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
[0.00683594 0.30957031 0.53173828 0.15185547]
[0.00585938 0.32568359 0.50390625 0.16455078]
Timer unit: 1e-07 s

Total time: 0.53064 s
File: C:/Users/usuario/qGAN/quantumGAN/performance_testing/performance_get_output_generator.py
Function: get_output_V1 at line 31

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    31                                           def get_output_V1():
    32         1         27.0     27.0      0.0      real_keys = {"00", "10", "01", "11"}
    33
    34         1        728.0    728.0      0.0      quantum = QuantumRegister(sum(num_qubits), name="q")
    35         1       1528.0   1528.0      0.0      qc = QuantumCircuit(sum(num_qubits))
    36
    37         1       1286.0   1286.0      0.0      init_dist = qiskit.QuantumCircuit(sum(num_qubits))
    38         1         42.0     42.0      0.0      assert latent_space_noise.shape[0] == sum(num_qubits)
    39
    40         3         42.0     14.0      0.0      for num_qubit in range(sum(num_qubits)):
    41         2       1979.0    989.5      0.0          init_dist.ry(latent_space_noise[num_qubit], num_qubit)
    42
    43         1         36.0     36.0      0.0      params = cast(np.ndarray, parameter_values)
    44
    45         1     144983.0 144983.0      2.7      qc.append(construct_circuit(params), quantum)
    46         1      33205.0  33205.0      0.6      final_circuit = qc.compose(init_dist, front=True)
    47         1       3750.0   3750.0      0.1      final_circuit.measure_all()
    48
    49         1       3933.0   3933.0      0.1      simulator = qiskit.Aer.get_backend("aer_simulator")
    50         1    4988157.0 4988157.0     94.0      final_circuit = qiskit.transpile(final_circuit, simulator)
    51         1     118460.0 118460.0      2.2      result = simulator.run(final_circuit, shots=shots).result()
    52         1       2017.0   2017.0      0.0      counts = result.get_counts(final_circuit)
    53
    54         1         12.0     12.0      0.0      try:
    55         1        123.0    123.0      0.0          pixels = np.array([counts["00"], counts["10"], counts["01"], counts["11"]])
    56
    57                                               except KeyError:
    58                                                   # dealing with the keys that qiskit doesn't include in the
    59                                                   # dictionary because they don't get any measurements
    60
    61                                                   keys = counts.keys()
    62                                                   missing_keys = real_keys.difference(keys)
    63                                                   # we use sets to get the missing keys
    64                                                   for key_missing in missing_keys:
    65                                                       counts[key_missing] = 0
    66
    67                                                   pixels = np.array([counts["00"], counts["10"], counts["01"], counts["11"]])
    68
    69         1        184.0    184.0      0.0      pixels = pixels / shots
    70         1       5904.0   5904.0      0.1      print(pixels)

Total time: 0.132975 s
File: C:/Users/usuario/qGAN/quantumGAN/performance_testing/performance_get_output_generator.py
Function: get_output_V2 at line 74

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    74                                           def get_output_V2():
    75         1         17.0     17.0      0.0      real_keys = {"00", "10", "01", "11"}
    76
    77         1        694.0    694.0      0.1      quantum = QuantumRegister(sum(num_qubits), name="q")
    78         1       1698.0   1698.0      0.1      qc = QuantumCircuit(sum(num_qubits))
    79
    80         1       1295.0   1295.0      0.1      init_dist = qiskit.QuantumCircuit(sum(num_qubits))
    81         1         28.0     28.0      0.0      assert latent_space_noise.shape[0] == sum(num_qubits)
    82
    83         3         39.0     13.0      0.0      for num_qubit in range(sum(num_qubits)):
    84         2       1837.0    918.5      0.1          init_dist.ry(latent_space_noise[num_qubit], num_qubit)
    85
    86         1         30.0     30.0      0.0      params = cast(np.ndarray, parameter_values)
    87
    88         1     101830.0 101830.0      7.7      qc.append(construct_circuit(params), quantum)
    89         1      42937.0  42937.0      3.2      final_circuit = qc.compose(init_dist, front=True)
    90         1       3175.0   3175.0      0.2      final_circuit.measure_all()
    91
    92         1    1070220.0 1070220.0     80.5      final_circuit = qiskit.transpile(final_circuit, simulator)
    93         1      98646.0  98646.0      7.4      result = simulator.run(final_circuit, shots=shots).result()
    94         1       1720.0   1720.0      0.1      counts = result.get_counts(final_circuit)
    95
    96         1         12.0     12.0      0.0      try:
    97         1        139.0    139.0      0.0          pixels = np.array([counts["00"], counts["10"], counts["01"], counts["11"]])
    98
    99                                               except KeyError:
   100                                                   # dealing with the keys that qiskit doesn't include in the
```

```
 86    101                                                 # dictionary because they don't get any measurements
 87    102
 88    103                                                 keys = counts.keys()
 89    104                                                 missing_keys = real_keys.difference(keys)
 90    105                                                 # we use sets to get the missing keys
 91    106                                                 for key_missing in missing_keys:
 92    107                                                     counts[key_missing] = 0
 93    108
 94    109                                                 pixels = np.array([counts["00"], counts["10"], counts["01"],
    counts["11"]])
 95    110
 96    111         1        161.0    161.0      0.0     pixels = pixels / shots
 97    112         1       5273.0   5273.0      0.4     print(pixels)
 98
 99
100 Process finished with exit code 0
101
```