README.md

GANs refined

In this week's lab, you used a GAN to generate images of people that look like celebrities.

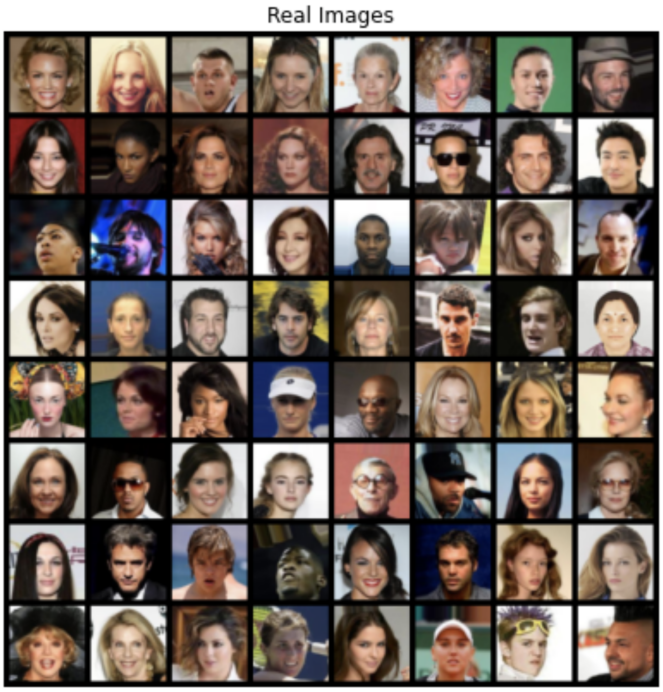
For this homework, you will re-run the lab with modifications to improve the quality of the generated images. The easiest improvements can be made by modifying values in the ***Inputs*** section

Submission:

1. What changes did you make to the original lab? Did your changes result in higher quality images?

I increased the number of epochs to 10 and increased the size of z latent vector (size of generator input, which subjectively did result in higher quality changes.

Objectively speaking, the average output of the discriminator for the all real batch converges closer to 0.5 as the generator gets better at creating fake images and the discriminator has a harder time discerning between real and fake images. When it gets too difficult to tell the images as fake or real, the discriminator guesses randomly (**D(x)** = 0.5).



```
[9/10][0/1583]  Loss_D: 0.2576  Loss_G: 3.2995  D(x): 0.8721    D(G(z)): 0.0897 / 0.0541
[9/10][50/1583] Loss_D: 0.3014  Loss_G: 2.5209  D(x): 0.8012    D(G(z)): 0.0536 / 0.1063
[9/10][100/1583]        Loss_D: 0.3521  Loss_G: 2.6138  D(x): 0.8572    D(G(z)): 0.1553 / 0.0931
[9/10][150/1583]        Loss_D: 1.5249  Loss_G: 5.6833  D(x): 0.9664    D(G(z)): 0.7058 / 0.0066
[9/10][200/1583]        Loss_D: 0.6453  Loss_G: 4.9540  D(x): 0.9539    D(G(z)): 0.4050 / 0.0110
[9/10][250/1583]        Loss_D: 0.2808  Loss_G: 2.5076  D(x): 0.8654    D(G(z)): 0.1132 / 0.1072
[9/10][300/1583]        Loss_D: 0.2567  Loss_G: 3.0096  D(x): 0.8673    D(G(z)): 0.0872 / 0.0663
[9/10][350/1583]        Loss_D: 0.4214  Loss_G: 3.7033  D(x): 0.8988    D(G(z)): 0.2317 / 0.0393
[9/10][400/1583]        Loss_D: 0.4434  Loss_G: 3.6313  D(x): 0.9417    D(G(z)): 0.2781 / 0.0389
[9/10][450/1583]        Loss_D: 0.3469  Loss_G: 3.7545  D(x): 0.9002    D(G(z)): 0.1929 / 0.0328
[9/10][500/1583]        Loss_D: 0.4160  Loss_G: 3.6928  D(x): 0.9103    D(G(z)): 0.2426 / 0.0365
[9/10][550/1583]        Loss_D: 0.4468  Loss_G: 4.2297  D(x): 0.9502    D(G(z)): 0.2890 / 0.0208
[9/10][600/1583]        Loss_D: 0.3189  Loss_G: 2.7320  D(x): 0.8230    D(G(z)): 0.0905 / 0.0897
[9/10][650/1583]        Loss_D: 0.9422  Loss_G: 3.5547  D(x): 0.7919    D(G(z)): 0.4024 / 0.0687
[9/10][700/1583]        Loss_D: 0.3060  Loss_G: 3.9613  D(x): 0.8998    D(G(z)): 0.1489 / 0.0294
[9/10][750/1583]        Loss_D: 0.5269  Loss_G: 3.7530  D(x): 0.8658    D(G(z)): 0.2650 / 0.0352
[9/10][800/1583]        Loss_D: 0.3410  Loss_G: 3.3201  D(x): 0.8984    D(G(z)): 0.1824 / 0.0533
[9/10][850/1583]        Loss_D: 0.4727  Loss_G: 2.8248  D(x): 0.8868    D(G(z)): 0.2616 / 0.0793
[9/10][900/1583]        Loss_D: 0.3446  Loss_G: 3.1823  D(x): 0.8539    D(G(z)): 0.1451 / 0.0628
[9/10][950/1583]        Loss_D: 0.2045  Loss_G: 3.4087  D(x): 0.8925    D(G(z)): 0.0733 / 0.0540
[9/10][1000/1583]       Loss_D: 1.2836  Loss_G: 1.1103  D(x): 0.4231    D(G(z)): 0.2034 / 0.3868
[9/10][1050/1583]       Loss_D: 0.4819  Loss_G: 3.6427  D(x): 0.9026    D(G(z)): 0.2801 / 0.0369
[9/10][1100/1583]       Loss_D: 0.4454  Loss_G: 3.9103  D(x): 0.8682    D(G(z)): 0.2259 / 0.0303
[9/10][1150/1583]       Loss_D: 0.6018  Loss_G: 2.0516  D(x): 0.6087    D(G(z)): 0.0211 / 0.1826
[9/10][1200/1583]       Loss_D: 0.8125  Loss_G: 0.1227  D(x): 0.5224    D(G(z)): 0.0298 / 0.8959
[9/10][1250/1583]       Loss_D: 0.4629  Loss_G: 3.9947  D(x): 0.9083    D(G(z)): 0.2677 / 0.0282
[9/10][1300/1583]       Loss_D: 0.6838  Loss_G: 2.5042  D(x): 0.7402    D(G(z)): 0.2522 / 0.1228
[9/10][1350/1583]       Loss_D: 0.7599  Loss_G: 2.3802  D(x): 0.6558    D(G(z)): 0.1916 / 0.1401
[9/10][1400/1583]       Loss_D: 0.3929  Loss_G: 2.1607  D(x): 0.7733    D(G(z)): 0.1023 / 0.1598
[9/10][1450/1583]       Loss_D: 1.1561  Loss_G: 3.9871  D(x): 0.9351    D(G(z)): 0.5981 / 0.0300
[9/10][1500/1583]       Loss_D: 0.9115  Loss_G: 1.6442  D(x): 0.5104    D(G(z)): 0.1217 / 0.2530
[9/10][1550/1583]       Loss_D: 0.2719  Loss_G: 3.2279  D(x): 0.8707    D(G(z)): 0.1069 / 0.0605
```

2. In your own words, how does the Discriminator improve its ability to detect fakes?

The Discriminator improves its ability to detect fakes by being fed both real and fake images and minimizing a trainable loss function that identifies whether the given image is real or fake. Through backpropagation, the Discriminator tells the Generator how to tweak each pixel so the image will be more realistic. As the Generator gets better, the Discriminator should theoretically also improve its ability to detect fakes.

3. Share a copy of the outpt image from the last step in the notebook (can be an upload to the ISVC Portal, or a link to the file in AWS Object Store).

Attached above.