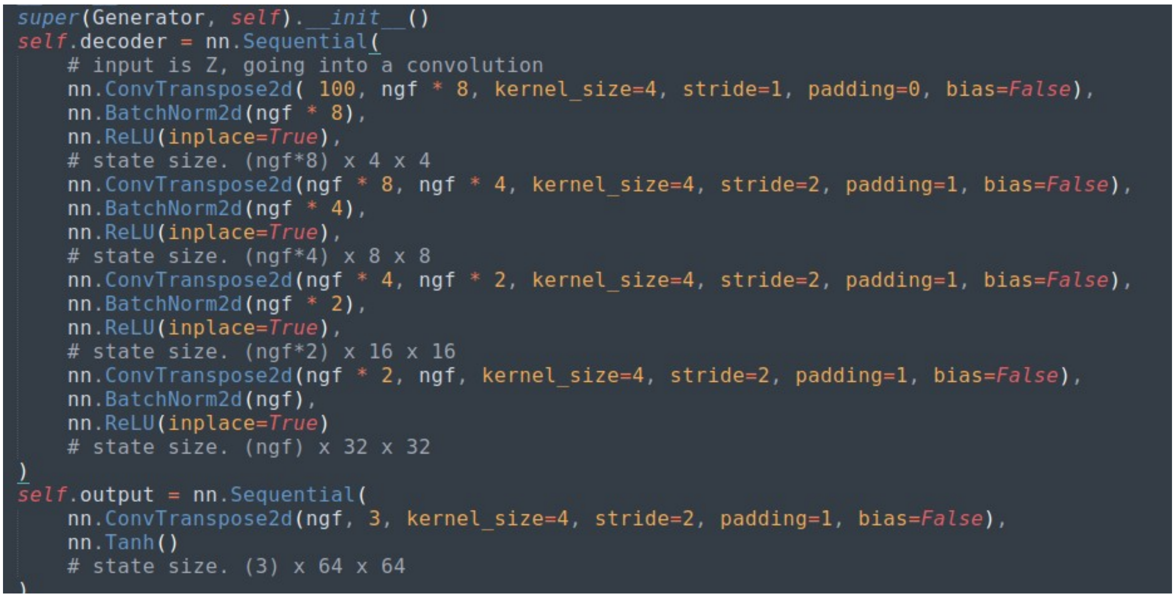
DLCV
HW3_REPORT

R07945010
許展銘

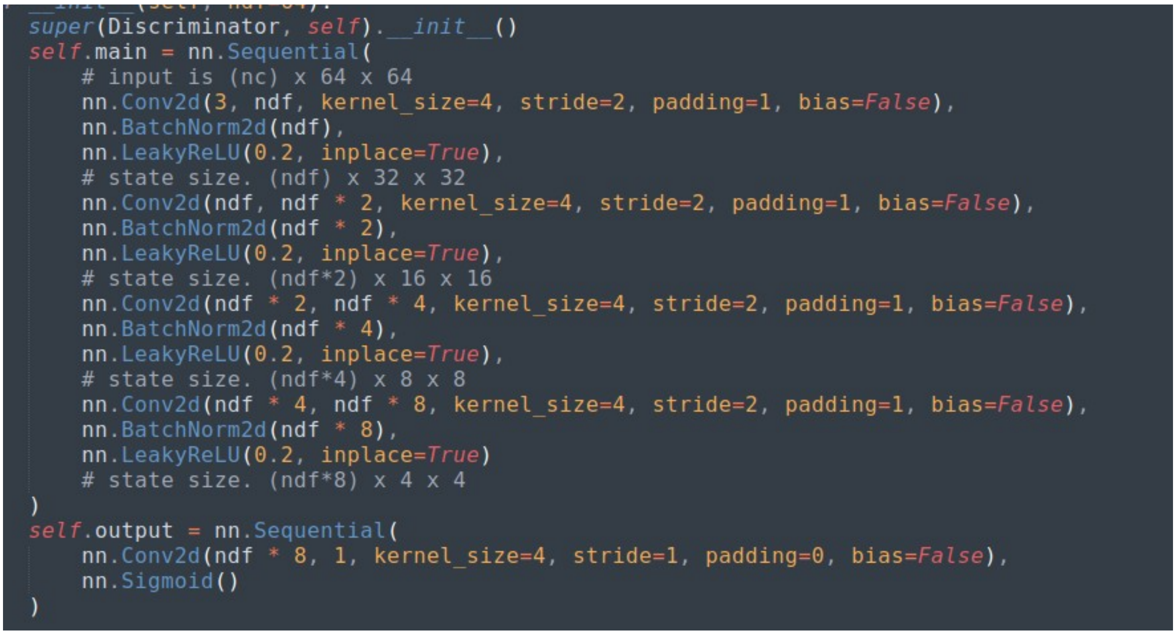
- GAN

1. Generator



```
super(Generator, self).__init__()
self.decoder = nn.Sequential(
    # input is Z, going into a convolution
    nn.ConvTranspose2d( 100, ngf * 8, kernel_size=4, stride=1, padding=0, bias=False),
    nn.BatchNorm2d(ngf * 8),
    nn.ReLU(inplace=True),
    # state size. (ngf*8) x 4 x 4
    nn.ConvTranspose2d(ngf * 8, ngf * 4, kernel_size=4, stride=2, padding=1, bias=False),
    nn.BatchNorm2d(ngf * 4),
    nn.ReLU(inplace=True),
    # state size. (ngf*4) x 8 x 8
    nn.ConvTranspose2d(ngf * 4, ngf * 2, kernel_size=4, stride=2, padding=1, bias=False),
    nn.BatchNorm2d(ngf * 2),
    nn.ReLU(inplace=True),
    # state size. (ngf*2) x 16 x 16
    nn.ConvTranspose2d(ngf * 2, ngf, kernel_size=4, stride=2, padding=1, bias=False),
    nn.BatchNorm2d(ngf),
    nn.ReLU(inplace=True)
    # state size. (ngf) x 32 x 32
)
self.output = nn.Sequential(
    nn.ConvTranspose2d(ngf, 3, kernel_size=4, stride=2, padding=1, bias=False),
    nn.Tanh()
    # state size. (3) x 64 x 64
```

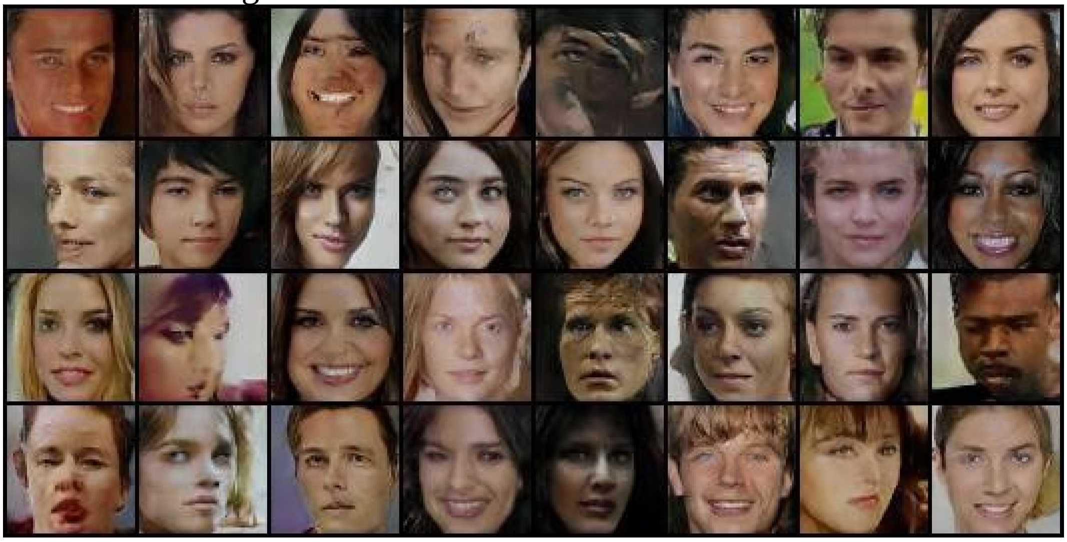
Discriminator



```
super(Discriminator, self).__init__()
self.main = nn.Sequential(
    # input is (nc) x 64 x 64
    nn.Conv2d(3, ndf, kernel_size=4, stride=2, padding=1, bias=False),
    nn.BatchNorm2d(ndf),
    nn.LeakyReLU(0.2, inplace=True),
    # state size. (ndf) x 32 x 32
    nn.Conv2d(ndf, ndf * 2, kernel_size=4, stride=2, padding=1, bias=False),
    nn.BatchNorm2d(ndf * 2),
    nn.LeakyReLU(0.2, inplace=True),
    # state size. (ndf*2) x 16 x 16
    nn.Conv2d(ndf * 2, ndf * 4, kernel_size=4, stride=2, padding=1, bias=False),
    nn.BatchNorm2d(ndf * 4),
    nn.LeakyReLU(0.2, inplace=True),
    # state size. (ndf*4) x 8 x 8
    nn.Conv2d(ndf * 4, ndf * 8, kernel_size=4, stride=2, padding=1, bias=False),
    nn.BatchNorm2d(ndf * 8),
    nn.LeakyReLU(0.2, inplace=True)
    # state size. (ndf*8) x 4 x 4
)
self.output = nn.Sequential(
    nn.Conv2d(ndf * 8, 1, kernel_size=4, stride=1, padding=0, bias=False),
    nn.Sigmoid()
)
```

The architecture was adopted from Pytorch DCGAN tutorial.

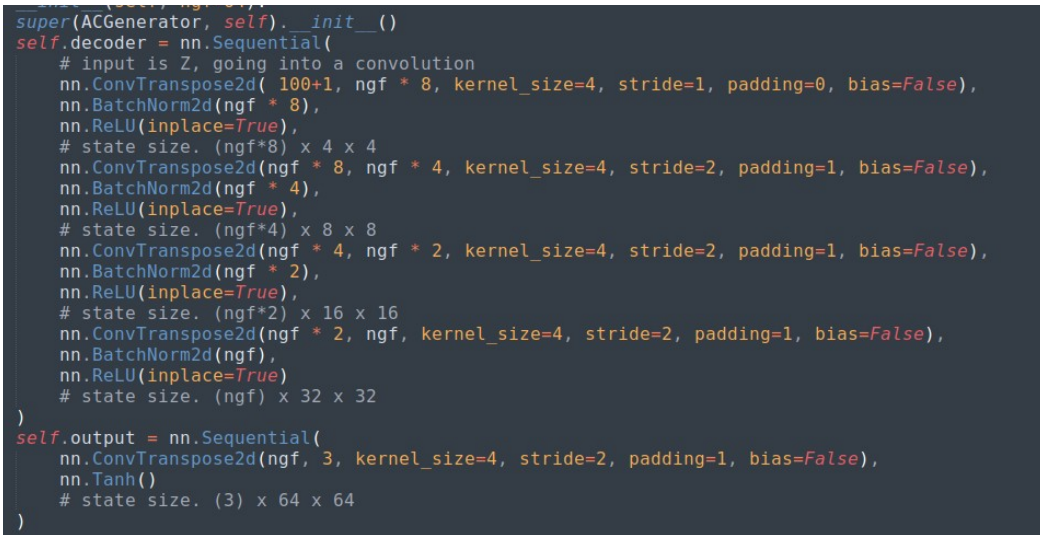
2. 32 random images from our model



3. While implementing GAN, we do not assign specified label to the random noise, so it comes out that most of the random generated images are in the same face category such as the “smile” label.

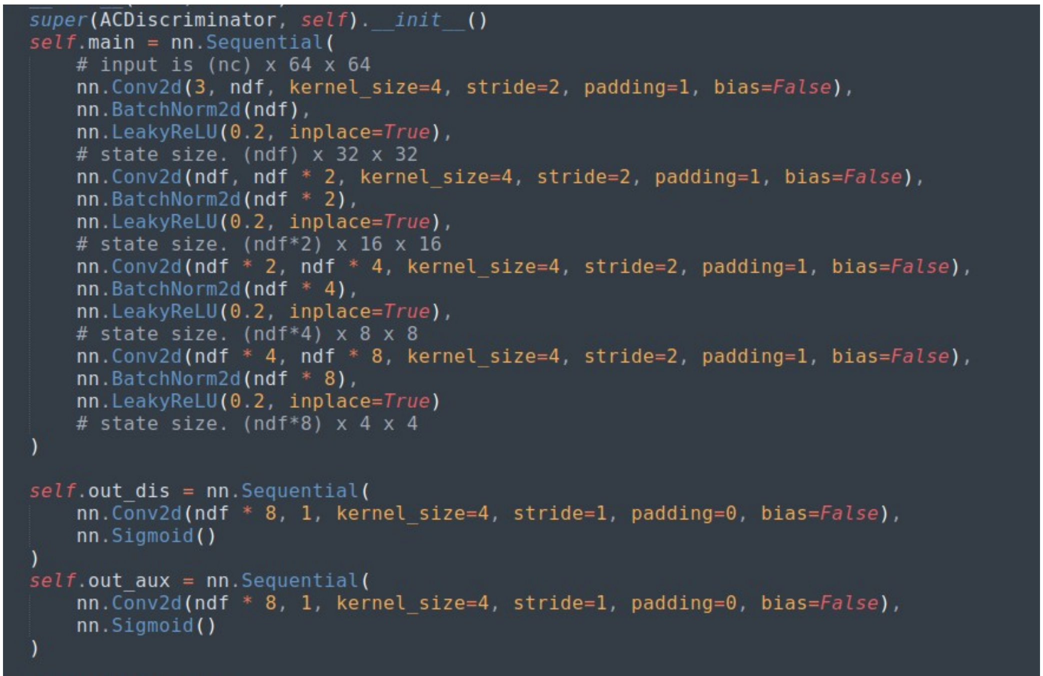
- ACGAN

  1. Generator



```
super(ACGenerator, self).__init__()
self.decoder = nn.Sequential(
    # input is Z, going into a convolution
    nn.ConvTranspose2d( 100+1, ngf * 8, kernel_size=4, stride=1, padding=0, bias=False),
    nn.BatchNorm2d(ngf * 8),
    nn.ReLU(inplace=True),
    # state size. (ngf*8) x 4 x 4
    nn.ConvTranspose2d(ngf * 8, ngf * 4, kernel_size=4, stride=2, padding=1, bias=False),
    nn.BatchNorm2d(ngf * 4),
    nn.ReLU(inplace=True),
    # state size. (ngf*4) x 8 x 8
    nn.ConvTranspose2d(ngf * 4, ngf * 2, kernel_size=4, stride=2, padding=1, bias=False),
    nn.BatchNorm2d(ngf * 2),
    nn.ReLU(inplace=True),
    # state size. (ngf*2) x 16 x 16
    nn.ConvTranspose2d(ngf * 2, ngf, kernel_size=4, stride=2, padding=1, bias=False),
    nn.BatchNorm2d(ngf),
    nn.ReLU(inplace=True)
    # state size. (ngf) x 32 x 32
)
self.output = nn.Sequential(
    nn.ConvTranspose2d(ngf, 3, kernel_size=4, stride=2, padding=1, bias=False),
    nn.Tanh()
    # state size. (3) x 64 x 64
)
```

  Discriminator

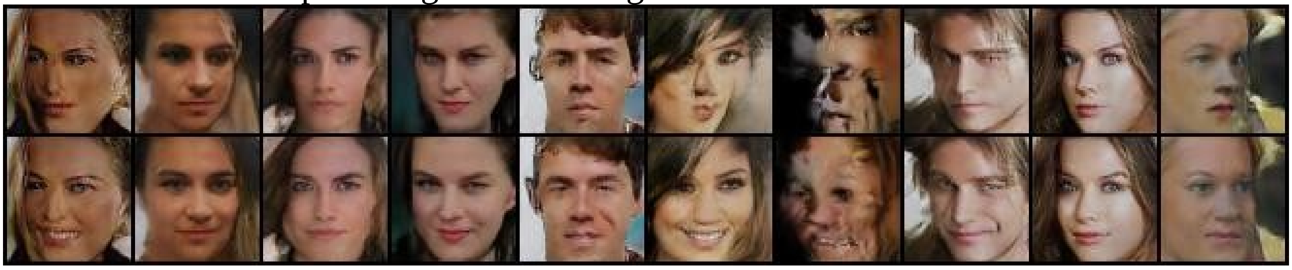


```
super(ACDiscriminator, self).__init__()
self.main = nn.Sequential(
    # input is (nc) x 64 x 64
    nn.Conv2d(3, ndf, kernel_size=4, stride=2, padding=1, bias=False),
    nn.BatchNorm2d(ndf),
    nn.LeakyReLU(0.2, inplace=True),
    # state size. (ndf) x 32 x 32
    nn.Conv2d(ndf, ndf * 2, kernel_size=4, stride=2, padding=1, bias=False),
    nn.BatchNorm2d(ndf * 2),
    nn.LeakyReLU(0.2, inplace=True),
    # state size. (ndf*2) x 16 x 16
    nn.Conv2d(ndf * 2, ndf * 4, kernel_size=4, stride=2, padding=1, bias=False),
    nn.BatchNorm2d(ndf * 4),
    nn.LeakyReLU(0.2, inplace=True),
    # state size. (ndf*4) x 8 x 8
    nn.Conv2d(ndf * 4, ndf * 8, kernel_size=4, stride=2, padding=1, bias=False),
    nn.BatchNorm2d(ndf * 8),
    nn.LeakyReLU(0.2, inplace=True)
    # state size. (ndf*8) x 4 x 4
)

self.out_dis = nn.Sequential(
    nn.Conv2d(ndf * 8, 1, kernel_size=4, stride=1, padding=0, bias=False),
    nn.Sigmoid()
)
self.out_aux = nn.Sequential(
    nn.Conv2d(ndf * 8, 1, kernel_size=4, stride=1, padding=0, bias=False),
    nn.Sigmoid()
)
```

The only difference in the generator between GAN and ACGAN is the input size(100>>101 since we add one “smiling” attribute).
And the difference in the discriminator between them is the output was separated to two parts, one for real/fake, one for attribute.

2. 10 random pairs of generated images from our model



3. With the specified label (smiling) trained together, we are able to feed the noise with such attribute to get the smiling generated face. Vice versa.