



PROJECT

Generate Faces

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!  

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 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/>

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!

Build the Neural Network

The function model_inputs is implemented correctly.

perfect!

The function discriminator is implemented correctly.

👍 Great work! You have used batch_normalization, leaky_relu and sigmoid activation on a dense layer!

The function generator is implemented correctly.

Awesome work!

✔ batch_normalization

✔ leaky_relu

✔ tanh activation

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

Just one remark on coding style: `reuse=True if is_train==False else False` You could write this simpler: `reuse = True if not is_train else False` (better programming style!)

The function model_loss is implemented correctly.

Great job on this difficult function! 👍

The function model_opt is implemented correctly.

Good idea using tf.control_dependencies!

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

👍 Great implementation! You have also rescaled batch_image data from [-0.5, 0.5] to [-1, 1]

The parameters are set reasonable numbers.

Absolutely reasonable values on your 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 unstable and you would need to reduce the learning rate. So batch size and learning rate 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.

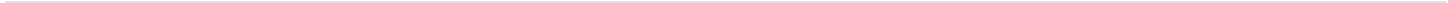
beta1: about 0.5 see: <https://arxiv.org/pdf/1511.06434.pdf>

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

Your GAN generates realistic looking images. You have proven to understand the architecture of GANs!

↓ DOWNLOAD PROJECT

RETURN TO PATH



[Student FAQ](#)