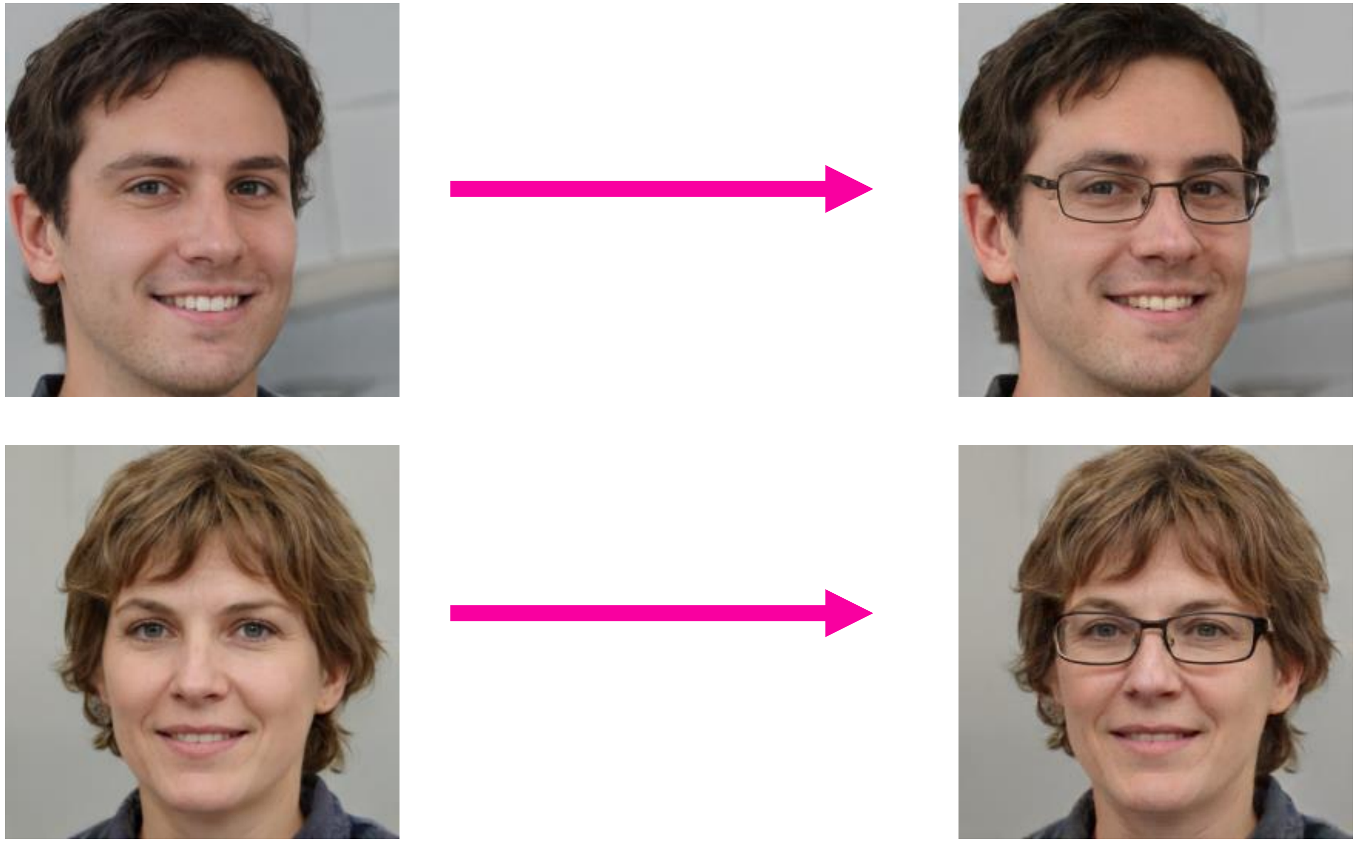

```python
fn_gan = nn.BCELoss().to(device)

# backward netD_a
pred_real_a = netD_a(input_a)
pred_fake_a = netD_a(output_a.detach())

loss_D_a_real = fn_gan(pred_real_a, torch.ones_like(pred_real_a))
loss_D_a_fake = fn_gan(pred_fake_a, torch.zeros_like(pred_fake_a))
loss_D_a = 0.5 * (loss_D_a_real + loss_D_a_fake)

# backward netD_b
pred_real_b = netD_b(input_b)
pred_fake_b = netD_b(output_b.detach())

loss_D_b_real = fn_gan(pred_real_b, torch.ones_like(pred_real_b))
loss_D_b_fake = fn_gan(pred_fake_b, torch.zeros_like(pred_fake_b))
loss_D_b = 0.5 * (loss_D_b_real + loss_D_b_fake)

loss_D = loss_D_a + loss_D_b

loss_G_a2b = fn_gan(pred_fake_a, torch.ones_like(pred_fake_a))
loss_G_b2a = fn_gan(pred_fake_b, torch.ones_like(pred_fake_b))
```

code – Identity loss

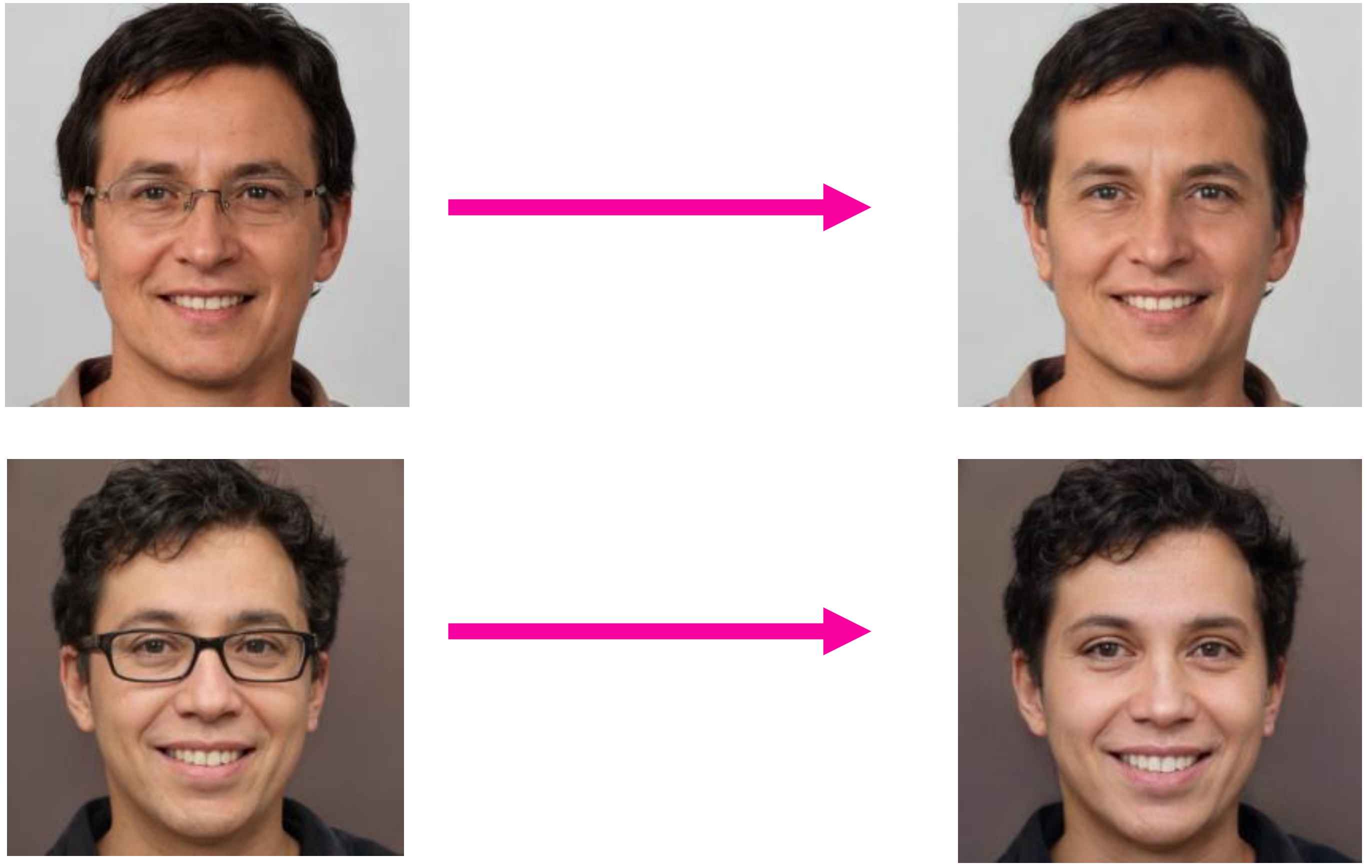



```
fn_ident = nn.L1Loss().to(device)

loss_ident_a = fn_ident(input_a, ident_a)
loss_ident_b = fn_ident(input_b, ident_b)
```

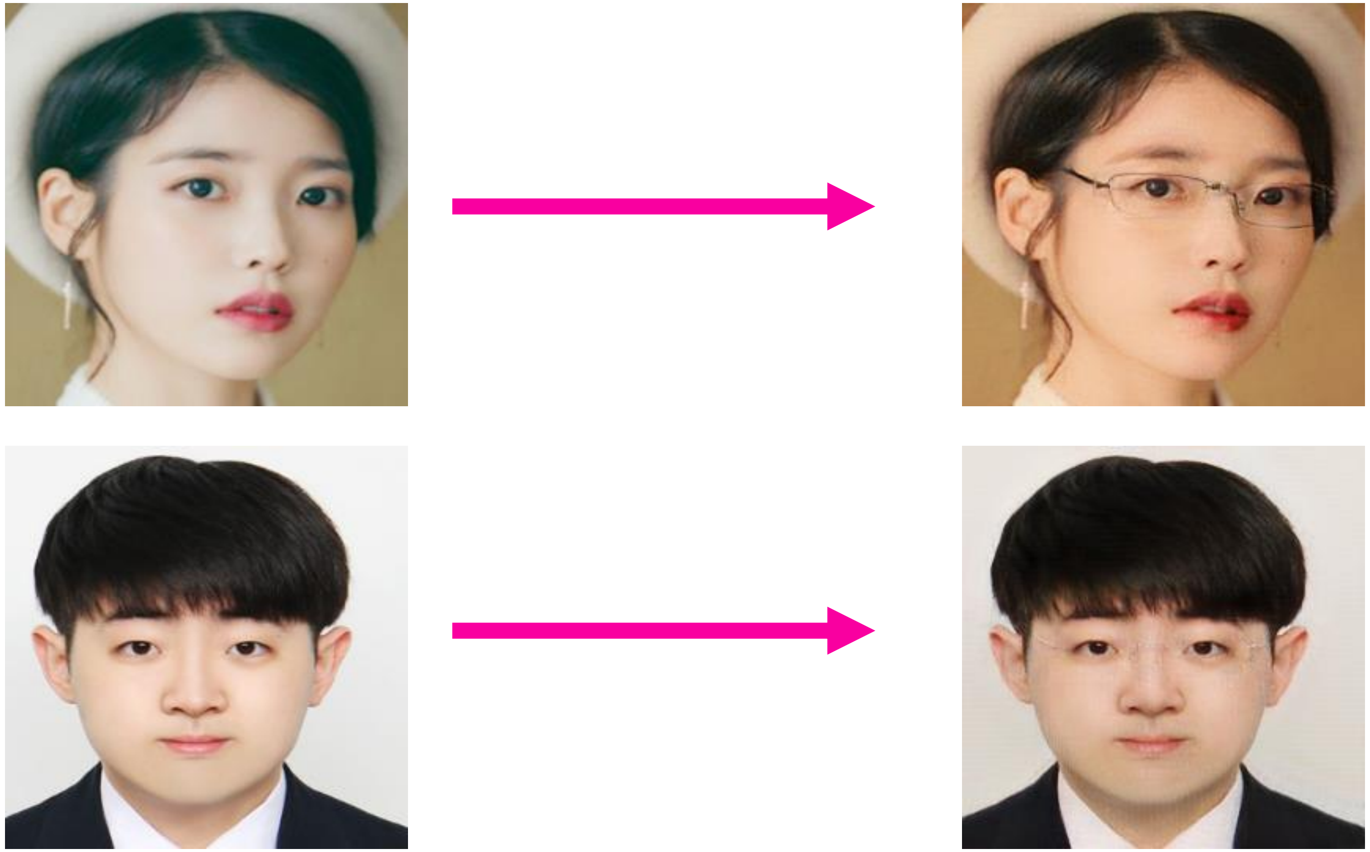
Results- From A to B



Results- From B to A



Applications



Discussion & Limitations

1. Impossible to make various kinds of glasses for each person.
- Because the weight of generators are fixed once the training is over

2. Low resolution & changes other than glasses
- Problems in scaling images

3. changes other than glasses
- May reflect actual infects of glasses including refraction.

4. Changes in hues
- Instance normalization
- Background becomes brighter by faces
- Faces becomes darker by background

Reference

https://openaccess.thecvf.com/content_ICCV_2017/papers/Zhu_Unpaired_Image-To-Image_Translation_ICCV_2017_paper.pdf
https://velog.io/@tobigs-gm1/Style-GAN
https://blog.lunit.io/2017/04/27/style-transfer/
https://arxiv.org/abs/1703.10593
https://www.kaggle.com/jeffheaton/glasses-or-no-glasses
http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html
http://www.kwangsiklee.com/2018/03/cyclegan%EC%9D%B4-%EB%AC%B4%EC%97%87%EC%9D%B8%EC%A7%80-%EC%95%8C%EC%95%84%EB%B3%B4%EC%9E%90/