Name: Chiranjeevi Konduru, Rishabh Agarwal
NetID(s): konduru4, ra26

GAN and LSGAN MNIST

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

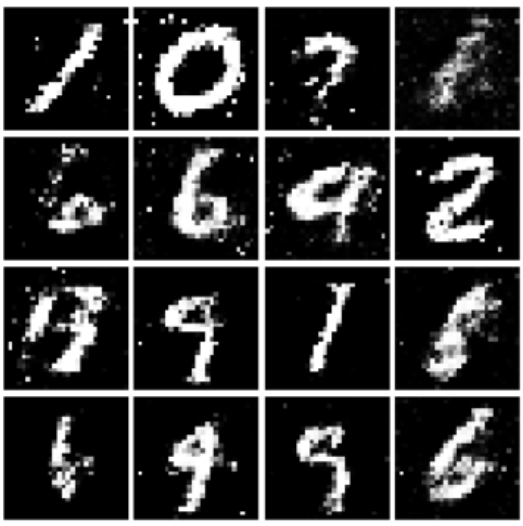
GAN – MNIST

```
Iter: 4500, D: 1.432, G:0.809
```

Learning rate	**$1e^{-3}$**
Noise_dim	**100**
Epochs	**10**
Optimizer	**Adam**
Batch_Size	**128**
loss	**GAN loss**

LSGAN – MNIST

```
Iter: 4500, D: 0.2226, G:0.179
```



Learning rate	**$1e^{-3}$**
Noise_dim	**100**
Epochs	**10**
Optimizer	**Adam**
Batch_Size	**128**
loss	**LSGAN loss**

GAN and LSGAN Cats

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

Note:

- These are results after 24000 iterations (total 24500 iterations), because I chose to show the 4x4 grid of images for every 1000 iterations, since showing the images for every 250 or 500 iterations is making PDF reach more than 100 pages and file size of 98 MB.
- **Data Transformation** techniques used on Cat dataset (**Applicable to all the GAN models where Cat Face Dataset is used**) are **RandomCrop**, **Resize**, **ColorJitter**, and **Gaussian Blur** (kernel_size = 3)

```
batch_size = 32
imsize = 64
cat_root = './cats'

cat_train = ImageFolder(root=cat_root, transform=transforms.Compose([
  transforms.ToTensor(),

  # Example use of RandomCrop:
  transforms.Resize(int(1.15 * imsize)),
  transforms.RandomCrop(imsize),
  transforms.ColorJitter(brightness=0.1, contrast=0.3, saturation=0.4, hue=0.05), #Color Jitter
  #transforms.RandomAffine(degrees=15, translate=(0.1, 0.1), scale=(0.9, 1.1), shear=10),
  transforms.GaussianBlur(kernel_size=3),
]))

cat_loader_train = DataLoader(cat_train, batch_size=batch_size, drop_last=True)
```

GAN – CAT (Final Output)

```
Iter: 24000, D: 0.03367, G:4.853
```

Hyperparameters:

Learning rate	**$1e^{-3}$**
Noise_dim	**100**
Epochs	**50**
Optimizer	**Adam**
Batch_Size	**32**
loss	**GAN loss**

LSGAN – CAT (Final Output)

```
Iter: 24000, D: 0.03709, G:0.5959
```

Hyperparameters:

Learning rate	**$1e^{-3}$**
Noise_dim	**100**
Epochs	**50**
Optimizer	**Adam**
Batch_Size	**32**
loss	**LSGAN loss**

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

- The discriminator and generator losses for LSGANs were quite low from the beginning and stable (no major fluctuations). However, for the Normal GAN the Generator loss was high from the range (1.8-5.6) and fluctuating frequently and Discriminator loss also started high (1.411 – epoch 1) but gradually decreased and was stable over the last epochs.
- The quality of images produced by LSGANs and regular GANs are almost similar, however for the few epochs in LSGANs the images had a greater detail (specifically comparing output after 15^{th} epoch in both the cases).
- LSGANs took noticeably lesser time to run(train) while having similar or better-quality outputs (in some cases).

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

- Yes, we have seen several cases of mode collapse, where the diversity of the generated images is poor and the same image (same face different color/ same eye color, etc.) or a small group of identical images repeatedly appear.

GAN Outputs:

There was one instance of mode collapse in **LS GANs**:

Extra Credit - Alternative GAN Formulation

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

Part 1: We **implemented** the **WGAN loss** for 1st part of extra credit.

Hyperparameters:

Learning rate	**$5e^{-4}$**
Noise_dim	**100**
Epochs	**40**
Optimizer	**Adam**
Batch_Size	**32**
Img_Size	**64**
loss	**WGAN loss**

Outputs:

```
EPOCH:  5
Iter: 2000, D: -5.292e+04, G:-1.935e+03
```

```
EPOCH:  39
Iter: 19000, D: -1.247e+06, G:-8.067e+04
```

Part 2: **Train your GAN on a different dataset**. I trained my LSGAN on CelebA dataset.

Data Transformations used (Applicable to all GAN models where CelebA dataset is used):
RandomCrop, **Resize**, **ColorJitter**, and **Gaussian Blur** (kernel_size = 5)

```
batch_size = 128
imsize = 64
faces_root = './celeb'
faces_train = ImageFolder(root=faces_root, transform=transforms.Compose([
   transforms.ToTensor(),

   # Example use of RandomCrop:
    transforms.Resize(int(1.1 * imsize)),
    transforms.RandomCrop(imsize),
    transforms.ColorJitter(brightness=0.1, contrast=0.3, saturation=0.4, hue=0.05),
    transforms.GaussianBlur(kernel_size=5)
 ]))

faces_loader_train = DataLoader(faces_train, batch_size=batch_size, drop_last=True)
```

Note: Tried adding few other transformations like Erasing and RandomAffine, the issue was with the black patches on images made it hard to generate outputs clearly. Even after 30 epochs the images had a black patch on them. So finally went with gaussian blur to smooth the high quality CelebA dataset.

Hyperparameters:

Learning rate	**$1e^{-4}$**
Noise_dim	**100**
Epochs	**30**
Optimizer	**Adam**
Batch_Size	**128**
Img_Size	**64**
loss	**LSGAN loss**

Outputs:

EPOCH: 6
Iter: 2000, D: 0.1882, G:0.1957

EPOCH: 20
Iter: 7000, D: 0.02141, G:0.5557

Final Output (30 Epoch)

Part 3: Spectral Normalization on CAT faces Dataset and Celeb Faces Dataset

As can be observed in our pictures, we think that spectrally normalizing our convolution layers helped to improve the quality of images we generated with our GAN. This is because Spectrally Normalized (SN) GANs are thought to produce images of higher quality than Normal DC GAN (for respective Epochs)
Reference: https://paperswithcode.com/method/spectral-normalization

Hyperparameters (Cat Data):

Learning rate	**$1e^{-4}$**
Noise_dim	**100**
Epochs	**50**
Batch_Size	**32**
Optimizer	**Adam**
Img_Size	**64**
loss	**LSGAN loss**

Outputs (Cat Faces Dataset)

EPOCH: 19
Iter: 9000, D: 0.06752, G:0.3259

EPOCH: 39
Iter: 19000, D: 0.01697, G:0.449

Hyperparameters (Celeb Faces Data):

Learning rate	**$1e^{-4}$**
Noise_dim	**100**
Epochs	**50**
Batch_Size	**128**
Optimizer	**Adam**
Img_Size	**64**
loss	**LSGAN loss**

Outputs (Celeb faces Dataset):

EPOCH: 20
Iter: 7000, D: 0.02863, G:0.4605

EPOCH: 37
Iter: 13500, D: 0.01392, G:0.4781