UDACITY

PROJECT

Generate Faces

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

I have considered your first submission. I am really sorry for previous review.

Congratulations You passed this Project and Nanodegree. I hope this was a great experience for you.

Some Links that will help you get better understanding of GAN's:

1. https://arxiv.org/pdf/1511.06434.pdf
2. https://blog.openai.com/generative-models/
3. https://www.youtube.com/watch?v=YpdP_0-IEOw
4. https://medium.com/@devnag/generative-adversarial-networks-gans-in-50-lines-of-code-pytorch-e81b79659e3f

Links from where you can see all variants of GAN's:

1. https://deephunt.in/the-gan-zoo-79597dc8c347
2. http://guimperarnau.com/blog/2017/03/Fantastic-GANs-and-where-to-find-them

Required Files and Tests

The project submission contains the project notebook, called "dlnd_face_generation.ipynb".

All the unit tests in project have passed.

Great work. Unit testing is very good practice to ensure that your code is free from bugs and prevent you from wasting a lot of time while debugging minor problems. Unit test also help in improving our code standards. For more details read this and I would recommend you to continue using unit testing in every module that you write in future, to keep it clean and speed up your development. Unit testing is highly motivated in industries.

It is not always that if you passed unit test your code is okay, there can be some errors.

Build the Neural Network

The function model_inputs is implemented correctly.

Good Work!!

(Following abstract is from Tensorflow documentation)
TensorFlow programs use a tensor data structure to represent all data -- only tensors are passed between operations in the computation graph. You can think of a TensorFlow tensor as an n-dimensional array or list. A tensor has a static type, a rank, and a shape.

In the TensorFlow system, tensors are described by a unit of dimensionality known as rank. Tensor rank is not the same as matrix rank. Tensor rank (sometimes referred to as order or degree or n-dimension) is the number of dimensions of the tensor. For example, the following tensor (defined as a Python list) has a rank of 2:

```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

A rank two tensor is what we typically think of as a matrix, a rank one tensor is a vector. For a rank two tensor you can access any element with the syntax t[i, j]. For a rank three tensor you would need to address an element with t[i, j, k].

Link! that might help you with better understanding of rank in Tensor.

The function discriminator is implemented correctly.

Good Work!! Perfectly implemented

Link to get an idea on implementing leaky relu.

Link to get an idea for implementing better DCGAN.

Batch normalization is recommended in DCGAN model (as mentioned in original paper). Batch normalization helps as we initialize the BatchNorm Parameters to transform the input to zero mean/unit variance distributions but during training they can learn that any other distribution might be better.
Why does this work?
Answer Well, we know that normalization (shifting inputs to zero-mean and unit variance) is often used as a pre-processing step to make the data comparable across features. As the data flows through a deep network, the weights and parameters adjust those values, sometimes making the data too big or too small again - a problem the authors refer to as "internal covariate shift". By normalizing the data in each mini-batch, this problem is largely avoided.

Link to know details about Batch Normalization.

The function generator is implemented correctly.

Github Link: Here you can find all good tips and tricks to make GANs work.

You should be comfortable with Deconvolution, Deconvolution layer is a very unfortunate name and should rather be called a transposed convolutional layer.

Link
Link

The function model_loss is implemented correctly.

Good Work

Why do we use label smoothing?
Answer: To encourage the discriminator to estimate soft probabilities rather than to extrapolate to extremely confident classification, we can use a technique called one-sided label smoothing.

Link to understand noise in details.

The function model_opt is implemented correctly.

Correctly implemented!! Good Work.

- You took care of all trainable variables. ☑
- Used Adam Optimizer. ☑

Important Note
When is_training parameter of batch_normalisation is true the moving_mean and moving_variance need to be updated, by default the update_ops are placed in tf.GraphKeys.UPDATE_OPS so they need to be added as a dependency to the g_train_op, thus the location chosen by you is correct. Here's a link for more details and a possible implementation.

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

Good Work!! You have done a very appreciable work here.

The parameters are set reasonable numbers.

Choice of Hyper Parameter is reasonable.

But I would suggest you should consider decreasing batch size because,

- If you choose a batch size too small then the gradients will become more unstable and 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.

I would recommend you to read http://leon.bottou.org/research/stochastic

I would suggest using beat1 in range [0.1 - 0.3]. Here's a good post explaining the importance of beta values and which value might be empirically better. Also try lowering it even further, ~0.1 might even produce better results.

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

Your results are good. Keep it up.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

Student FAQ