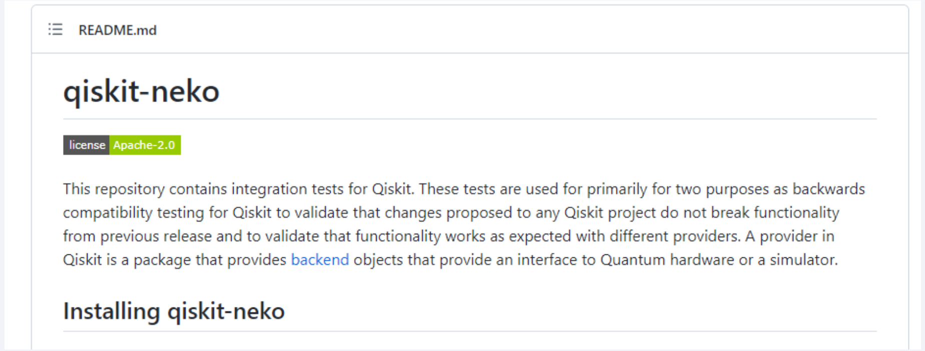
QAMP Spring 2022

Mentee : Tanya Garg
Mentor: Matthew Treinish

Expand qiskit-neko testing

Qiskit-neko

- New effort to add a proper integration test suite to the overall qiskit project.
- Designed to be run in CI to ensure that a proposed change to any qiskit project doesn't break backwards compatibility or break an interface that another qiskit project is currently using



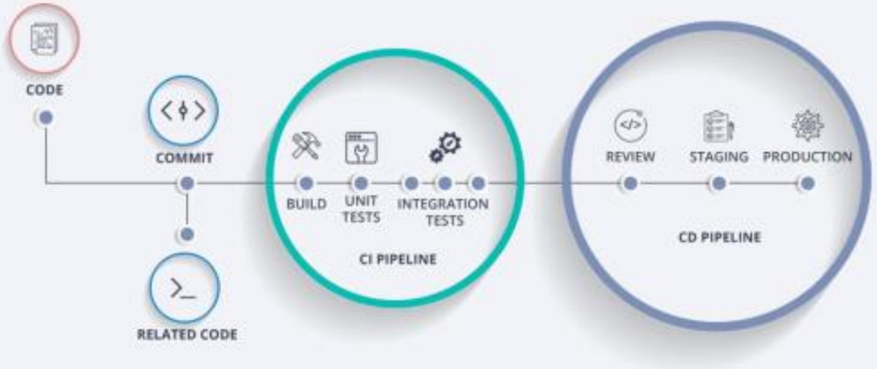
README.md

qiskit-neko

license Apache-2.0

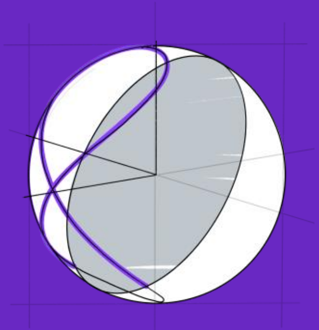
This repository contains integration tests for Qiskit. These tests are used for primarily for two purposes as backwards compatibility testing for Qiskit to validate that changes proposed to any Qiskit project do not break functionality from previous release and to validate that functionality works as expected with different providers. A provider in Qiskit is a package that provides backend objects that provide an interface to Quantum hardware or a simulator.

Installing qiskit-neko

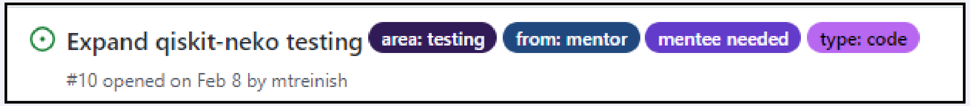


Initial Problem Statement

- Add tests for different sub sections of qiskit such as circuits, nature, machine learning and visualizations.
- Start running qiskit-neko in CI with proper integration with qiskit.



Delivered


Expand qiskit-neko testing area: testing from: mentor mentee needed type: code
#10 opened on Feb 8 by mtreinish

- Added a few tests covering the qiskit-machine learning classes. Namely,
  - Quantum Neural Networks (Using OPFlow)
  - Quantum Neural Networks (Using TwoLayer)
  - Variational Quantum Classifier

- Added a github action to automate running neko in other major qiskit repositories such as terra and nature

Workflow:

Basic Prep:

- Understood the code base of qiskit-neko.
- Refined unittest and CI skills through documentations

Initial PR:

- Initiated and merged (Added ML tests #3) to add tests on qiskit machine learning.
- Added tests related to QNN and VQC based on the tutorials.

Adding Github Action:

- Initiated (Added Action to Run Tests #5; Under Review) to add a custom action for automated running of neko tests.

Integration With Qiskit:

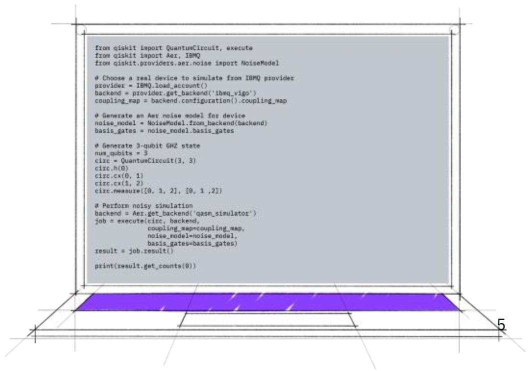
- The repo will be incorporated under the qiskit framework with the addition of the action in other major qiskit repos

Challenges

And what I learnt from them…

- Not experienced enough with software development practices and github actions.
  Understood the thought process that goes behind writing tests + Gained knowledge about github actions and workflows.

- Creating threshold testing conditions for qiskit machine learning where circuits are measured probabilistically giving different answers in each iteration.
  Learnt how to tackle this problem by using the delta parameter.



```
from qiskit import QuantumCircuit, execute
from qiskit import Aer, IBMQ
from qiskit.providers.aer.noise import NoiseModel

# Choose a real device to simulate from IBMQ provider
provider = IBMQ.load_account()
backend = provider.get_backend('ibmq_vigo')
coupling_map = backend.configuration().coupling_map

# Generate an Aer noise model for device
noise_model = NoiseModel.from_backend(backend)
basis_gates = noise_model.basis_gates

# Generate 3-qubit GHZ state
num_qubits = 3
circ = QuantumCircuit(3, 3)
circ.h(0)
circ.cx(0, 1)
circ.cx(1, 2)
circ.measure([0, 1, 2], [0, 1 ,2])

# Perform noisy simulation
backend = Aer.get_backend('qasm_simulator')
job = execute(circ, backend,
              coupling_map=coupling_map,
              noise_model=noise_model,
              basis_gates=basis_gates)
result = job.result()

print(result.get_counts(0))
```