DESIGN AUTOMATION CONFERENCE 59

Quantum Design-Program-Compilation (QuDPC) Tutorial
TorchQuantum Library

Torch Quantum

Hanrui Wang[1], Jiaqi Gu[2], Zirui Li[3], Zhiding Liang[4], Weiwen Jiang[5], Yiyu Shi[4], David Z. Pan[2], Yongshan Ding[6], Frederic T. Chong[7], Song Han[1]

[1]MIT [2]UT Austin [3]SJTU [4]University of Notre Dame [5]George Mason University [6]Yale University [7]University of Chicago

HAN LAB Hardware, AI and Neural-nets TEXAS The University of Texas at Austin Yale GEORGE MASON UNIVERSITY

Open-source: TorchQuantum

- TorchQuantum — An open-source library for differentiable quantum simulation and quantum machine learning
- https://github.com/mit-han-lab/torchquantum

Torch Quantum

Open-source: TorchQuantum

- Features
  - Easy construction and simulation of quantum circuits in **PyTorch**
  - **Dynamic** computation graph for easy debugging
  - **Gradient** support via autograd
  - **Batch** mode inference and training on CPU/GPU.
  - Easy **deployment** on real quantum devices such as IBMQ
  - Easy **hybrid** classical-quantum model construction
  - (coming soon) **pulse-level** simulation
  - Tutorials, videos and example projects of QML and using ML to optimize quantum computer system problems

Open-source: TorchQuantum

- Statevector

```
_state = torch.zeros(2 ** self.n_wires, dtype=C_DTYPE)
_state[0] = 1 + 0j
```

- Quantum Gates

```
'cnot': torch.tensor([[1, 0, 0, 0],
                      [0, 1, 0, 0],
                      [0, 0, 0, 1],
                      [0, 0, 1, 0]], dtype=C_DTYPE),
```

Open-source: TorchQuantum

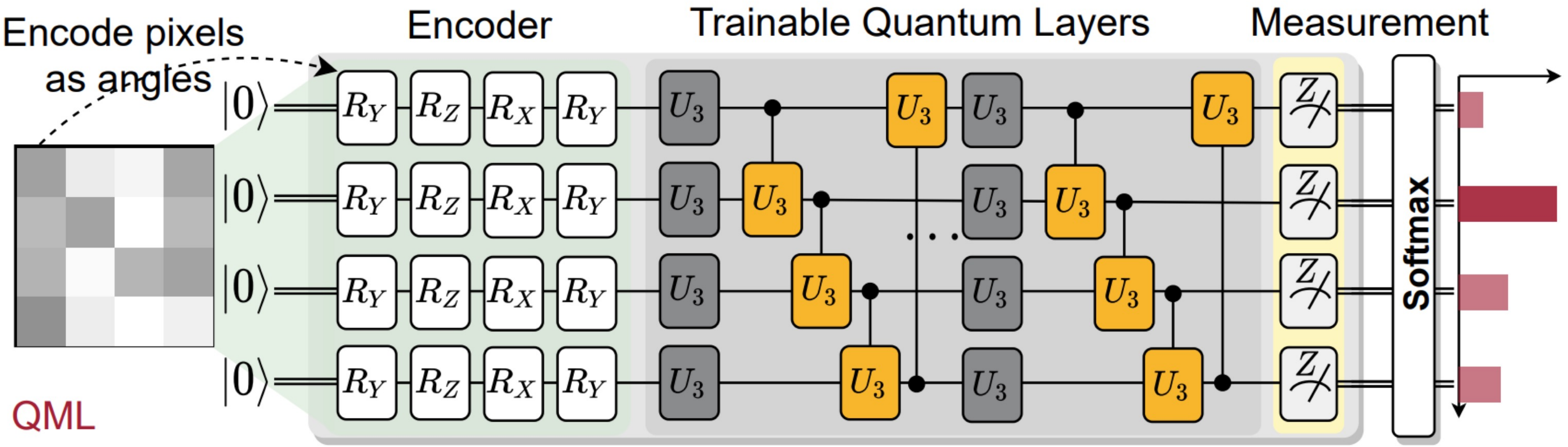
- Quantum Gates

```
def crx_matrix(params):
    theta = params.type(C_DTYPE)
    co = torch.cos(theta / 2)
    jsi = 1j * torch.sin(-theta / 2)

    matrix = torch.tensor([[1, 0, 0, 0],
                           [0, 1, 0, 0],
                           [0, 0, 0, 0],
                           [0, 0, 0, 0]], dtype=C_DTYPE, device=params.device
                          ).unsqueeze(0).repeat(co.shape[0], 1, 1)
    matrix[:, 2, 2] = co[:, 0]
    matrix[:, 2, 3] = jsi[:, 0]
    matrix[:, 3, 2] = jsi[:, 0]
    matrix[:, 3, 3] = co[:, 0]

    return matrix.squeeze(0)
```

- Matrix-vector multiplication: torch.einsum and torch.bmm

Open-source: TorchQuantum

- Example of QNN



Open-source: TorchQuantum

Initialize a quantum device

Specify encoder gates

Specify trainable gates

```
import torch.nn as nn
import torch.nn.functional as F
import torchquantum as tq
import torchquantum.functional as tqf

class QFCModel(nn.Module):
  def __init__(self):
    super().__init__()
    self.n_wires = 4
    self.q_device = tq.QuantumDevice(n_wires=self.n_wires)
    self.measure = tq.MeasureAll(tq.PauliZ)

    self.encoder_gates = [tqf.rx] * 4 + [tqf.ry] * 4 + \
                         [tqf.rz] * 4 + [tqf.rx] * 4
    self.rx0 = tq.RX(has_params=True, trainable=True)
    self.ry0 = tq.RY(has_params=True, trainable=True)
    self.rz0 = tq.RZ(has_params=True, trainable=True)
    self.crx0 = tq.CRX(has_params=True, trainable=True)
```

Reset statevector

Encode classical pixels

Apply the trainable gates

Apply some non-trainable gates

Measure to get classical values

```python
def forward(self, x):
  bsz = x.shape[0]
  # down-sample the image
  x = F.avg_pool2d(x, 6).view(bsz, 16)

  # reset qubit states
  self.q_device.reset_states(bsz)

  # encode the classical image to quantum domain
  for k, gate in enumerate(self.encoder_gates):
    gate(self.q_device, wires=k % self.n_wires, params=x[:, k])

  # add some trainable gates (need to instantiate ahead of time)
  self.rx0(self.q_device, wires=0)
  self.ry0(self.q_device, wires=1)
  self.rz0(self.q_device, wires=3)
  self.crx0(self.q_device, wires=[0, 2])

  # add some more non-parameterized gates (add on-the-fly)
  tqf.hadamard(self.q_device, wires=3)
  tqf.sx(self.q_device, wires=2)
  tqf.cnot(self.q_device, wires=[3, 0])
  tqf.qubitunitary(self.q_device0, wires=[1, 2], params=[[1, 0, 0, 0],
                                                          [0, 1, 0, 0],
                                                          [0, 0, 0, 1j],
                                                          [0, 0, -1j, 0]])

  # perform measurement to get expectations (back to classical domain)
  x = self.measure(self.q_device).reshape(bsz, 2, 2)

  # classification
  x = x.sum(-1).squeeze()
  x = F.log_softmax(x, dim=1)

  return x
```

Open-source: TorchQuantum

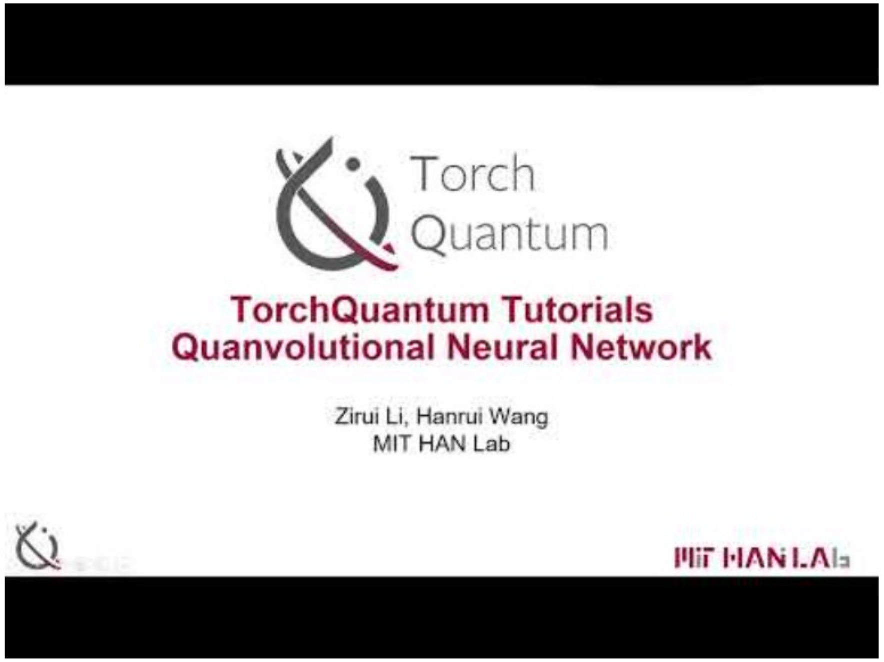
- Tutorial Colab and videos

Quantum Convolution (Quanvolution) for MNIST image classification

Authors: Zirui Li, Hanrui Wang

Use Colab to run this example: Open in Colab

See this tutorial video for detailed explanations:



Referece: Quanvolutional Neural Networks: Powering Image Recognition with Quantum Circuits

Speedup over existing frameworks

- **2 orders of magnitude** faster than other frameworks
  - Tensorized computation
  - Well optimized codes on CPU/GPU

WonJoon_YUN

Thanks! Mr. Wang!
Actually, our research team has studied quantum multi-agent reinforcement learning with leveraging torchquantum, which reduces 99% computational cost (6 days in other libraries but in <40 minutes in torchquantum). It enables feasibility studies on many existing QML research. Thank you and the torchquantum developers!!

Current status

- Users from around 20 research institutes/companies
- Over 250 stars on GitHub
- We will hold a full tutorial session on TorchQuantum in Quantum Week (QCE) this year

Torch Quantum

https://github.com/mit-han-lab/torchquantum

qmlsys.mit.edu

Quantum Computer Systems Lecture Series

A lecture series on quantum computer systems, software and cross-stack optimizations.
Talks will cover quantum computing basics and state-of-the-art research topics.

Speakers

Prof. Yongshan Ding, *Yale*
Prof. Yufei Ding, *UCSB*
Prof. Weiwen Jiang, *GMU*
Prof. Gushu Li, *UPenn*
Prof. Tongyang Li, *PKU*
Prof. Prineha Narang, *Harvard*
Prof. Tirthak Patel, *Rice*
Prof. Chen Qian, *UCSC*
Prof. Sabre Kais, *Purdue*
Prof. Robert Wille, *TUM*
Dr. Junyu Liu, *UChicago*
Dr. Gokul Ravi, *UChicago*
Jinglei Cheng, *USC*
Siyuan Niu, *Montpellier*
Bochen Tan, *UCLA*
Wei Tang, *Princeton*
Hanrui Wang, *MIT*
Zhixin Song, *Gatech*
•••

Topics

Quantum Basics
NISQ Algorithms
Quantum Compilation
Noise Mitigation
Quantum ML
Architecture Design
Program Debugging
Error Correction
Classical Simulation
Measurement
•••

Thursdays
10:30AM ET

Website
Mailing-list
TorchQuantum library

To attend, visit sites.nd.edu/quantum
For TorchQuantum library, visit qmlsys.mit.edu/ or scan QC code
Organized by Zhiding Liang (Notre Dame), Hanrui Wang (MIT)