File - performance_get_output_generator

```
C:\Users\usuario\Anaconda3\envs\tfq\python.exe C:/Users/usuario/qGAN/quantumGAN/performance_testing/performance_get_output_generator.py
[0.26757812 0.55615234 0.16699219 0.00927734]
[0.27587891 0.54003906 0.17089844 0.01318359]
[0.26123047 0.55761719 0.17236328 0.00878906]
Timer unit: 1e-07 s

Total time: 0.854987 s
File: C:/Users/usuario/qGAN/quantumGAN/performance_testing/performance_get_output_generator.py
Function: get_output_V1 at line 34

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    34                                           def get_output_V1():
    35         2         48.0     24.0      0.0      for noise in batch_noise:
    36         1         15.0     15.0      0.0          real_keys = {"00", "10", "01", "11"}
    37
    38         1        634.0    634.0      0.0          quantum = QuantumRegister(sum(num_qubits), name="q")
    39         1       1420.0   1420.0      0.0          qc = QuantumCircuit(sum(num_qubits))
    40
    41         1       1201.0   1201.0      0.0          init_dist = qiskit.QuantumCircuit(sum(num_qubits))
    42         1         35.0     35.0      0.0          assert noise.shape[0] == sum(num_qubits)
    43
    44         3         39.0     13.0      0.0          for num_qubit in range(sum(num_qubits)):
    45         2       1824.0    912.0      0.0              init_dist.ry(noise[num_qubit], num_qubit)
    46
    47         1         32.0     32.0      0.0          params = cast(np.ndarray, parameter_values)
    48
    49         1     133930.0 133930.0      1.6          qc.append(construct_circuit(params), quantum)
    50         1      29445.0  29445.0      0.3          final_circuit = qc.compose(init_dist, front=True)
    51         1       2592.0   2592.0      0.0          final_circuit.measure_all()
    52
    53         1    3930743.0 3930743.0     46.0          simulator_1 = qiskit.Aer.get_backend("aer_simulator")
    54         1    4335256.0 4335256.0     50.7          final_circuit = qiskit.transpile(final_circuit, simulator_1)
    55         1      99570.0  99570.0      1.2          result = simulator_1.run(final_circuit, shots=shots).result()
    56         1       2824.0   2824.0      0.0          counts = result.get_counts(final_circuit)
    57
    58         1         12.0     12.0      0.0          try:
    59         1        221.0    221.0      0.0              pixels = np.array([counts["00"], counts["10"], counts["01
"], counts["11"]])
    60
    61                                                   except KeyError:
    62                                                       # dealing with the keys that qiskit doesn't include in the
    63                                                       # dictionary because they don't get any measurements
    64
    65                                                       keys = counts.keys()
    66                                                       missing_keys = real_keys.difference(keys)
    67                                                       # we use sets to get the missing keys
    68                                                       for key_missing in missing_keys:
    69                                                           counts[key_missing] = 0
    70
    71                                                       pixels = np.array([counts["00"], counts["10"], counts["01
"], counts["11"]])
    72
    73         1        336.0    336.0      0.0          pixels = pixels / shots
    74         1       9692.0   9692.0      0.1          print(pixels)

Total time: 0.127569 s
File: C:/Users/usuario/qGAN/quantumGAN/performance_testing/performance_get_output_generator.py
Function: get_output_V2 at line 77

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    77                                           def get_output_V2():
    78         1       2921.0   2921.0      0.2      simulator = qiskit.Aer.get_backend("aer_simulator")
    79         2         45.0     22.5      0.0      for noise in batch_noise:
    80         1         13.0     13.0      0.0          real_keys = {"00", "10", "01", "11"}
    81
    82         1        761.0    761.0      0.1          quantum = QuantumRegister(sum(num_qubits), name="q")
    83         1       1665.0   1665.0      0.1          qc = QuantumCircuit(sum(num_qubits))
    84
    85         1       1241.0   1241.0      0.1          init_dist = qiskit.QuantumCircuit(sum(num_qubits))
    86         1         29.0     29.0      0.0          assert noise.shape[0] == sum(num_qubits)
    87
    88         3         55.0     18.3      0.0          for num_qubit in range(sum(num_qubits)):
    89         2       2022.0   1011.0      0.2              init_dist.ry(noise[num_qubit], num_qubit)
    90
    91         1         31.0     31.0      0.0          params = cast(np.ndarray, parameter_values)
    92
    93         1     103099.0 103099.0      8.1          qc.append(construct_circuit(params), quantum)
    94         1      60099.0  60099.0      4.7          final_circuit = qc.compose(init_dist, front=True)
    95         1       3281.0   3281.0      0.3          final_circuit.measure_all()
    96
    97         1    1022653.0 1022653.0     80.2          final_circuit = qiskit.transpile(final_circuit, simulator)
    98         1      70847.0  70847.0      5.6          result = simulator.run(final_circuit, shots=shots).result()
    99         1       1612.0   1612.0      0.1          counts = result.get_counts(final_circuit)
   100
   101         1         12.0     12.0      0.0          try:
   102         1        143.0    143.0      0.0              pixels = np.array([counts["00"], counts["10"], counts["01
```

```
 86 "], counts["11"]])
 87    103
 88    104                                                 except KeyError:
 89    105                                                     # dealing with the keys that qiskit doesn't include in the
 90    106                                                     # dictionary because they don't get any measurements
 91    107
 92    108                                                     keys = counts.keys()
 93    109                                                     missing_keys = real_keys.difference(keys)
 94    110                                                     # we use sets to get the missing keys
 95    111                                                     for key_missing in missing_keys:
 96    112                                                         counts[key_missing] = 0
 97    113
 98    114                                                     pixels = np.array([counts["00"], counts["10"], counts["01
    "], counts["11"]])
 99    115
100    116         1        208.0    208.0      0.0         pixels = pixels / shots
101    117         1       4952.0   4952.0      0.4         print(pixels)
102
103 Total time: 0.138738 s
104 File: C:/Users/usuario/qGAN/quantumGAN/performance_testing/performance_get_output_generator.py
105 Function: get_output_V3 at line 119
106
107 Line #      Hits         Time  Per Hit   % Time  Line Contents
108 ==============================================================
109    119                                           def get_output_V3():
110    120         1       2565.0   2565.0      0.2     simulator = qiskit.Aer.get_backend("aer_simulator")
111    121         1         15.0     15.0      0.0     real_keys = {"00", "10", "01", "11"}
112    122
113    123         1        592.0    592.0      0.0     quantum = QuantumRegister(sum(num_qubits), name="q")
114    124         1       1421.0   1421.0      0.1     qc = QuantumCircuit(sum(num_qubits))
115    125         1       1392.0   1392.0      0.1     init_dist = qiskit.QuantumCircuit(sum(num_qubits))
116    126
117    127         2         35.0     17.5      0.0     for noise in batch_noise:
118    128         1         24.0     24.0      0.0         assert noise.shape[0] == sum(num_qubits)
119    129
120    130         3         41.0     13.7      0.0         for num_qubit in range(sum(num_qubits)):
121    131         2       1879.0    939.5      0.1             init_dist.ry(noise[num_qubit], num_qubit)
122    132
123    133         1         27.0     27.0      0.0         params = cast(np.ndarray, parameter_values)
124    134
125    135         1     117366.0 117366.0      8.5         qc.append(construct_circuit(params), quantum)
126    136         1      29305.0  29305.0      2.1         final_circuit = qc.compose(init_dist, front=True)
127    137         1       2734.0   2734.0      0.2         final_circuit.measure_all()
128    138
129    139         1    1137173.0 1137173.0     82.0         final_circuit = qiskit.transpile(final_circuit, simulator)
130    140         1      82403.0  82403.0      5.9         result = simulator.run(final_circuit, shots=shots).result()
131    141         1       1798.0   1798.0      0.1         counts = result.get_counts(final_circuit)
132    142
133    143         1         25.0     25.0      0.0         try:
134    144         1        315.0    315.0      0.0             pixels = np.array([counts["00"], counts["10"], counts["01
    "], counts["11"]])
135    145
136    146                                                 except KeyError:
137    147                                                     # dealing with the keys that qiskit doesn't include in the
138    148                                                     # dictionary because they don't get any measurements
139    149
140    150                                                     keys = counts.keys()
141    151                                                     missing_keys = real_keys.difference(keys)
142    152                                                     # we use sets to get the missing keys
143    153                                                     for key_missing in missing_keys:
144    154                                                         counts[key_missing] = 0
145    155
146    156                                                     pixels = np.array([counts["00"], counts["10"], counts["01
    "], counts["11"]])
147    157
148    158         1        325.0    325.0      0.0         pixels = pixels / shots
149    159         1       7948.0   7948.0      0.6         print(pixels)
150
151
152 Process finished with exit code 0
153
```