UDACITY

‹ Return to Classroom

DISCUSS ON STUDENT HUB

Generate Faces

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Congratulations on completing the GAN project. As you might have experienced GANs are advanced and complex topic. In this project, a few more changes such as dropout layers in discriminator, bigger generator, etc., will get you better results. You have done an awesome job so far.

Here some links to know GAN's better:
https://arxiv.org/pdf/1511.06434.pdf
https://blog.openai.com/generative-models/
https://www.youtube.com/watch?v=YpdP_0-IEOw
https://medium.com/@devnag/generative-adversarial-networks-gans-in-50-lines-of-code-pytorch-e81b79659e3f
https://deephunt.in/the-gan-zoo-79597dc8c347
http://guimperarnau.com/blog/2017/03/Fantastic-GANs-and-where-to-nd-them
Image Completion with Deep Learning in TensorFlow
http://bamos.github.io/2016/08/09/deep-completion/
Wasserstein GAN implementation in TensorFlow and Pytorch
https://wiseodd.github.io/techblog/2017/02/04/wasserstein-gan/
Stability of GAN
http://www.araya.org/archives/1183
Generative Adversarial Networks (GANs) - Computerphile
https://www.youtube.com/watch?v=Sw9r8CL98N0
GAN - intro Ian Goodfellow

https://www.youtube.com/watch?v=YpdP_0-IEOw&t=250s

Required Files and Tests

The project submission contains the project notebook, called “dlnd_face_generation.ipynb”.

All the unit tests in project have passed.

Data Loading and Processing

The function `get_dataloader` should transform image data into resized, Tensor image types and return a DataLoader that batches all the training data into an appropriate size.

Nice job resizing the images and creating a DataLoader.

Suggestion:

- Reduce batch size.
  A batch size of 32 more is appropriate for this architecture and learning rate. The batch size and learning rate often found linked, smaller batch size with a lower learning rate is more appropriate.

Pre-process the images by creating a `scale` function that scales images into a given pixel range. This function should be used later, in the training loop.

Images are scaled correctly. We are using tanh activation at the last layer of generator, so the pixels in the generated images are in the range of -1.0 to 1.0. Scaling the input images (real images) also into the same range will make the training more effective and faster.

Build the Adversarial Networks

The Discriminator class is implemented correctly; it outputs one value that will determine whether an image is real or fake.

Good:

- Used a sequence of three convolutional layers. The size(3-4 layers) is appropriate
- Used strided convolution instead of max pooling to downsample, making the network to learn its own pooling function.
- Used batch normalization starting from second layer.
- Leaky relu and batch norm promote healthy gradient flow.

Suggestion:

- Add dropout layers right after each leaky ReLU. I see commented it out. In fact, dropouts in discriminator help it generalize it better.
  See example here: https://github.com/udacity/deep-learning-v2-pytorch/blob/master/gan-mnist/MNIST_GAN_Solution.ipynb.

The Generator class is implemented correctly; it outputs an image of the same shape as the processed training data.

Good:

- Used a series of three strided covolutional transpose layers.
- Use ReLU activation
- Used tanh at the output to return in the range -1 to 1.

Suggestions:

- Make the generator at least one more layer bigger than discriminator or reduce the size of the discriminator
- Try bigger value of conv_dim, for generator.
  Example:

```
d_conv_dim = 32
g_conv_dim = 64
```

- Try different values of z_size between 64-256 for varied types of faces.

Tips and Tricks to make GANs work.
https://github.com/soumith/ganhacks

This function should initialize the weights of any convolutional or linear layer with weights taken from a normal distribution with a mean = 0 and standard deviation = 0.02.

Weights are initialized correctly.

Suggestion:

- Try xavier initializer
  Example: `init.xaviernormal(m.weight.data, gain=0.2)`

Ref: https://prateekvjoshi.com/2016/03/29/understanding-xavier-initialization-in-deep-neural-networks/

Optimization Strategy

The loss functions take in the outputs from a discriminator and return the real or fake loss.

Use label smoothing for calculating real_loss. This will prevent discriminator from being too strong by encouraging it to estimate soft probabilities.

Example: `labels = torch.ones(batch_size)*0.9`

More info:
http://www.inference.vc/instance-noise-a-trick-for-stabilising-gan-training/

There are optimizers for updating the weights of the discriminator and generator. These optimizers should have appropriate hyperparameters.

Nice choice of Adam, best optimizer for this scenario. Learning rate in the range of 0.0002-0.0008 works well with a batch size of 16-32. The batch size and learning rate are linked. The smaller batch size should go with low learning rate.

Check this article to know more about optimization in GAN.
https://towardsdatascience.com/understanding-and-optimizing-gans-going-back-to-first-principles-e5df8835ae18

Training and Results

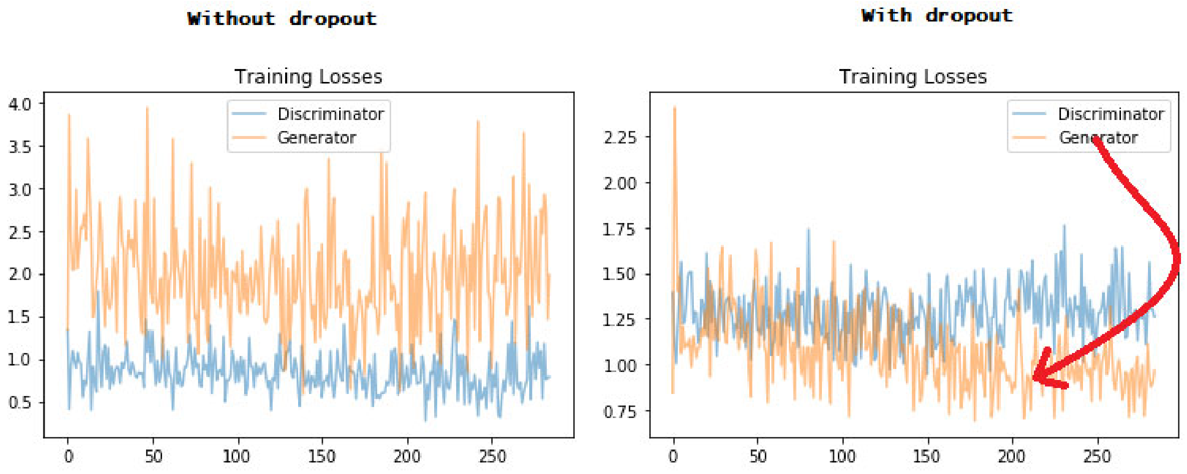
Real training images should be scaled appropriately. The training loop should alternate between training the discriminator and generator networks.

Nice job assembling everything together and make it work. GPU is used appropriately. 5-10 epochs are sufficient considering the low resolution of images. There won't be much significant improvement with increasing epochs.

There is not an exact answer here, but the models should be deep enough to recognize facial features and the optimizers should have parameters that help wth model convergence.

The two important factors must consider are dropouts and label smoothing. Adding dropouts after each leaky relu in discriminator will make it stable training and bring down the generator loss lower than discriminator loss.

See the significance of dropouts:



The project generates realistic faces. It should be obvious that generated sample images look like faces.

The generated images look like faces. Well done! In our case, with some tuning such as drop out layers in discriminator, bigger generator, and more importantly label smoothing, you can improve the results.

The question about model improvement is answered.

Excellent observations. GANs are a power-play between discriminator and generator. We want the generator to be slightly more dominant so that it can fool the discriminator with fake images. In other words, the generated images should look like real images.

With dropout layers and label smoothing, 3-5 epochs are sufficient in our case.

The diversity of the input data is significant as well as the number of feature maps and model. The racial bias is a huge concern in the AI community, check out the news articles below:

https://www.forbes.com/sites/parmyolson/2018/02/26/artificial-intelligence-ai-bias-google/#6303f72b1a01
https://www.nature.com/articles/d41586-018-05707-8
https://www.independent.co.uk/life-style/gadgets-and-tech/amazon-ai-sexist-recruitment-tool-algorithm-a8579161.html
https://www.theverge.com/2018/7/26/17615634/amazon-rekognition-aclu-mug-shot-congress-facial-recognition

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review