File - performance_test (1)

```
C:\Users\usuario\Anaconda3\envs\tfq\python.exe C:/Users/usuario/qGAN/quantumGAN/performance_testing/performance_test.py
[0.07987741 0.32571303 0.58140782 0.55737798 0.51180307 0.9932685 ]
Epoch 0: Loss: [-0.3917592] [0.4385706  0.         0.42198345 0.        ] [0.22265625 0.52490234 0.19384766 0.05859375]
[0.20872809] [0.21132026]
[0.29309806 0.25148647 0.35858132 0.00397947 0.29342985 0.94847022]
Epoch 0: Loss: [-0.36056637] [0.45520778 0.         0.45257102 0.        ] [0.25830078 0.13427734 0.08056641 0.02685547]
[0.25883474] [0.26574875]
Timer unit: 1e-07 s

Total time: 27.8785 s
File: C:/Users/usuario/qGAN/quantumGAN/performance_testing/performance_test.py
Function: mainV1 at line 13

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    13                                           def mainV1():
    14
    15         1         35.0     35.0      0.0     seed = 71
    16         1         43.0     43.0      0.0     np.random.seed = seed
    17
    18         1         23.0     23.0      0.0     num_qubits = [2]
    19         1         20.0     20.0      0.0     batch_size = 10
    20         1         22.0     22.0      0.0     entangler_map = [[0, 1]]
    21
    22         1        527.0    527.0      0.0     randoms = np.random.normal(-np.pi * .01, np.pi * .01, 2)
    23
    24         1       2210.0   2210.0      0.0     init_dist = qiskit.QuantumCircuit(2)
    25         1       1326.0   1326.0      0.0     init_dist.ry(randoms[0], 0)
    26         1        723.0    723.0      0.0     init_dist.ry(randoms[1], 1)
    27
    28         1   20574804.0 20574804.0      7.4   ansatz = TwoLocal(int(np.sum(num_qubits)), 'rx', 'cz', entanglement=
   entangler_map, reps=2, insert_barriers=True)
    29
    30         1         35.0     35.0      0.0     train_data = []
    31        16        338.0     21.1      0.0     for _ in range(15):
    32        15       1660.0    110.7      0.0         x2 = np.random.uniform(.5, .4, (2,))
    33        15       1418.0     94.5      0.0         fake_datapoint = np.random.uniform(-np.pi * .01, np.pi * .01, (2
   ,))
    34        15        963.0     64.2      0.0         real_datapoint = np.array([x2[1], 0., x2[0], 0])
    35        15        359.0     23.9      0.0         train_data.append((real_datapoint, fake_datapoint))
    36
    37         1     113597.0 113597.0      0.0     g_circuit = ansatz.compose(init_dist, front=True)
    38
    39         1         41.0     41.0      0.0     discriminator = Network(training_data=train_data,
    40         1         21.0     21.0      0.0                             mini_batch_size=batch_size,
    41         1         23.0     23.0      0.0                             sizes=[4, 16, 8, 1],
    42         1       1590.0   1590.0      0.0                             loss_BCE=True)
    43         1         29.0     29.0      0.0     generator = PerformanceQuantumGeneratorV1(training_data=train_data,
    44         1         21.0     21.0      0.0                                   mini_batch_size=batch_size,
    45         1        949.0    949.0      0.0                                   num_qubits=num_qubits,
    46         1         33.0     33.0      0.0                                   generator_circuit=g_circuit,
    47         1         21.0     21.0      0.0                                   shots=2048,
    48         1      15218.0  15218.0      0.0                                   learning_rate=.1)
    49         1         65.0     65.0      0.0     generator.set_discriminator(discriminator)
    50
    51         1         30.0     30.0      0.0     def train():
    52                                                 for o in range(num_epochs):
    53                                                     mini_batches = discriminator.create_mini_batches()
    54                                                     for mini_batch in mini_batches:
    55                                                         output_real = mini_batch[0][0]
    56                                                         output_fake = generator.get_output(latent_space_noise=
   mini_batch[0][1],
    57                                                                                            params=None)
    58                                                         generator.set_mini_batch(mini_batch)
    59                                                         generator.shots = 2048
    60                                                         lp_wrapper_gen = lp(generator.train_mini_batch)
    61                                                         lp_wrapper_gen()
    62                                                         discriminator.train_mini_batch(generator.mini_batch, .1
   , o)
    63                                                     print("Epoch {}: Loss: {}".format(o, discriminator.ret["loss
   "][-1]), output_real, output_fake)
    64                                                     print(discriminator.ret["label real"][-1], discriminator.ret
   ["label fake"][-1])
    65
    66         1        474.0    474.0      0.0     lp_wrapper_train = lp(train)
    67         1  258068683.0 258068683.0     92.6      lp_wrapper_train()

Total time: 25.8068 s
File: C:/Users/usuario/qGAN/quantumGAN/performance_testing/performance_test.py
Function: train at line 51

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    51                                           def train():
    52         2         38.0     19.0      0.0         for o in range(num_epochs):
    53         1        896.0    896.0      0.0             mini_batches = discriminator.create_mini_batches()
    54         3         46.0     15.3      0.0             for mini_batch in mini_batches:
    55         2         26.0     13.0      0.0                 output_real = mini_batch[0][0]
    56         2         52.0     26.0      0.0                 output_fake = generator.get_output(latent_space_noise=
```

File - performance_test (1)

```
 83 mini_batch[0][1],
 84     57         2    5730521.0 2865260.5      2.2                                                params=None)
 85     58         2        221.0    110.5      0.0                generator.set_mini_batch(mini_batch)
 86     59         2         25.0     12.5      0.0                generator.shots = 2048
 87     60         2       1077.0    538.5      0.0                lp_wrapper_gen = lp(generator.train_mini_batch)
 88     61         2  252253003.0 126126501.5     97.7                    lp_wrapper_gen()
 89     62         2      64549.0  32274.5      0.0                discriminator.train_mini_batch(generator.mini_batch, .1
    , o)
 90     63         1      11856.0  11856.0      0.0            print("Epoch {}: Loss: {}".format(o, discriminator.ret["
    loss"][-1]), output_real, output_fake)
 91     64         1       5862.0   5862.0      0.0            print(discriminator.ret["label real"][-1], discriminator.
    ret["label fake"][-1])
 92
 93 Total time: 25.0799 s
 94 File: C:/Users/usuario/qGAN/quantumGAN/performance_testing/performance_test.py
 95 Function: mainV2 at line 69
 96
 97 Line #      Hits         Time  Per Hit   % Time  Line Contents
 98 ==============================================================
 99     69                                           def mainV2():
100     70
101     71         1         23.0     23.0      0.0    seed = 71
102     72         1         53.0     53.0      0.0    np.random.seed = seed
103     73
104     74         1         22.0     22.0      0.0    num_qubits = [2]
105     75         1         21.0     21.0      0.0    batch_size = 10
106     76         1         22.0     22.0      0.0    entangler_map = [[0, 1]]
107     77
108     78         1        282.0    282.0      0.0    randoms = np.random.normal(-np.pi * .01, np.pi * .01, 2)
109     79
110     80         1       1925.0   1925.0      0.0    init_dist = qiskit.QuantumCircuit(2)
111     81         1       1215.0   1215.0      0.0    init_dist.ry(randoms[0], 0)
112     82         1        751.0    751.0      0.0    init_dist.ry(randoms[1], 1)
113     83
114     84         1      39721.0  39721.0      0.0    ansatz = TwoLocal(int(np.sum(num_qubits)), 'rx', 'cz', entanglement
    =entangler_map, reps=2, insert_barriers=True)
115     85
116     86         1         34.0     34.0      0.0    train_data = []
117     87        16        992.0     62.0      0.0    for _ in range(15):
118     88        15       3626.0    241.7      0.0        x2 = np.random.uniform(.5, .4, (2,))
119     89        15       1539.0    102.6      0.0        fake_datapoint = np.random.uniform(-np.pi * .01, np.pi * .01, (
    2,))
120     90        15       1426.0     95.1      0.0        real_datapoint = np.array([x2[1], 0., x2[0], 0])
121     91        15        389.0     25.9      0.0        train_data.append((real_datapoint, fake_datapoint))
122     92
123     93         1     103183.0 103183.0      0.0    g_circuit = ansatz.compose(init_dist, front=True)
124     94
125     95         1         31.0     31.0      0.0    discriminator = Network(training_data=train_data,
126     96         1         21.0     21.0      0.0                            mini_batch_size=batch_size,
127     97         1         23.0     23.0      0.0                            sizes=[4, 16, 8, 1],
128     98         1        708.0    708.0      0.0                            loss_BCE=True)
129     99         1         25.0     25.0      0.0    generator = PerformanceQuantumGeneratorV2(training_data=train_data,
130    100         1         21.0     21.0      0.0                                 mini_batch_size=batch_size,
131    101         1         21.0     21.0      0.0                                 num_qubits=num_qubits,
132    102         1         20.0     20.0      0.0                                 generator_circuit=g_circuit,
133    103         1         21.0     21.0      0.0                                 shots=2048,
134    104         1       7375.0   7375.0      0.0                                 learning_rate=.1)
135    105         1         42.0     42.0      0.0    generator.set_discriminator(discriminator)
136    106
137    107         1         25.0     25.0      0.0    def train():
138    108                                               for o in range(num_epochs):
139    109                                                   mini_batches = discriminator.create_mini_batches()
140    110                                                   for mini_batch in mini_batches:
141    111                                                       output_real = mini_batch[0][0]
142    112                                                       output_fake = generator.get_output(latent_space_noise=
    mini_batch[0][1],
143    113                                                                                          params=None)
144    114                                                       generator.set_mini_batch(mini_batch)
145    115                                                       generator.shots = 2048
146    116                                                       lp_wrapper_gen = lp(generator.train_mini_batch)
147    117                                                       lp_wrapper_gen()
148    118                                                       discriminator.train_mini_batch(generator.mini_batch, .1
    , o)
149    119                                                       print("Epoch {}: Loss: {}".format(o, discriminator.ret["
    loss"][-1]), output_real, output_fake)
150    120                                                       print(discriminator.ret["label real"][-1], discriminator.
    ret["label fake"][-1])
151    121
152    122         1        399.0    399.0      0.0    lp_wrapper_train = lp(train)
153    123         1  250635340.0 250635340.0     99.9    lp_wrapper_train()
154
155 Total time: 25.0635 s
156 File: C:/Users/usuario/qGAN/quantumGAN/performance_testing/performance_test.py
157 Function: train at line 107
158
159 Line #      Hits         Time  Per Hit   % Time  Line Contents
160 ==============================================================
161    107                                           def train():
162    108         2         43.0     21.5      0.0        for o in range(num_epochs):
```

File - performance_test (1)

```
163    109         1        797.0    797.0      0.0              mini_batches = discriminator.create_mini_batches()
164    110         3         85.0     28.3      0.0              for mini_batch in mini_batches:
165    111         2         24.0     12.0      0.0                  output_real = mini_batch[0][0]
166    112         2         42.0     21.0      0.0                  output_fake = generator.get_output(latent_space_noise=
    mini_batch[0][1],
167    113         2    2497559.0 1248779.5      1.0                                                     params=None)
168    114         2        176.0     88.0      0.0                  generator.set_mini_batch(mini_batch)
169    115         2         26.0     13.0      0.0                  generator.shots = 2048
170    116         2       1155.0    577.5      0.0                  lp_wrapper_gen = lp(generator.train_mini_batch)
171    117         2  248026048.0 124013024.0     99.0                      lp_wrapper_gen()
172    118         2      90591.0  45295.5      0.0                  discriminator.train_mini_batch(generator.mini_batch, .1
    , o)
173    119         1      12335.0  12335.0      0.0              print("Epoch {}: Loss: {}".format(o, discriminator.ret["
    loss"][-1]), output_real, output_fake)
174    120         1       5816.0   5816.0      0.0              print(discriminator.ret["label real"][-1], discriminator.
    ret["label fake"][-1])
175
176 Total time: 25.2231 s
177 File: C:\Users\usuario\qGAN\quantumGAN\performance_testing\performance_quantum_generator.py
178 Function: train_mini_batch at line 119
179
180 Line #      Hits         Time  Per Hit   % Time  Line Contents
181 ==============================================================
182    119                                           def train_mini_batch(self):
183    120         2        175.0     87.5      0.0          nabla_theta = np.zeros(self.parameter_values.shape)
184    121         2         24.0     12.0      0.0          new_images = []
185    122
186    123        17        713.0     41.9      0.0          for _, noise in self.mini_batch:
187    124       105       2259.0     21.5      0.0              for index in range(len(self.parameter_values)):
188    125        90       7241.0     80.5      0.0                  perturbation_vector = np.zeros(len(self.
    parameter_values))
189    126        90       2127.0     23.6      0.0                  perturbation_vector[index] = 1
190    127
191    128        90       7970.0     88.6      0.0                  pos_params = self.parameter_values + (np.pi / 4) *
    perturbation_vector
192    129        90       5422.0     60.2      0.0                  neg_params = self.parameter_values - (np.pi / 4) *
    perturbation_vector
193    130
194    131        90  116753569.0 1297261.9     46.3                  pos_result = self.get_output(noise, params=pos_params)
195    132        90  116231184.0 1291457.6     46.1                  neg_result = self.get_output(noise, params=neg_params)
196    133
197    134        90      85408.0    949.0      0.0                  pos_result = self.discriminator.predict(pos_result)
198    135        90      48613.0    540.1      0.0                  neg_result = self.discriminator.predict(neg_result)
199    136        90      40700.0    452.2      0.0                  gradient = self.BCE(pos_result, np.array([1.])) - self.
    BCE(neg_result, np.array([1.]))
200    137        90      13989.0    155.4      0.0                  nabla_theta[index] += gradient
201    138        15   19030946.0 1268729.7      7.5              new_images.append(self.get_output(noise))
202    139
203    140        14        234.0     16.7      0.0          for index in range(len(self.parameter_values)):
204    141        12        386.0     32.2      0.0              self.parameter_values[index] -= (self.learning_rate / self.
    mini_batch_size) * nabla_theta[index]
205    142
206    143         2        212.0    106.0      0.0          self.mini_batch = [(datapoint[0], fake_image) for datapoint,
    fake_image in zip(self.mini_batch, new_images)]
207
208 Total time: 24.8004 s
209 File: C:\Users\usuario\qGAN\quantumGAN\performance_testing\performance_quantum_generator.py
210 Function: train_mini_batch at line 256
211
212 Line #      Hits         Time  Per Hit   % Time  Line Contents
213 ==============================================================
214    256                                           def train_mini_batch(self):
215    257         2        106.0     53.0      0.0          nabla_theta = np.zeros(self.parameter_values.shape)
216    258         2         23.0     11.5      0.0          new_images = []
217    259
218    260        17        600.0     35.3      0.0          for _, noise in self.mini_batch:
219    261       105       1813.0     17.3      0.0              for index in range(len(self.parameter_values)):
220    262        90       6555.0     72.8      0.0                  perturbation_vector = np.zeros(len(self.
    parameter_values))
221    263        90       1857.0     20.6      0.0                  perturbation_vector[index] = 1
222    264
223    265        90       7497.0     83.3      0.0                  pos_params = self.parameter_values + (np.pi / 4) *
    perturbation_vector
224    266        90       6570.0     73.0      0.0                  neg_params = self.parameter_values - (np.pi / 4) *
    perturbation_vector
225    267
226    268        90  114874234.0 1276380.4     46.3                  pos_result = self.get_output(noise, params=pos_params)
227    269        90  114285126.0 1269834.7     46.1                  neg_result = self.get_output(noise, params=neg_params)
228    270
229    271        90      86064.0    956.3      0.0                  pos_result = self.discriminator.predict(pos_result)
230    272        90      52408.0    582.3      0.0                  neg_result = self.discriminator.predict(neg_result)
231    273        90      42069.0    467.4      0.0                  gradient = self.BCE(pos_result, np.array([1.])) - self.
    BCE(neg_result, np.array([1.]))
232    274        90      15079.0    167.5      0.0                  nabla_theta[index] += gradient
233    275        15   18623731.0 1241582.1      7.5              new_images.append(self.get_output(noise))
234    276
235    277        14        186.0     13.3      0.0          for index in range(len(self.parameter_values)):
236    278        12        391.0     32.6      0.0              self.parameter_values[index] -= (self.learning_rate / self.
    mini_batch_size) * nabla_theta[index]
```

File - performance_test (1)

```
237    279
238    280         2        181.0     90.5      0.0         self.mini_batch = [(datapoint[0], fake_image) for datapoint,
    fake_image in zip(self.mini_batch, new_images)]
239
240
241 Process finished with exit code 0
242
```