UDACITY

PROJECT SPECIFICATION

Generate Faces

Required Files and Tests

CRITERIA	MEETS SPECIFICATIONS
Have all project files been included with the submission?	The project submission contains the project notebook, called “dlnd_face_generation.ipynb”.
Have all the unit tests in the project passed?	All the unit tests in project have passed.

Data Loading and Processing

CRITERIA	MEETS SPECIFICATIONS
Has `get_dataloader` been implemented?	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.

CRITERIA	MEETS SPECIFICATIONS
Has the `scale` function been implemented?	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.

Build the Adversarial Networks

CRITERIA	MEETS SPECIFICATIONS
Does the **discriminator** discriminate between real and fake mages?	The Discriminator class is implemented correctly; it outputs one value that will determine whether an image is real or fake.
Does the **generator** generate fake mages?	The Generator class is implemented correctly; it outputs an image of the same shape as the processed training data.
Is the weight initialization function implemented correctly?	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.

Optimization Strategy

CRITERIA	MEETS SPECIFICATIONS
Are the `real_loss` and `fake_loss` functions implemented correctly?	The loss functions take in the outputs from a discriminator and return the real or fake loss.
Are appropriate optimizers defined for the networks?	There are optimizers for updating the weights of the discriminator and generator. These optimizers should have appropriate hyperparameters.

Training and Results

CRITERIA	MEETS SPECIFICATIONS
Are all adversarial networks trained correctly?	Real training images should be scaled appropriately. The training loop should alternate between training the discriminator and generator networks.
Do all models and optimizers have reasonable hyperparameters?	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.

CRITERIA	MEETS SPECIFICATIONS
Does the project generate realistic faces?	The project generates realistic faces. It should be obvious that generated sample images look like faces.
How could your model improve?	The **question** about model improvement is answered.

Suggestions to Make Your Project Stand Out!

- Create a deeper model and use it to generate larger (say 128x128) images of faces.
- Read existing literature to see if you can use padding and normalization techniques to generate higher-resolution images.
- Implement a learning rate that evolves over time as they did in this CycleGAN Github repo.
- See if you can extend this model and use a CycleGAN to learn to swap different kinds of faces. For example, learn a mapping between faces that have and do not have eye/lip makeup, as they did in this paper.