UDACITY

PROJECT

Generate Faces

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

> Congratulations! This is V 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".

Good, all files are in place.

All the unit tests in project have passed.

Viola!. you passed all unit tests. 👍

Build the Neural Network

The function model_inputs is implemented correctly.

Good job, model inputs are implemented correctly.

Often I find students confused between `tf.Variable` and `tf.placeholder`. This link gives correct usecase for both.

The function discriminator is implemented correctly.

Well done!, Good work.
Good implementation of CNN and its layers.
Good use of `Batch Normalisation` and `Leaky Relu` in the CNN which will improve efficiency of discriminator.
Suggestion : `Dropout` with keep probability of 0.7 or 0.8 and `Xavier's Initialisation` of weights will further improve the network

Xavier can be implemented by passing `tf.contrib.layers.xavier_initializer()` as the value for the `kernel_initializer` parameter in `tf.layers.conv2d`. This blog explains concepts of Xavier initializer clearly
Dropout can be implemented by adding dropout layer right after Leaky Relu of each layer by function `tf.nn.dropout`

The function generator is implemented correctly.

Well done!, Good work.
Good implementation of CNN and its layers.
Good use of `Batch Normalisation` and `Leaky Relu` in the CNN which will help generator to produce good images.
Suggestion : `Dropout` with probability less than 0.8 and `Xavier's Initialisation` of weights will further improve the network

Xavier can be implemented by passing `tf.contrib.layers.xavier_initializer()` as the value for the `kernel_initializer` parameter in `tf.layers.conv2d`. This blog explains concepts of Xavier initializer clearly
Dropout can be implemented by adding dropout layer right after Leaky Relu of each layer by function `tf.nn.dropout`

The function model_loss is implemented correctly.

Nice work!
In `tf.reduce_mean` function, Multiplication of real labels of discriminator by 0.9 to smoothen them is good effort. 👍

The function model_opt is implemented correctly.

Excellent! `Adam optimiser` is implemented correctly, for optimising the GAN.
You can learn more about optimisers here : http://sebastianruder.com/optimizing-gradient-descent/

You can learn more about these `update_ops` and its influence on `batch norm` layers here : http://ruishu.io/2016/12/27/batchnorm/.

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**

Excellent! Good job in implementing train function 👍.
Small details like sampling `z` from -1 to +1 , multiplying the `batch_image` by 2 as we need to `normalise real images from -1 to +1` are handled very well 👏.

The parameters are set reasonable numbers.

Various parameter are set in correct way 👏.You can furthur improve your parameters with these suggestions.

Always set `parameters` in powers of 2 like 4,8,16,32,64..... This will help tensorflow to optimise the computation. Batch size influences the quality of image generated by GAN.
Suggestions :

1. For celebA dataset, as it contains large images `Batch size` of 16 or 32 would be good
2. For MNIST dataset, 28*28 black and white images `Batch size` of 32 or 64 would be good
3. In GAN, `learning rate` of `0.0002` is preferred. `Beta` equal to or less than `0.5` is preferred. `You can tweak them for better results` .
4. Size of `z` vector needs to be `100` or `128` for optimal results

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

Good results 👍👏

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

Student FAQ