‹ Return to "Deep Learning" in the classroom

Generate Faces

REVIEW | CODE REVIEW | HISTORY

Meets Specifications

> Congratulations!🎉 This is Very Good submission 😄.
> Very few people achieve this task with such perfection 👍, Kudos!

Here are some intermediate to advanced articles which may furthur your understanding of GANs:
(please have a look)
Original DCGAN paper : https://arxiv.org/pdf/1511.06434.pdf
GAN Training hacks: https://github.com/soumith/ganhacks
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 (pytorch): https://github.com/carpedm20/DiscoGAN-pytorch
WGAN (Intro) : http://wiseodd.github.io/techblog/2017/02/04/wasserstein-gan/
WGAN (pytorch) : https://github.com/martinarjovsky/WassersteinGAN
For Advances Learners: https://blog.openai.com/generative-models/
http://bamos.github.io/2016/08/09/deep-completion/

> Happy DeepLearning 😄, All the best for your future endeavours in ai 😀 🍻

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.**

✓ **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.

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 Generator class is implemented correctly; it outputs an image of the same shape as the processed training data.
- ✓ 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.
- ✓ There are optimizers for updating the weights of the discriminator and generator. These optimizers should have appropriate hyperparameters.

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.

✓ The question about model improvement is answered.

DOWNLOAD PROJECT

RETURN TO PATH