UDACITY

Return to Classroom

Face Generation

REVIEW

HISTORY

Meets Specifications

good job :)

Section 1: Code quality

Scripts have an intuitive, easy-to-follow structure with code separated into logical functions. Naming for variables and functions follows the PEP8 style guidelines.

Section 2: Generator and discriminator design

The generator should take a batched 1d latent vector as input and output a batch of RGB images (3 channels).

The discriminator should take as input a batch of images and output a score for each image in the batch.

Generator and Discriminator are inheriting from the torch `Module` class. The layers are defined in the `init` method and called in the `forward` method.

Section 3: Data pipeline

(Provide specific details on how the student will fulfill the criteria.)
The `get_transform` function should output a `Compose` of different torchvision (or non torchvision) transforms.

The custom dataset should have the `__len__` and the `__get_item__` methods implemented and working. The dataset should return a tensor image in the -1 / 1 range.

Section 4: Loss implementation and training

Both loss functions are accomplishing their roles: the discriminator should be trained to separate fake and real images and the generator should be trained to fool the discriminator. No specific functions are required.

Two optimizers are created, one for the generator and one for the discriminator. They are both using low learning rates.

The `discriminator_step` and `generator_step` functions are correctly implemented. The model is training for enough epochs.

The student makes reasonable decisions based on initial results.

The generated samples should have face attributes (eyes, nose, mouth, hair) and a rough resemblance to a face.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START