```
C:\Users\usuario\Anaconda3\envs\tfq\python.exe C:/Users/usuario/qGAN/quantumGAN/performance_testing/performance_testv2.py
[0.25740648 0.57659044 0.23477071 0.55527819 0.61069801 0.4174805 ]
Epoch 0: Loss: [-0.2692237] [0.42004681 0.         0.47987748 0.        ] [0.45263672 0.40625    0.12011719 0.02099609]
[0.52914619] [0.4530131]
Epoch 1: Loss: [-0.21593399] [0.47485789 0.         0.45674713 0.        ] [0.40917969 0.56005859 0.01953125 0.01123047]
[0.60691333] [0.3904556]
Epoch 2: Loss: [-0.14973108] [0.47529203 0.         0.4991709  0.        ] [0.14013672 0.85644531 0.         0.00341797]
[0.70792182] [0.29115295]
Epoch 3: Loss: [-0.09087498] [0.48988611 0.         0.40917071 0.        ] [7.81250000e-03 9.91699219e-01 0.00000000e+00 4.88281250e-04]
[0.82428565] [0.20168865]
Epoch 4: Loss: [-0.06340145] [0.43582923 0.         0.45221299 0.        ] [0. 1. 0. 0.]
[0.87536304] [0.14688242]
Epoch 5: Loss: [-0.04789987] [0.47813589 0.         0.4975698  0.        ] [4.88281250e-04 9.99511719e-01 0.00000000e+00 0.00000000e+00]
[0.90274115] [0.11154177]
Epoch 6: Loss: [-0.03950784] [0.46009184 0.         0.4241394  0.        ] [0. 1. 0. 0.]
[0.91462706] [0.08853442]
Epoch 7: Loss: [-0.03095257] [0.48469709 0.         0.45676017 0.        ] [0. 1. 0. 0.]
[0.93511231] [0.0726769]
Epoch 8: Loss: [-0.02610199] [0.4905251  0.         0.48932336 0.        ] [4.88281250e-04 9.99511719e-01 0.00000000e+00 0.00000000e+00]
[0.94426326] [0.06091927]
Epoch 9: Loss: [-0.02159365] [0.45058751 0.         0.40946602 0.        ] [0. 1. 0. 0.]
[0.95523335] [0.05222942]
Timer unit: 1e-07 s

Total time: 1179.62 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         52.0     52.0      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         19.0     19.0      0.0      entangler_map = [[0, 1]]
    21
    22         1        619.0    619.0      0.0      randoms = np.random.normal(-np.pi * .01, np.pi * .01, 2)
    23
    24         1       2182.0   2182.0      0.0      init_dist = qiskit.QuantumCircuit(2)
    25         1       1322.0   1322.0      0.0      init_dist.ry(randoms[0], 0)
    26         1        727.0    727.0      0.0      init_dist.ry(randoms[1], 1)
    27
    28         1   24781433.0 24781433.0      0.2      ansatz = TwoLocal(int(np.sum(num_qubits)), 'rx', 'cz', entanglement=entangler_map, reps=2, insert_barriers=True)
    29
    30         1         34.0     34.0      0.0      train_data = []
    31       201       3857.0     19.2      0.0      for _ in range(200):
    32       200      20460.0    102.3      0.0          x2 = np.random.uniform(.5, .4, (2,))
    33       200      18288.0     91.4      0.0          fake_datapoint = np.random.uniform(-np.pi * .01, np.pi * .01, (2,))
    34       200      50922.0    254.6      0.0          real_datapoint = np.array([x2[1], 0., x2[0], 0])
    35       200       4840.0     24.2      0.0          train_data.append((real_datapoint, fake_datapoint))
    36
    37         1     158443.0 158443.0      0.0      g_circuit = ansatz.compose(init_dist, front=True)
    38
    39         1         48.0     48.0      0.0      discriminator = Network(training_data=train_data,
    40         1         20.0     20.0      0.0                              mini_batch_size=batch_size,
    41         1         20.0     20.0      0.0                              sizes=[4, 16, 8, 1],
    42         1       1046.0   1046.0      0.0                              loss_BCE=True)
    43         1         35.0     35.0      0.0      generator = PerformanceQuantumGeneratorV3(training_data=train_data,
    44         1         21.0     21.0      0.0                                                mini_batch_size=batch_size,
    45         1         18.0     18.0      0.0                                                num_qubits=num_qubits,
    46         1         27.0     27.0      0.0                                                generator_circuit=g_circuit,
    47         1         18.0     18.0      0.0                                                shots=2048,
    48         1       9193.0   9193.0      0.0                                                learning_rate=.1)
    49         1         71.0     71.0      0.0      generator.set_discriminator(discriminator)
    50
    51        11        331.0     30.1      0.0      for o in range(num_epochs):
    52        10      84271.0   8427.1      0.0          mini_batches = discriminator.create_mini_batches()
    53       210       8767.0     41.7      0.0          for mini_batch in mini_batches:
    54       200       5446.0     27.2      0.0              output_real = mini_batch[0][0]
    55       200       6725.0     33.6      0.0              output_fake = generator.get_output(latent_space_noise=mini_batch[0][1],
    56       200   97835143.0 489175.7      0.8                                                 params=None)
    57       200      33291.0    166.5      0.0              generator.set_mini_batch(mini_batch)
    58       200       4605.0     23.0      0.0              generator.shots = 2048
    59       200     121132.0    605.7      0.0              lp_wrapper_gen = lp(generator.train_mini_batch)
    60       200 11663610880.0 58318054.4     98.9                  lp_wrapper_gen()
    61       200    9279044.0  46395.2      0.1              discriminator.train_mini_batch(generator.mini_batch, .1, o)
    62        10     129592.0  12959.2      0.0          print("Epoch {}: Loss: {}".format(o, discriminator.ret["loss"][-1]), output_real, output_fake)
    63        10      62874.0   6287.4      0.0          print(discriminator.ret["label real"][-1], discriminator.ret["label fake"][-1])

Total time: 1166.08 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       200      13767.0     68.8      0.0          nabla_theta = np.zeros(self.parameter_values.shape)
   261       200       2497.0     12.5      0.0          new_images = []
   262
   263      2200      82688.0     37.6      0.0          for _, noise in self.mini_batch:
   264     14000     275512.0     19.7      0.0              for index in range(len(self.parameter_values)):
   265     12000     920383.0     76.7      0.0                  perturbation_vector = np.zeros(len(self.parameter_values))
   266     12000     288224.0     24.0      0.0                  perturbation_vector[index] = 1
   267
   268     12000    1148790.0     95.7      0.0                  pos_params = self.parameter_values + (np.pi / 4) * perturbation_vector
   269     12000     731346.0     60.9      0.0                  neg_params = self.parameter_values - (np.pi / 4) * perturbation_vector
   270
   271     12000 5368421470.0 447368.5     46.0                  pos_result = self.get_output(noise, params=pos_params)
   272     12000 5360074218.0 446672.9     46.0                  neg_result = self.get_output(noise, params=neg_params)
   273
   274     12000   11865238.0    988.8      0.1                  pos_result = self.discriminator.predict(pos_result)
   275     12000    6070037.0    505.8      0.1                  neg_result = self.discriminator.predict(neg_result)
   276     12000    5600952.0    466.7      0.0                  gradient = self.BCE(pos_result, np.array([1.])) - self.BCE(neg_result, np.array([1.]))
   277     12000    2315916.0    193.0      0.0                  nabla_theta[index] += gradient
   278      2000  902844468.0 451422.2      7.7              new_images.append(self.get_output(noise))
   279
   280      1400      22608.0     16.1      0.0          for index in range(len(self.parameter_values)):
   281      1200      45594.0     38.0      0.0              self.parameter_values[index] -= (self.learning_rate / self.mini_batch_size) * nabla_theta[index]
   282
   283       200      47914.0    239.6      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

```