QML project

Ludovico Venturi

Indice

1 Introduction

1.1 Barren Plateaus

- where the gradient of a cost function vanishes exponentially with system size, rendering the architecture untrainable for large problem sizes
- Random circuits are often proposed as initial guesses for exploring the space of quantum states
- Due to exponential dimension of Hilbert space and the gradient estimation complexity on more than a few qubits
- For a wide class of PQCs, the probability that the gradient along any reasonable direction is non-zero to some fixed precision is exponentially small as a function of the number of qubits[1]

1.1.1 Barren Plateaus on QCNNs

- The variance of the gradient vanishes no faster than polynomially[2] so QCNNs do not exhibit *barren plateaus*.
- It is guaranteed that randomly initialized QCNNs are trainable, unlike many other QNN architectures.

1.2 QCNNs

Introduced in[3]

1.3 NAS

1.4 HierarQcal

HierarQcal is an open-source python package[4] that simplifies the process of creating general quantum convolutional neural networks (QCNNs) by enabling an hierarchical design process. It makes automatic generation of QCNN circuits easy and it facilitates QCNN search space design for neural architecture search (NAS). The package includes primitives such as *convolutions, pooling* and *dense layers* that can be stacked together hierarchically to form complex QCNN circuit architectures.

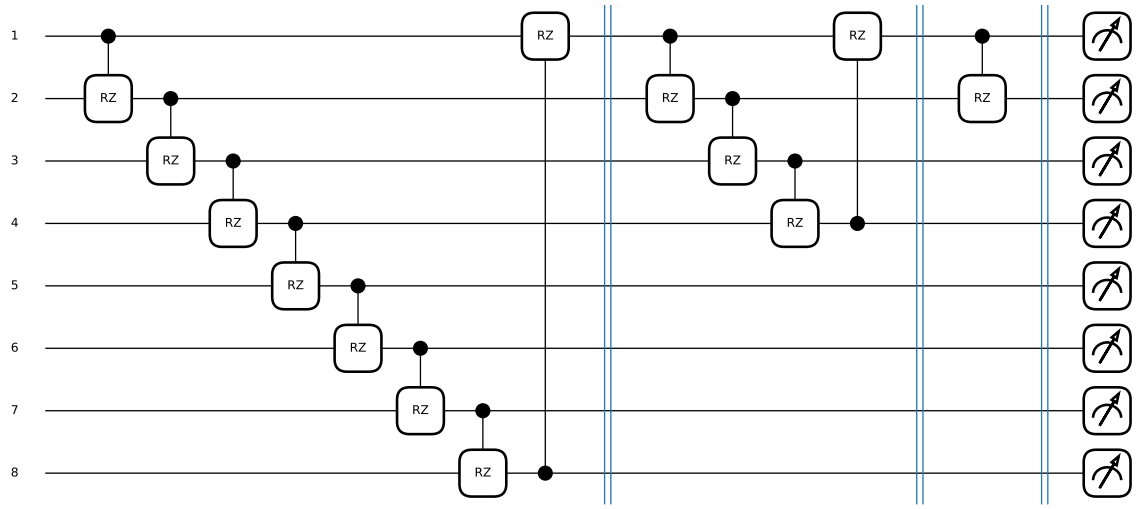
1.4.1 Implementation

```
import numpy as np
import sympy as sp
import matplotlib.pyplot as plt
from hierarqcal import (
    Qhierarchy,
    Qcycle,
    Qpermute,
    Qmask,
    Qunmask,
    Qpivot,
    Qinit,
    Qmotif,
    Qmotifs,
    plot_motif,
    plot_circuit,
    Qunitary,
)
```

```
import pennylane as qml
from hierarqcal.pennylane.pennylane_circuits import V2, U2, V4

def get_circuit(hierq):
    dev = qml.device("default.qubit", wires=hierq.tail.Q)

    @qml.qnode(dev)
    def circuit():
```

```
        if isinstance(next(hierq.get_symbols(), False), sp.Symbol):
            # Pennylane doesn't support symbolic parameters, so if
             ↪ no symbols were set (i.e. they are still symbolic),
             ↪ we initialize them randomly
            hierq.set_symbols(np.random.uniform(0, 2 * np.pi,
 ↪ hierq.n_symbols))
        hierq(
            backend="pennylane"
        )  # This executes the compute graph in order
        return [qml.expval(qml.PauliZ(wire)) for wire in
         ↪ hierq.tail.Q]

    return circuit

def draw_circuit(circuit, **kwargs):
    fig, ax = qml.draw_mpl(circuit)(**kwargs)
```

```
m = Qcycle(1) + Qmask("right")
heirq = Qinit(8) + m*3
circuit = get_circuit(heirq)
draw_circuit(circuit)
```



References

1. McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R. & Neven, H. Barren plateaus in quantum neural network training landscapes. *Nat. Commun.* **9**, 1–6 (2018).
2. Pesah, A. *et al.* Absence of Barren Plateaus in Quantum Convolutional Neural Networks. *Phys. Rev. X* **11**, 041011 (2021).
3. Cong, I., Choi, S. & Lukin, M. D. Quantum convolutional neural networks. *Nat. Phys.* **15**, 1273–1278 (2019).
4. matt-lourens. hierarqcal. *GitHub* (2024).