UDACITY

Return to "Deep Learning" in the classroom

Generate Faces

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Congratulations!

This is a great submission. Your implementation is quite good although a few changes can help you to further improve the results. I have added hints to help you out.

To further learn about GANs work in real world, please consider going through these urls:

GANs in the fashion industry:

- https://www.cs.toronto.edu/~urtasun/publications/zhu_etal_iccv17.pdf
- https://www.livemint.com/Companies/tchtq74FOkMM43szMIHh0M/At-Myntra-machines-tell-designers-how-to-make-clothes.html

Music generation using GANs:

- https://www.quora.com/Can-music-be-generated-using-generative-adversarial-networks

A list of all named GANs!

- https://deephunt.in/the-gan-zoo-79597dc8c347

Required Files and Tests

The project submission contains the project notebook, called "dlnd_face_generation.ipynb".

The IPython notebook is rightly submitted.

All the unit tests in project have passed.

Awesome!

The unit tests are running flawlessly. 👌

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.

The dataloader() method is well implemented. You have used the transforms correctly. The img_size, batch_size are well chosen.

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.

The rescaling of images is precisely done. Final pixel values of images lie in the range of -1 to 1.
Thumbs up for plotting the hisograms. 👍

Build the Adversarial Networks

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

The discriminator network is rightly coded. You have used the helper function rightly to build the network.

- The convolution layer and batch normalization are rightly used in the helper function.
- The usage of leaky relu in the `forward` method is correctly done.
- Final fully connected layer is rightly used.

Suggestions

To enhance the power of discriminator, I highly recommend you to consider adding a few more hidden layers. This increases the depth and helps it to learn better to discriminate.

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

The generator method is well implemented.
You have used the deconvolution layers & batch normalization rightly.

Suggestions

- It would be better to use dropout layers with the deconvolution layers.
- Consider adding 1-2 more hidden layers to the network. This improves the learning ability of the network and helps it to generate improved images.

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.

The weights initialization steps are perfectly done. Initializing them in a normal curve helps to make it better.

Optimization Strategy

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

The losses are rightly computed.

Suggestion (Important)

It's recommended to use label smoothening. This can definitely help the model. Please consider going through the Tips and tricks provided in this repo: https://github.com/soumith/ganhacks

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

The Adam Optimizer is rightly used to build the network.

```
lr = 0.0002
beta1=0.5
beta2=0.999 # default value
```

The values of hyperparameters are quite good. Although a few more changes can make it better.
The value of beta1 as 0.5 seems a bit high. It's noticed that network starts to generate better images and discriminate better with lower decay rate.

Training and Results

Training and Results

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

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 architecture of the network is acceptable. Although I would like to suggest you to try adding 1-2 more layers in the network. You can try adding layers with stride=1, padding=1/same.

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

Good Job!

The generated images are quite good and easily recognizable. Although there's still space for improvement. I recommend you to keep working on it and experiment the things that researchers have used to build GANs(go through the GANs listed on top)

The question about model improvement is answered.

Good to go through your observations:

- The dataset is biased; it is made of "celebrity" faces that are mostly white
    - This is actually one of the issues. If we get a balanced dataset with different racial faces then our model can generate better images.
- Model size; larger models have the opportunity to learn more features in a data feature space
    - You should try adding 1-2 more hidden layers. No need to go for bigger images, you can add hidden layers with stride=1, padding=1 (same). This will increase the depth i.e, number of filters but will maintain the width and height.

DOWNLOAD PROJECT