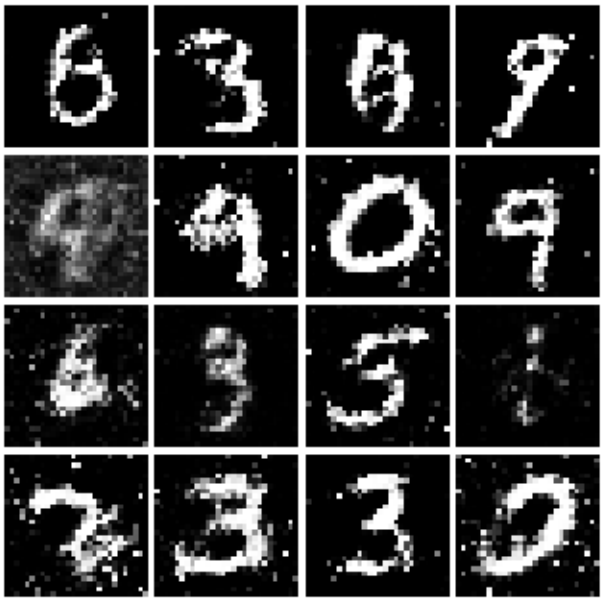
Name: Qi Long, Yihong Yang
NetID(s): qilong2, yihongy3

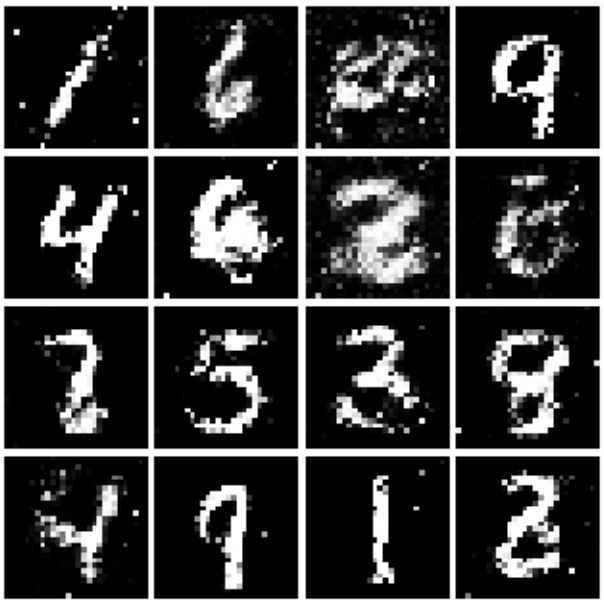
GAN and LSGAN MNIST

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

1. GAN:



2. LSGAN:



GAN and LSGAN Cats

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

1. GAN

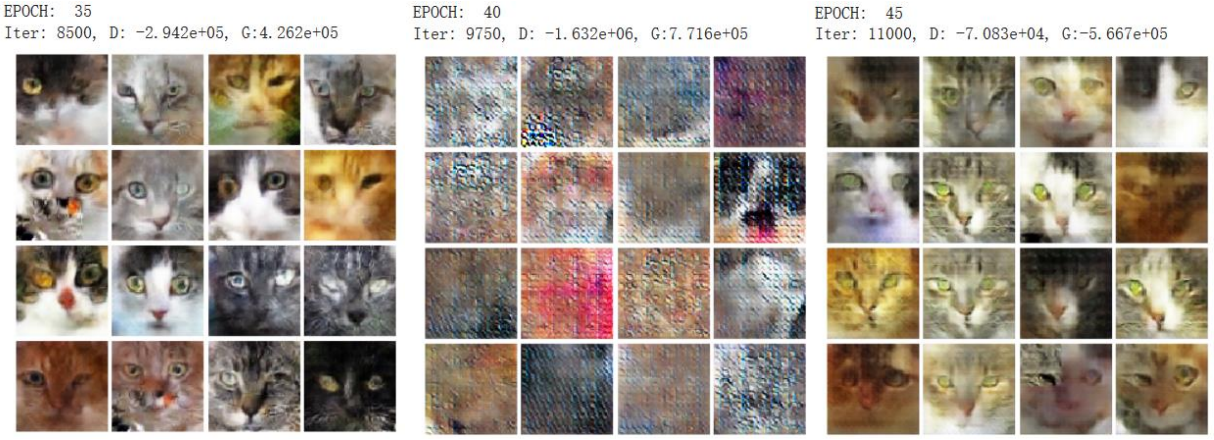
2. LSGAN

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

1. Quality of output:
   LSGAN's final output has overall higher quality than GAN, GAN has some generated pictures containing nothing (top right one / third one of last row above) but most of LSGAN outputs have basic features of cats.
2. Training Process:
   GAN's training process is less stable. At Iteration 6750, GAN's generator loss rises from 2 to 8 in the following iterations and the quality of generated pictures reaches highest at middle of the training. Once LSGAN reaches relatively good outputs, it can remain stable (after 10750 iteration) and improve slightly each iteration.

Do you notice any instances of mode collapse in your GAN training? Show some instances of mode collapse from your training output.

During training process using WGAN Loss in Extra Credit 1, the collapse happens that after it reached relatively good outputs, it outputs noises for some epochs and after that recovered for later epochs.



Extra Credit - Alternative GAN Formulation

Explain what you did (describing all model changes and hyperparameter settings) and provide output images.

Extra Credit 1 – Alternative WGAN Loss

1. Implementation: define WGAN Loss in losses.py.

```
def w_discriminator_loss(scores_real, scores_fake):

    loss_real = torch.mean(scores_real)
    loss_fake = torch.mean(scores_fake)
    loss = loss_fake - loss_real

    return loss


def w_generator_loss(scores_fake):

    loss_fake = torch.mean(scores_fake)
    loss = -loss_fake

    return loss
```

Hyperparameters stay the same.

2. Results:

Extra Credit 2 – Spectral Normalization

1. Implementation: include spectral normalization (torch.nn.utils.spectral_norm) in model.py.

```
### EC2: Spectral Normalization
self.ec2_layers = torch.nn.Sequential(
    torch.nn.utils.spectral_norm(torch.nn.Conv2d(input_channels, 128, 4, 2, 1)),  # in_ch, out_ch, kernel_size, stride, padding
    torch.nn.LeakyReLU(0.2),
    torch.nn.utils.spectral_norm(torch.nn.Conv2d(128, 256, 4, 2, 1)),
    torch.nn.BatchNorm2d(256),
    torch.nn.LeakyReLU(0.2),
    torch.nn.utils.spectral_norm(torch.nn.Conv2d(256, 512, 4, 2, 1)),
    torch.nn.BatchNorm2d(512),
    torch.nn.LeakyReLU(0.2),
    torch.nn.utils.spectral_norm(torch.nn.Conv2d(512, 1024, 4, 2, 1)),
    torch.nn.BatchNorm2d(1024),
    torch.nn.LeakyReLU(0.2),
    torch.nn.Conv2d(1024, 1, 4, 1, 0),
)
```

Loss: Original GAN
Hyperparameters stays the same.

2. Results:

By comparison, outputs of this improved model are better than original GAN. The quality of final output is better, and the training process is more stable.

Extra Credit 3 – Another Dataset

1. Dataset: Animal Faces https://www.kaggle.com/datasets/andrewmvd/animal-faces/data.

2. Implementation: same as original GAN.
3. Results: