UDACITY

‹ Return to "Deep Learning" in the classroom

DISCUSS ON STUDENT HUB

Generate Faces

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Well Done !!! This is a great first submission. Congratulations on completing the project.

In order to gain more intuition about GANs in general, I would suggest you take a look at this blog post. Also, if you want to gain intuition about the convolution and transpose convolution arithmetic, I would suggest referring to this paper. For more advanced techniques on training GANs, you can refer to this paper.

All the best for your next project. Keep learning.

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.

`model_inputs` has correctly defined the placeholder tensors for real input, z input and the learning rate.

The function discriminator is implemented correctly.

Well Done !!! The function discriminator is implemented correctly. Below are the good points of the architecture chosen:

- You have used Leaky ReLU as the activation function for the convolution layers which helps with the gradient flow. It helps alleviate the problem of sparse gradients.
- You are using the same size filters across all the layers.
- You have used batch normalization which stabilizes GAN training.
- You have used Sigmoid as the activation function for the output layer which produces probability-like values between 0 and 1.

However, you could improve the architecture:

- Instead of using random normal initializer for conv layers, I would recommend using Xavier initialization.
- Try adding dropout layers with a small dropout rate between 10% and 30%. This should help improve the stability of the model.

The function generator is implemented correctly.

Well Done Implementing the generator. Conceptually, the generator is implemented correctly. However, you could improve it further for better results:

- Similar to the discriminator, please use Xavier initialization instead of the random normal initialization.
- Please add more convolution transpose layers. Using a simple 3-layer generator is not going to help generate better images. Apart from this, generators which are sufficiently larger than the discriminator in terms of depth, help generate better samples. So, I would recommend adding at least 2 more layers to the generator.

The function model_loss is implemented correctly.

Good Job !!! However, you should have used one-sided label smoothing for the discriminator real loss and not for the generator loss as discussed in this paper.

The function model_opt is implemented correctly.

Nice work using control dependency on update ops before optimization. Normally, we scale the inputs to a neural network between 0 and 1. However, as the data moves through multiple layers, it starts shifting from

that distribution and the deeper layers start getting data in a different range as input. This is known as internal covariate shift. To combat this, a technique known as Batch normalization was introduced, where the inputs to the layer are scaled between 0 and 1. The moving mean and variance need to be updated for the technique. In Tensorflow, these operations are part of update ops and these need to be updated prior to the optimization. So, we add control dependencies on the update ops before optimizing the network. Please refer to this post for more information.

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**

You have correctly pieced together all the components to create the model. Also, you have correctly scaled the input images to the same range as the generated images. Well Done !!!

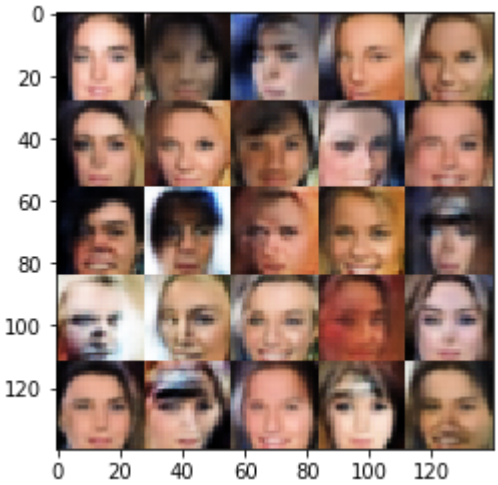
The parameters are set reasonable numbers.

The parameters are good but can be tweaked a little.

- Batch size is a little high. Using higher batch sizes might not give good results as GANs are difficult to train. Please use batch sizes of either 32 or 64.
- The value for z_dim is fine. However, if you wish to generate more varied face shapes, you can use a value of up to 200.
- Learning rate is fine. However, you could lower it a little for the face generation training.
- Did you try other values for beta1? I would recommend trying out other values in the range 0.3 - 0.6 and see if that helps generate better images.

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

The image generated does look like a face. However, it could be much clear. You can attempt at generating faces like below:



DOWNLOAD PROJECT

RETURN TO PATH

Rate this review