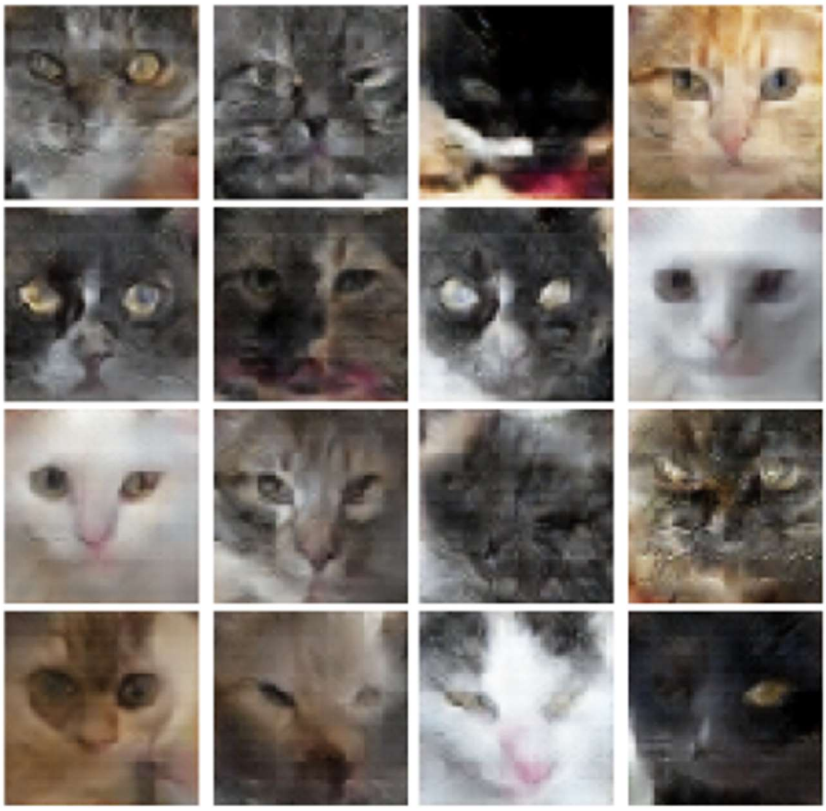
GAN and LSGAN results

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:

Iter: 14750, D: 0.1852, G:0.2654



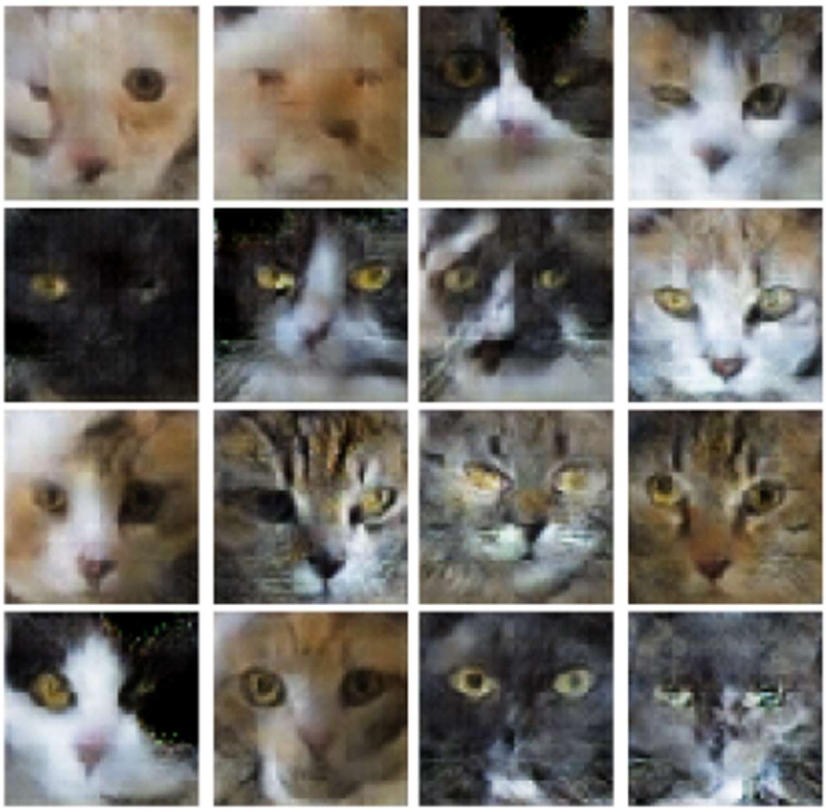
Discuss any differences you observed in quality of output or behavior during training of the two GAN models.

GAN and LSGAN's behaviors during training are similar, with not too severe mode collapse in some epochs. In addition, the outputs of LSGAN are slightly smoother than GAN's. The main difference I have observed is that the Discriminator ang Generator losses for GAN are higher than LSGAN's.

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. There are a lot of mode collapse examples during the early stages of the training for the GAN, for example:

EPOCH: 15
Iter: 7000, D: 1.17, G:1.127

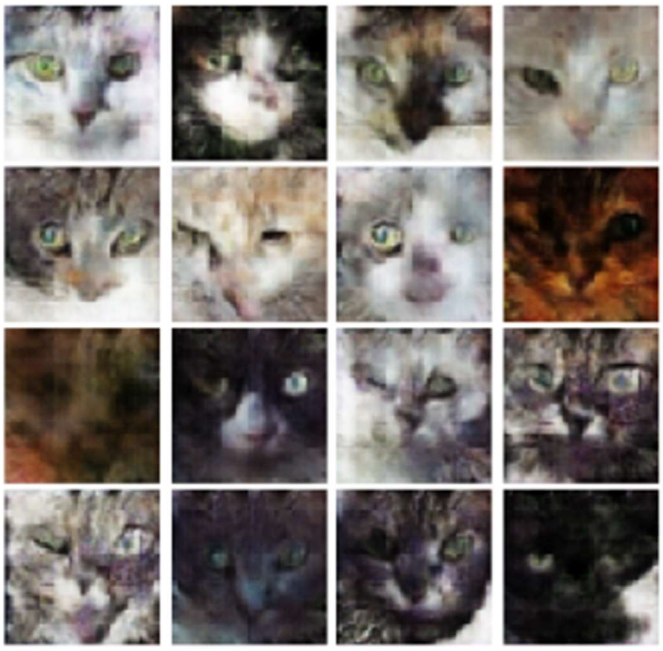


Even though the majority of the photos are decent enough, the top left photos show mode collapse.

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.

Spectral normalization helps the training a lot. The first image shown is an output from GAN implementation with only Batchnorm at 10 epochs:

Iter: 4750, D: 1.556, G:4.231



In contrast, here is the output with SpectralNorm:

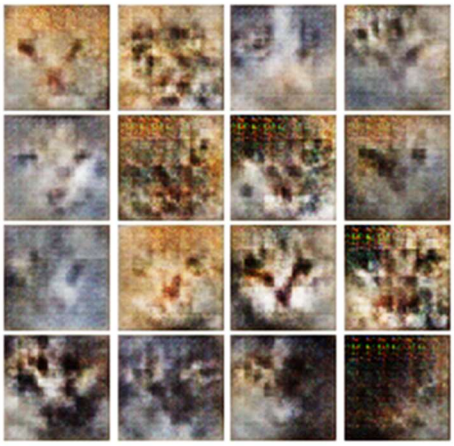
Iter: 4750, D: 1.292, G:0.8204

It is observable that SpectralNorm provides clearer details and more defined features.

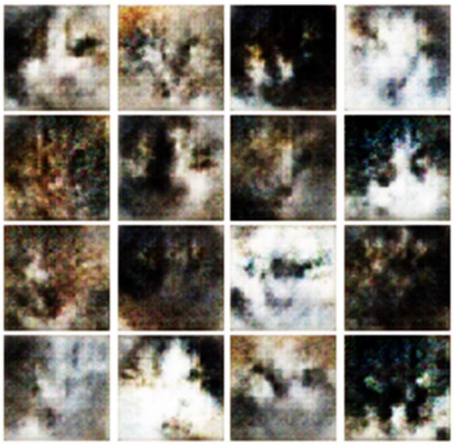
Extra credit: If you completed the extra credit for this portion, explain what you did (describing all model changes and hyperparameter settings) and provide output images.

I implemented WGAN for extra credit. Here are some examples:

```
EPOCH:  20
Iter: 9500, D: 0.1862, G:-0.06252
```



```
Iter: 9750, D: 0.1762, G:-0.04792
```



Although it did not quite achieve the same quality as both GAN and LSGAN, I believe if I had trained this for more than 100 epochs, my results would have been better. As far as I read about WGAN, it generally takes longer to train the model to get decent results as, computationally speaking, the Discriminator runs 5 times more than the Generator (5 was basically a hyperparameter). As far as the model, I did not change anything. But for train.py, I added a compute_grad_penalty function to calculate the gradient penalty to be combined with the real and fake losses of the Discriminator. I also changed the losses, since the WGAN paper does not implement log in their losses. I also added a weight clamping function to regulate the Discriminator weights. I also changed the optimizer to be RMSprop with a learning rate of 5e-5, instead of ADAM, as suggested in the paper.