

Name: Adel Alkhamisy

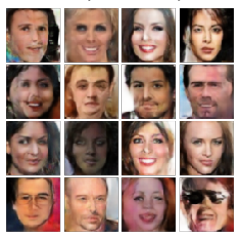
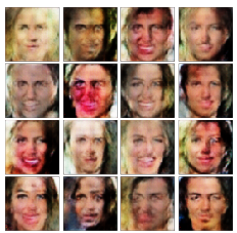
NetID:aalkhami

Team name on Kaggle leaderboard: aalkhami

Part 1:

Answer the following questions briefly (no more than a few sentences), and provide output images where requested.

Show final results from training both your GAN and LSGAN (give the final 4x4 grid of images for both):

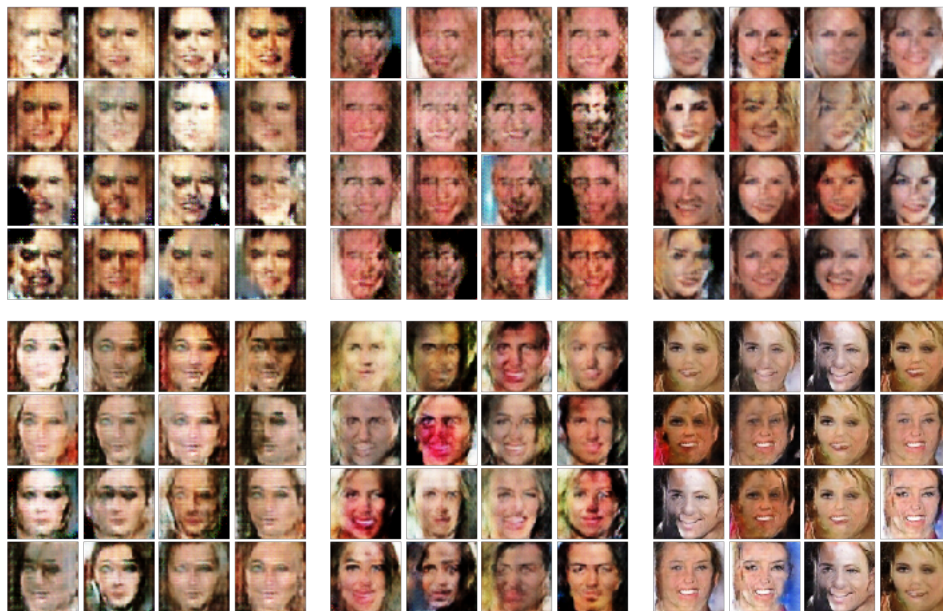
GAN		LSGAN	
<p>EPOCH: 1 Iter: 0, D: 0.6938, G:1.057</p> 	<p>EPOCH: 12 Iter: 10950, D: 0.3549, G:1.977</p> 	<p>EPOCH: 1 Iter: 0, D: 0.5227, G:0.4657</p> 	<p>EPOCH: 26 Iter: 25000, D: 0.1039, G:0.2582</p> 
<p>EPOCH: 3 Iter: 2100, D: 0.3796, G:1.269</p> 	<p>EPOCH: 14 Iter: 13350, D: 0.2588, G:2.246</p> 	<p>EPOCH: 3 Iter: 2500, D: 0.09941, G:0.4422</p> 	<p>EPOCH: 33 Iter: 32500, D: 0.0263, G:0.4661</p> 
<p>EPOCH: 6 Iter: 4950, D: 0.397, G:1.008</p> 	<p>EPOCH: 15 Iter: 14400, D: 0.2402, G:3.172</p> 	<p>EPOCH: 6 Iter: 5000, D: 0.2511, G:0.1332</p> 	<p>EPOCH: 41 Iter: 40000, D: 0.2551, G:0.1263</p> 
<p>EPOCH: 8 Iter: 7050, D: 0.4932, G:2.143</p> 	<p>EPOCH: 27 Iter: 25650, D: 0.3831, G:6.237</p> 	<p>EPOCH: 21 Iter: 20000, D: 0.0595, G:0.484</p> 	<p>EPOCH: 49 Iter: 47500, D: 0.05492, G:0.4089</p> 

Discuss any differences you observed in quality of output or behavior during training of the two GAN models.

The sigmoid cross-entropy loss function used by the original GAN can result in vanishing gradients and unstable training. Mode collapse is a common outcome of this, when the generator produces outputs with little output diversity. The least squares loss function used by LSGAN, on the other hand, helps to address these problems by punishing the model based on the squared difference between the predicted and actual values. This strategy promotes the generator to generate samples that are more varied and of higher quality, while also enhancing training stability and lowering the risk of mode collapse. Compared to the original GAN model, LSGAN generally gives superior outcomes and a more consistent training environment.

**Do you notice any instances of mode collapse in your GAN training (especially early in training)?
Show some instances of mode collapse (if any) from your training output.**

Yes, I notice some instances of model collapse, the below is a few examples:



Discuss briefly how/whether spectral normalization helps generate higher quality images in your implementation. Ideally, you should show samples from models with and without normalization.

The process of "spectral normalization" limits the spectral norm of the weights in the convolutional layers of the discriminator, which stabilizes the training of GANs. In order to avoid unstable training dynamics and mode collapse, it seeks to prevent the Discriminator from being very strong or dominating during training. Convolutional layers of the Discriminator are normalized spectrally. It aids in maintaining a balance between the Generator and Discriminator during training by limiting the spectral norm of the weights. The Generator learns to produce higher quality images by receiving more valuable gradient information from the Discriminator while the balance is preserved.

Without spectral norm



with spectral norm

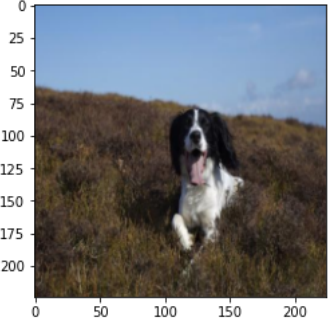

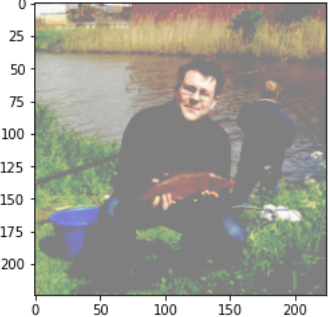
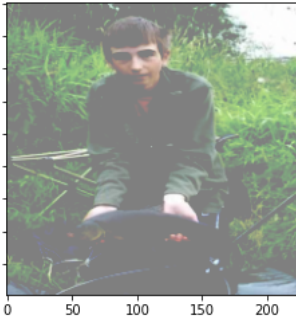


Extra credit: If you completed the extra credit for this portion, explain what you did (describing all model changes and hyperparameter settings) and provide output images.

First, I changed the **scale_size=64** to **scale_size=128**, and, the first convolution layer in the discrimination from `SpectralNorm(nn.Conv2d(input_channels, 64, 4, 2, 1))` to `SpectralNorm(nn.Conv2d(input_channels, 128, 4, 2, 1))` to handle 128x128 resolution CelebA data. The number of output channels increases progressively from 128 to 2048. In the generator, the first deconvolutional layer, changed to `nn.ConvTranspose2d(noise_dim, 2048, 4, 1, 0)` to to upsample the input noise tensor and generate an image. A series of six transpose convolutional layers that progressively increase the spatial dimensions of the generated image while decreasing the number of output channels from 2048 to the final number of output channels.

Part 2 (Adversarial Attack):

Show some sample images after your attack methods.

fgsm attack	iterative gradient sign attack
<p>Before</p> 	<p>Before</p> 
<p>After</p> 	<p>After</p> 
<p>Before</p> 	<p>Before</p> 
<p>After</p> 	<p>After</p> 

Discuss why your attack/defense mechanisms work or do not work.

In this experiment, input transformations were used to protect a pre-trained model from the FGSM attack. The adversarial perturbations were partially reduced by applying Gaussian blur and resizing to the input images, which resulted in barely any difference in the accuracy of the model before and after the attack. The results show that the input transformations were able to create an additional layer of defense against the FGSM attack in this particular case.