NTU-IBM Q System
2020 Q-Camp

Predicting molecules ground state energies and handwriting recognition with parametrized quantum circuit by using

Group members: 郭庭愷、劉峻瑋、張中瀚、高魁駿、李孟璋

Coach: 蔡沛愷

Reference

Article

Hybrid Quantum-Classical Neural Network for Calculating Ground State Energies of Molecules

It combines the VQC and classical neural network

The qiskit official documentation : https://qiskit.org/textbook/ch-machine-learning/machine-learning-qiskit-pytorch.html

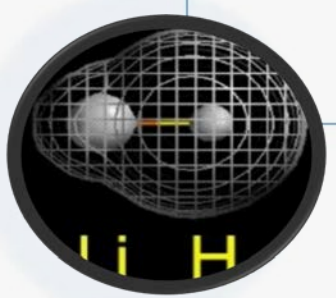
The metalearner using quantum circuit+LSTM:
https://github.com/BoschSamuel/QizGloria/tree/master/Notebooks

Our Task

To predict the ground state energy of two-atom molecule (hydrogen-like) with difference bond length

How to generate the data?

different bond lengths



LiH molecule

Use exact diagonization to solve ground state as **ground truth (dim= 1)**

Transform the Hamiltonian of system to the sum of 4-pauli operators, totally 100 terms

The number of training samples =3500 (bond lengths from 0.5 to 4.0 angstrom)

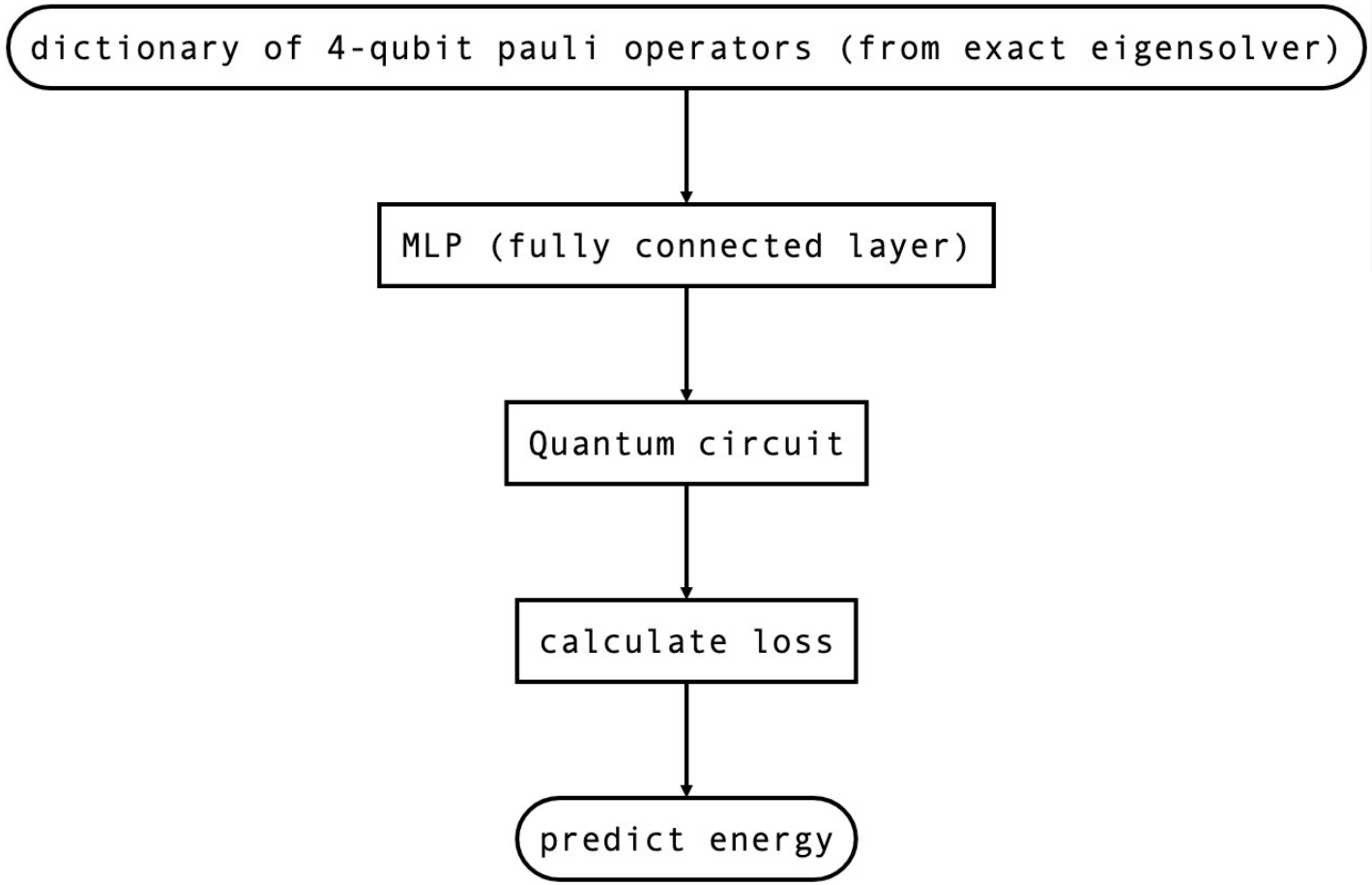
Gather the coefficients of Pauli terms to a **feature vectors (dim=100x1)**

$$\hat{H}_{molecule} = \sum_{i=1,\ldots,100} c_i \hat{P}_i$$

$$\mathrm{P} = \mathrm{IIZI},\ldots,\mathrm{IXII},\ldots,\ \mathrm{YIII},\ldots$$

...

Flow and Data Structure




```
├── LiH (Hydrogen-like molecules)
    ├── train
        ├── class1(IIII)
            ├── coefficient
        ├── class2(IIIZ)
            ├── coefficient
        ├── class3 (IIZI)
            ├── coefficient
        ├── ...
    ├── val
    ├── test
 -> save to csv
```

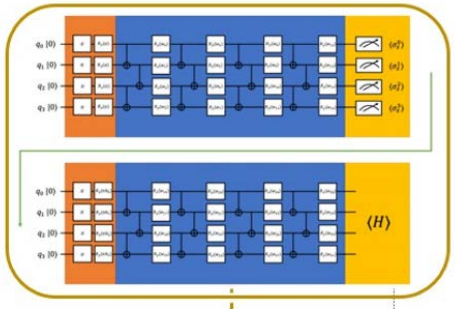
AQGD Code Snippet

```
for k in range(NUM_QUBITS):
    shift_right = input_numbers.detach().clone()
    shift_right[k] = shift_right[k] + SHIFT
    shift_left = input_numbers.detach().clone()
    shift_left[k] = shift_left[k] - SHIFT
```

Ref:

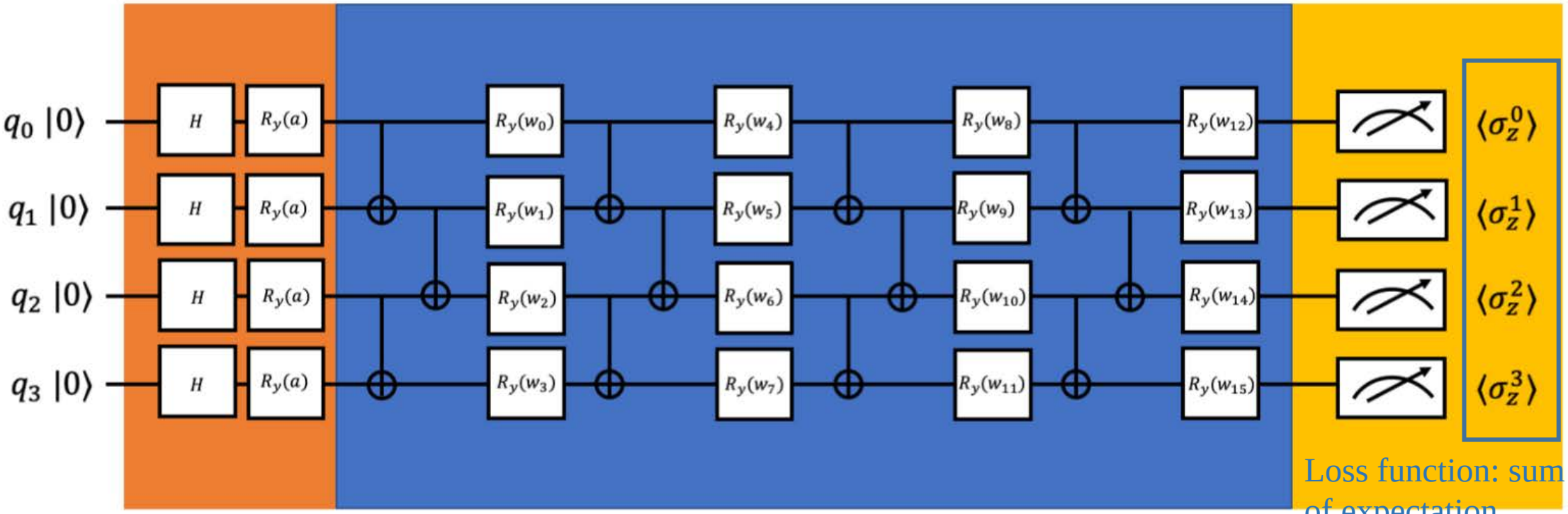
Valuating analytic gradient on quantum hardware, PRA 99,032331 (2019)

Recall:



Parametrized Quantum Circuit (PQC)

We changed our plan and focus on the quantum block of original model



Loss function: sum of expectation

$$L(W) = \sum_i \langle\sigma_i\rangle$$

Optimizer: **AQGD**

W'

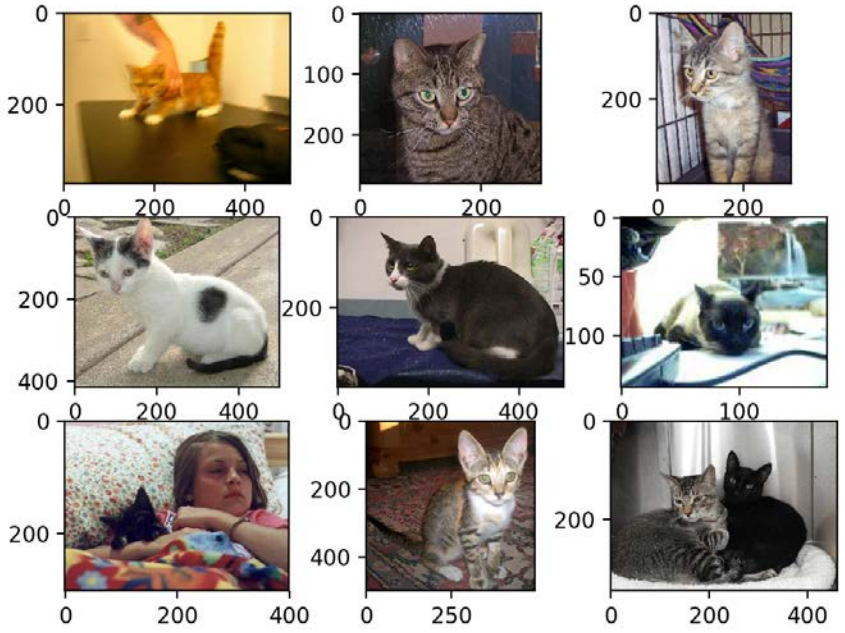
PQC+AQGD

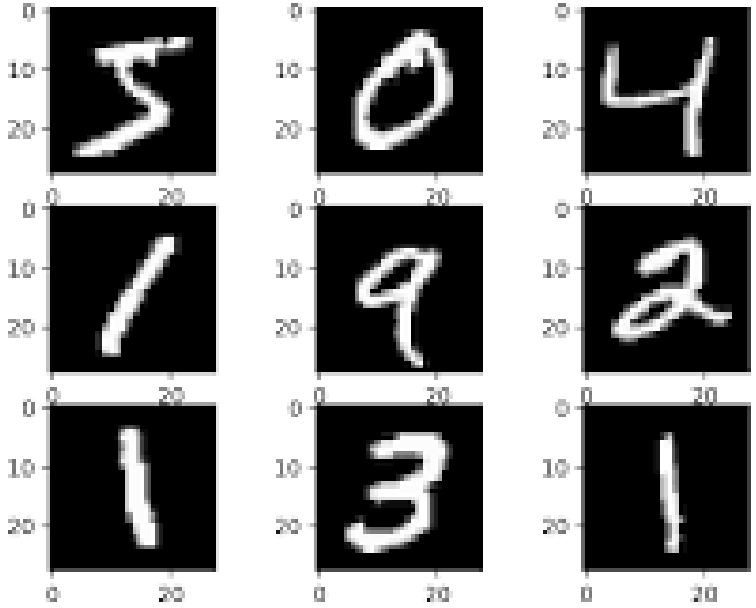
If we only use the single-qubit rotation gate in the VQE, We can **exactly** derive the gradient of loss function by **AQGD**
formulation rather than approximated gradient.

$$W' = W - \nabla W \eta, W = [w_0, ..., w_{15}]^T$$

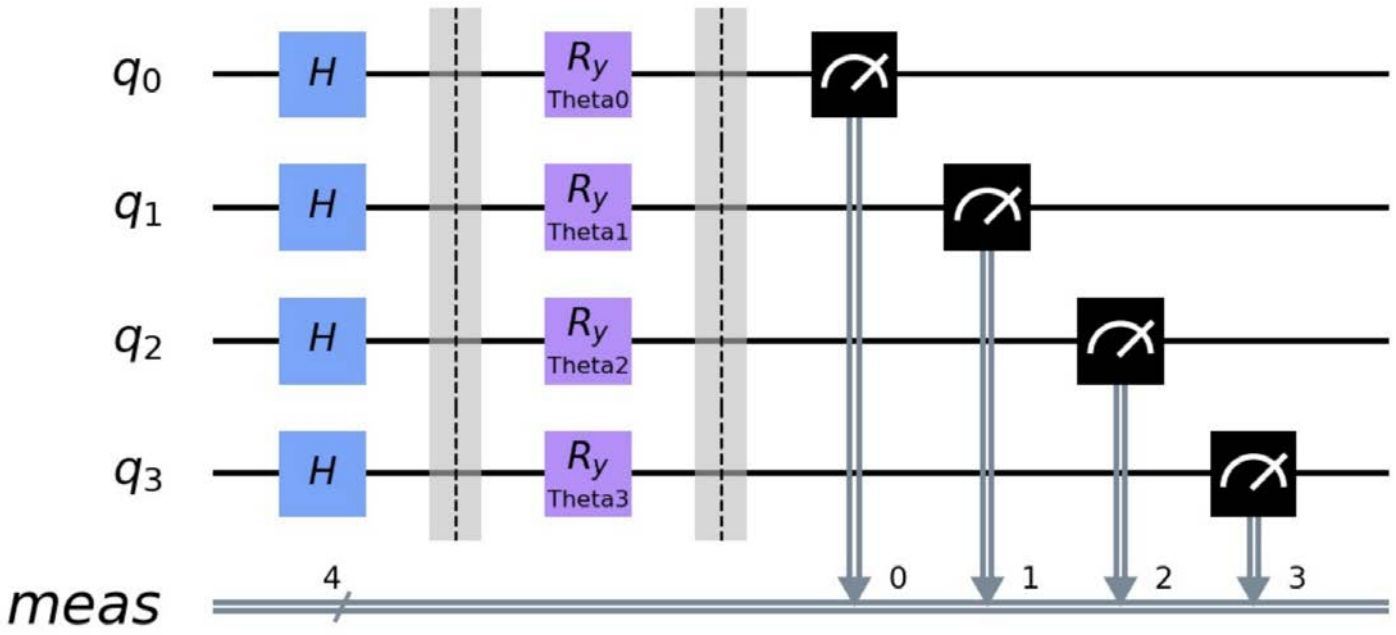
$$r\left(f(\mu + s) - f(\mu - s)\right) \qquad s = \frac{\pi}{2}, r = \frac{1}{2}$$

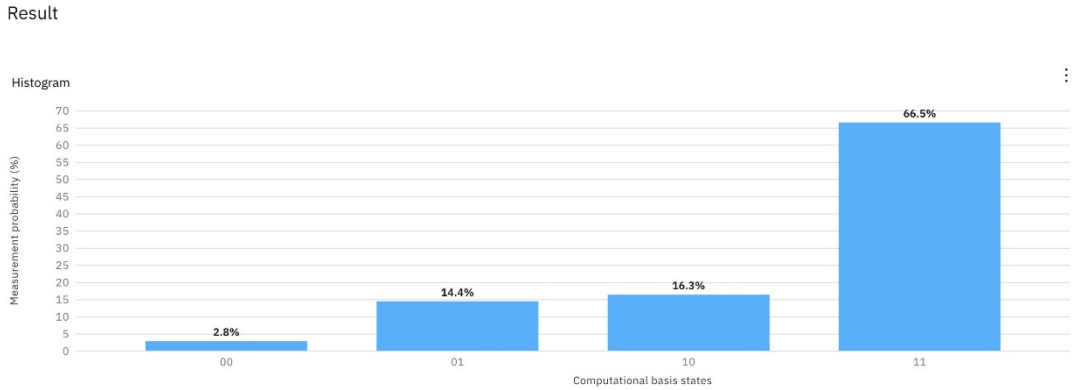
We also apply model to the MNIST and Dog-and-Cat dataset

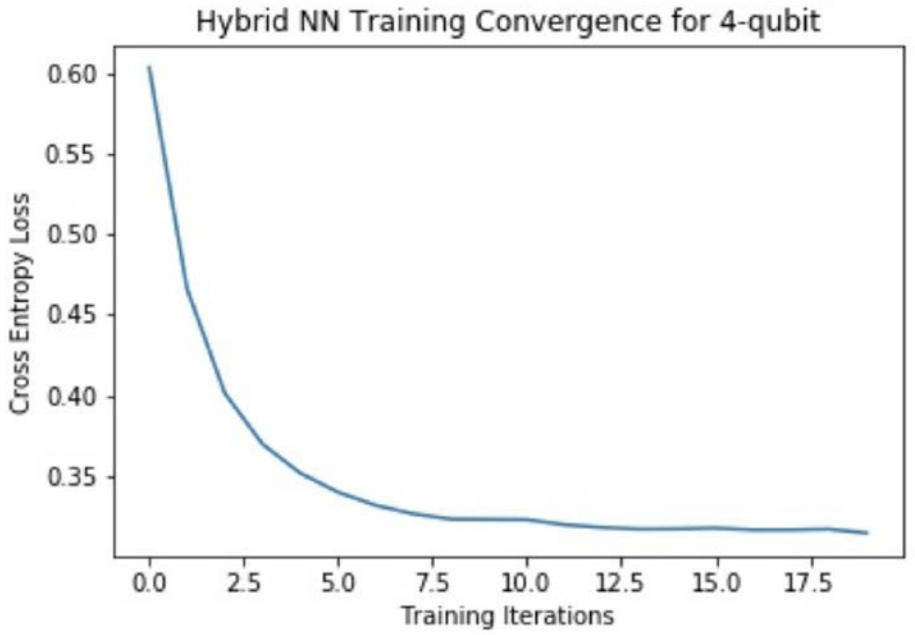




RyN: apply Hadamard gates and Ry gates to N-qubit and measure the qubits

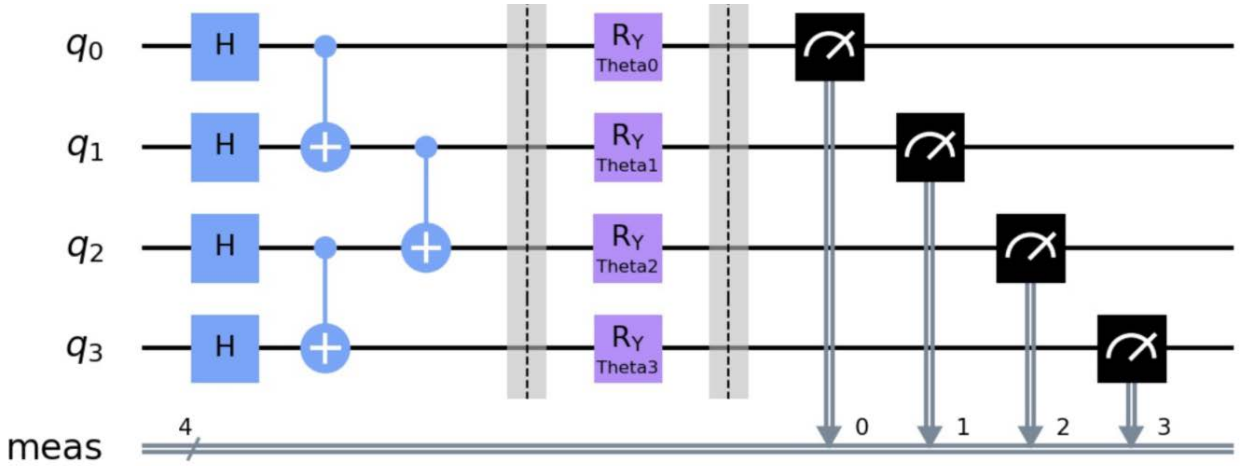


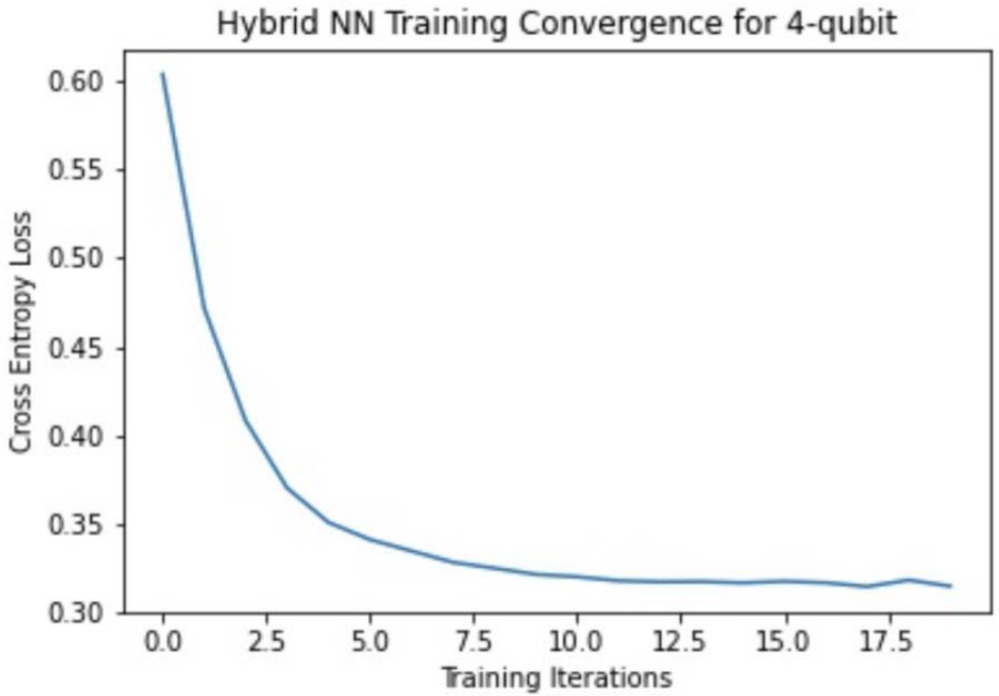




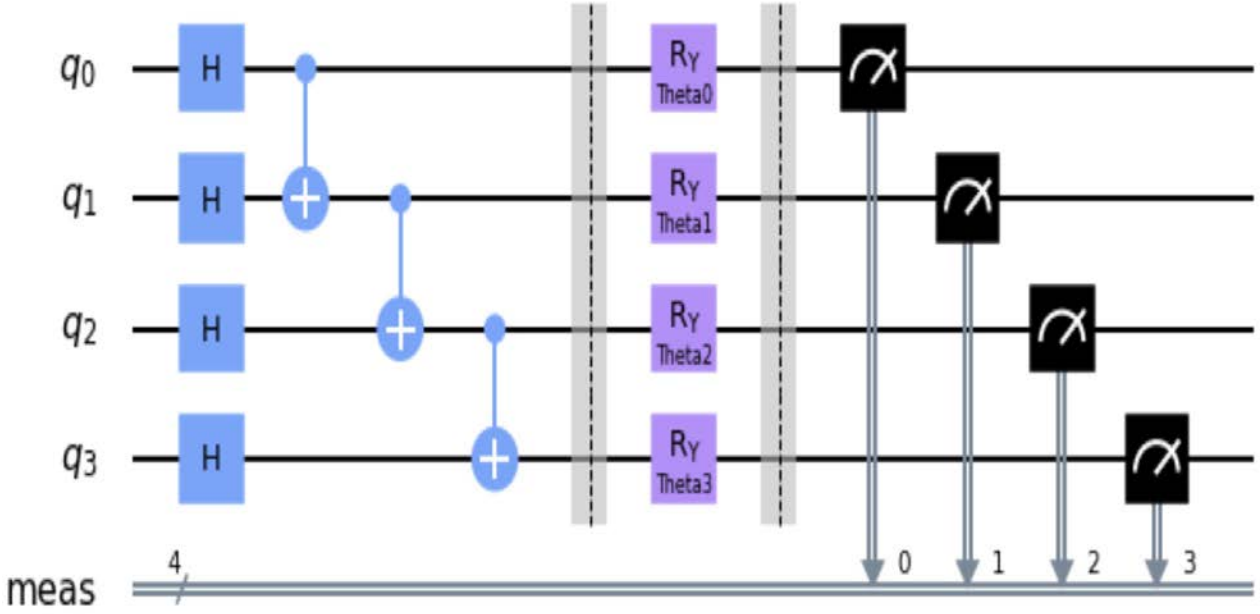
Ladder Circuit(Using staggered CNOT gates to make more entangled)

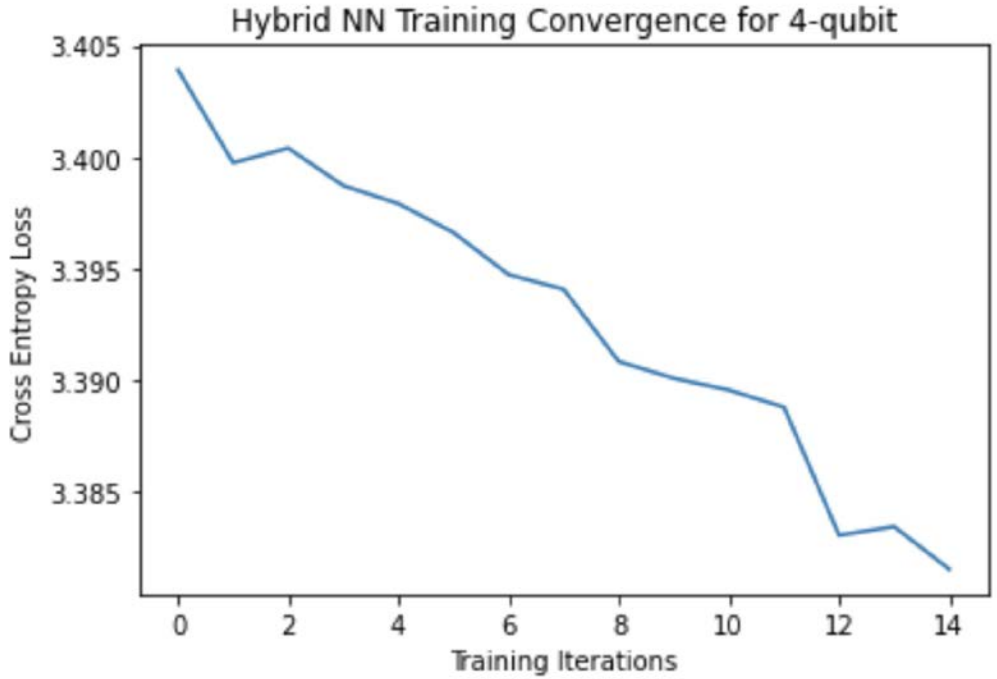
The original circuit block



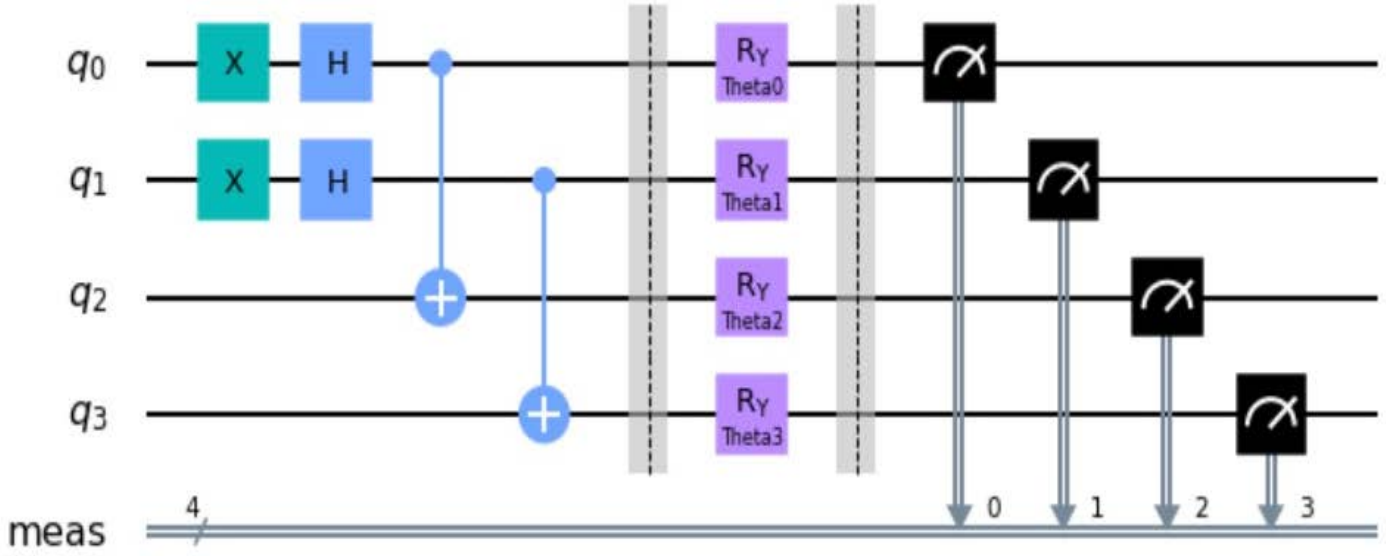


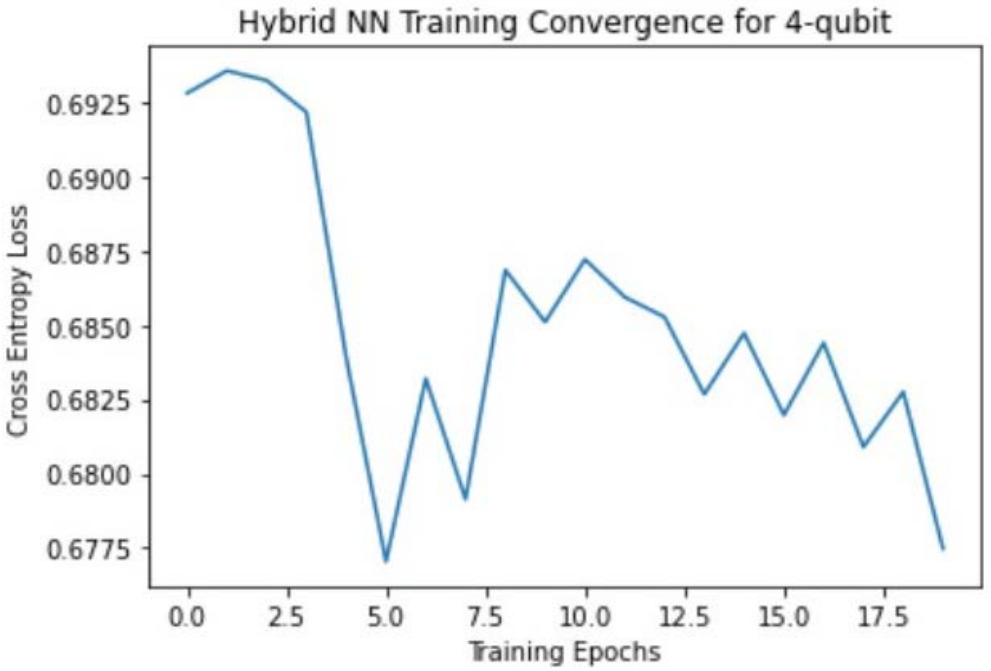
The GHZ version



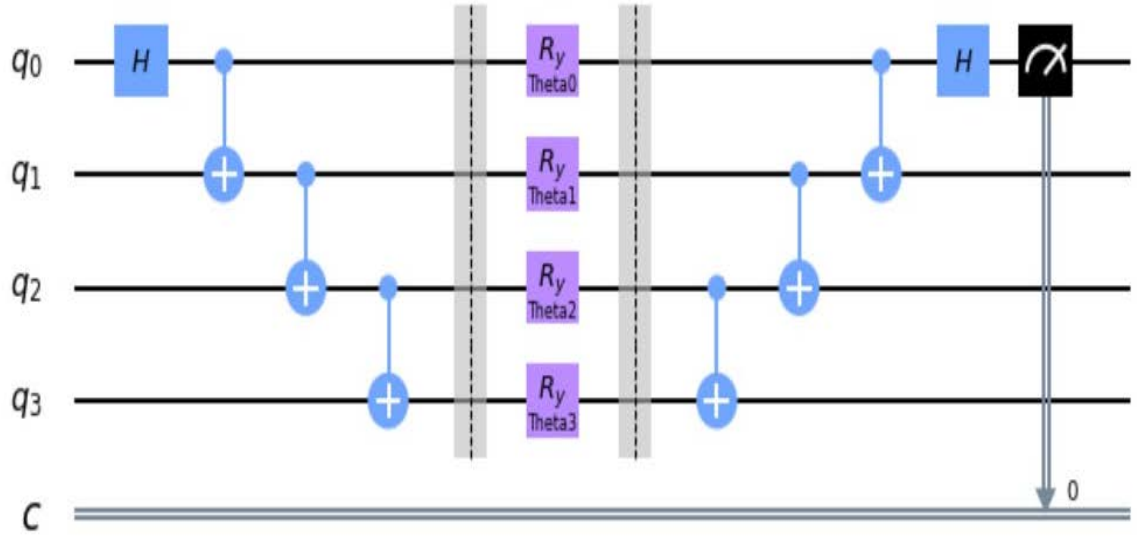


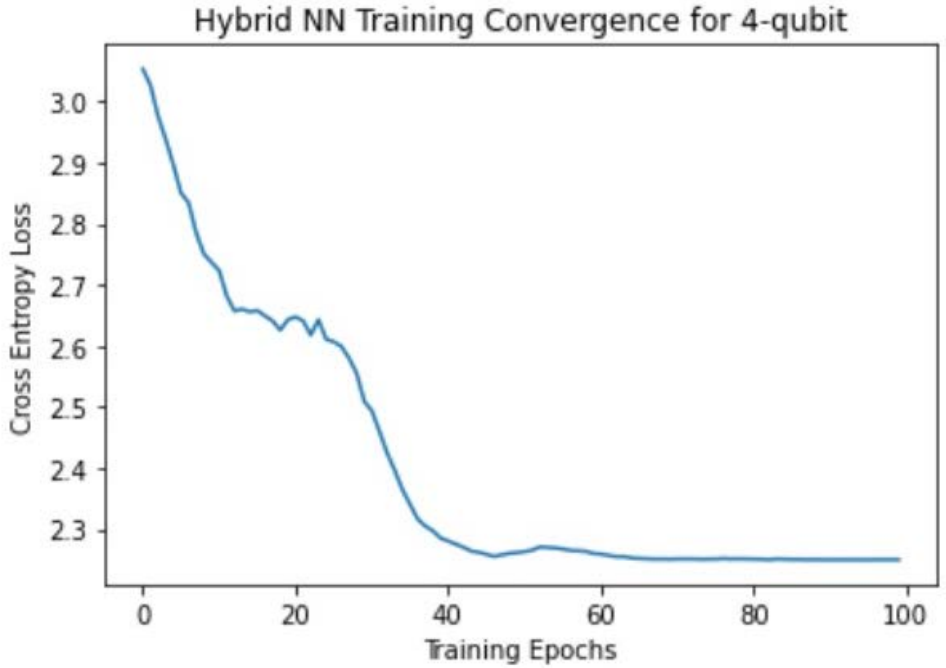
Sub-block of MPS-inspired ansatz





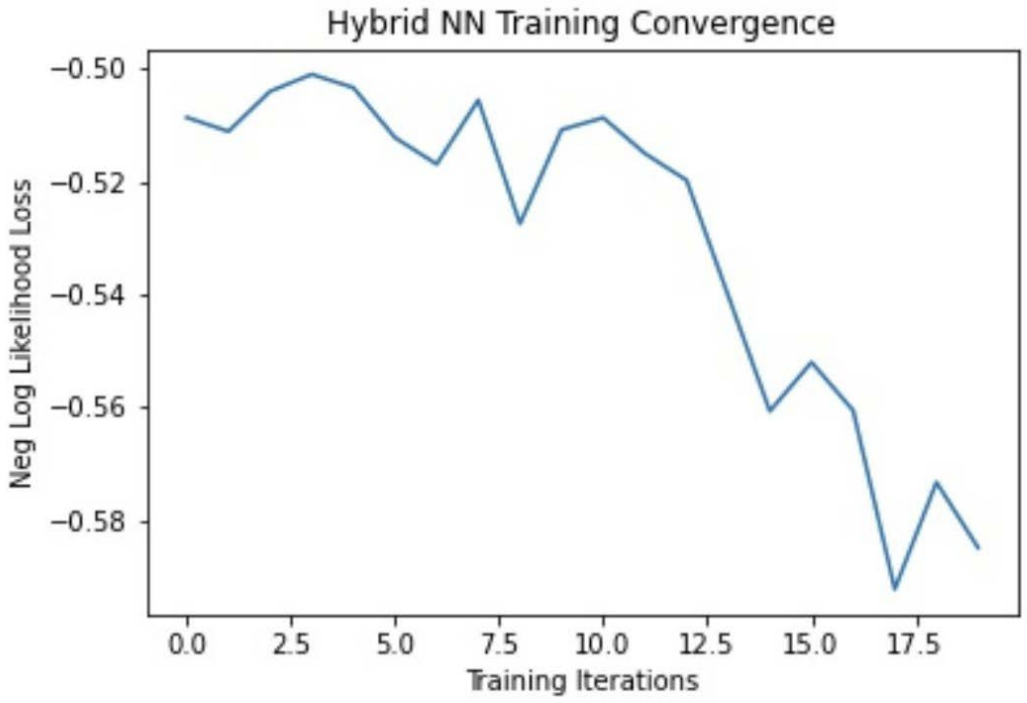
Bell state


q_0
q_1
q_2
q_3
c
H
R_y
Theta0
R_y
Theta1
R_y
Theta2
R_y
Theta3
H
0

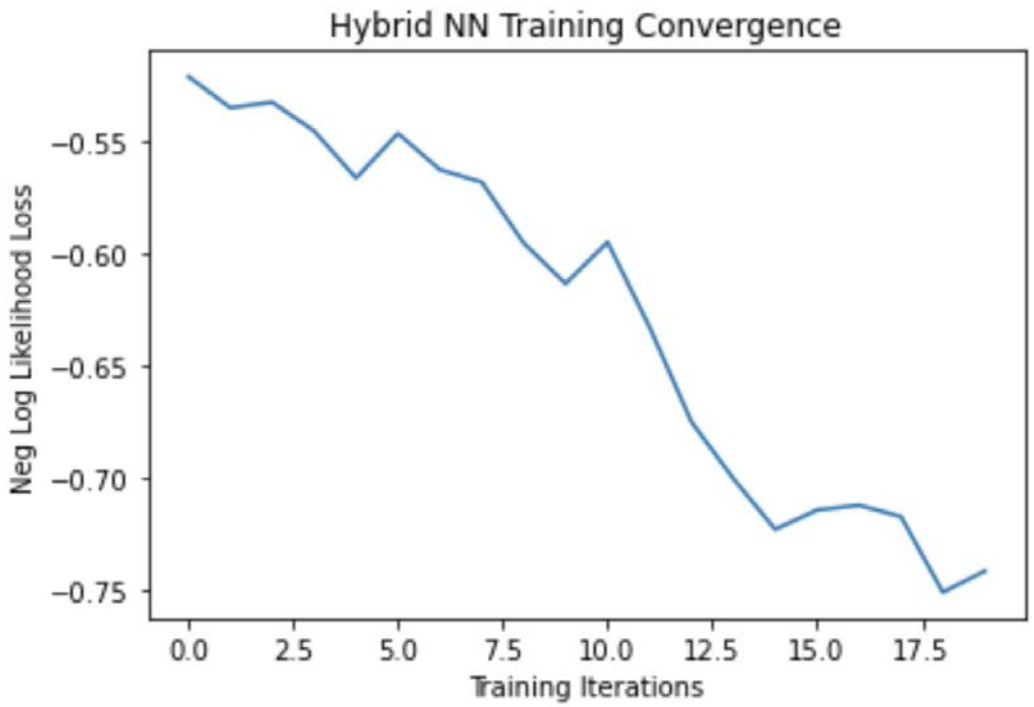

Hybrid NN Training Convergence for 4-qubit
Cross Entropy Loss
3.0
2.9
2.8
2.7
2.6
2.5
2.4
2.3
0
20
40
60
80
100
Training Epochs

U3(unitary operation to three phase angles) gate run on MNIST dataset

Run on CPU, Accuracy: 75%



Run on GPU, Accuracy: 87%



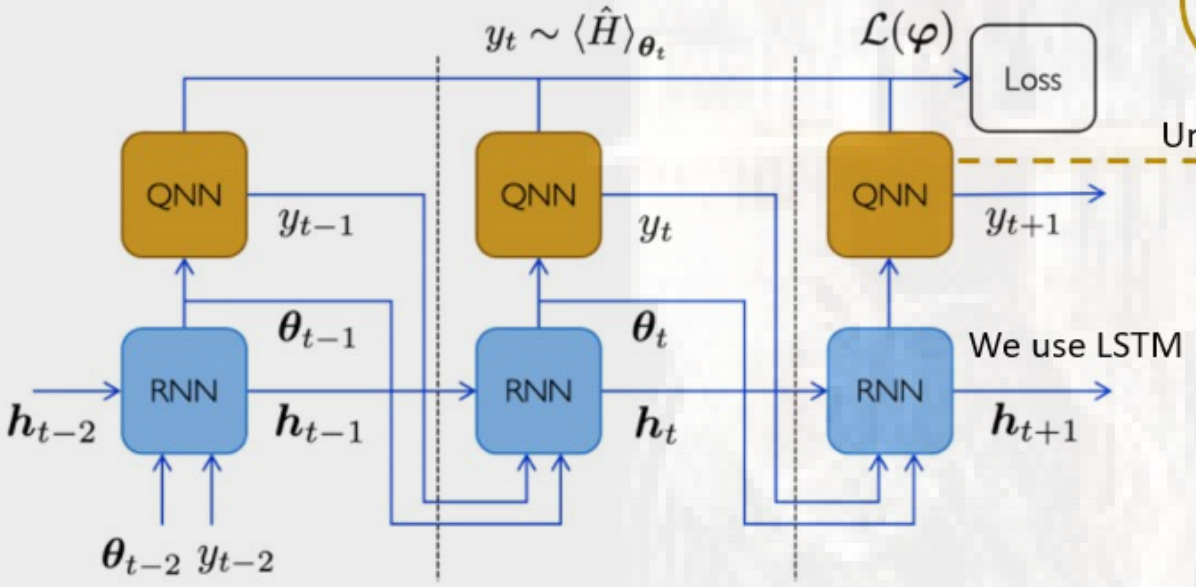
Accuracy Comparison Between Difference RyN Gates Circuit

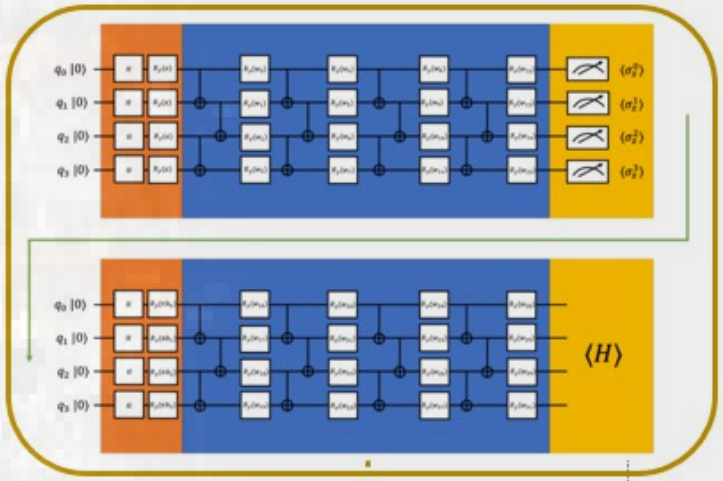
Accuracy Of Our Hybrid Quantum Circuits.	
Model	MNIST $\in 0, 1$
ryN	99.9%
Ladder	99.9%
GHZ	85%
MPS-inspired	52.7%
Bell state	78%

In Progress

The architecture of model: LSTM meta-learner

It was our original plan. But we met the runtime problem in the back-propagation step.





Unfold the QNN block

(actually is aVQE)

Meta-learning is for **few-shot** learning. It can rapidly decrease the training loss by fewer samples and epochs because it utilize the memory of parameters of quantum circuit from the previous training steps.

In Progress:

Code Cat&Dog

```python
class QiskitCircuit():
    def __init__(self, n_qubits, backend, shots):
        # --- Circuit definition ---
        self.circuit = qiskit.QuantumCircuit(n_qubits)
        self.n_qubits = n_qubits
        self.thetas = {
            k: Parameter('Theta' + str(k))
            for k in range(self.n_qubits)
        }

        all_qubits = [i for i in range(n_qubits)]
        self.circuit.h(all_qubits)
        self.circuit.cx(0, 1)
        self.circuit.cx(2, 3)
        self.circuit.cx(1, 2)
        self.circuit.barrier()
        for k in range(n_qubits):
            self.circuit.ry(self.thetas[k], k)
```

```python
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.vgg = models.vgg19_bn(pretrained = True)
        for param in self.vgg.features.parameters():
            param.requires_grad = False

        self.vgg.classifier = nn.Sequential(nn.Linear(25088, 4096),
                                            nn.ReLU(),
                                            nn.Dropout(0.5),
                                            nn.Linear(4096, 512),
                                            nn.ReLU(),
                                            nn.Dropout(0.5),
                                            nn.Linear(512, 256)
                                            )

        #self.conv1 = nn.Conv2d(3, 10, kernel_size=5)
        #self.conv2 = nn.Conv2d(10, 20, kernel_size=5)
        #self.conv2_drop = nn.Dropout2d()
        #self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(256, NUM_QUBITS)
        self.qc = TorchCircuit.apply
        self.fc3 = nn.Linear(2**NUM_QUBITS, 2)
        #self.qc = TorchCircuit.apply
```

Summaries

- It seems sequential entanglement, exhibiting more benenificial to converge loss than that in long range one.
- more entaglement, more powerful
  - Explaination: if only product state in the circuit， it is more similar to separately predict the bit of the binary representation of 0~9 rather than the whole number.

In The Future

- Implement hybrid neural network model with VQE to predict the ground state energy of different hydrogen-like molecules.(in progress)
- Try training our neural nets on the actual IBM machine (have attempted)
- Compare results from an actual hardware with that from the simulator
- Implement different image dataset on complex classical model + quantum circuit (in progress)

Thanks for your listening!