UDACITY

< Return to "Deep Learning" in the classroom

Generate Faces

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

👏🎓 **Great job!**

I have had a lot of fun reviewing your work! I hope you also had fun during your ND of deep learning!
You have shown your deep understanding on the architecture of GANs!

Just in case you want to do some further studies, I'll send a bunch of links.
Regards 😬

Here there is a great video about GAN: https://www.youtube.com/watch?v=X1mUN6dD8uE
I recommend checking the Google Python Style Guide, there are great tips about how to improve coding, in general: https://google.github.io/styleguide/pyguide.html
GANs for beginners video: https://www.oreilly.com/learning/generative-adversarial-networks-for-beginners

Advanced tips:
How to Train a GAN: https://github.com/soumith/ganhacks
How to to select the batch_size vs the number of epochs:
https://stats.stackexchange.com/questions/164876/tradeoff-batch-size-vs-number-of-iterations-to-train-a-neural-network
GAN stability: http://www.araya.org/archives/1183
MNIST GAN with Keras: https://medium.com/towards-data-science/gan-by-example-using-keras-on-tensorflow-backend-1a6d515a60d0
DCGAN : https://github.com/yihui-he/GAN-MNIST, https://github.com/carpedm20/DCGAN-tensorflow
DiscoGAN, Discover Cross-Domain Relations with Generative Adversarial Networks:
https://github.com/carpedm20/DiscoGAN-pytorch
beta1 values: https://arxiv.org/pdf/1511.06434.pdf
WGANs: https://paper.dropbox.com/doc/Wasserstein-GAN-GvU0p2V9ThzdwY3BbhoP7

Good articles :
https://blog.openai.com/generative-models/
https://medium.com/@ageitgey/abusing-generative-adversarial-networks-to-make-8-bit-pixel-art-e45d9b96cee7

Do you want your deep net to sing? Have a look at this paper:
http://www.creativeai.net/posts/W2C3baXvf2yJSLbY6/a-neural-parametric-singing-synthesizer
An app called FaceApp uses a CNN to make you smile in a picture or change genders:
http://www.digitaltrends.com/photography/faceapp-neural-net-image-editing/

Here you see examples of Stanfords' student projects on deep learning: https://cs230.stanford.edu/proj-spring-2018.html

Required Files and Tests

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

all files ready!

All the unit tests in project have passed.

all tests 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.

correct size

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.

correctly scaled! 👍

Build the Adversarial Networks

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

Great work! You have used batch_normalization, leaky relus and sigmoid activation on a dense layer!

Original Paper on batch normalization: https://arxiv.org/pdf/1502.03167.pdf

Beyond the intuitive reasons, there are good mathematical reasons why it helps the network learn better, too. It helps combat what the authors call internal covariate shift. This discussion is best handled in the paper and in Deep Learning(http://www.deeplearningbook.org) a book you can read online written by Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Specifically, check out the batch normalization section of Chapter 8: Optimization for Training Deep Models (http://www.deeplearningbook.org/contents/optimization.html).

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

Awesome work!
- [x] batch_normalization
- [x] leaky relus
- [x] tanh activation

kernel size: https://www.quora.com/How-can-I-decide-the-kernel-size-output-maps-and-layers-of-CNN

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.

Optimization Strategy

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

Great job on this difficult function!

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

My suggestions on hyperparams on this project are:
batch_size: 16, 32, 64

```
* If you choose a batch size too small then the gradients will become more unstab
le and you would need to reduce the learning rate. So batch size and learning rat
e are linked.
* Also if one use a batch size too big then the gradients will become less noisy
 but it will take longer to converge.
```

z_dim: 100-128
learning_rate: 0.0002 - 0.0008
Lowering the learning rate would require more epochs (in this project you are asked not to modify nb of epochs), but could ultimately achieve better accuracy.
beta1: about 0.5 see: https://arxiv.org/pdf/1511.06434.pdf (Student also reported good results with values like 0.2 or 0.3)

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

Your GAN generates solid images. You have proven to understand the basic concepts on GAN architectures!

The question about model improvement is answered.

DOWNLOAD PROJECT

RETURN TO PATH