Meets Specifications

Congratulations!  You've passed this project. Fantastic Work here!! This is a great submission. Your concepts of DCGAN are crystal clear. I've suggested a few more tips.

Also, keep studying about the topic as this is just the beginning. I have also given some more **tips** to further improve your project.

Moreover, here're a few resources to help you continue this wonderful journey:

* How to Train a GAN: <https://github.com/soumith/ganhacks>
* Stability of GANs: <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>
* <https://blog.openai.com/generative-models/>
* <https://medium.com/@ageitgey/abusing-generative-adversarial-networks-to-make-8-bit-pixel-art-e45d9b96cee7>

I really hope you enjoyed studying Deep Learning, the hottest topic in AI right now, here with Udacity 

Until next time! Have an amazing time working with neural nets.

**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.**