Machine Learning with full PyTorch & Qiskit capabilities

Team QizGloria

Patrick Huembeli, Amira Abbas, Samuel Bosch, Isaac Turteltaub, Karel Dumon

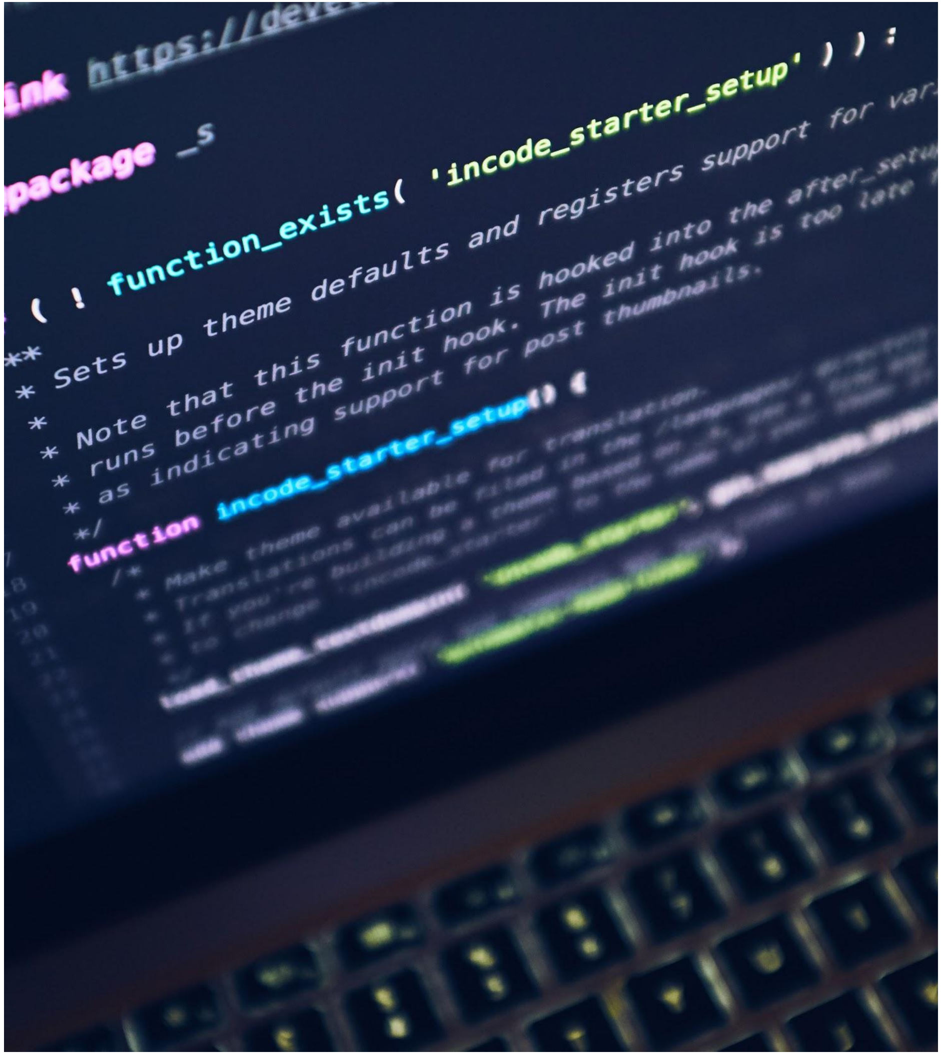
IBM Coach: Christa Zoufal

Our mission

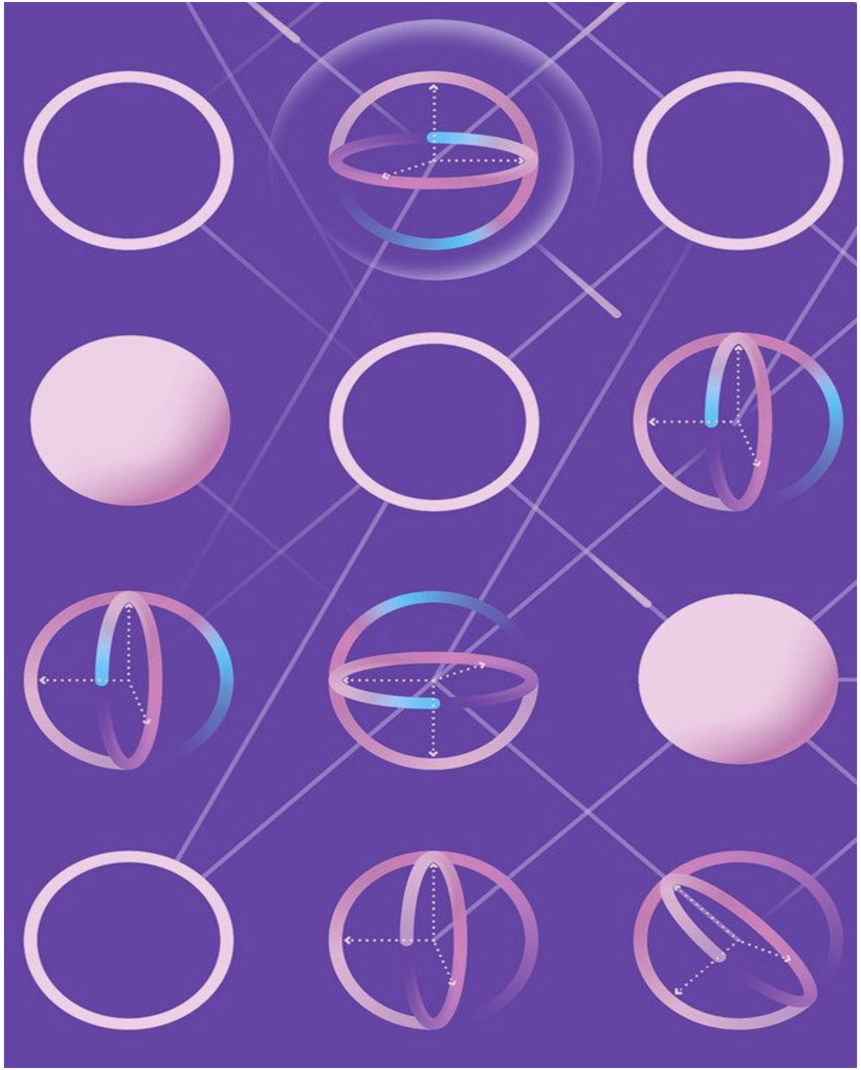
- Closer integration of Pytorch & Qiskit beyond existing tools

- Enable seamless co-training of quantum circuits & neural networks

- Encourage classical ML engineers to use quantum nodes



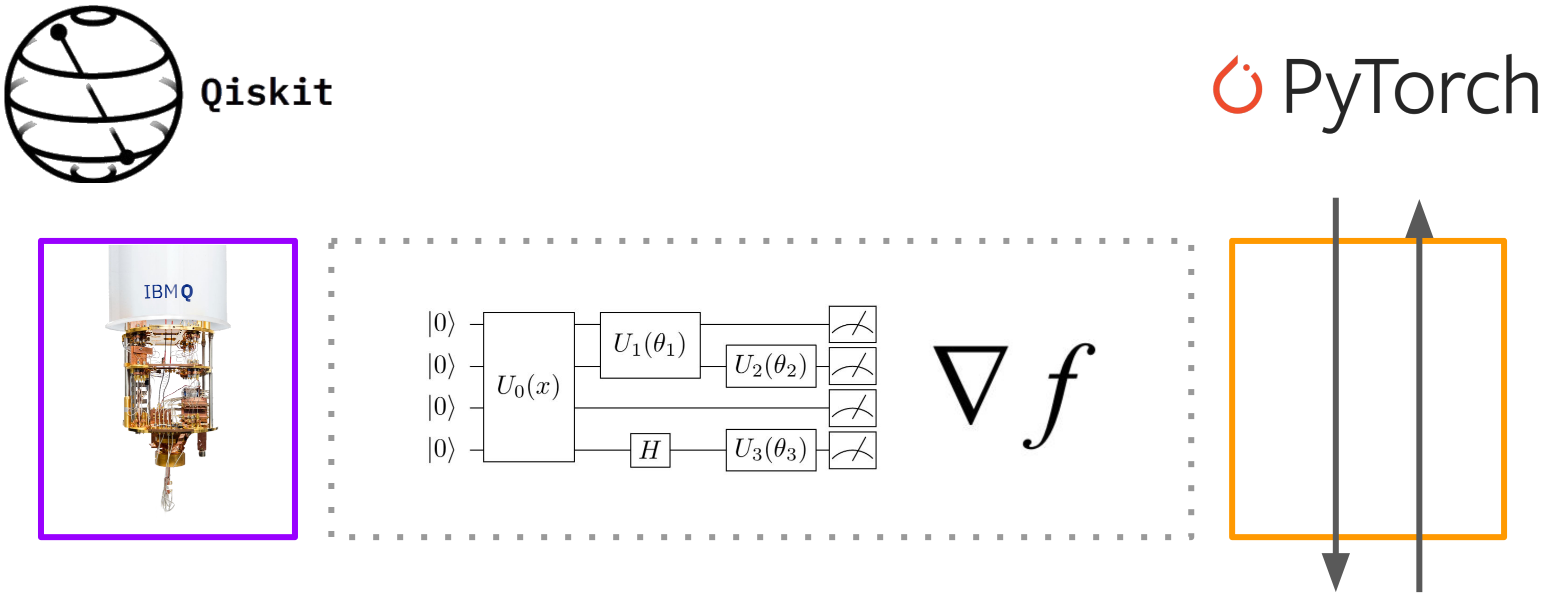
```
ink https://deve
package _s
( ! function_exists( 'incode_starter_setup' ) ) :
**
 * Sets up theme defaults and registers support for var
 *
 * Note that this function is hooked into the after_setu
 * runs before the init hook. The init hook is too late
 * as indicating support for post thumbnails.
 */
function incode_starter_setup() {
	/*
	 * Make theme available for translation.
	 * Translations can be filed in the /languages/
	 * If you're building a theme based on _s,
	 * to change 'incode_starter' to the name of
```



Why is this cool?

- PyTorch neural networks & tools
- More options for optimizers (RMSprop; Momentum; etc.)
- Full Qiskit capabilities (circuit definition, transpiler, Aqua,...)
- Back-end management by Qiskit (QPU, simulators)
- Can now parallelize your optimization code
- Bridges the gap between the QML and classical ML community

How does it work?

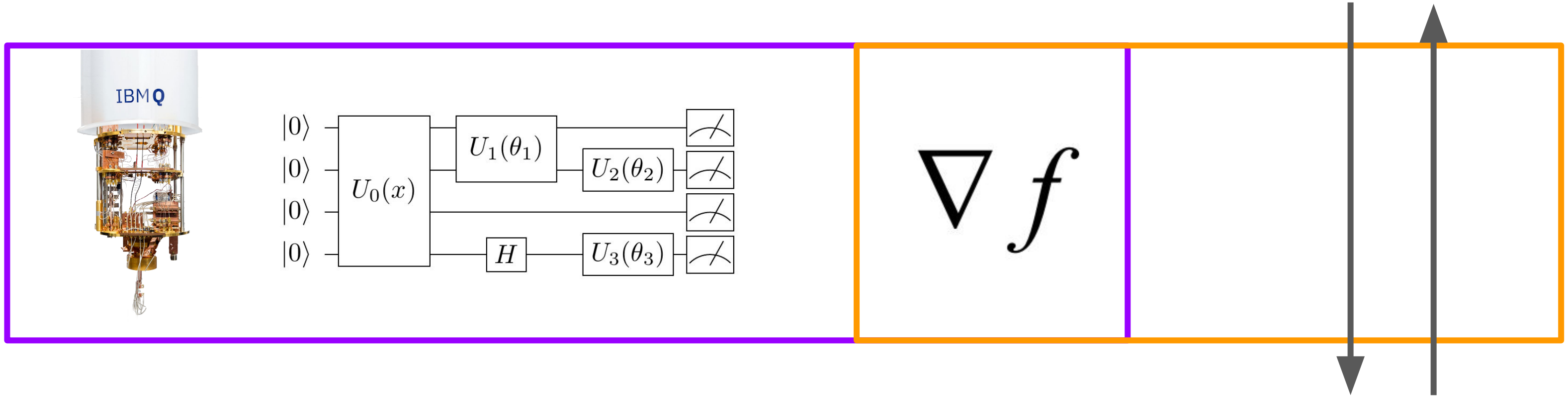


other frameworks:
blocked from Qiskit-tools!

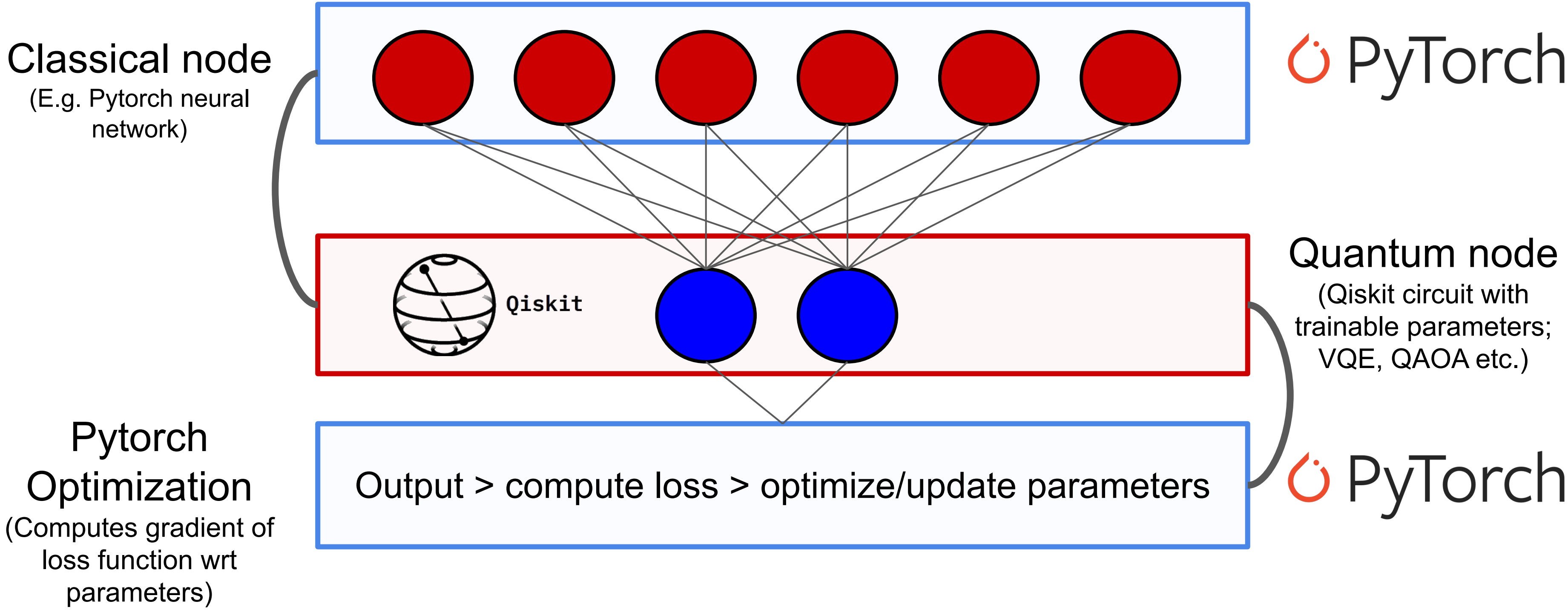
How does it work?

Qiskit

PyTorch



How does it work?



Classical node

(E.g. Pytorch neural network)

PyTorch

Qiskit

Quantum node

(Qiskit circuit with trainable parameters; VQE, QAOA etc.)

Pytorch Optimization

(Computes gradient of loss function wrt parameters)

Output > compute loss > optimize/update parameters

PyTorch

How does it work?

Qiskit

PyTorch

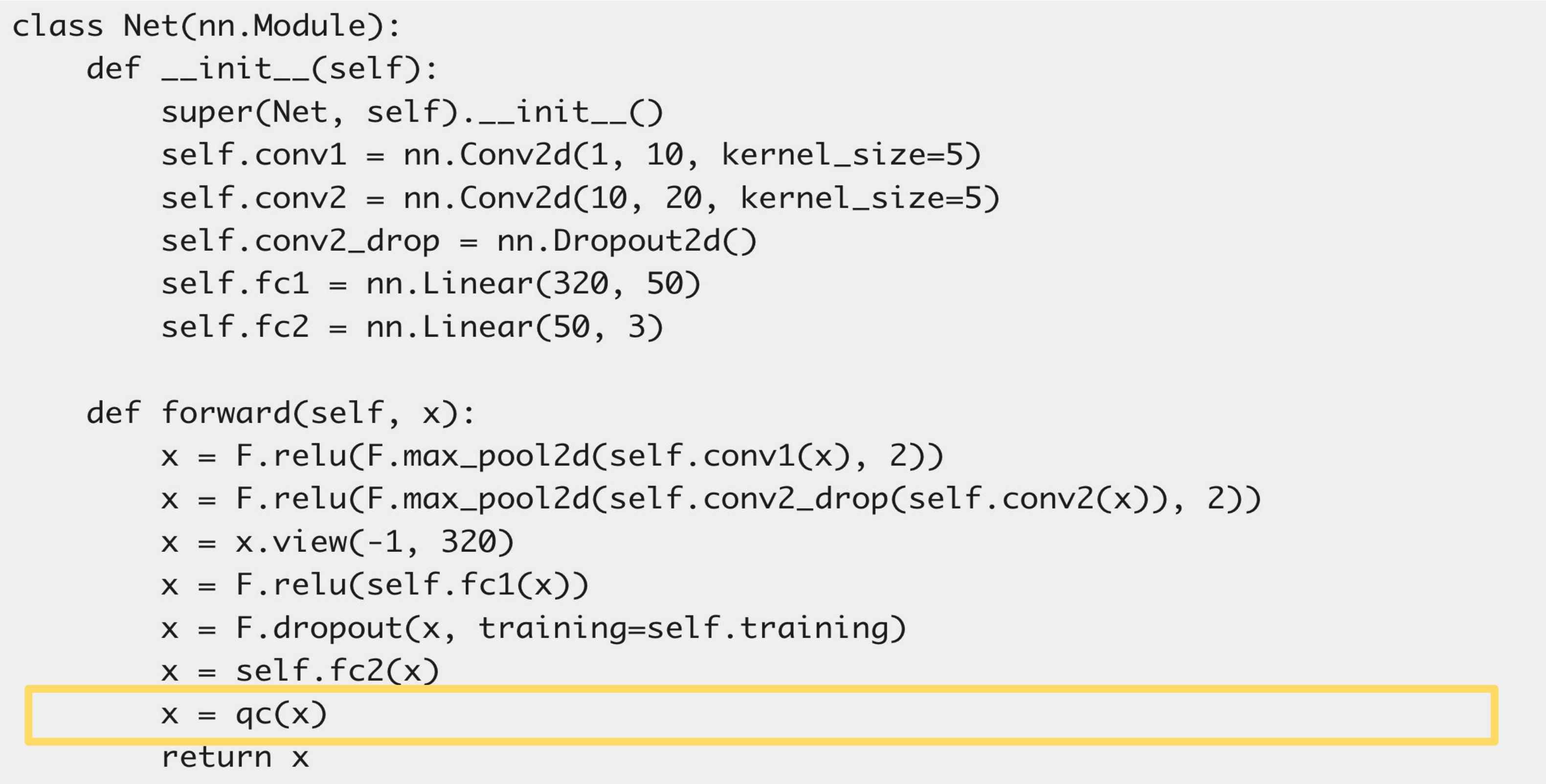
QiskitCircuit

circuit definition (Terra, Aqua)
parameter binding
expectation value evaluation
back-end management

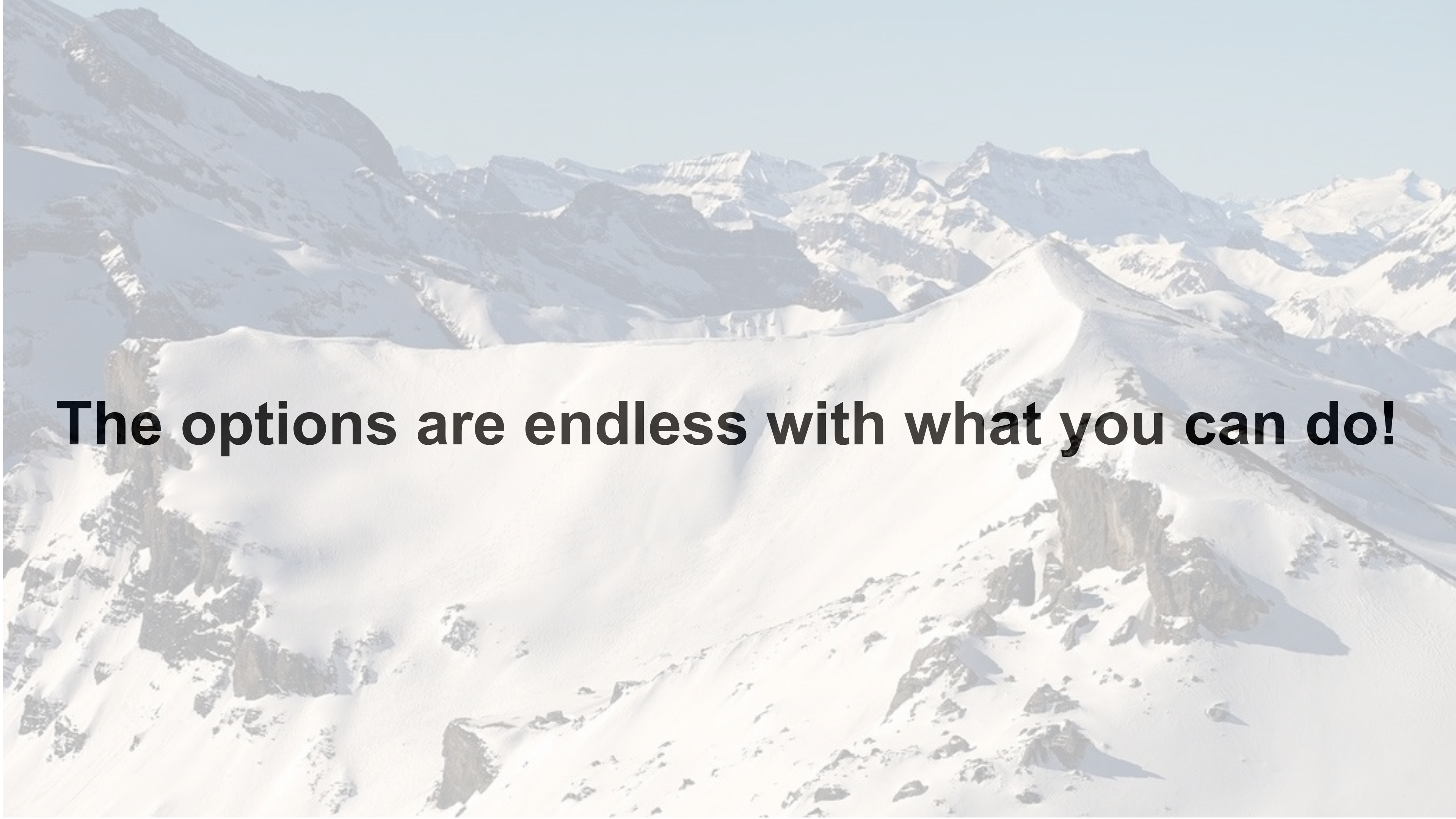
TorchCircuit

tensorization
parallelization
forward pass
backward pass (finite diff, aqgd)

Seamless integration of Pytorch and Qiskit



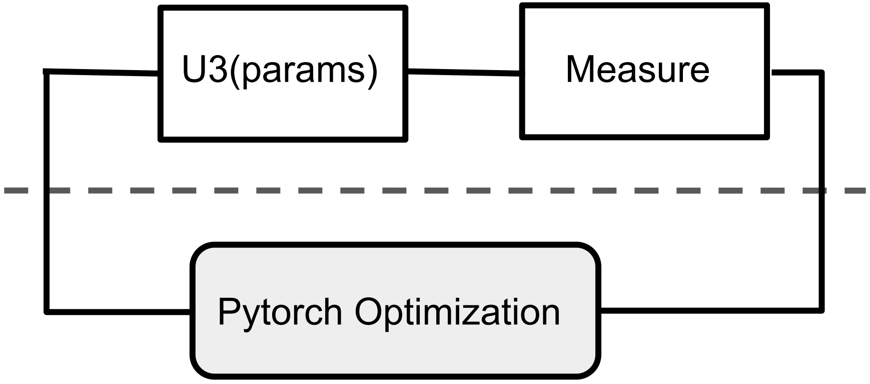
```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 10, kernel_size=5)
        self.conv2 = nn.Conv2d(10, 20, kernel_size=5)
        self.conv2_drop = nn.Dropout2d()
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 3)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
        x = x.view(-1, 320)
        x = F.relu(self.fc1(x))
        x = F.dropout(x, training=self.training)
        x = self.fc2(x)
        x = qc(x)
        return x
```



The options are endless with what you can do!

Hello World!

- Learn how to rotate 1 qubit to get a defined σ_z expectation
- Used U3 rotation in qiskit

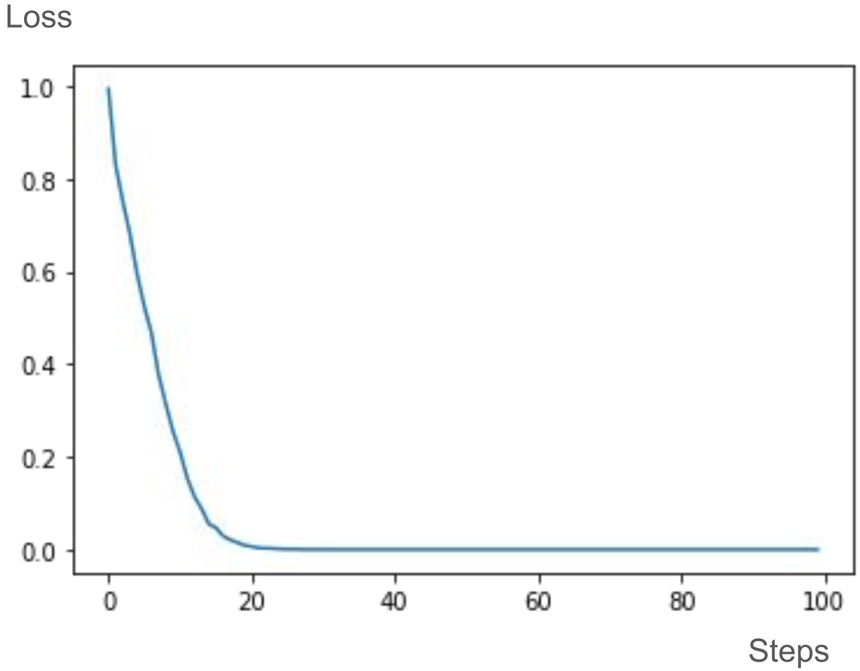


U3(params)

Measure

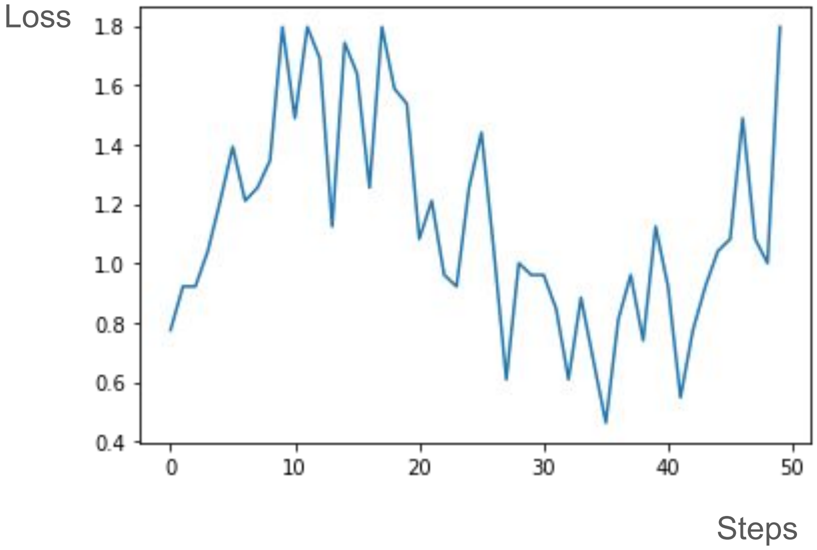
Pytorch Optimization

Update params

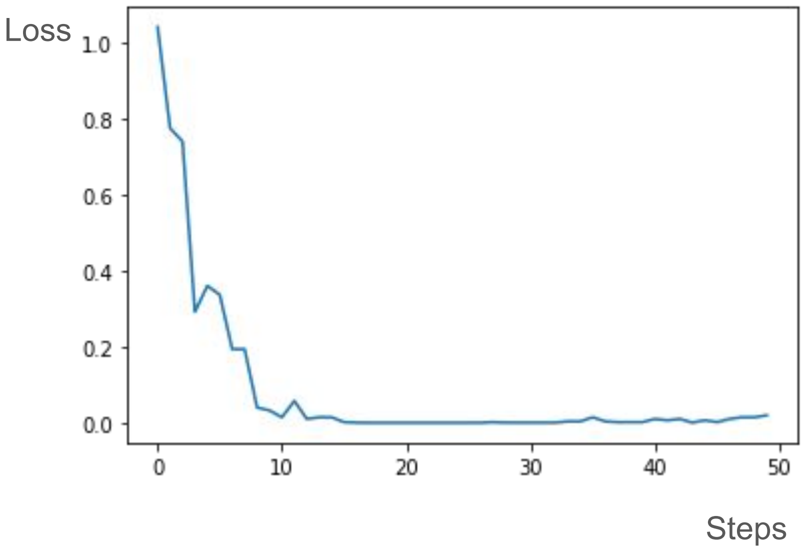


Details:
Finite difference gradient estimation; shots = 10 000

Hello World! - analytical gradients

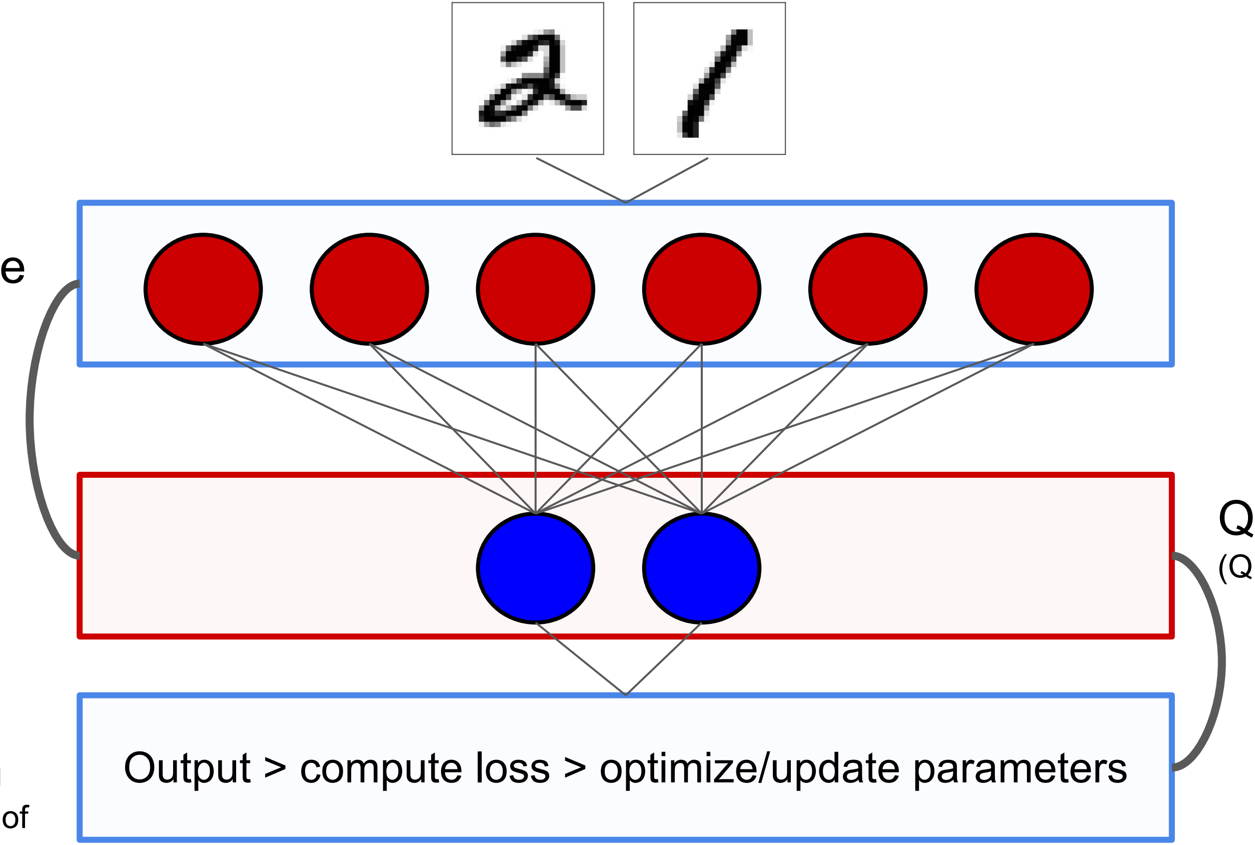


Details:
Finite difference gradient estimation; shots = 100



Details:
Analytical gradient; shots = 100

MNIST



Classical node
ConvNet

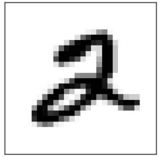
Quantum node
(Qiskit circuit: Rx & Ry rotations)

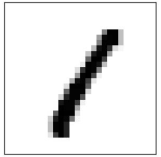
Output > compute loss > optimize/update parameters

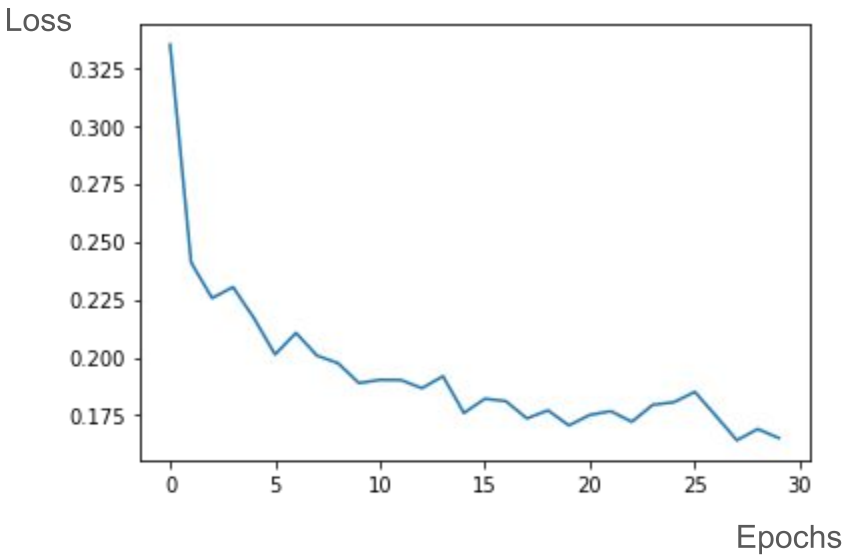
Pytorch Optimization
(Computes gradient of loss function wrt two parameters)

Sigma z exp > NLL/cross entropy loss > Pytorch Adam optimizer

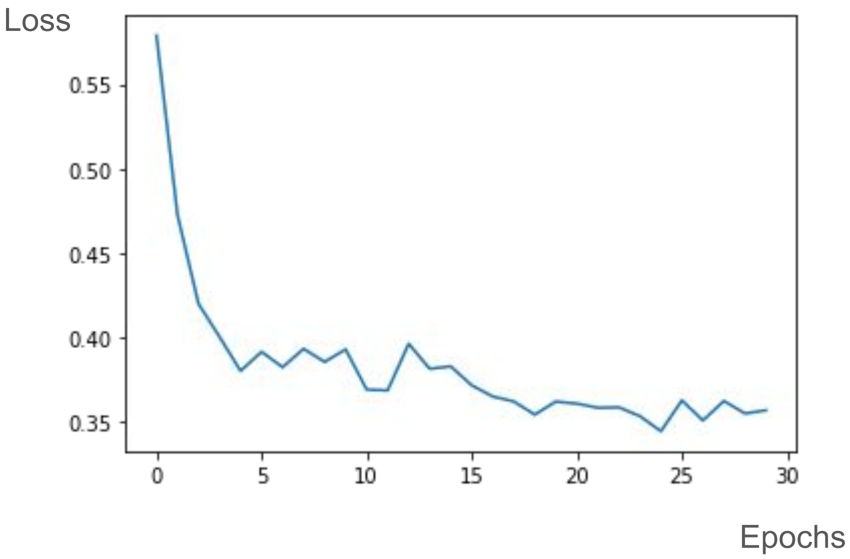
MNIST







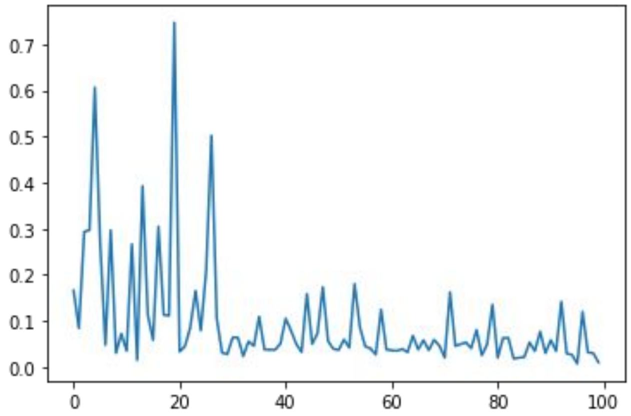
Details:
Analytical gradients
Negative-log-likelihood-loss
Shots = 100
(200 data samples per epoch)

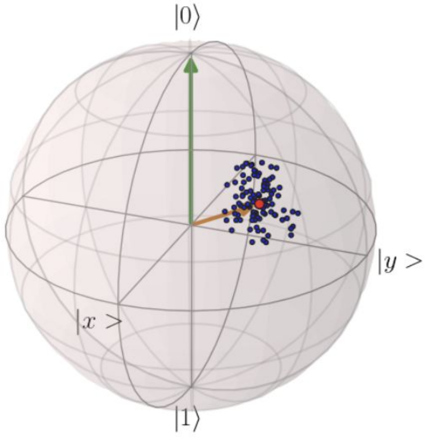


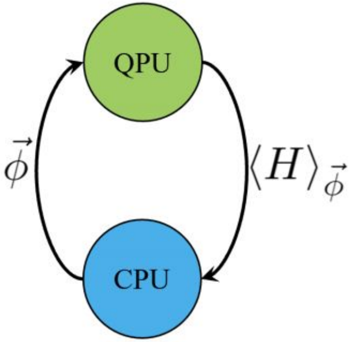
Details:
Finite difference gradient estimation
Cross-entropy loss for MNIST
Shots = 10 000

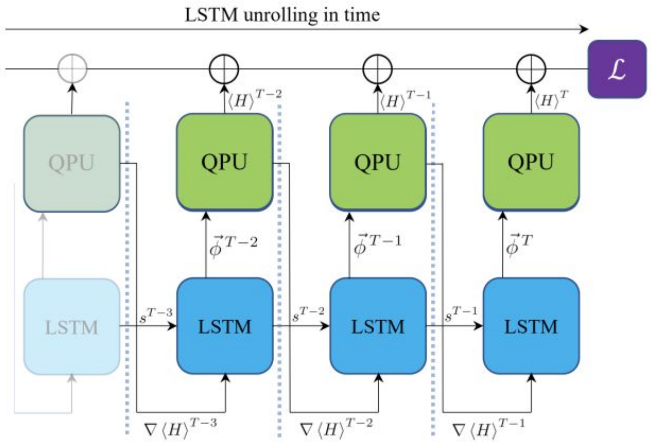
Other implementations we did

Meta-learning for neural optimizer for single qubit rotation



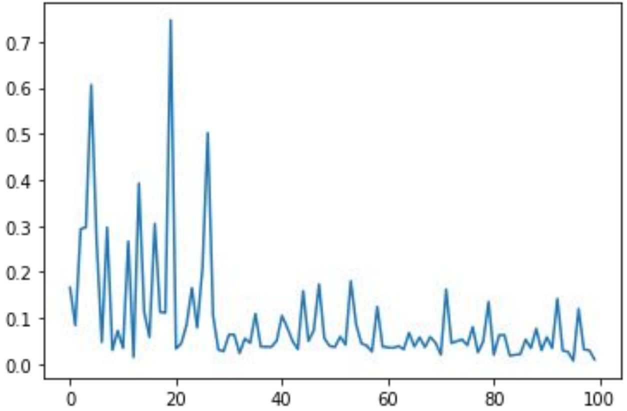




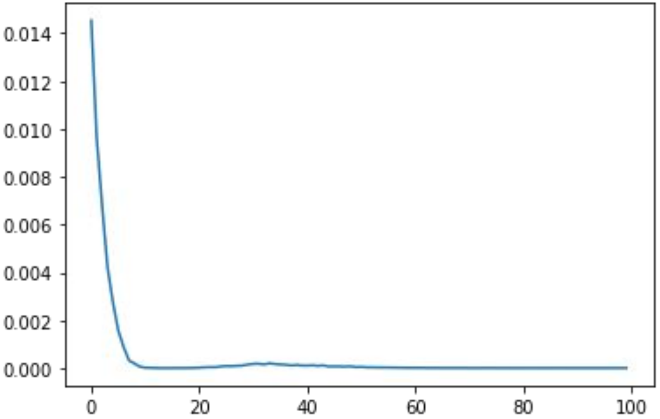


Wilson, Max, et al. "Optimizing quantum heuristics with meta-learning." *arXiv preprint arXiv:1908.03185* (2019).

Other implementations we did

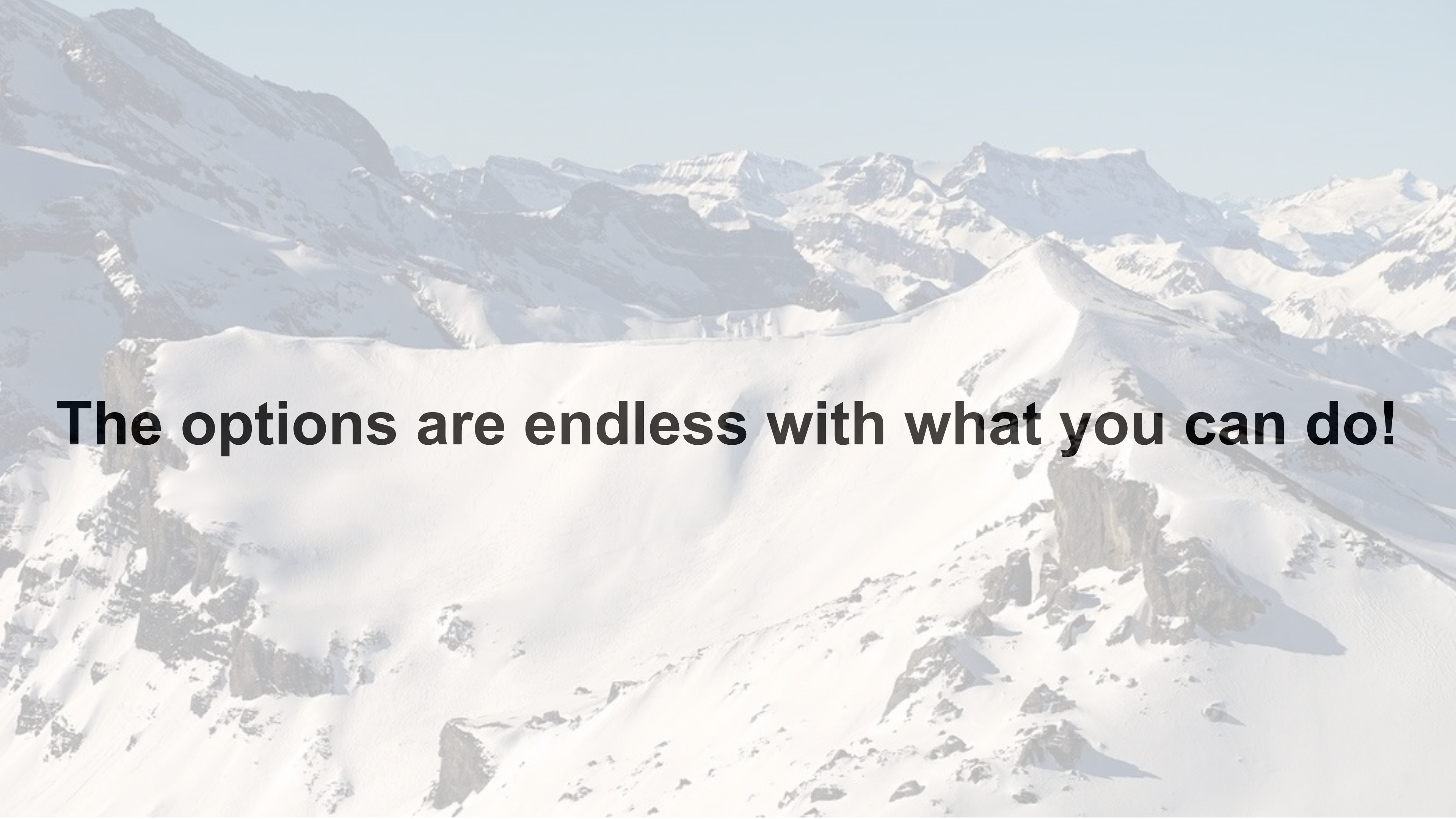


Meta-learning for neural optimizer for single qubit rotation



2 qubit example: QAOA with Pytorch optimizers

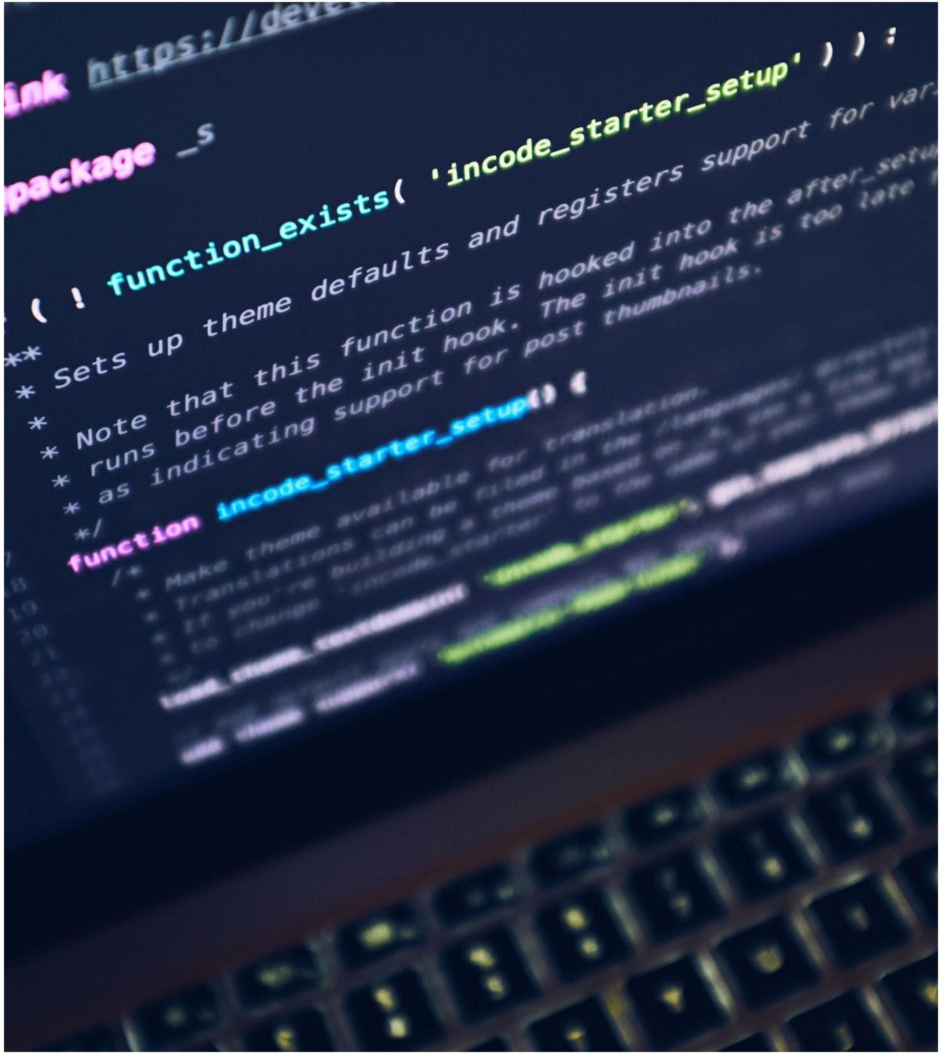
Wilson, Max, et al. "Optimizing quantum heuristics with meta-learning." *arXiv preprint arXiv:1908.03185* (2019).



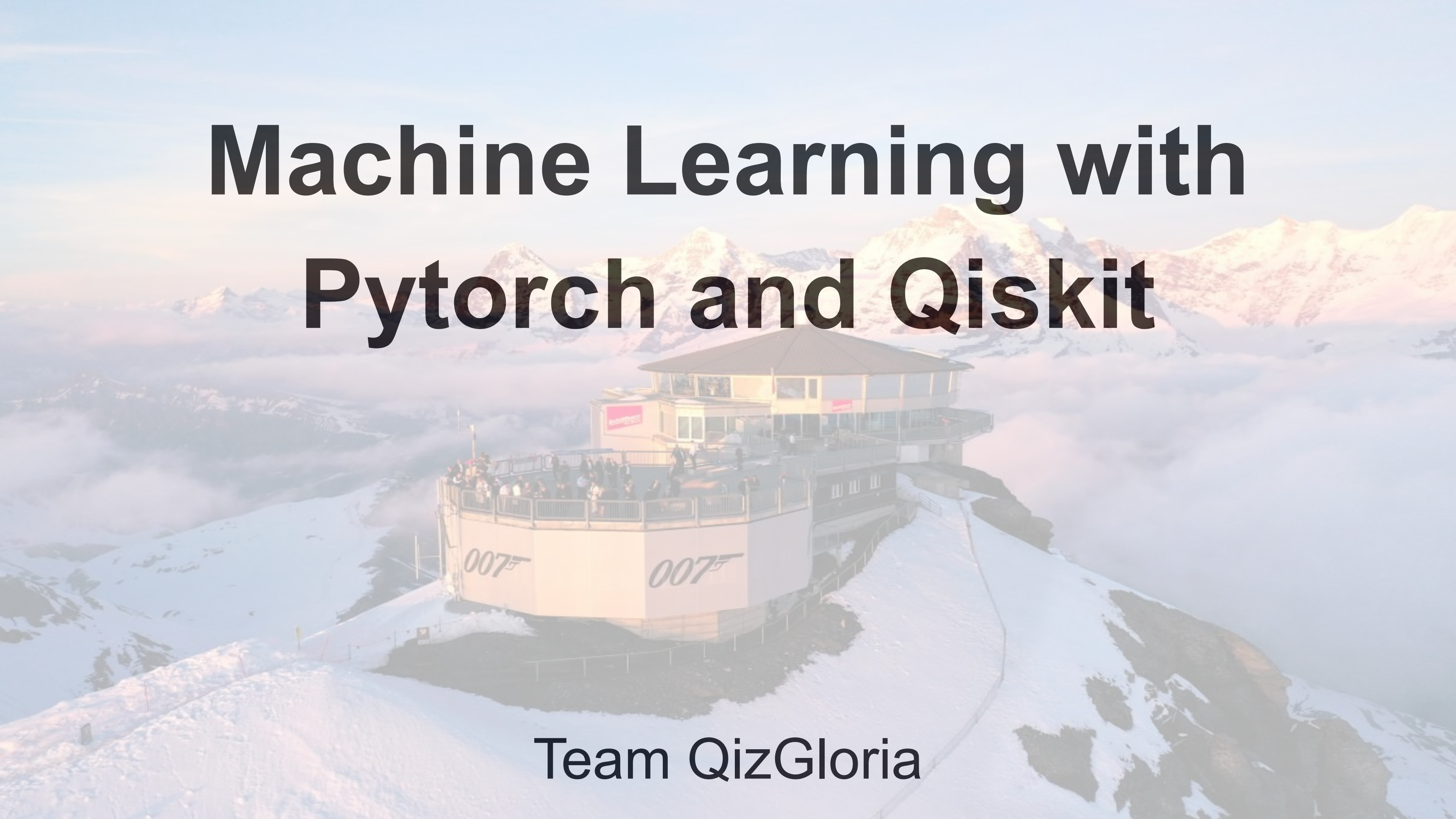
The options are endless with what you can do!

Our mission

- Closer integration of Pytorch & Qiskit beyond existing tools
- Enable seamless co-training of quantum circuits & neural networks
- Encourage classical ML engineers to use quantum nodes



```
ink https://deve
package _s
( ! function_exists( 'incode_starter_setup' ) ) :
**
 * Sets up theme defaults and registers support for var
 *
 * Note that this function is hooked into the after_setu
 * runs before the init hook. The init hook is too late
 * as indicating support for post thumbnails.
 */
function incode_starter_setup() {
	/*
	 * Make theme available for translation.
	 * Translations can be filed in the /languages/
```



Machine Learning with Pytorch and Qiskit

Team QizGloria