UDACITY

‹ Return to Classroom

# Face Generation

REVIEW

HISTORY

## Meets Specifications

Congratulations! Your project meets all the requirements and it is able to sample decent faces. Keep up the good work!

PD: Do not worry about the reference, your code speaks for itself.

## 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.**

Good work. Code follows good programming practices, it is clear and uses type hints.

## 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 generator takes 1D latent vectors and outputs them to 3-channel RGB images.
✓ The generator uses transpose convolutions.
✓ The generator uses an activation function to map the output to (-1, 1).
✓ The generator is not too complex nor too simple.

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

✓ The discriminator takes images and outputs a score for each one.
✓ The discriminator downsamples the images using convolutions. You can try pooling methods too, as an alternative.
✓ The discriminator is not too complex nor too simple.

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

Both discriminator and generator inherit from the Module class and are implemented correctly.

## 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.**

Correct transform data pipeline. No data augmentation was used and Normalize is used correctly.

**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.**

Nicely done!

## 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.**

Correct implementation. Gradient penalty was used.

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

Both optimizers correctly use the Adam optimizer with proper parameters.

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

Proper and easy-to-read implementation. Five epochs are more than enough for this exercise.

**The student makes reasonable decisions based on initial results.**

Nice analysis and sensible suggestions to try in the future.

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

The model clearly generates recognizable faces, with the problems described in the questions section.

DOWNLOAD PROJECT

RETURN TO PATH