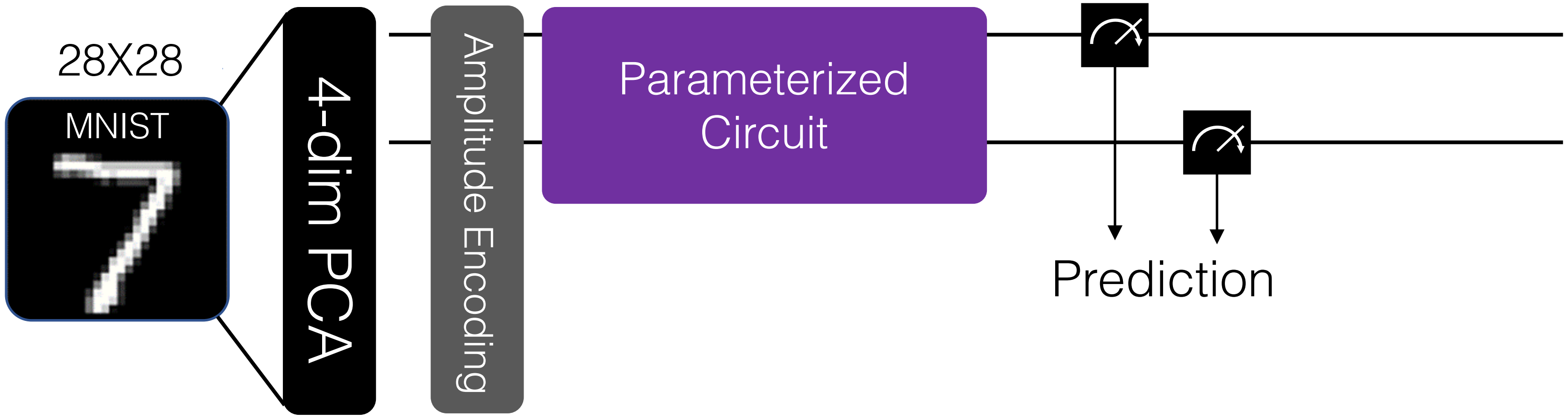
Quantum NAS

DS4QISKIT 1조

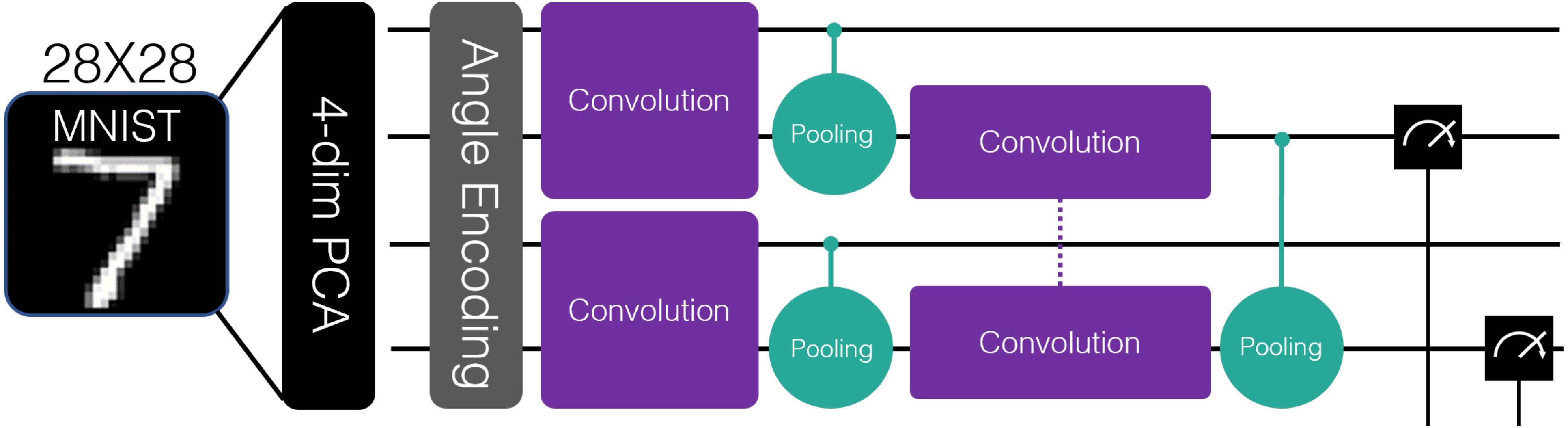
Intuition

Quantum Neural Network



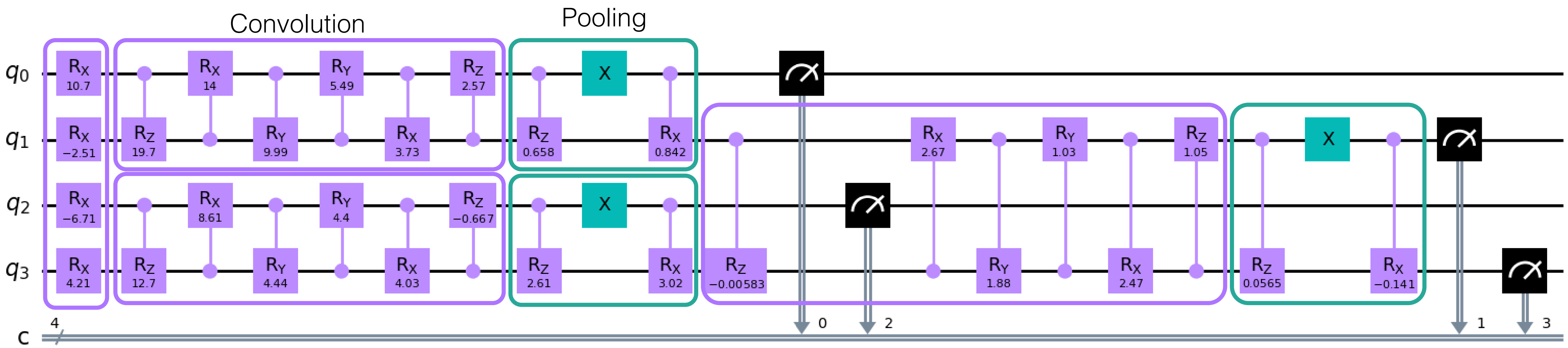
Intuition

Quantum Convolutional Neural Network

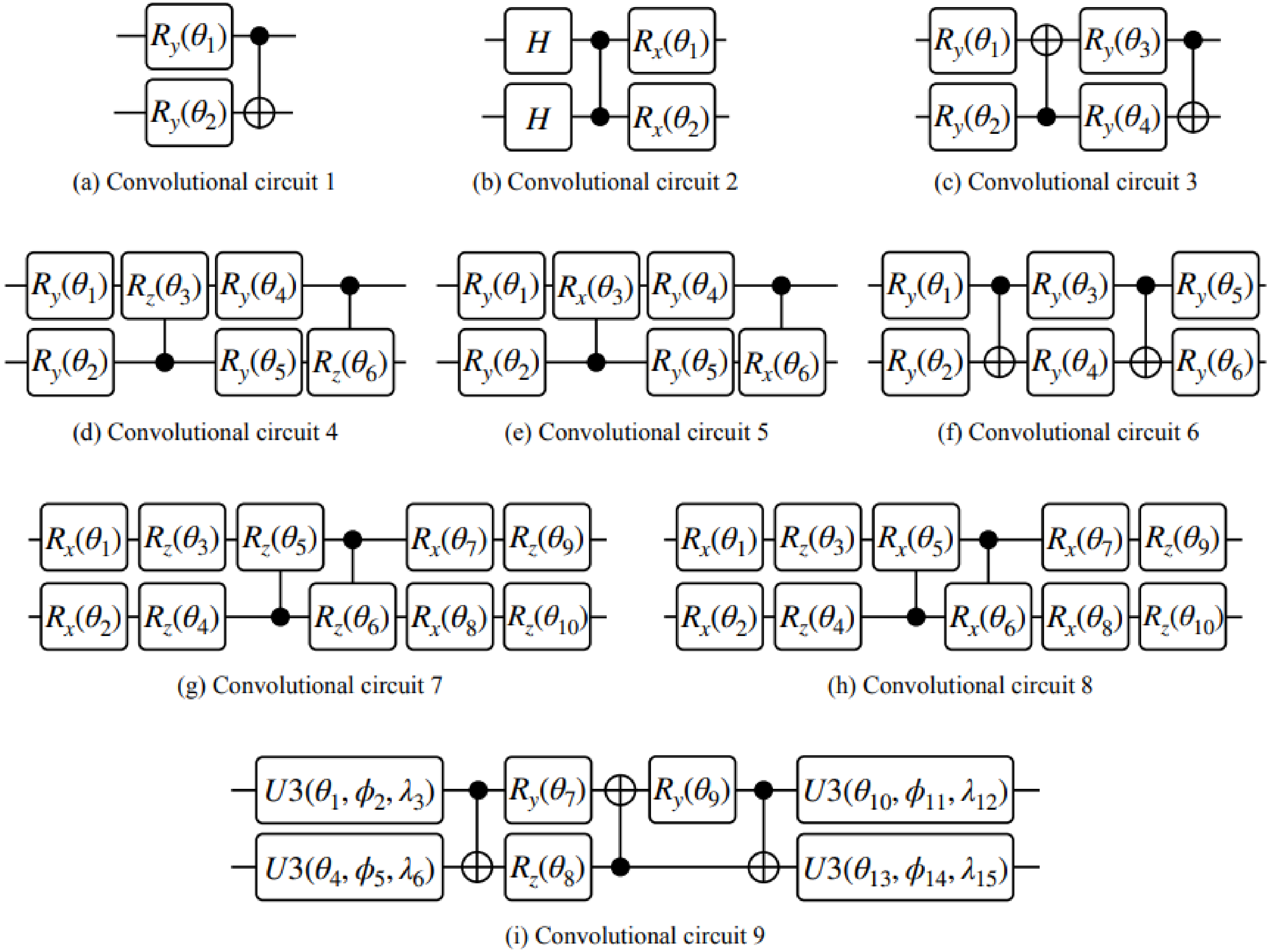


Intuition

Quantum Convolutional Neural Network



Intuition



Parameterized quantum circuits used in the convolutional layer.

Hur, Tak, Leeseok Kim, and Daniel K. Park. "Quantum convolutional neural network for classical data classification." *Quantum Machine Intelligence* 4.1 (2022): 1-18.

Can we search the "best circuit architecture"?

Heuristic Search

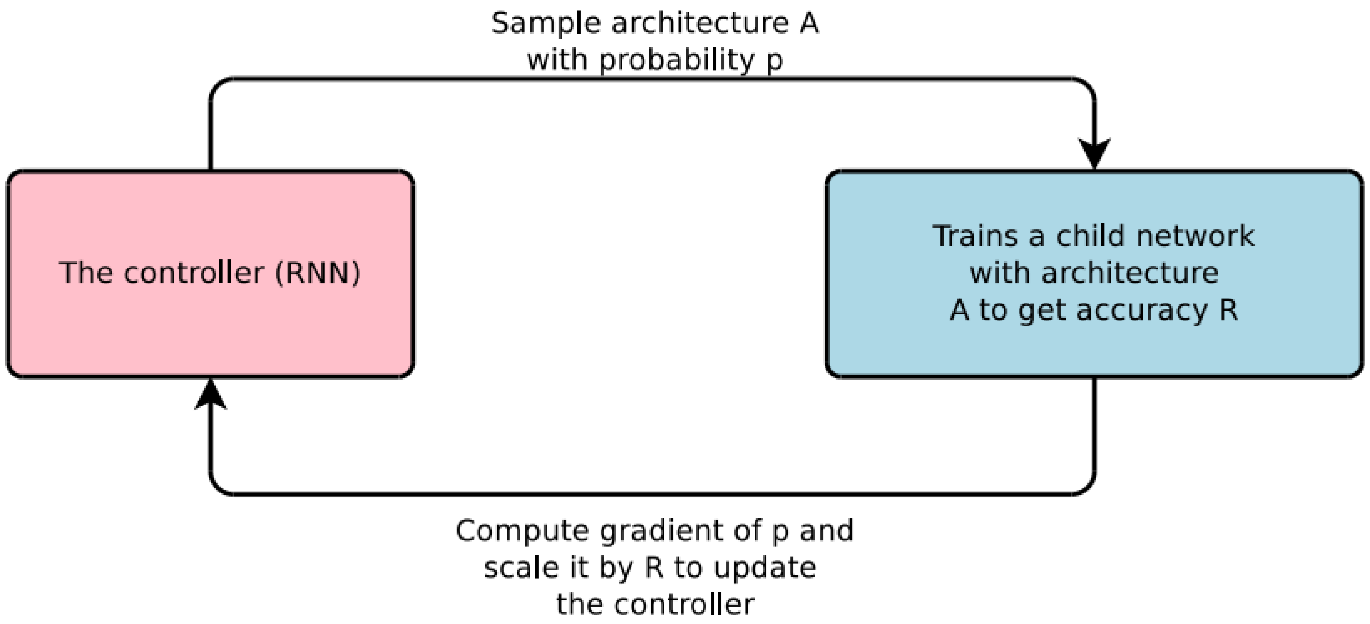
Metric-based
Entanglement & Expressibility

DL/RL

Intuition

Neural Architecture Search(NAS)

Fundamental Concept of RNN-controller based NAS (2016, Zoph et al.)



Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).

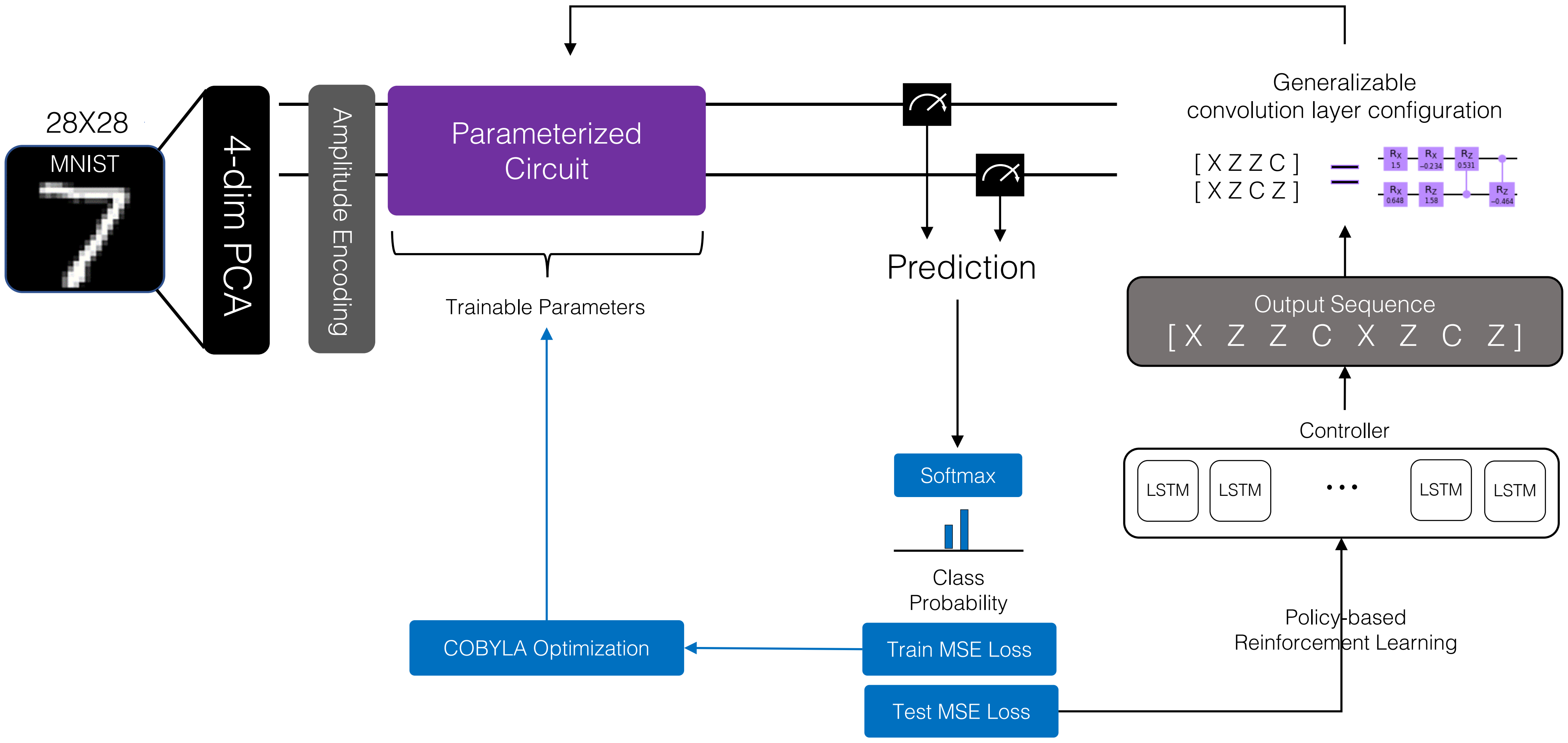
NAS algorithm can give 'accuracy score & loss' of configured model as a 'reward' to RNN controller, to perform reinforcement learning.

Updated RNN Network returns Configuration of CNN Network that performs classification task.

(Example for CNN-NAS)
1. RNN Network > Return [2,4,4]
2. Configure CNN Network with 3 layers with filter number [2,4,4]
3. Train & assess accuracy of configured CNN Network
> Return loss 0.042 & accuracy 0.976
4. Update RNN Network via Reinforcement Learning

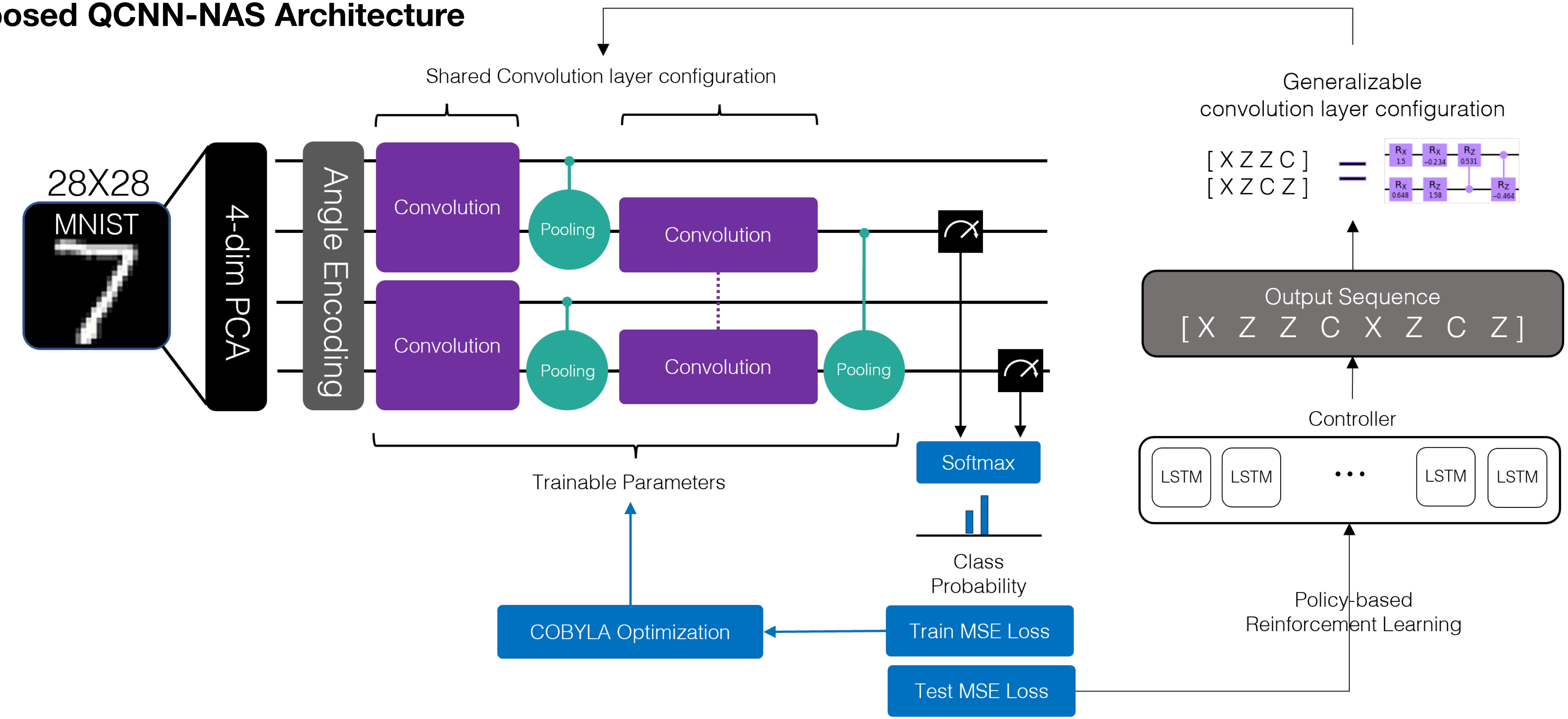
Method

Proposed QNN-NAS Architecture



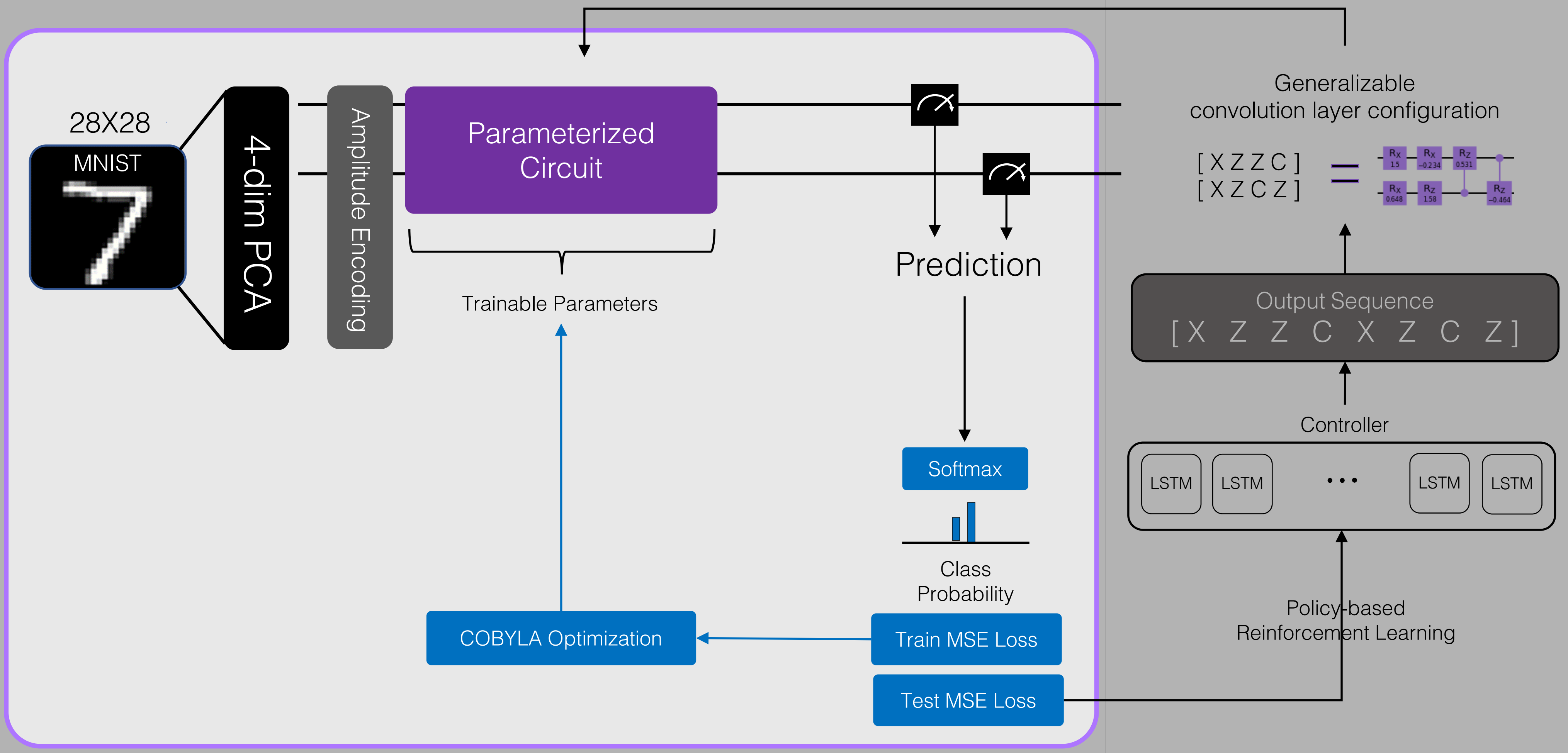
Method

Proposed QCNN-NAS Architecture



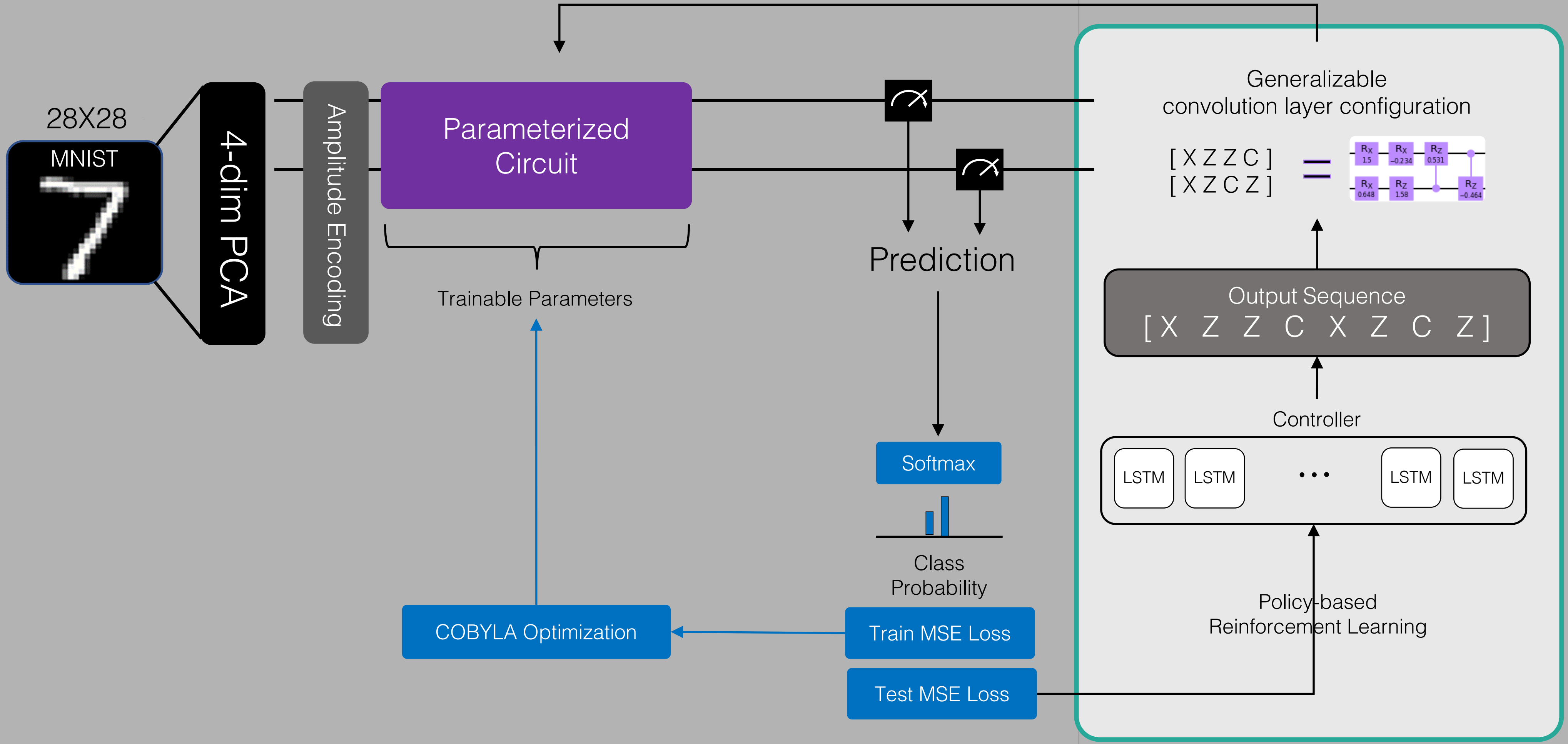
Method

Configurable QNN



Method

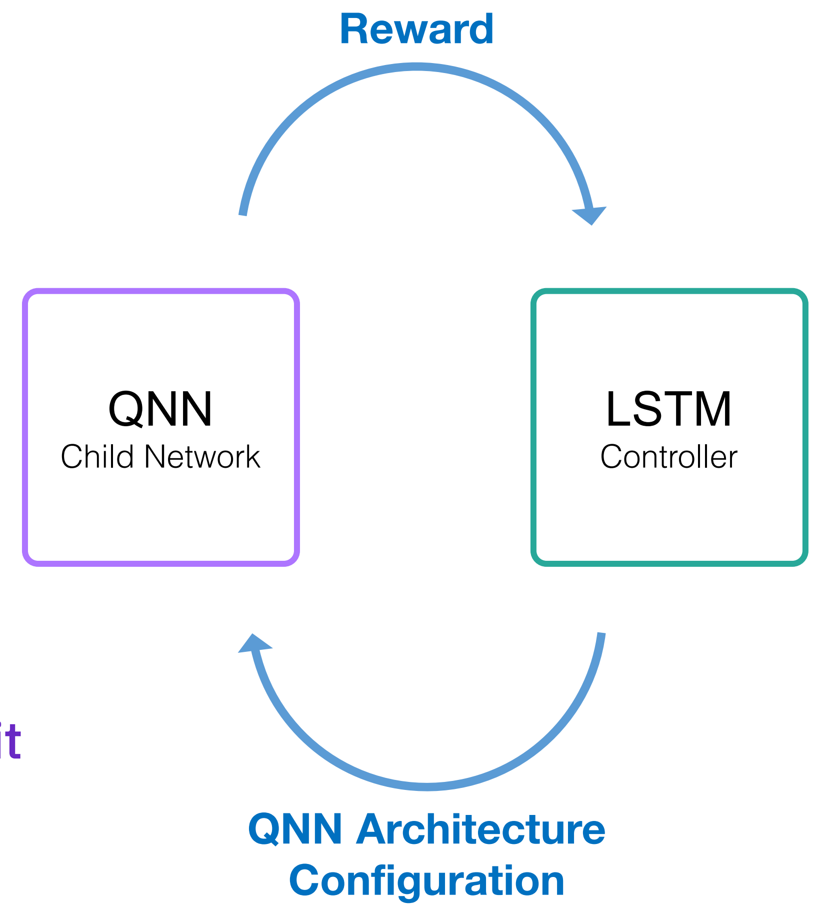
Controller LSTM



Method

Overall Architecture

Trained via
Classical optimization
COBYLA



Trained via
Reinforcement Learning

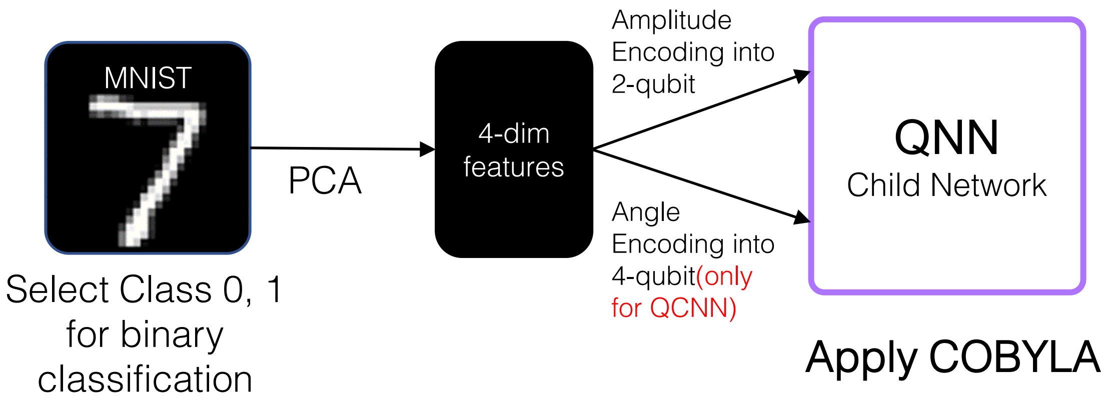
Backend simulator with Qiskit

Plugin with Pennylane

Reinforcement Learning on PyTorch

Results

QNN Configuration & Accuracy (Without NAS)



Apply COBYLA optimization algorithm with

Train set size = 10 (Very small)
Validation set size = 500 (Bigger for robust assessment)

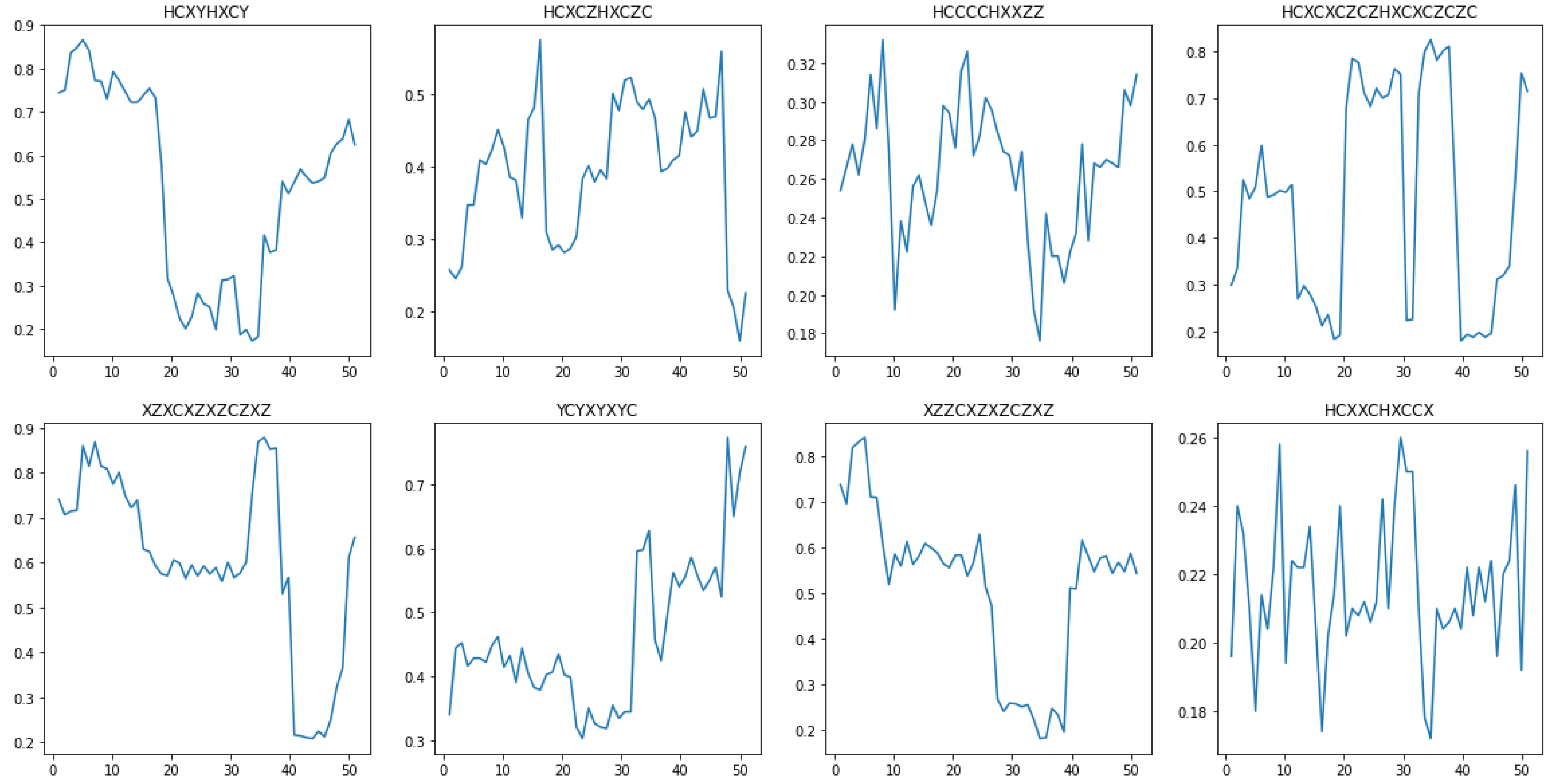
Select minimum Validation loss parameter for testing

Test set size = 2000 (Biggest)

Results

QNN Configuration & Accuracy (Without NAS)

Fails to converge with COBYLA optimization



Validation loss
(= 1-accuracy)
of various QNN circuits.

Results

QNN Configuration & Accuracy (Without NAS)

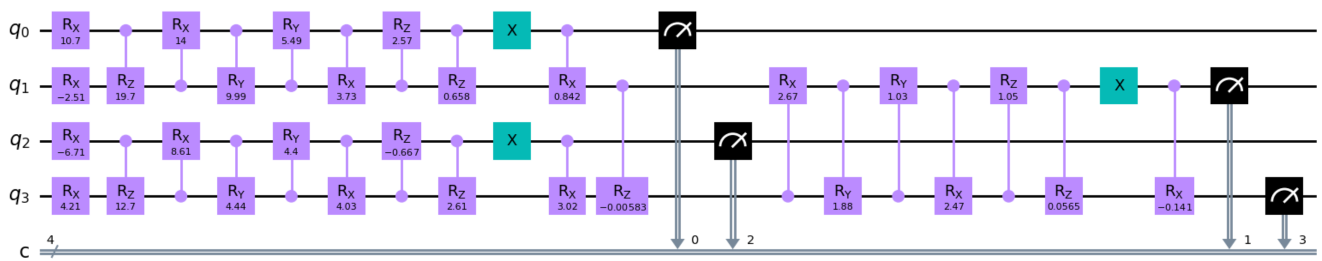
Fails to converge with COBYLA optimization but shows consistency on test dataset

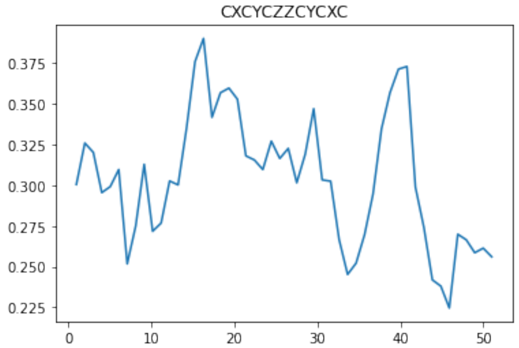
Configuration	Best valid loss	Test loss	# of params
HCXY HXCY	0.172	0.311	4
HCXCZ HXCZC	0.16	0.1955	4
HCCCC HXXZZ	0.176	0.349	4
HCXCXCZCZ HXCXCZCZC	0.18	0.267	8
XZXCXZ XZCZXZ	0.208	0.2125	10
YCYX YXYC	0.302	0.42	8
XZZCXZ XZCZXZ	0.18	0.2785	10
HCXXC HXCCX	0.172	0.3425	4

Results

QCNN Configuration & Accuracy (Without NAS)

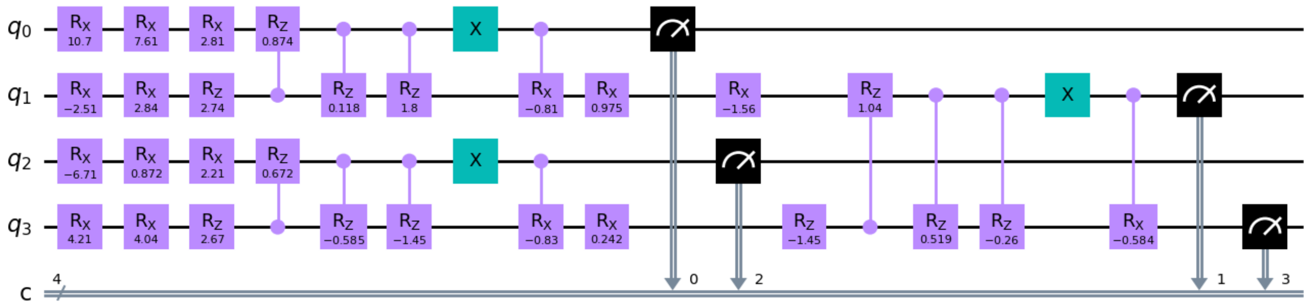
Fails to converge with COBYLA optimization

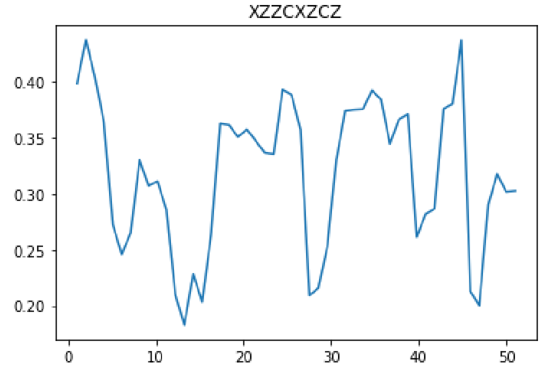




Convolution circuit config
CXCYCZ
ZCYCXC

Minimum validation loss : 0.225
Test loss : 0.291





Convolution circuit config
XZZC
XZCZ

Minimum validation loss : 0.183
Test loss : 0.247

Validation loss
(= 1-accuracy)
of various QCNN circuits.

Results

QCNN Configuration & Accuracy (Without NAS)

Fails to converge with COBYLA optimization but shows consistency on test dataset

Our observation

1. Length of configuration matters (Too short configuration shows poor performance)

   4-length configuration takes half training time compared with 5-length configuration

2. Entanglement(like CX- XC) matters

3. These symmetric circuits are quite preforming well(60~80% accuracy), but overfitting exists

"Let's find the best architecture in constrained-length (5) architecture space!"

Fixed Length(5) architecture

Only select elements (C, H, X, Y, Z) with LSTM

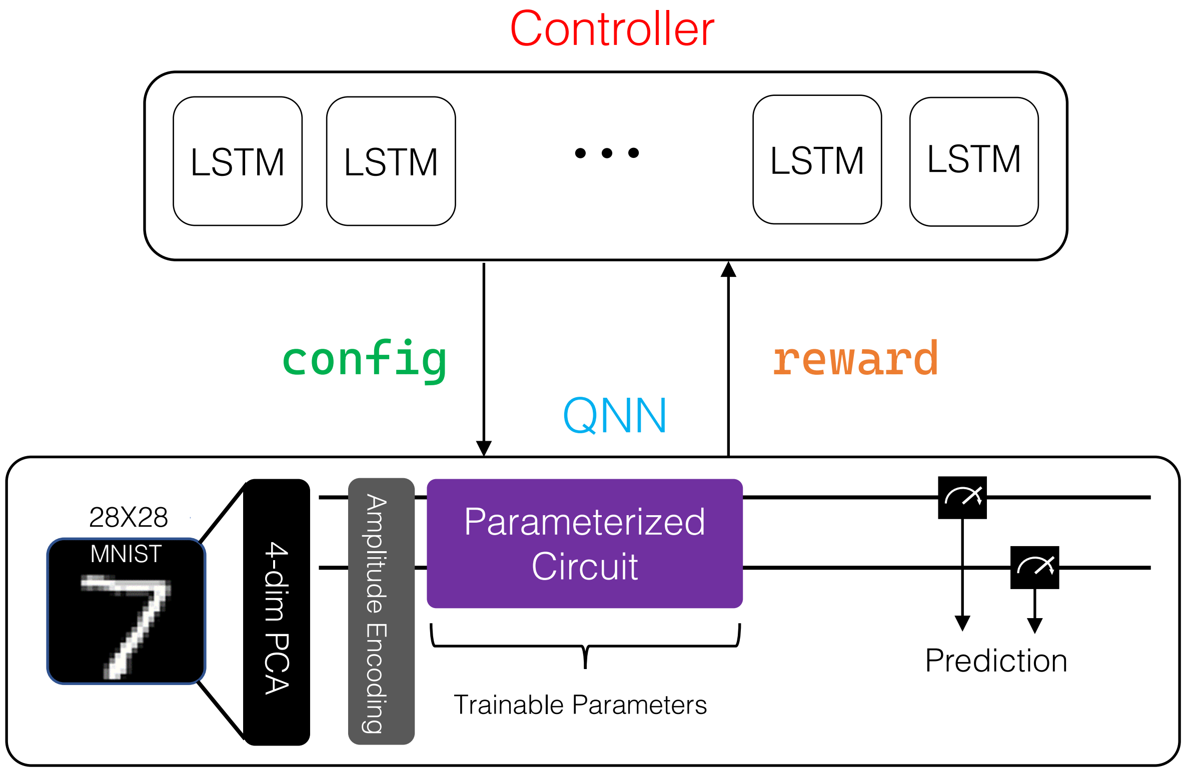
[H X Y Z X]
[H C C X C]

Total 10 selections

Results

QNN Configuration & Accuracy (With NAS)

Evolution of quantum neural network architecture



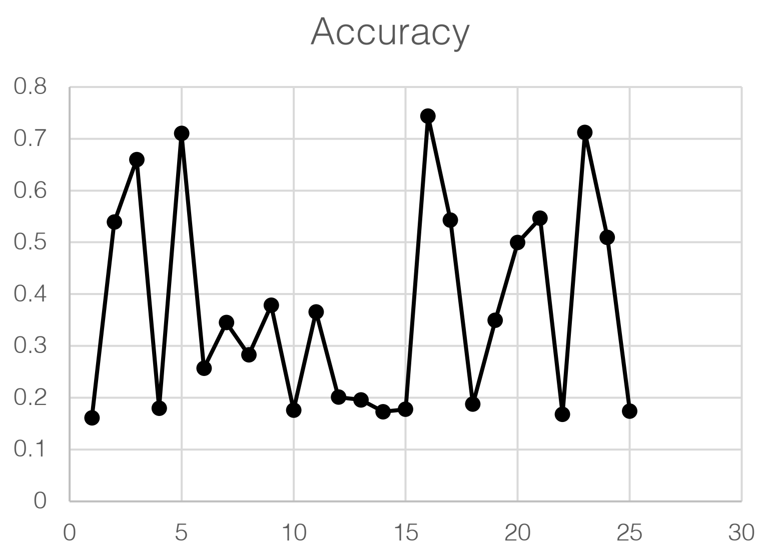

```
Agent = Controller()

config = []
for 10 iterations:
    input = Agent.lstm(input, hidden state)
    output = Agent.linear(input)
    config.append(output.argmax())

Model = QNN(config)

for 50 iterations:
    param = Model.train(param)
    valid_loss = Model.eval(param)
    if best:
        best_param = param

reward = Model.test(best_param)
controller_loss = Agent.update(reward)
```

Results

QNN Configuration & Accuracy (With NAS)

Evolution of quantum neural network architecture
But failed to converge…

Epoch	wire1	wire2	accuracy
1	YYCZY	CCYYX	0.161167
2	HYZHZ	CYYHZ	0.539167
3	HZZHX	XXXCH	0.659833
4	ZCXZY	CXZYC	0.179667
5	XCYYY	HXCHC	0.710333
6	XYCXX	XXXCZ	0.257
7	XYCYY	YYHCH	0.345167
8	XXZXX	XXYZH	0.282667
9	YXCYY	ZYYYY	0.3785
10	XXYCH	CYYXC	0.176167
11	YCCYX	XYYHZ	0.365833
12	XXXZX	YXHCY	0.201167
13	XZYHY	ZHYXC	0.195667
14	XZCZY	CYZYZ	0.172833
15	XXXCC	CXZHY	0.1775
16	ZCCZZ	HCHYZ	0.7435
17	HXCHZ	CZYYC	0.542833
18	CXHYX	YXYYC	0.187333
19	YXXXC	ZCCYY	0.3495
20	XZHYX	YYYHC	0.4995
21	XHYCX	YYCHC	0.546333
22	XXHXH	ZYCHC	0.168
23	HCXHZ	YHXXC	0.712167
24	XHYZY	YZHCY	0.509833
25	XYCXZ	CHXYX	0.174167



Challenges & Future plans

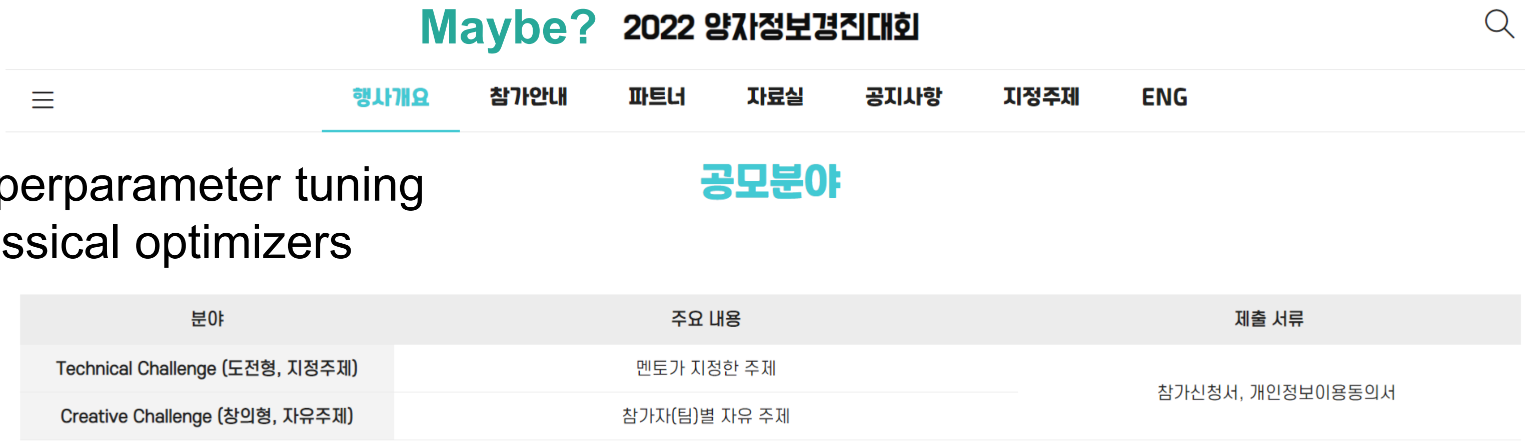
Current Challenges

1. AER simulator(about 1 hr/epoch, total 1day training for 25 epoch)
    - Not available at real NISQ device
1. Robustness problem : QNN does not converge
2. Robustness problem : LSTM policy reward does not converge
3. Searching on fixed space (5^5 dimension)
    - Autoregressive RNN : pennylane plugin does not support variable parameter inputs

Further study

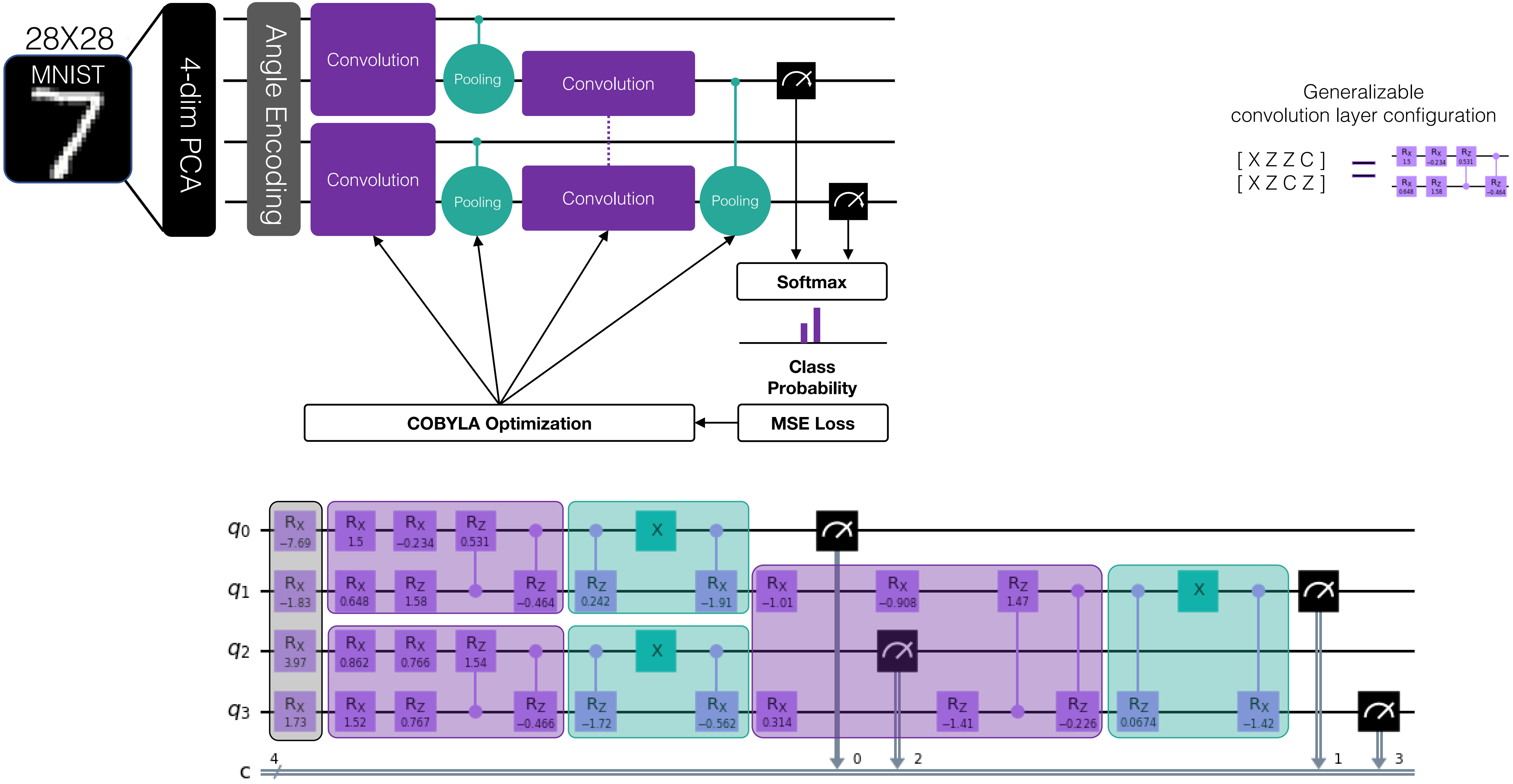
1. COBYLA optimization rho hyperparameter tuning
2. Search for another robust classical optimizers
3. QNN to QCNN

Maybe?

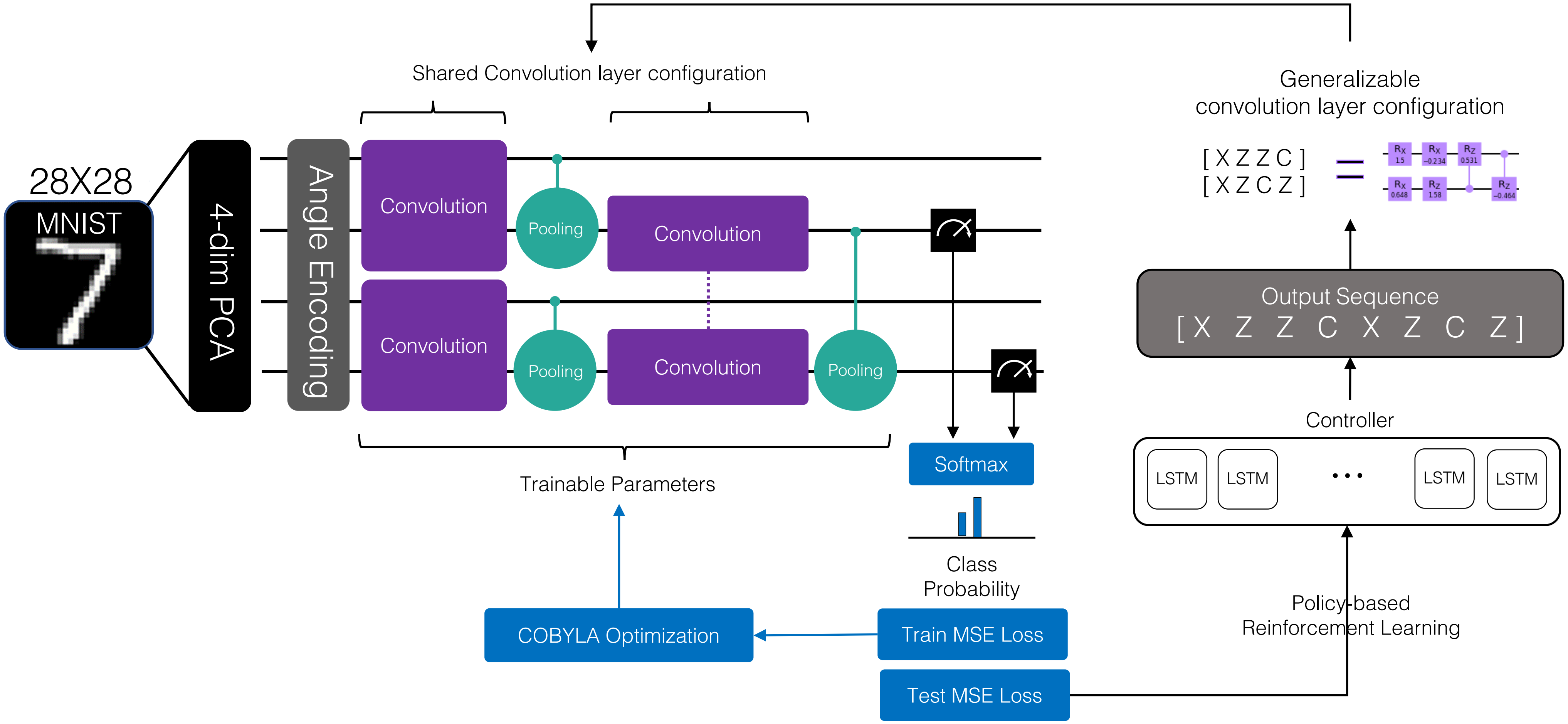


분야	주요 내용	제출 서류
Technical Challenge (도전형, 지정주제)	멘토가 지정한 주제	참가신청서, 개인정보이용동의서
Creative Challenge (창의형, 자유주제)	참가자(팀)별 자유 주제	

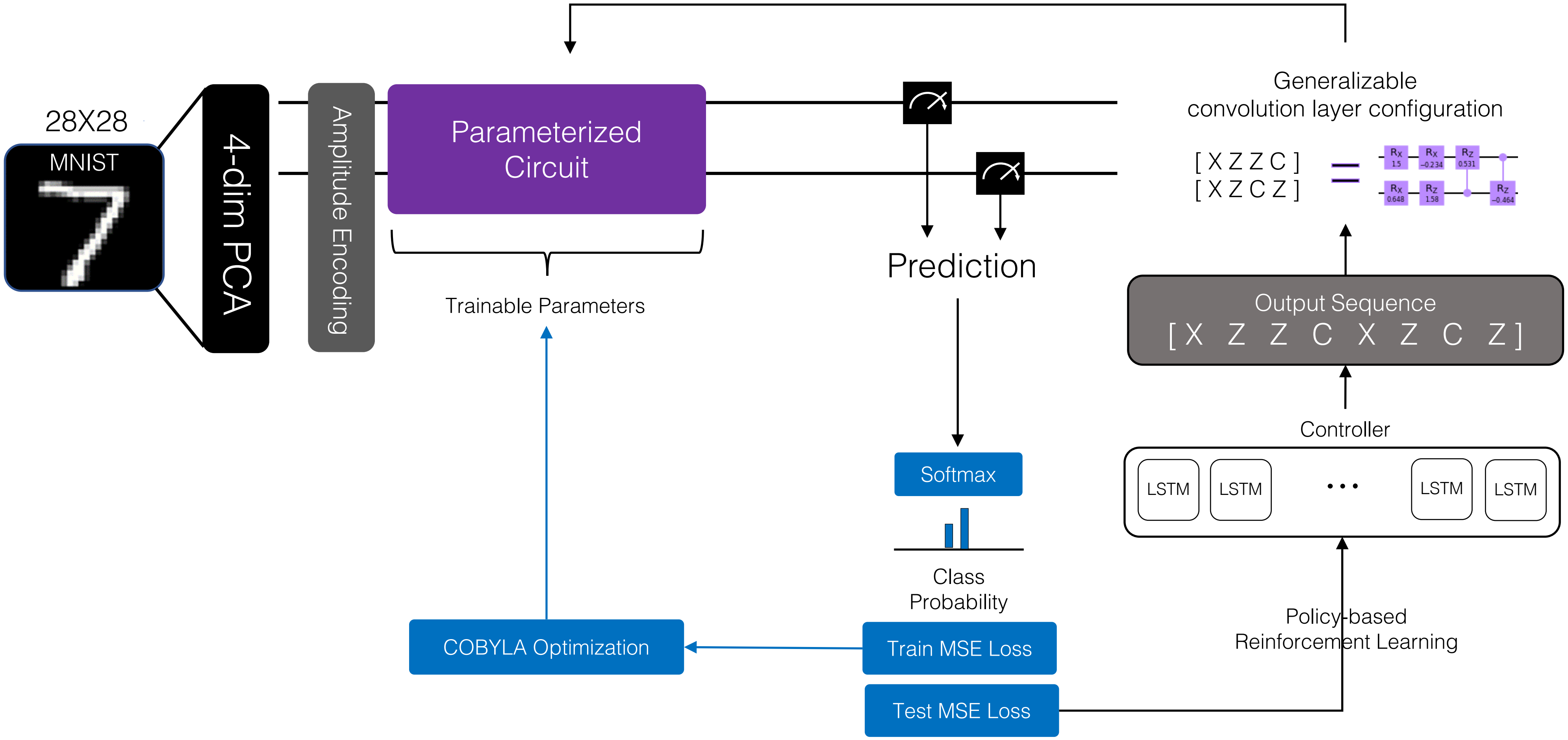
Appendix



28X28
MNIST
4-dim PCA
Angle Encoding
Convolution
Convolution
Pooling
Pooling
Convolution
Convolution
Pooling
Softmax
Class Probability
COBYLA Optimization
MSE Loss
Generalizable convolution layer configuration
[X Z Z C]
[X Z C Z]
q0
q1
q2
q3
c
4
Rx -7.69
Rx -1.83
Rx 3.97
Rx 1.73
Rx 1.5
Rx -0.234
Rz 0.531
Rx 0.648
Rz 1.58
Rz -0.464
Rx 0.862
Rx 0.766
Rz 1.54
Rx 1.52
Rz 0.767
Rz -0.466
X
Rz 0.242
Rx -1.91
X
Rz -1.72
Rx -0.562
Rx -1.01
Rx -0.908
Rz 1.47
Rx 0.314
Rz -1.41
Rz -0.226
X
Rz 0.0674
Rx -1.42
0
2
1
3



Shared Convolution layer configuration
Generalizable
convolution layer configuration
28X28
MNIST
4-dim PCA
Angle Encoding
Convolution
Convolution
Pooling
Convolution
Pooling
Convolution
Pooling
[X Z Z C]
[X Z C Z]
Output Sequence
[X Z Z C X Z C Z]
Controller
LSTM
LSTM
LSTM
LSTM
Trainable Parameters
Softmax
Class
Probability
COBYLA Optimization
Train MSE Loss
Test MSE Loss
Policy-based
Reinforcement Learning


28X28
MNIST
4-dim PCA
Amplitude Encoding
Parameterized Circuit
Trainable Parameters
Prediction
Generalizable convolution layer configuration
[X Z Z C]
[X Z C Z]
=
Output Sequence
[X Z Z C X Z C Z]
Controller
LSTM
LSTM
LSTM
LSTM
Softmax
Class Probability
Policy-based Reinforcement Learning
COBYLA Optimization
Train MSE Loss
Test MSE Loss