Meets Specifications

You did great on this submission! Congratulations on finishing the final project 

Some resources to further improve your project:

* This repository <https://github.com/carpedm20/DCGAN-tensorflow> contains a very similar DCGAN Tensorflow implementation, you can use this as a reference for tweaking your GAN further.
* This blog post <http://guimperarnau.com/blog/2017/03/Fantastic-GANs-and-where-to-find-them> discusses the major developments in the improvement of GAN-generated images, you could try to implement the discussed techniques.

**Required Files and Tests**

**The project submission contains the project notebook, called “dlnd\_face\_generation.ipynb”.**

**All the unit tests in project have passed.**

**Build the Neural Network**

**The function model\_inputs is implemented correctly.**

Good work! You have created the placeholders with the right shapes and data types and given them some nice names.

**The function discriminator is implemented correctly.**

Perfect implementation of the discriminator, good to see you have used batch normalization and leaky relu!

**The function generator is implemented correctly.**

Also the generator is correctly implemented 

Note that often it turns out the discriminator is much better at its job than the generator, in that case you can choose to make the generator 'smarter' than the discriminator by having fewer layers in the discriminator.

**The function model\_loss is implemented correctly.**

Also well done here! You could consider to at some label smoothing here by multiplying the ones in d\_loss\_real by a number a bit smaller than 1.

**The function model\_opt is implemented correctly.**

Good job! You wrapped the training step with a dependency to tf.GraphKeys.UPDATE\_OPS, this is essential for the batch normalization to work properly and get good results from your GAN. Check out <http://ruishu.io/2016/12/27/batchnorm/> for more information on this topic.

**Neural Network Training**

**The function train is implemented correctly.**

* **It should build the model using model\_inputs, model\_loss, and model\_opt.**
* **It should show output of the generator using the show\_generator\_output function**

Your training code looks good, just one thing to improve: the learning rate is hardcoded, it's better to use the learning rate parameter.

**The parameters are set reasonable numbers.**

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