Analyze and improve performance of Qiskit Machine Learning

Mentee: Cenk Tüysüz

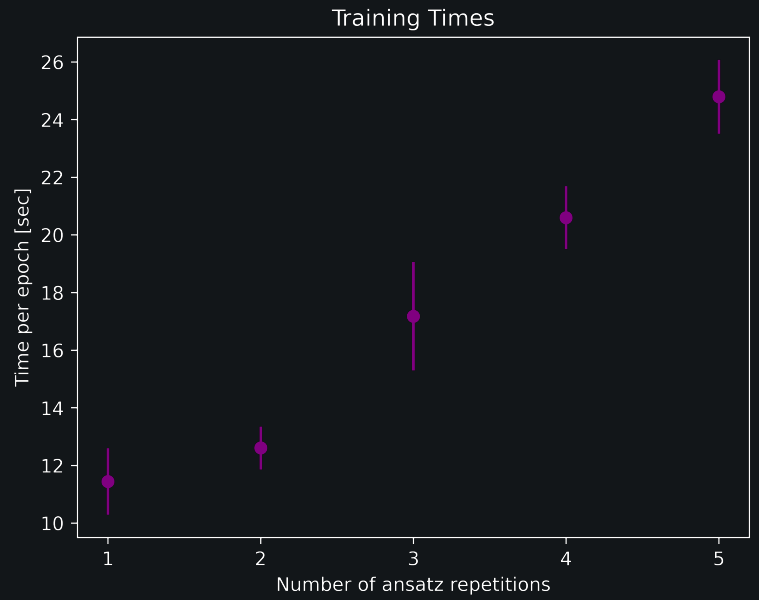
Mentor: Anton Dekusar

Qiskit

Identifying the problem

Current version of Qiskit QML is relatively slow. This project aims to identify sources of bottlenecks and produce solutions to them in order to speed-up the overall performance of the module.

As a first step, performance of VQC was investigated. IRIS dataset classification performance with different ansatz repetitions were considered.



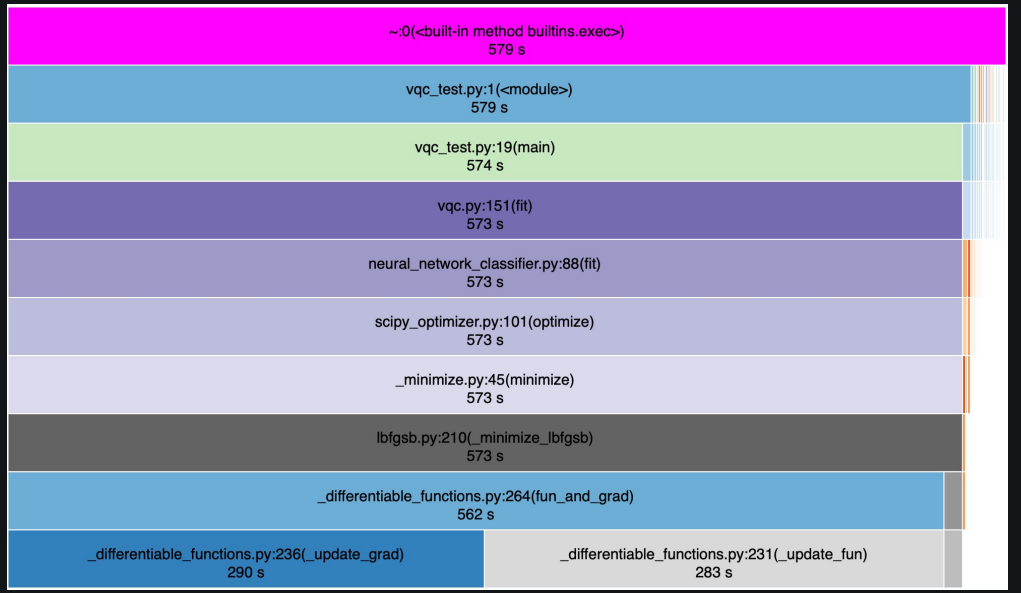
Looking at profiler outputs to detect potential bottlenecks.

Qiskit

A profiler is a tool that allows us to observe how many times a function is called and for how long it runs .

We can use it to identify bottlenecks by looking at;

- Unexpectedly long execution times
- Unexpectedly many number of calls



A profiler output example.*

*The profiler output is more detailed; we are showing a portion of it here for visual purposes.

Qiskit

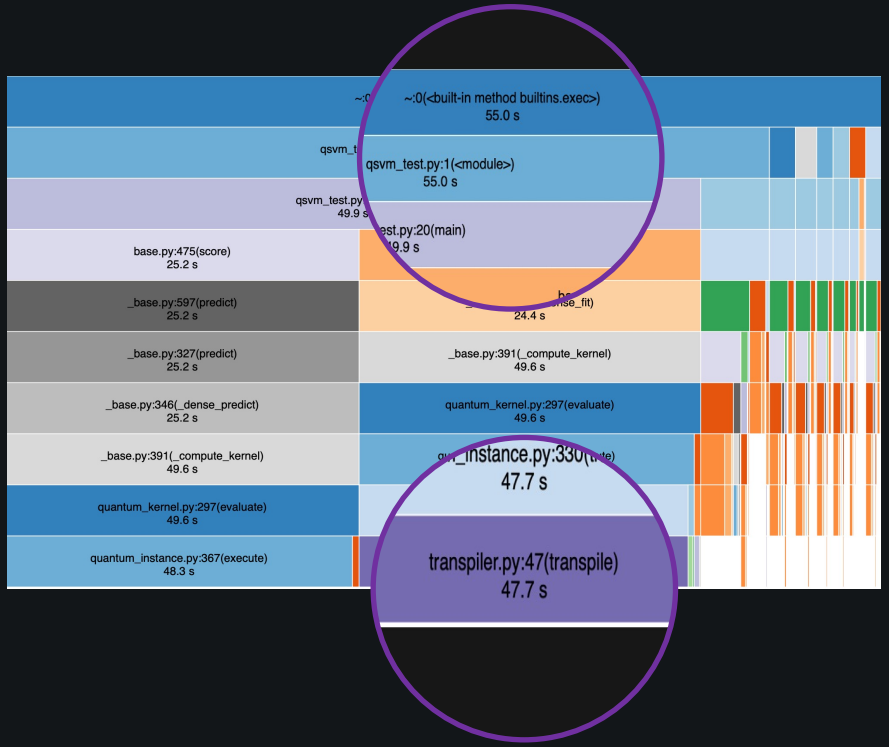
Let’s have a look at how we can produce the profiler output!

Tests with QML algorithms

A test scenario using Quantum Support Vector Machine Classifier (QSVC) from Qiskit QML was deployed.

Profiler results showed that the transpiler takes unexpectedly long and is called many times.

The transpiler is the operation that adapts the Quantum circuit to Quantum hardware according to its qubit connectivity and native gate set.



The Fix

When training a QSVC algorithm, circuits are being transpiled in *quantum_kernel.py* but the necessary *had_transpiled=True* flag was not being passed when the circuits are to be executed.

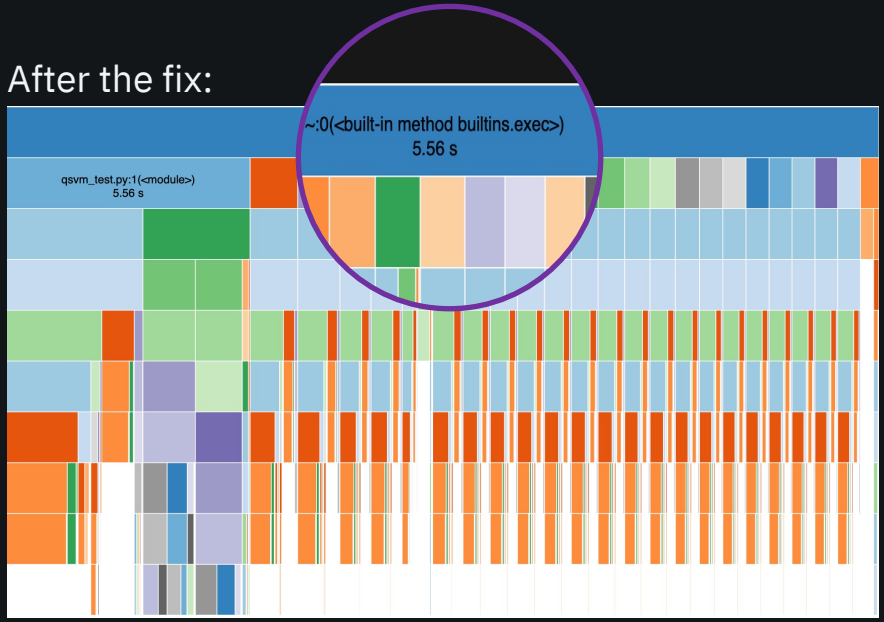
This makes the transpilation operation re-run many times and this slows down the QSVC algorithm.

We were able to reduce total run time by almost 10 times;

55.0 seconds → 5.56 seconds

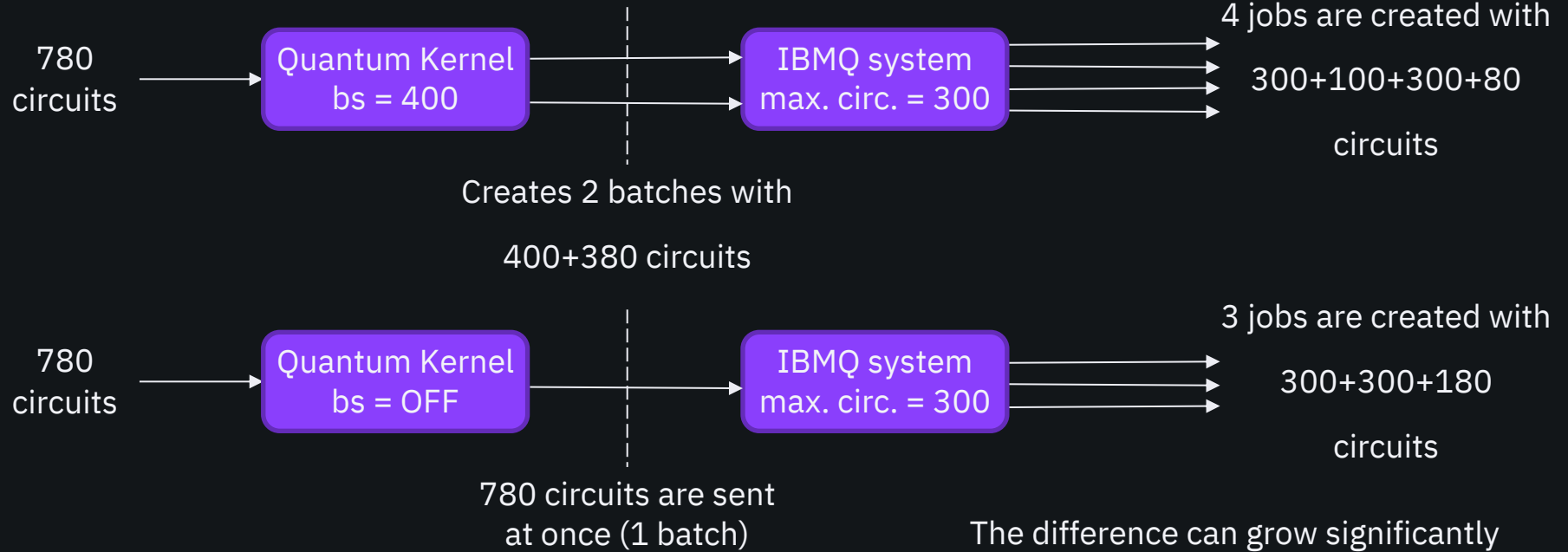
This issue was solved in PR #247

After the fix:



A similar problem was also observed for *CircuitQNN*, and it was fixed by Anton Dekusar in PR #243

Minor performance issue with batch system of Quantum Kernel



The difference can grow significantly with more circuits and worse choice of batch size!

Minor performance issue with batch system of Quantum Kernel

Qiskit

QuantumKernel has a batch feature that allows users to separate *batch_size* of circuits for different jobs.

The *IBMQ* system also has a batch system that uses a pre-set *max_circuits* value.

This feature can create unwanted number of jobs when the user is not careful about the choice of *batch_size* and *max_circuits* of the device. This would become a problem only at large numbers of circuit executions.

To prevent this becoming a problem, we decided to turn of the batch feature of *QuantumKernel* by default. This way advanced users can still leverage the system.

The change is currently in progress under PR#270

Thank you for your time.

Questions?