File - performance_testv2

```
C:\Users\usuario\Anaconda3\envs\tfq\python.exe C:/Users/usuario/qGAN/quantumGAN/performance_testing/performance_testv2.py
[0.30546899 0.84043945 0.66654435 0.96177867 0.78028236 0.64278492]
Epoch 0: Loss: [-0.43007618] [0.43246163 0.         0.40837062 0.        ] [0.01123047 0.73681641 0.20996094 0.04199219] [0.16602186] [0.16884436]
Epoch 1: Loss: [-0.36970994] [0.41347372 0.         0.41008435 0.        ] [0.00927734 0.71777344 0.23339844 0.03955078] [0.23815121] [0.23488393]
Epoch 2: Loss: [-0.33187365] [0.41347372 0.         0.41008435 0.        ] [0.01171875 0.72265625 0.22216797 0.04345703] [0.30729966] [0.29418543]
Epoch 3: Loss: [-0.31119143] [0.44583337 0.         0.40754034 0.        ] [0.00830078 0.70996094 0.22949219 0.05224609] [0.36283901] [0.3424888]
Epoch 4: Loss: [-0.29763083] [0.47735513 0.         0.47265937 0.        ] [0.00830078 0.72265625 0.22314453 0.04589844] [0.40938011] [0.37968598]
Epoch 5: Loss: [-0.28999453] [0.4502122  0.         0.48988201 0.        ] [0.01074219 0.70507812 0.24365234 0.04052734] [0.44409205] [0.40770516]
Epoch 6: Loss: [-0.28415198] [0.49098103 0.         0.4739671  0.        ] [0.01318359 0.71337891 0.22900391 0.04443359] [0.47184181] [0.42733635]
Epoch 7: Loss: [-0.27939384] [0.45667907 0.         0.42744342 0.        ] [0.00976562 0.74560547 0.20751953 0.03710938] [0.49510754] [0.44215598]
Epoch 8: Loss: [-0.27692851] [0.48965458 0.         0.4887946  0.        ] [0.01123047 0.7421875  0.21142578 0.03515625] [0.50967907] [0.45191719]
Epoch 9: Loss: [-0.27288438] [0.41347372 0.         0.41008435 0.        ] [0.00683594 0.72021484 0.2265625  0.04638672] [0.52672079] [0.45968032]
Timer unit: 1e-07 s

Total time: 149.96 s
File: C:/Users/usuario/qGAN/quantumGAN/performance_testing/performance_testv2.py
Function: mainV2 at line 14

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    14                                           def mainV2():
    15         1         32.0     32.0      0.0      seed = 71
    16         1         40.0     40.0      0.0      np.random.seed = seed
    17
    18         1         21.0     21.0      0.0      num_qubits = [2]
    19         1         18.0     18.0      0.0      batch_size = 10
    20         1         20.0     20.0      0.0      entangler_map = [[0, 1]]
    21
    22         1        577.0    577.0      0.0      randoms = np.random.normal(-np.pi * .01, np.pi * .01, 2)
    23
    24         1       2105.0   2105.0      0.0      init_dist = qiskit.QuantumCircuit(2)
    25         1       1376.0   1376.0      0.0      init_dist.ry(randoms[0], 0)
    26         1        719.0    719.0      0.0      init_dist.ry(randoms[1], 1)
    27
    28         1   29932945.0 29932945.0      2.0      ansatz = TwoLocal(int(np.sum(num_qubits)), 'rx', 'cz', entanglement=entangler_map, reps=2, insert_barriers=True)
    29
    30         1         47.0     47.0      0.0      train_data = []
    31        21        432.0     20.6      0.0      for _ in range(20):
    32        20       2280.0    114.0      0.0          x2 = np.random.uniform(.5, .4, (2,))
    33        20       1614.0     80.7      0.0          fake_datapoint = np.random.uniform(-np.pi * .01, np.pi * .01, (2,))
    34        20       1285.0     64.2      0.0          real_datapoint = np.array([x2[1], 0., x2[0], 0])
    35        20        445.0     22.2      0.0          train_data.append((real_datapoint, fake_datapoint))
    36
    37         1     158275.0 158275.0      0.0      g_circuit = ansatz.compose(init_dist, front=True)
    38
    39         1         38.0     38.0      0.0      discriminator = Network(training_data=train_data,
    40         1         20.0     20.0      0.0                              mini_batch_size=batch_size,
    41         1         19.0     19.0      0.0                              sizes=[4, 16, 8, 1],
    42         1       1289.0   1289.0      0.0                              loss_BCE=True)
    43         1         27.0     27.0      0.0      generator = PerformanceQuantumGeneratorV3(training_data=train_data,
    44         1         19.0     19.0      0.0                                                mini_batch_size=batch_size,
    45         1         22.0     22.0      0.0                                                num_qubits=num_qubits,
    46         1         18.0     18.0      0.0                                                generator_circuit=g_circuit,
    47         1         18.0     18.0      0.0                                                shots=2048,
    48         1       9346.0   9346.0      0.0                                                learning_rate=.1)
    49         1         45.0     45.0      0.0      generator.set_discriminator(discriminator)
    50
    51        11        339.0     30.8      0.0      for o in range(num_epochs):
    52        10      11057.0   1105.7      0.0          mini_batches = discriminator.create_mini_batches()
    53        30       1213.0     40.4      0.0          for mini_batch in mini_batches:
    54        20        511.0     25.6      0.0              output_real = mini_batch[0][0]
    55        20        609.0     30.4      0.0              output_fake = generator.get_output(latent_space_noise=mini_batch[0][1],
    56        20   16182693.0 809134.7      1.1                                                 params=None)
    57        20       3563.0    178.2      0.0              generator.set_mini_batch(mini_batch)
    58        20        447.0     22.4      0.0              generator.shots = 2048
    59        20      14255.0    712.8      0.0              lp_wrapper_gen = lp(generator.train_mini_batch)
    60        20 1451901992.0 72595099.6     96.8              lp_wrapper_gen()
    61        20    1053274.0  52663.7      0.1              discriminator.train_mini_batch(generator.mini_batch, .1, o)
    62        10     242369.0  24236.9      0.0          print("Epoch {}: Loss: {}".format(o, discriminator.ret["loss"][-1]), output_real, output_fake)
    63        10      71173.0   7117.3      0.0          print(discriminator.ret["label real"][-1], discriminator.ret["label fake"][-1])
```

File - performance_testv2

```
Total time: 145.16 s
File: C:\Users\usuario\qGAN\quantumGAN\performance_testing\performance_quantum_generator.py
Function: train_mini_batch at line 259

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    259                                           def train_mini_batch(self):
    260        20       2497.0    124.8      0.0          nabla_theta = np.zeros(self.parameter_values.shape)
    261        20        317.0     15.8      0.0          new_images = []
    262
    263       220       9445.0     42.9      0.0          for _, noise in self.mini_batch:
    264      1400      27275.0     19.5      0.0              for index in range(len(self.parameter_values)):
    265      1200     102566.0     85.5      0.0                  perturbation_vector = np.zeros(len(self.
   parameter_values))
    266      1200      48403.0     40.3      0.0                  perturbation_vector[index] = 1
    267
    268      1200     125322.0    104.4      0.0                  pos_params = self.parameter_values + (np.pi / 4) *
   perturbation_vector
    269      1200      79244.0     66.0      0.0                  neg_params = self.parameter_values - (np.pi / 4) *
   perturbation_vector
    270
    271      1200  668005512.0 556671.3     46.0                  pos_result = self.get_output(noise, params=pos_params)
    272      1200  668153156.0 556794.3     46.0                  neg_result = self.get_output(noise, params=neg_params)
    273
    274      1200    1492999.0   1244.2      0.1                  pos_result = self.discriminator.predict(pos_result)
    275      1200     675461.0    562.9      0.0                  neg_result = self.discriminator.predict(neg_result)
    276      1200     653894.0    544.9      0.0                  gradient = self.BCE(pos_result, np.array([1.])) - self.
   BCE(neg_result, np.array([1.]))
    277      1200     301943.0    251.6      0.0                  nabla_theta[index] += gradient
    278       200  111911533.0 559557.7      7.7              new_images.append(self.get_output(noise))
    279
    280       140       1886.0     13.5      0.0          for index in range(len(self.parameter_values)):
    281       120       4233.0     35.3      0.0              self.parameter_values[index] -= (self.learning_rate / self.
   mini_batch_size) * nabla_theta[index]
    282
    283        20       2627.0    131.3      0.0          self.mini_batch = [(datapoint[0], fake_image) for datapoint,
   fake_image in zip(self.mini_batch, new_images)]


Process finished with exit code 0

```