UDACITY

‹ Return to Classroom

Generate Faces

REVIEW

CODE REVIEW

HISTORY

▾ problem_unittests.py

```
from unittest.mock import MagicMock, patch
import numpy as np
import torch


def _print_success_message():
    print('Tests Passed')


class AssertTest(object):
    def __init__(self, params):
        self.assert_param_message = '\n'.join([str(k) + ': ' + str(v) + '' for k, v i

    def test(self, assert_condition, assert_message):
        assert assert_condition, assert_message + '\n\nUnit Test Function Parameters\


def test_discriminator(Discriminator):
    batch_size = 50
    conv_dim=10
    D = Discriminator(conv_dim)

    # create random image input
    x = torch.from_numpy(np.random.randint(1, size=(batch_size, 3, 32, 32))*2 -1).flo

    train_on_gpu = torch.cuda.is_available()
    if train_on_gpu:
```

```python
        x.cuda()

    output = D(x)
    assert_test = AssertTest({
        'Conv Dim': conv_dim,
        'Batch Size': batch_size,
        'Input': x})

    correct_output_size = (batch_size, 1)
    assert_condition = output.size() == correct_output_size
    assert_message = 'Wrong output size. Expected type {}. Got type {}'.format(correc
    assert_test.test(assert_condition, assert_message)

    _print_success_message()

def test_generator(Generator):
    batch_size = 50
    z_size = 25
    conv_dim=10
    G = Generator(z_size, conv_dim)

    # create random input
    z = np.random.uniform(-1, 1, size=(batch_size, z_size))
    z = torch.from_numpy(z).float()

    train_on_gpu = torch.cuda.is_available()
    if train_on_gpu:
        z.cuda()
    #b = torch.LongTensor(a)
    #nn_input = torch.autograd.Variable(b)

    output = G(z)
    assert_test = AssertTest({
        'Z size': z_size,
        'Conv Dim': conv_dim,
        'Batch Size': batch_size,
        'Input': z})

    correct_output_size = (batch_size, 3, 32, 32)
    assert_condition = output.size() == correct_output_size
    assert_message = 'Wrong output size. Expected type {}. Got type {}'.format(correc
    assert_test.test(assert_condition, assert_message)

    _print_success_message()

```

RETURN TO PATH