UDACITY

PROJECT

Generate Faces

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

It's an outstanding work! Keep it up!

I suggest you check out the following Ian Goodfellow's tutorial on GANs. It has a number of great tips.

Congratulations on almost finishing Deep Learning Nanodegree Foundations program! I hope you've thoroughly enjoyed this journey :)

Required Files and Tests

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

All the unit tests in project have passed.

Well done! All unit tests have passed!

Build the Neural Network

The function model_inputs is implemented correctly.

Good job!

It is not necessary to pass a shape parameter when defining a scalar placeholder (like learning rate).

The function discriminator is implemented correctly.

Well done on using batch normalization and a leaky ReLU (rather than a vanilla ReLU). This is important since it helps the gradient flow through the network, which in turn is crucial for the network's ability to learn.

BTW, note that since we repeatedly use a leaky ReLU activation function, it would make sense to factor it out into a separate function.

The function generator is implemented correctly.

Perfect!

The function model_loss is implemented correctly.

Well done on multiplying `labels` (for `d_loss_real`) by a smoothing factor (0.9, for instance). This helps optimizing this loss for the following reason: initially the generator network does not produce anything close to the real input images; hence, the discriminator quickly learns to distinguish between real inputs and generated inputs - outputting a probability close to 1; hence cross-entropy loss will involve the following computation: `log(some_very_small_number)`, which can be unstable.

The function model_opt is implemented correctly.

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

OPTIONAL
You should pass the learning rate placeholder (created by `model_inputs` function; `lr`, in your case) to `model_opt` rather than the float. It is not a mistake per se. However, it reduces model's flexibility. Suppose, you want to tweak the value of the learning rate after N iterations. If you use a placeholder, you can easily pass an appropriate value of the learning rate to the `feed_dict` parameter of `sess.run` to achieve that.

Note that you then pass `lr:learning_rate` to the `feed_dict`, which is redundant since you didn't use a placeholder to create a model, but rather a float `learning_rate`.

The parameters are set reasonable numbers.

It's a great set of hyperparameters! I hope that you played with different values to see how it affects training.

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

Well done! Your model does a remarkable job generating realistic faces.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

Student FAQ