Generate Faces

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Required Files and Tests

✓

The project submission contains the project notebook, called “dlnd_face_generation.ipynb”.

Good 👏

✓

All the unit tests in project have passed.

Great 👍 all unit tests are passing successfully.

Data Loading and Processing

✓

The function `get_dataloader` should transform image data into resized, Tensor image types and return a

DataLoader that batches all the training data into an appropriate size.

Good 👏 you have used Image Folder wrapper. But also remember the DataLoader approach that used in the assignment of dog breed classification to understand the basics of the framework.

✓

Pre-process the images by creating a `scale` function that scales images into a given pixel range. This function should be used later, in the training loop.

Great 👍

Build the Adversarial Networks

✓

The Discriminator class is implemented correctly; it outputs one value that will determine whether an image is real or fake.

good 👌 you can also refer explaination from here

✓

The Generator class is implemented correctly; it outputs an image of the same shape as the processed training data.

good 👌 you can also refer explaination from here

✓

This function should initialize the weights of any convolutional or linear layer with weights taken from a normal distribution with a mean = 0 and standard deviation = 0.02.

Good 👏

```
if classname.find("Conv") != -1 or classname.find('Linear') != -1:
    torch.nn.init.normal_(m.weight.data, 0.0, 0.02)
elif classname.find("BatchNorm2d") != -1:
    torch.nn.init.normal_(m.weight.data, 1.0, 0.02)
    torch.nn.init.constant_(m.bias.data, 0.0)
```

Optimization Strategy

✓

The loss functions take in the outputs from a discriminator and return the real or fake loss.

Good that you have used BCE.
This is reference to accelerate your model but this is just a reference for future caselink

✓

There are optimizers for updating the weights of the discriminator and generator. These optimizers should have appropriate hyperparameters.

Great 👍
for reference, you can visit the site.

Training and Results

✓

Real training images should be scaled appropriately. The training loop should alternate between training the discriminator and generator networks.

Well done 👍

✓

There is not an exact answer here, but the models should be deep enough to recognize facial features and the optimizers should have parameters that help wth model convergence.

Good 👏the models are deep enough to recognize facial features

✓

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

Generator is creating proper realistic faces

✓

The question about model improvement is answered.

superb 🔥

DOWNLOAD PROJECT

RETURN TO PATH