ICSC
Centro Nazionale di Ricerca in HPC,
Big Data and Quantum Computing

QML:
QUICK INTRODUCTION

Laura Cappelli & Stefano Giagu

INFN

Università degli Studi di Ferrara

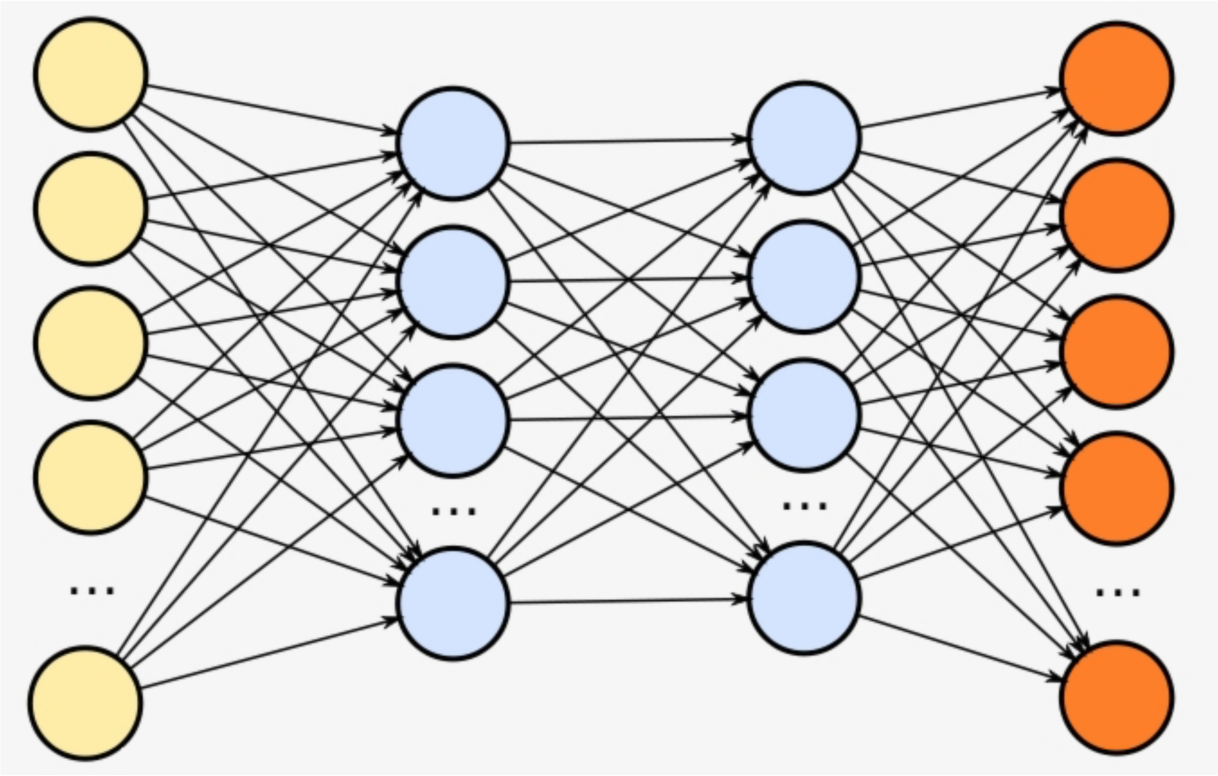
SAPIENZA UNIVERSITÀ DI ROMA

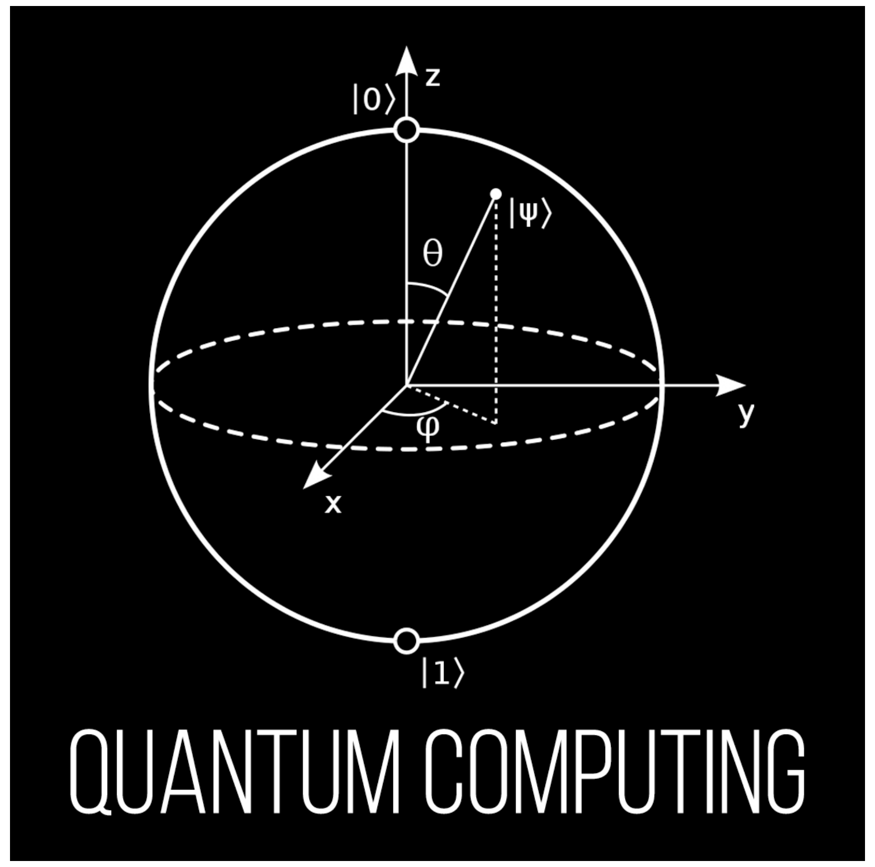
1st AI-INFN Advanced Hackathon - Padova November 26-28, 2024

INTRODUCTION

QML = ML + QC

- Quantum Machine Learning is an emerging design paradigm to program gate-based quantum computers akin to classical ML





- motivated by the rapid development of increasingly performing quantum computers, which made interesting to understand whether QML can actually offer significant improvements compared to classical algorithms in particular using current noisy devices (NISQ)

- aim of this lecture is to give a quick introduction to the most important elements of QC and QML, following an intuitive more than a formal approach

MINIMAL INTRODUCTORY REFERENCES

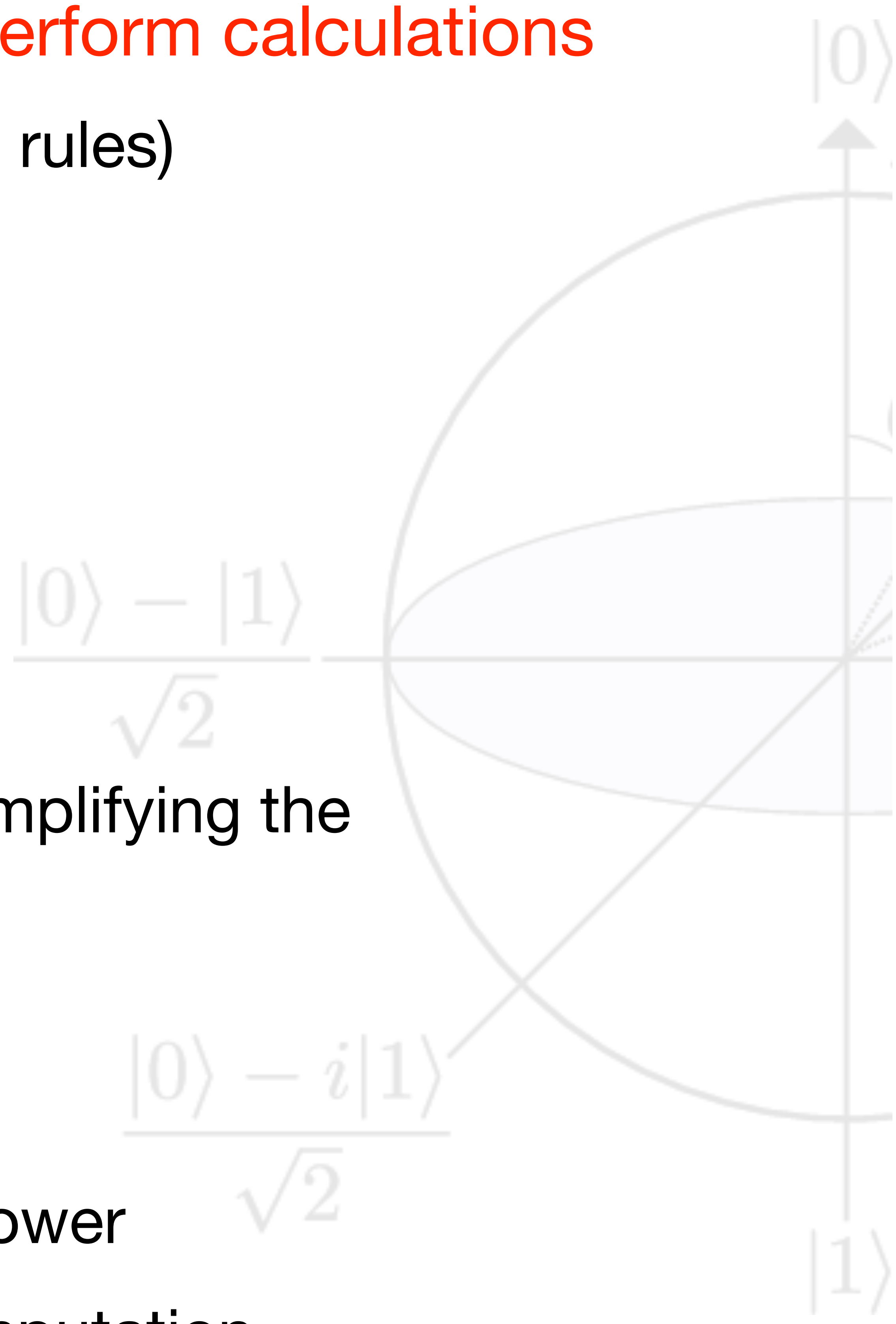
- M.A. Nielsen, I.L.Chuang, "Quantum Computation and Quantum Information", 10th annyv. edition, Cambridge (2010)
- M. Schuld, F. Petruccione, "Machine Learning with Quantum Computers" (2nd edition), Springer (2021)
- O. Simeone (2022), "An Introduction to Quantum Machine Learning for Engineers", arXiv:2205.09510 [quant-ph]
- Qiskit QC platform: https://www.ibm.com/quantum/qiskit
- Pennylane QC platform: https://pennylane.ai/

QUANTUM COMPUTING

- Computing paradigm that explicitly leverage quantum mechanical properties of matters to perform calculations
  - in contraposition to CC where QM enters only indirectly (eg. semiconductors in CPUs follow QM rules)
  - QC are not “faster” computers wrt CC, but systems that do computation in different ways
  - QC are not general purpose machines that speed up any problem wrt CC

- Takes advantage of:
  - **Superposition and Entanglement** → exponential representation power
  - **Constructive/destructive interference** → guide the computation toward the correct solution amplifying the probabilities of correct answers and reducing the probabilities of incorrect ones

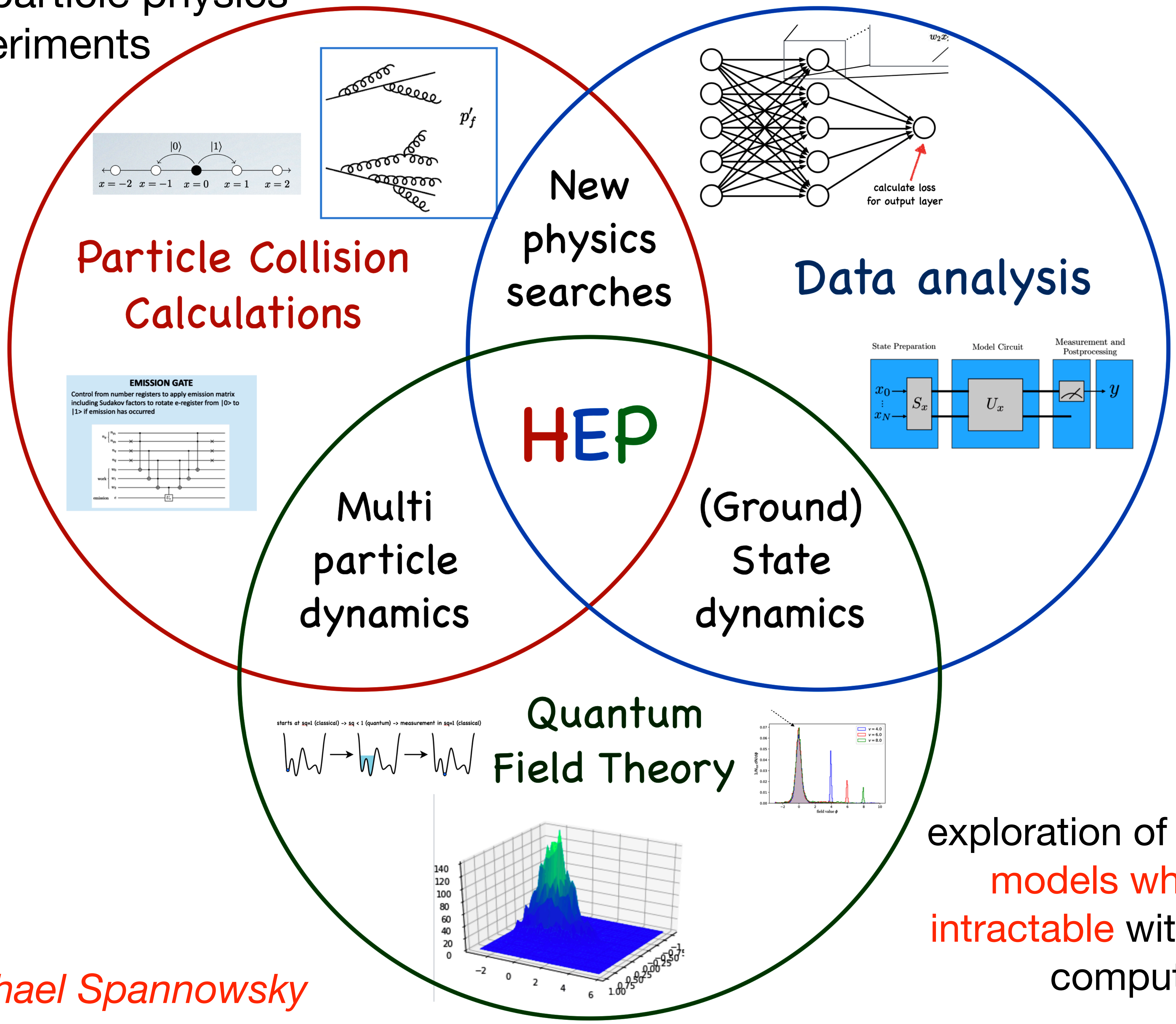
- … and is affected by others useful/problematic features of QM:
  - Quantum operations (gates) as **unitary transformations** → reversible computing / expressive power
  - Output is the result of a **quantum state measurement** according to Born rule → stochastic computation
  - **No-cloning** theorem → information security / complex & resource hungry error-correction
  - **Quantum state coherence and isolation** → computation stability and errors



WHY QUANTUM COMPUTING?

major application areas in HEP that may benefit from QC/QML

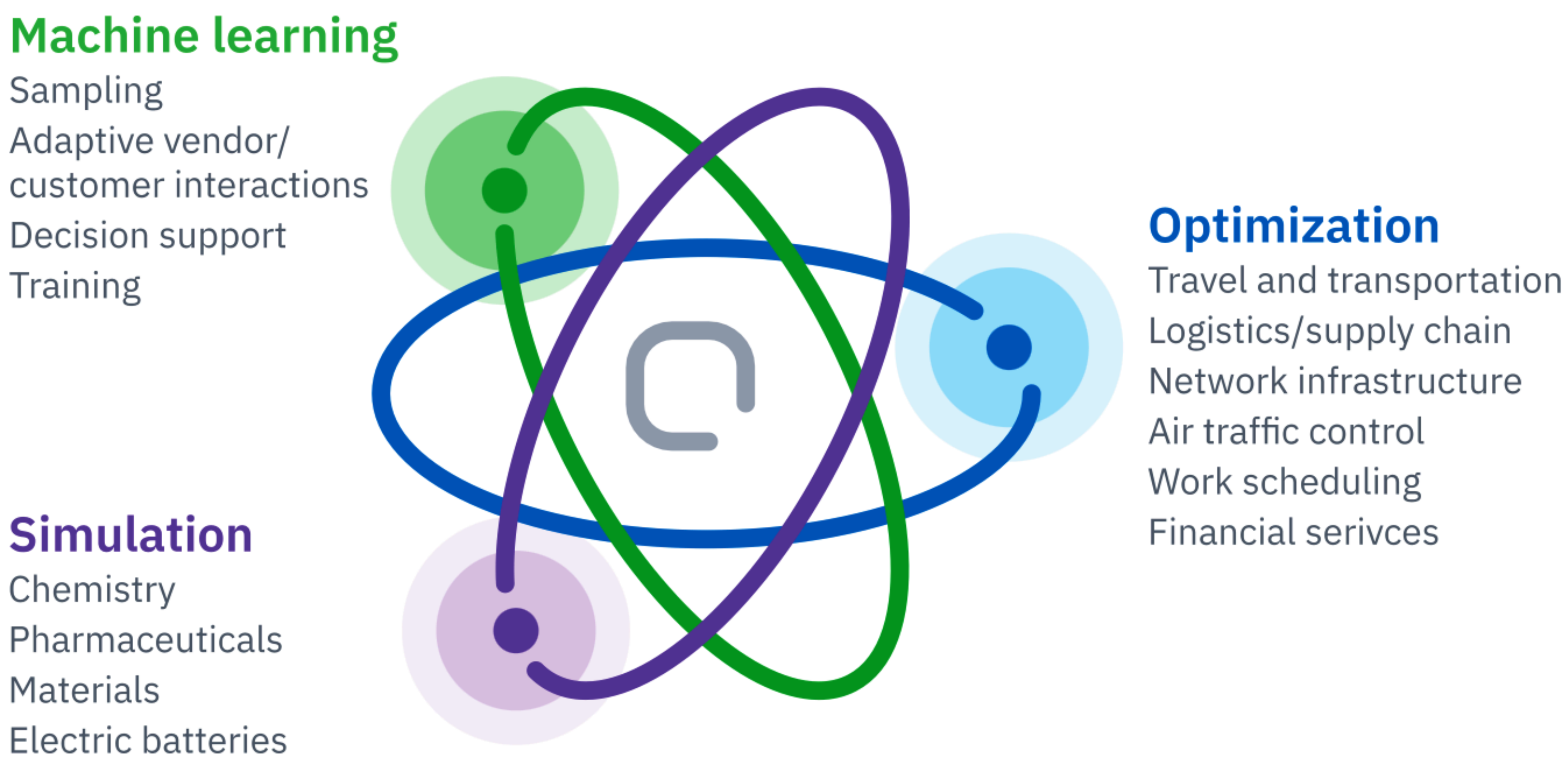
simulation of particle collisions and and data analysis in particle physics experiments



exploration of theoretical models which are intractable with classical computers

figure: Michael Spannowsky

many others interesting use-cases outside fundamental research …



QC use-cases in 2023 for IBM

ELEMENTS OF A QUANTUM CIRCUIT MODEL OF COMPUTATION

- in a general way any computation (classical or quantum) is based on three fundamental elements:

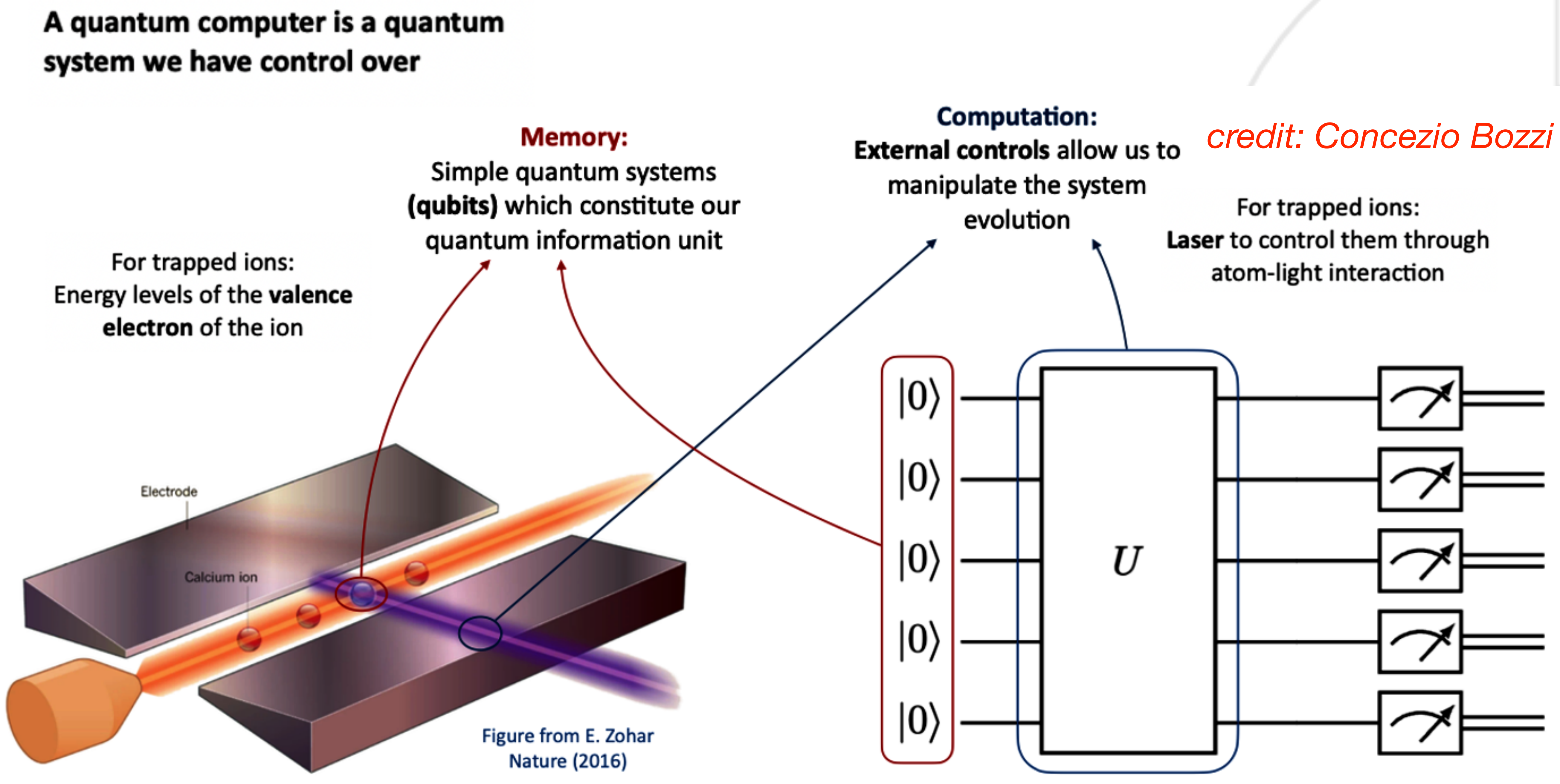
input data → operations on data → output results

- in a quantum circuit these elements are described by:

- **qubits** (quantum bits): basic unit of quantum information
  - store information as classical bits in a CC

- **quantum logic gates**: operators that transform quantum data
  - the building blocks of QC, like classical logic gates in CC

- **quantum measurement**: the operation that allows to access classically the resulting quantum state
  - reading out information from a quantum system generally change the state and destroy the computation that we are performing
  - can't predict the exact outcome of a quantum measurement, due to the probabilistic nature of QM all we can predict are only likelihood



quantum circuit representation

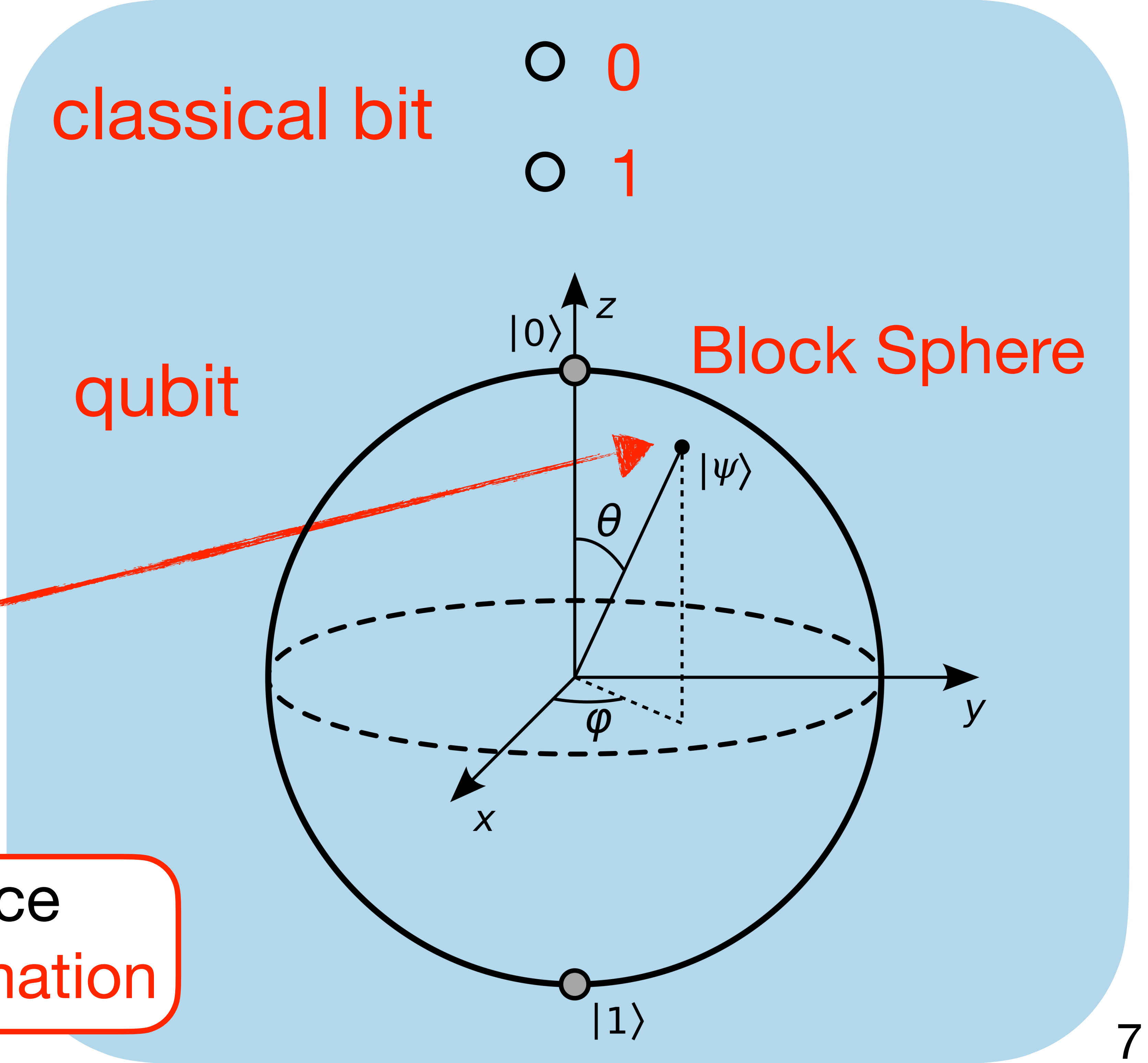
QUANTUM BIT

- basic unit of quantum computation representation
  - **classical bit**: binary (“0 or 1”)
  - a generic quantum state (**qubit**) $|\psi\rangle$ can be written in a superposition of a Hilbert space basis: can “take” infinitely many different values, it is continuous (when we read it we always find 0 or 1)

computational basis typically used as canonical basis:

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \qquad |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$$

$$|\psi\rangle = \begin{bmatrix}a_0\\a_1\end{bmatrix} = a_0\begin{bmatrix}1\\0\end{bmatrix} + a_1\begin{bmatrix}0\\1\end{bmatrix} = a_0|0\rangle + a_1|1\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle$$



Extending this to a system of n qubits forms a 2^n-dimensional Hilbert Space
eg. 2^n complex numbers → a quantum system can contain much more information

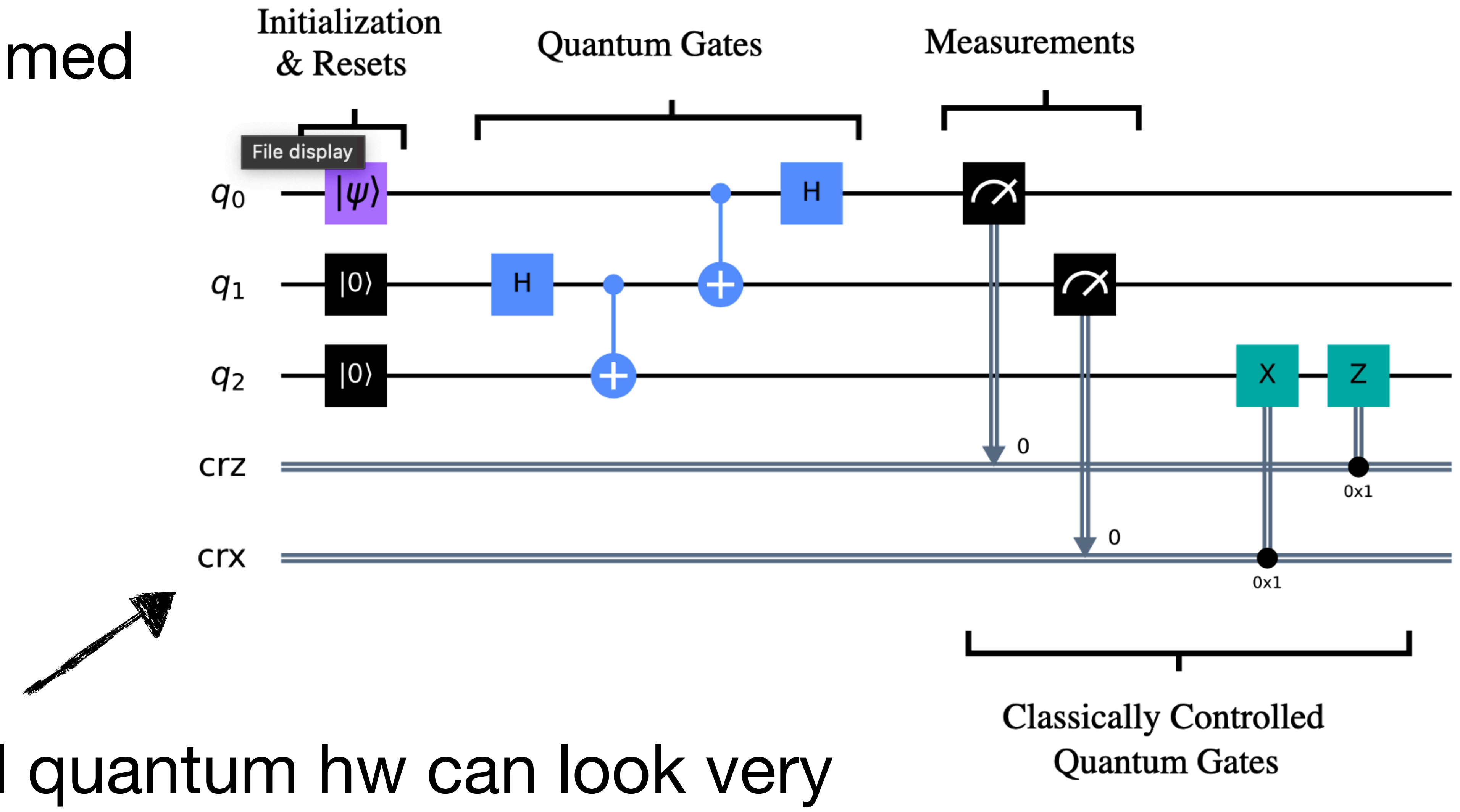
OPERATIONS OVER QUBITS: QUANTUM GATES&CIRCUITS

- quantum computation proceeds by applying physical operations on a quantum state of qubits inducing a change in the state, in a similar way as we operate on classical bits through logical operations
- a state-changing operator is called a quantum gate, and it is represented by a complex unitary matrix, eg length-preserving, linear transformation matrix, which represent a rotation on the Bloch sphere:

$$|\psi'\rangle = U|\psi\rangle \quad \text{with } U \text{ unitary matrix:} \quad U : U^{\dagger}U = I$$

Operator	Gate(s)		Matrix
Pauli-X (X) (NOT gate)	X	$\oplus$	$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$
Pauli-Y (Y)	Y		$\begin{bmatrix}0 & -i\\i & 0\end{bmatrix}$
Pauli-Z (Z)	Z		$\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$
Hadamard (H)	H		$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$
Phase (S, P)	S		$\begin{bmatrix}1 & 0\\0 & i\end{bmatrix}$
$\pi/8$ (T)	T		$\begin{bmatrix}1 & 0\\0 & e^{i\pi/4}\end{bmatrix}$
Controlled Not (CNOT, CX)			$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$
Controlled Z (CZ)	Z		$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{bmatrix}$
SWAP			$\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}$
Identity/idle	I		$\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$

- **quantum circuit** = a collection of quantum gates that operates on qubits
- quantum software is programmed by building these circuits



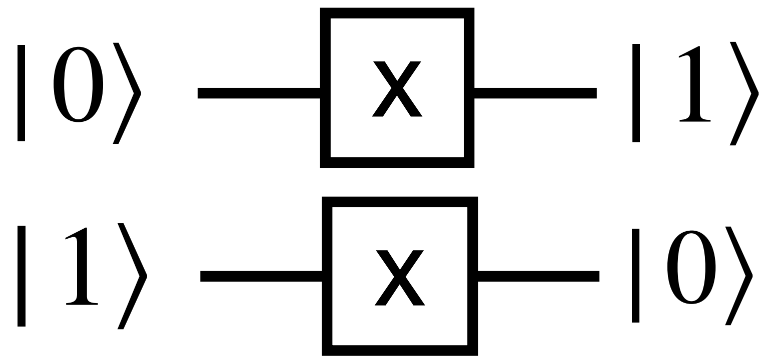
NOTE: when ported to the real quantum hw can look very different from the initial design (circuit adaptation, transpiling)

ONE-QUBIT GATES

- there is an uncountably infinite number of possible quantum gates, however all the interesting quantum logic can be approximated by the composition of a relatively small number gates acting on 1 or 2 qubits (Solvay-Kitaev theorem: a finite set of gates can approximate any unitary operation)
- most notable quantum gates acting on single qubits:

Pauli X-gate (or NOT gate)

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

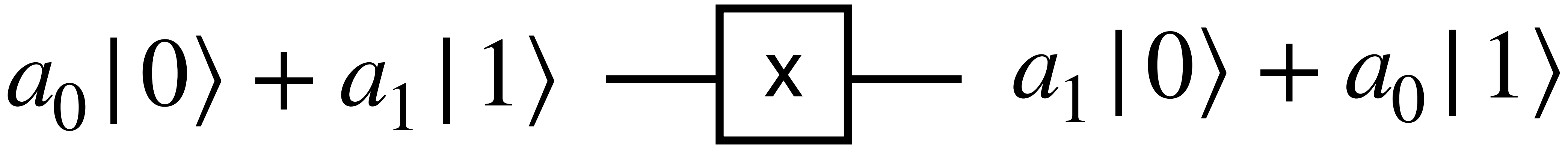


acts like a classical NOT gate

corresponds to a rotation around the x axis of the Bloch sphere by π radians

$$|\psi\rangle = a_0|0\rangle + a_1|1\rangle$$

$$X|\psi\rangle = a_1|0\rangle + a_0|1\rangle$$



ONE-QUBIT GATES

Pauli Z-gate

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$|0\rangle \;—\boxed{Z}—\; |0\rangle$$

$$|1\rangle \;—\boxed{Z}—\; -|1\rangle$$

reverse the phase of $|1\rangle$

Hadamard H-gate

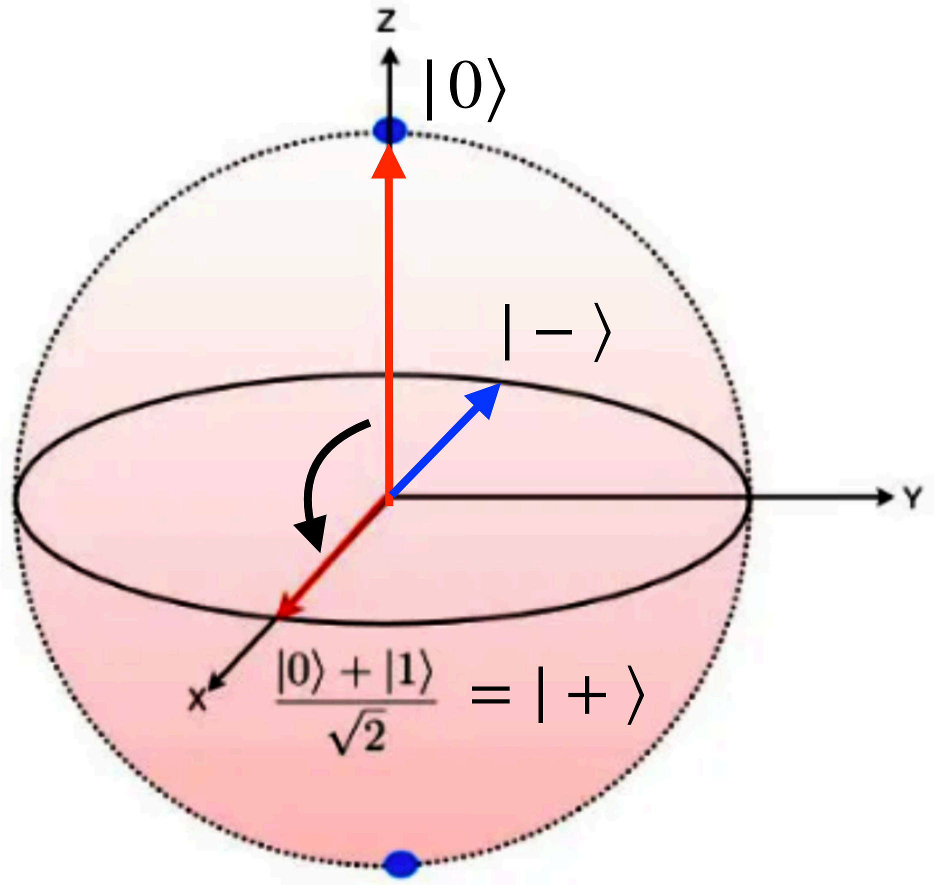
create superposition between states

is a one-qubit rotation, that maps the basis states to an equal superposition state of the two states

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$|0\rangle \;—\boxed{H}—\; \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$$

$$|1\rangle \;—\boxed{H}—\; \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$$



INTERFERENCE

- let's apply the H gate to the $|0\rangle$ state:

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) = |+\rangle$$

- if we apply again the H gate to the resulting $|+\rangle$ state:

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) = \frac{1}{2}\left(|0\rangle + \cancel{|1\rangle} + |0\rangle - \cancel{|1\rangle}\right) = |0\rangle$$

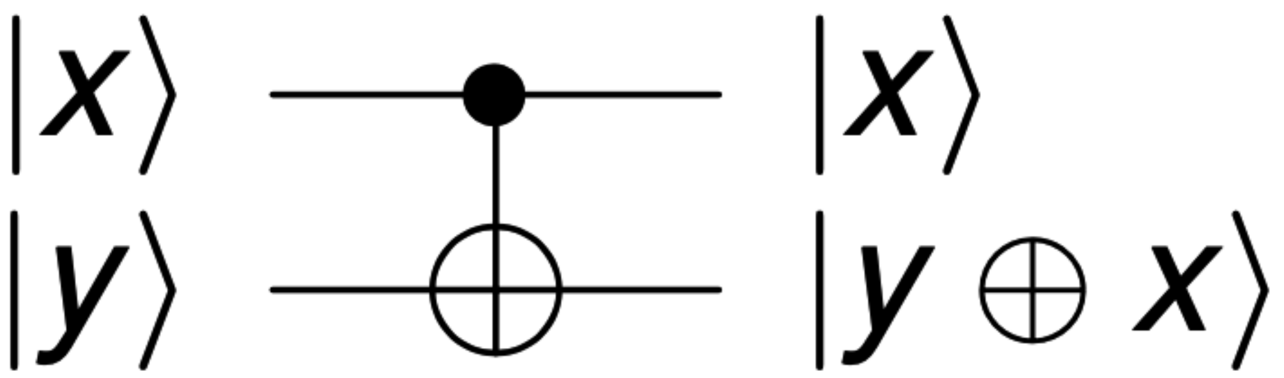
this an example of interference in QM: the probability amplitudes of the $|1\rangle$ state destructively interfere and vanish from the superposition, the output state is now deterministic $|0\rangle$ with probability 1

CNOT GATE

- the Controlled-NOT gate (CNOT, CX) is a two-qubit gate represented by the unitary matrix:

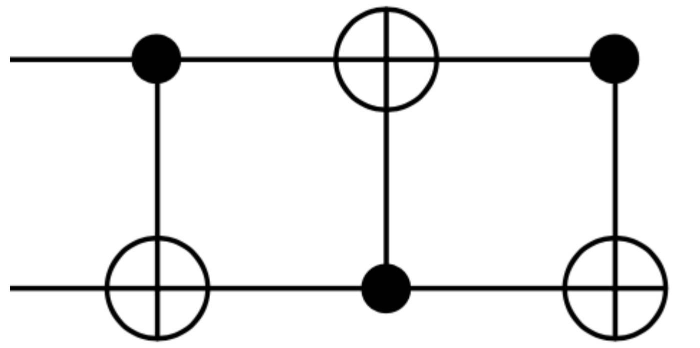
$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

  - if the first qubit is $|0\rangle \Rightarrow$ nothing change
  - if the first qubit is $|1\rangle \Rightarrow$ the second qubit is flipped and the first doesn't change



$|01\rangle \rightarrow |01\rangle$ $\quad |11\rangle \rightarrow |10\rangle$

$|00\rangle \rightarrow |00\rangle$ $\quad |10\rangle \rightarrow |11\rangle$



$|01\rangle \rightarrow |01\rangle \rightarrow |11\rangle \rightarrow |10\rangle$

allows to build other controlled gates, swap states, create entangled states …

ENTANGLEMENT

- a state $|\psi\rangle$ is called a product state if can be written in the form: $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$ (ex. $|00\rangle = |0\rangle \otimes |0\rangle$)
- there are states that cannot be written in this form though they live in the same space of product states, they are called entangled states

example

$$|\psi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

cannot be written as $|\psi_1\rangle \otimes |\psi_2\rangle$

proof

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \otimes \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} a_0 \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \\ a_1 \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} a_0 b_0 \\ a_0 b_1 \\ a_1 b_0 \\ a_1 b_1 \end{bmatrix} = a_0 b_0 |00\rangle + a_0 b_1 |10\rangle + a_1 b_0 |01\rangle + a_1 b_1 |11\rangle$$

$a_0 b_0 = 1/\sqrt{2}$ → $a_0 \neq 0$

$a_0 b_1 = 0$ → $b_1 = 0$

$a_1 b_0 = 0$

$a_1 b_1 = 1/\sqrt{2}$

inconsistent

MEASURING A QUBIT

- the last stage of a quantum circuit is a measurement, a read-out of computational results classically
- quantum measurement behave very similarly to a random variable in classical probability

1. the result of the measurement of a qubit $|\psi\rangle$ is a random value between 0 and 1
2. the probability of getting 0 or 1 is given by the Born Rule:

   if we measure the state: $|\psi\rangle = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = a_0 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + a_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = a_0|0\rangle + a_1|1\rangle$

   - we get outcome 0 with probability $|a_0|^2$
   - we get outcome 1 with probability $|a_1|^2$

3. after the measurement $|\psi\rangle$ the superposition is destroyed: the new state will be $|0\rangle$ or $|1\rangle$ depending on the outcome we have obtained (this his called collapse of the wave function)
4. we cannot perform several independent measurements of $|\psi\rangle$ because it is not possible to clone a quantum state (no-cloning theorem)

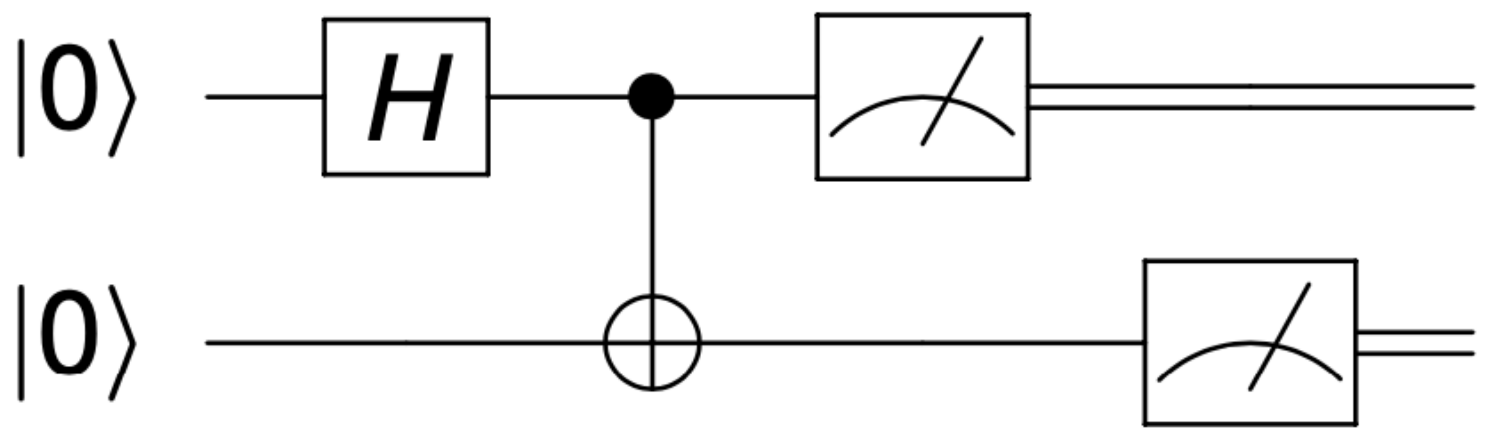
MEASURING TWO-QUBIT STATES

- given the superposition state: $|\psi\rangle = a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle$

- if we **measure both qubits** we will obtain:
  - 00 with probability $|a_{00}|^2$ and the post-measurement state will be $|00\rangle$
  - 01 with probability $|a_{01}|^2$ and the post-measurement state will be $|01\rangle$
  - 10 with probability $|a_{10}|^2$ and the post-measurement state will be $|10\rangle$
  - 11 with probability $|a_{11}|^2$ and the post-measurement state will be $|11\rangle$

- if we **measure only one of the two qubits**, for example the first one, we will obtain:
  - 0 with probability $|a_{00}|^2 + |a_{01}|^2$ and the post-measurement state will be $|\psi'\rangle = \dfrac{a_{00}|00\rangle + a_{01}|01\rangle}{\sqrt{2}}$
  - 1 with probability $|a_{10}|^2 + |a_{11}|^2$ and the post-measurement state will be $|\psi'\rangle = \dfrac{a_{10}|10\rangle + a_{11}|11\rangle}{\sqrt{2}}$

ENTANGLED STATES



- consider the following simple circuit:
- initial state: $|00\rangle$
- after H gate on first qubit: $\frac{|00\rangle + |10\rangle}{\sqrt{2}}$
- after the CNOT gate: $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$
- we measure the first qubit and we will get 0 or 1 with 50% probability
    - suppose we obtain 0, then the new state will be: $|00\rangle$
        - then if now we measure the second qubit we obtain 0 with probability 1
- if instead we measure 1 in the first qubit, then we will measure 1 also in the second with probability 1

in an entangled state, the quantum state of each qubit cannot be described independently of the state of the others, including when the qubits are separated by a large distance values (spooky action at distance)

QUANTUM ALGORITHMS

- an entire zoo of sophisticated quantum algorithms that can offer speedups over classical algorithms has been studied and proposed in literature:
  - Shor's (proved exponential speedup in factoring prime numbers), Grover's (polynomial (quadratic) speedup for searching in an unsorted db), Quantum MC, Quantum Fourier Transform …

- assume fault-tolerant quantum processors with many qubits
- current quantum computers support only $O(10^{2\div3})$ qubits, noisy and not all necessarily able to interact with each others: noisy intermediate-scale quantum devices (NISQ):
  - no error correction: can produce only approximate results of computations
  - algorithms limited to use only a few qubits and gates with deep impact on quantum algorithmic design and achievable performance

- narrow down the objective: finding problems that can be solved by NISQ devices while possibly exhibiting some kind of utility wrt classical algorithms:
  - ex. find the ground-state energy of a many-body system (ex. a molecule) with quantum annealers
  - **ex. Quantum Machine Learning**

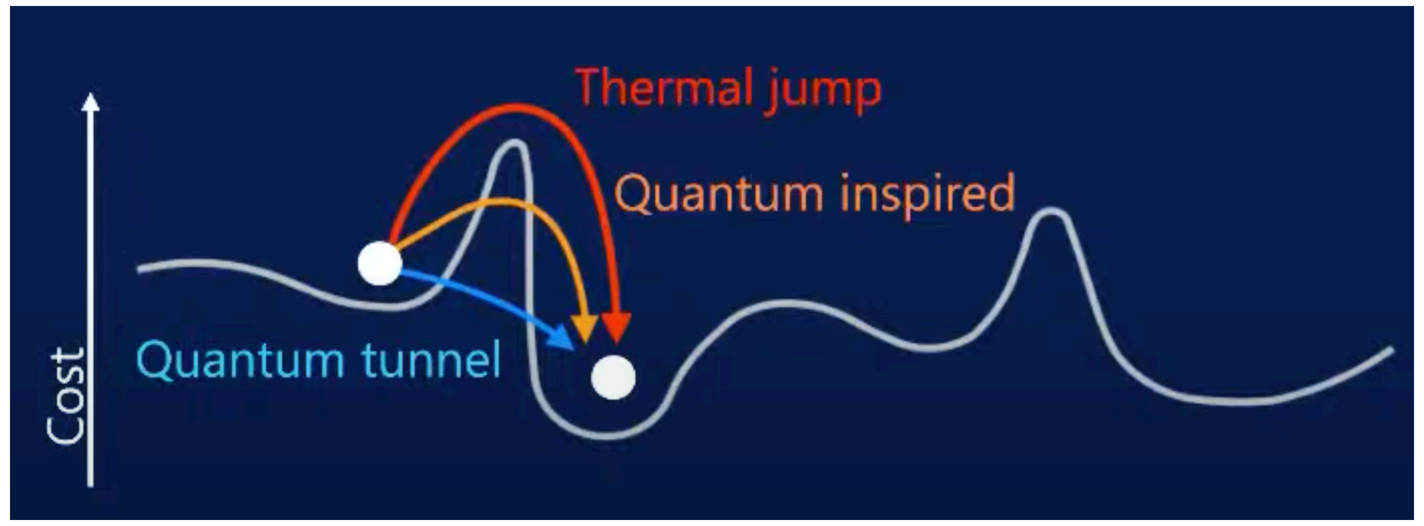


figure: M.Troyer - Quantum Colloquium 2021 - Simons Institute

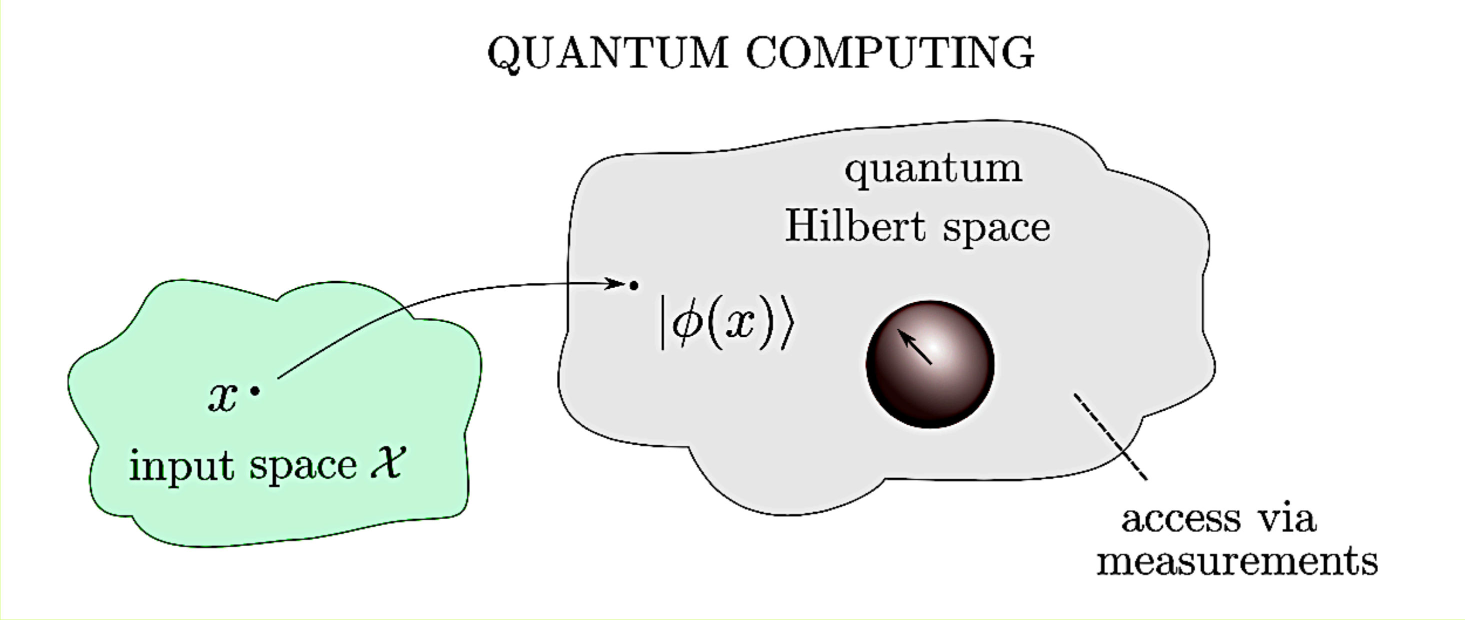
QUANTUM MACHINE LEARNING

A set of hybrid algorithms inspired by classical ML that all share the same common idea:

- a parametric quantum circuit (ansatz) implements an algorithm and returns results via measurement operations
- a classical objective function, encapsulate the problem-specific goal, is used to find the parameters of the PQC on a classical computer

Kernel Based

ex. Quantum-Support Vector Machines

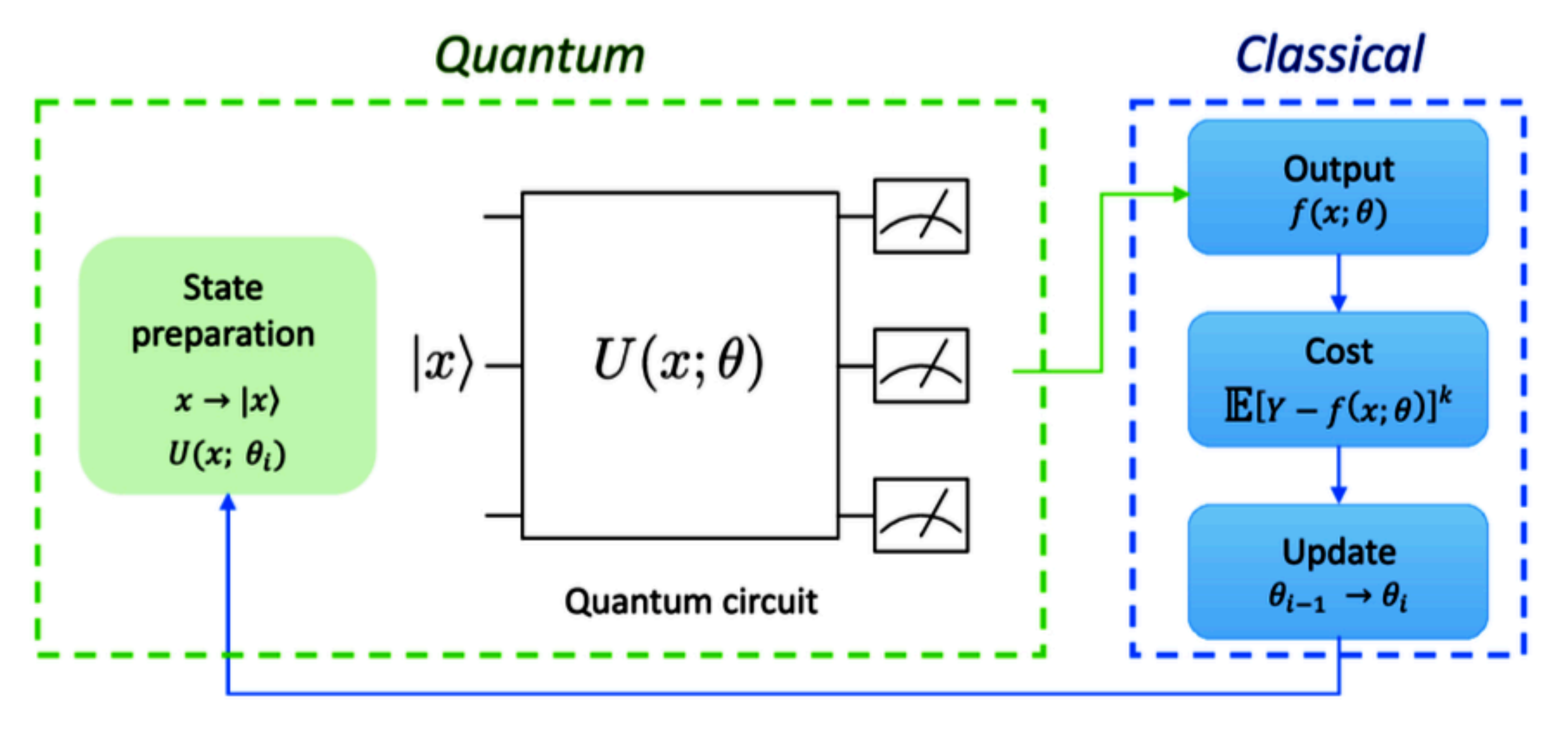


$$|\phi(x)\rangle = U_\phi(x)|0\rangle$$
$$|\phi(x')\rangle = U_\phi(x')|0\rangle$$
$$|\langle\phi(x)|\phi(x')\rangle|^2 = K(x,x')$$

- advantages of PQCs: one circuit can represent an entire family of different algorithms, less sensitive to noise

Variational Q-Algorithms

ex. Quantum-NN



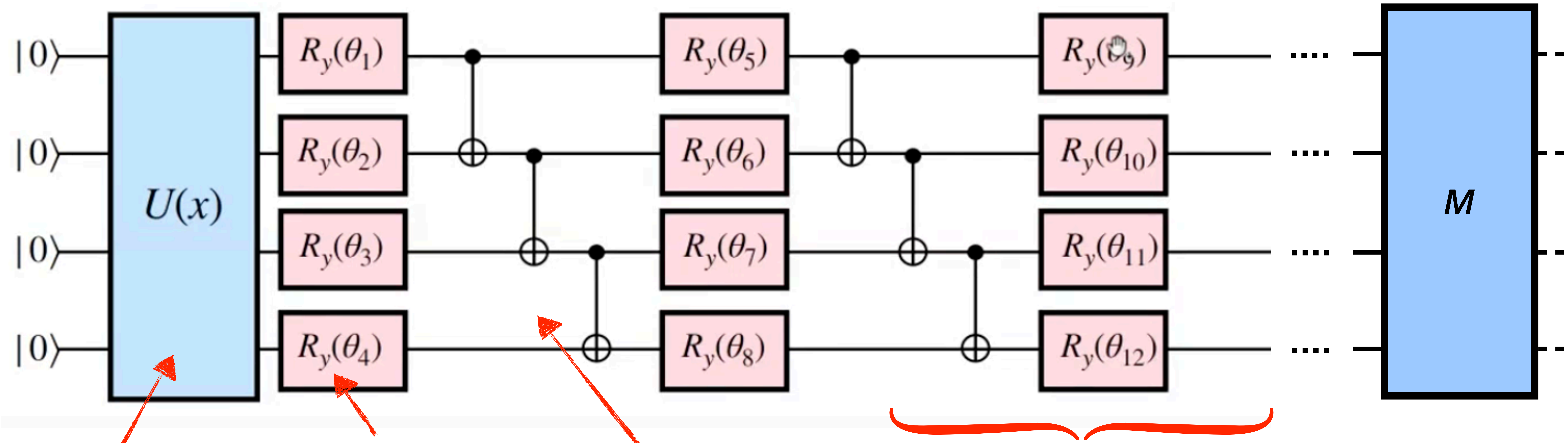
Energy Based

Finds the ground-state of a quantum system using a quantum-classical hybrid approach (ex. Variational Quantum Eigensolver (VQE))

QUANTM NEURAL NETWORKS

- **typical example: a QNN**

$$\hat{y} \doteq f(x, \theta) = \langle M \rangle = $$
$$= \langle x | V^{\dagger}(\theta) M V(\theta) | x \rangle$$



$\hat{y}$

$L(y, \hat{y} | \theta)$

$$\theta^* = \arg\min \sum_{i \in D} L(y_i, f(x_i, \theta))$$

encoding classical input x

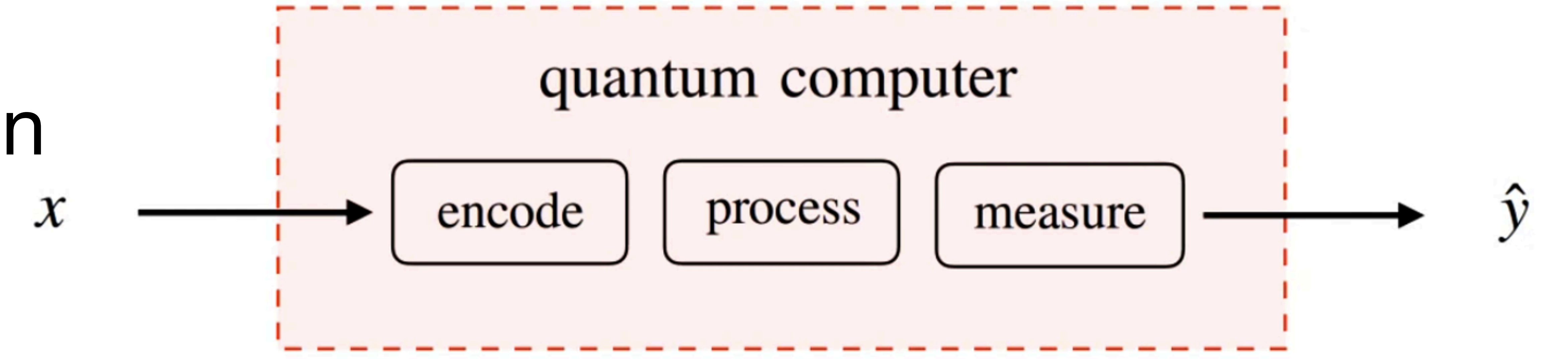
parametrised rotations (along y in this example)

entanglement block via CNOT gates creates non-classical correlations that hopefully will increase the expressibility of the circuit

repeated N-times (N-layers) to increase expressibility

ENCODING CLASSICAL DATA

- encoding of classical data is a crucial step in implementing a QML algorithm

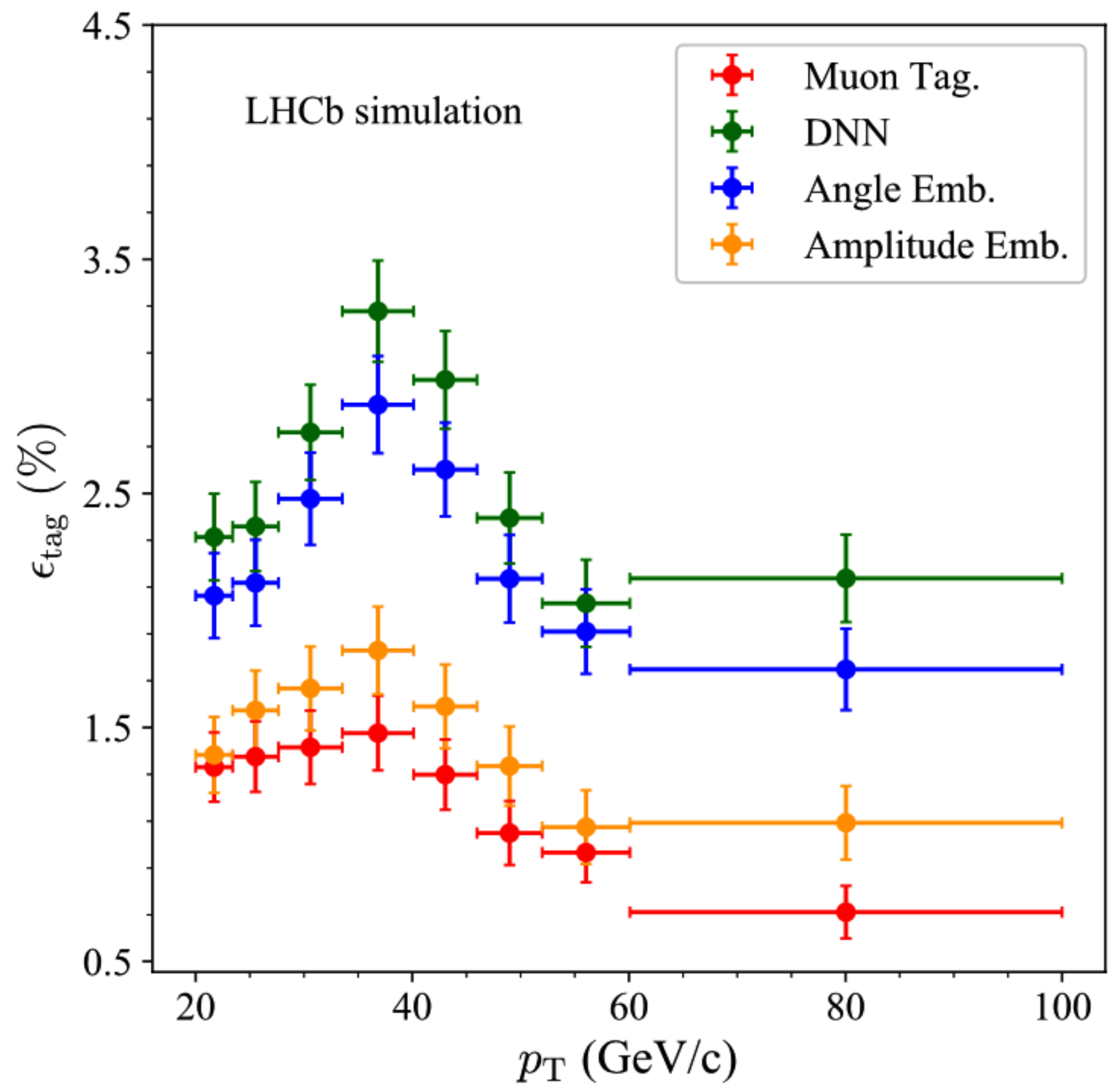


- several way to do it, ranging from conceptually simple (ex angular encoding) but resources hungry (# qubits), to more efficient but also more complex (tradeoff between compression and circuit depth)

- example: amplitude encoding

$$x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \rightarrow |x\rangle = \frac{1}{\|x\|}(x_0|0\rangle + x_1|1\rangle) = \frac{1}{\|x\|}\sum_{i=0}^{N} x_i|i\rangle$$

- fewer qubits needed: exponential compression $n_q \propto O(\log N)$
- more complex preparation and readout: # of gates $n_g \propto O(\text{poly}(N))$

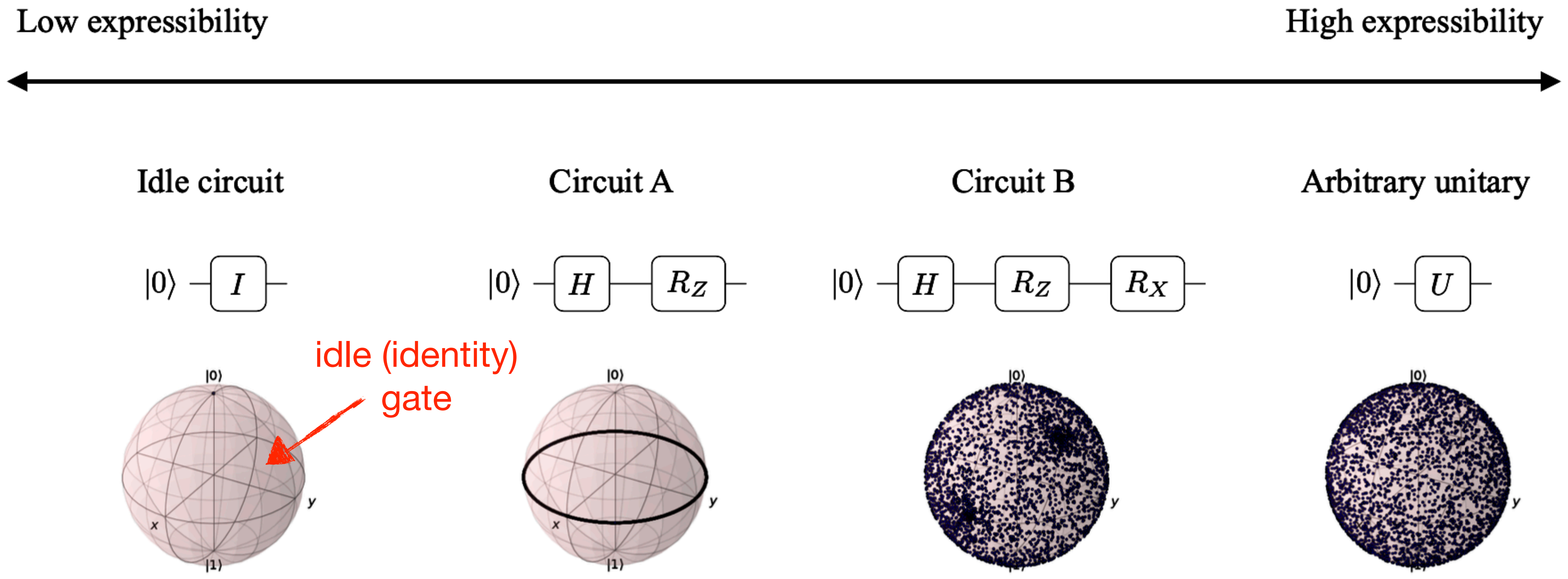


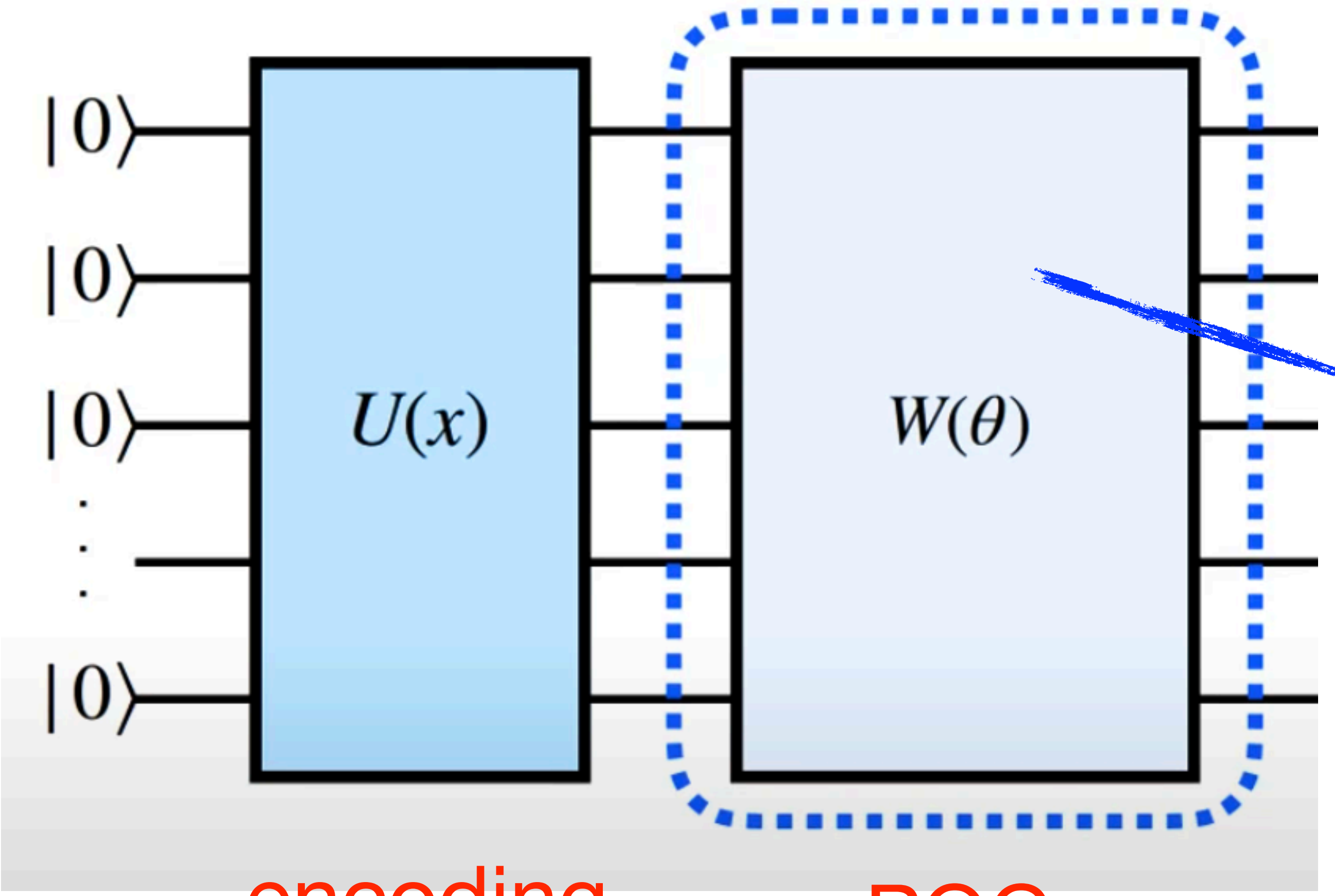
A.Gianelle et al, JHEP 2022, 14

EXPRESSIVE POWER OF PARAMETRIZED QUANTUM CIRCUITS

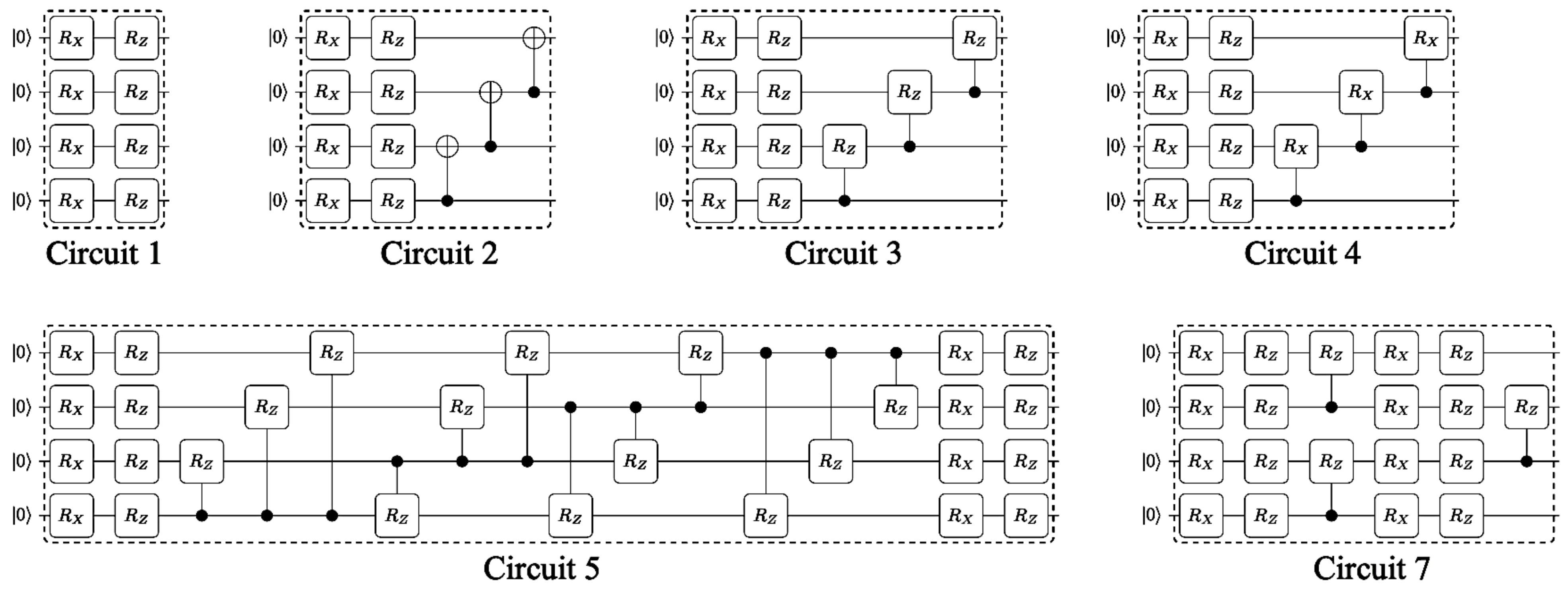
- several studies in literature on how to choose the circuit ansatz in order to maximise expressibility and entangling capabilities (eg ability to efficiently represent the solution space and to capture non-trivial correlation in the quantum data)

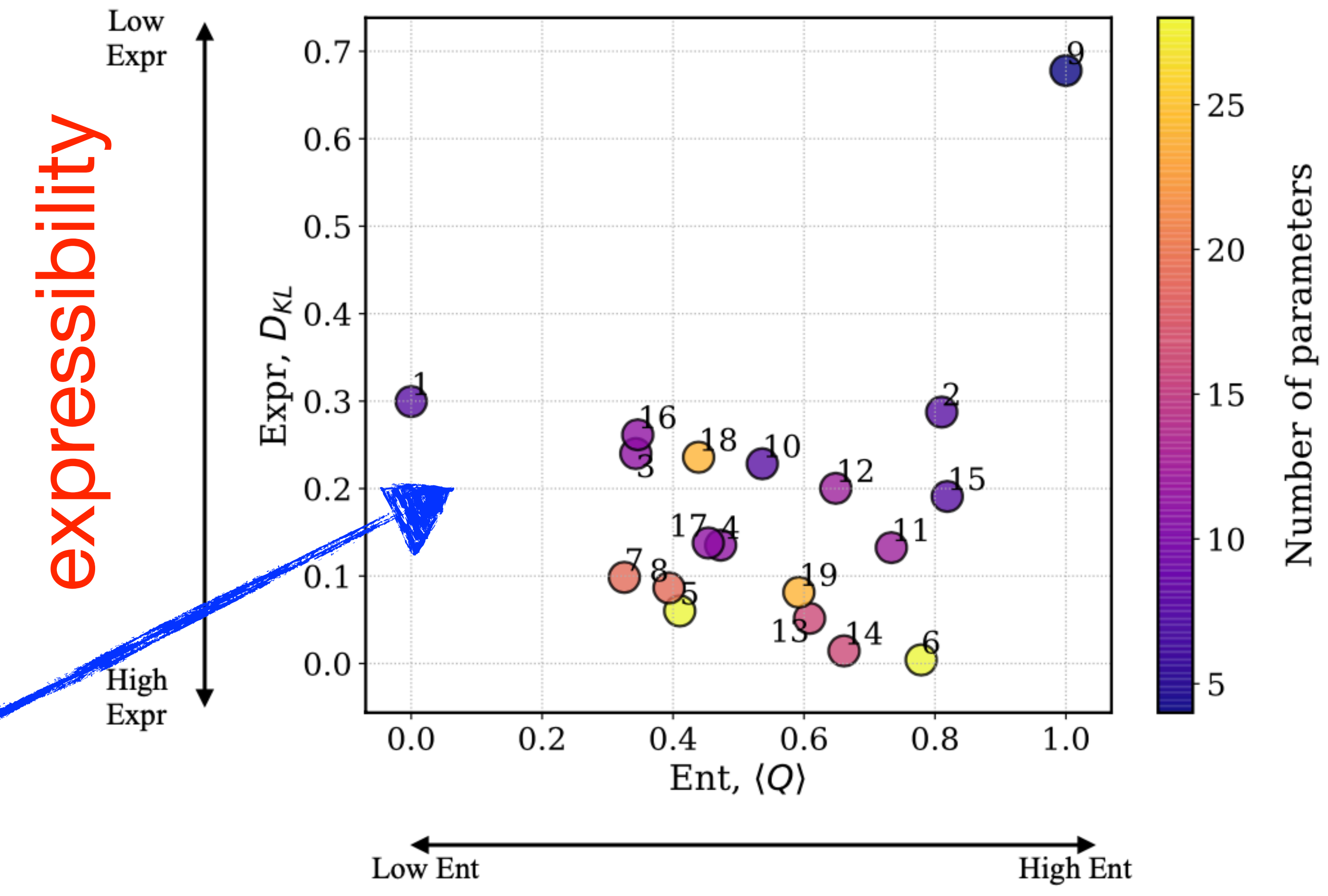
- expressibility: circuit’s ability to generate (pure) states that are well representative of the Hilbert space
- in the case of a single qubit, the expressibility corresponds to the circuit’s ability to explore the Bloch sphere





encoding PQC





entangling

S. Sim et al: arXiv:1905.10876

EXPRESSIBILITY VS TRAINABILITY: THE BARREN PLATEAUS PROBLEM

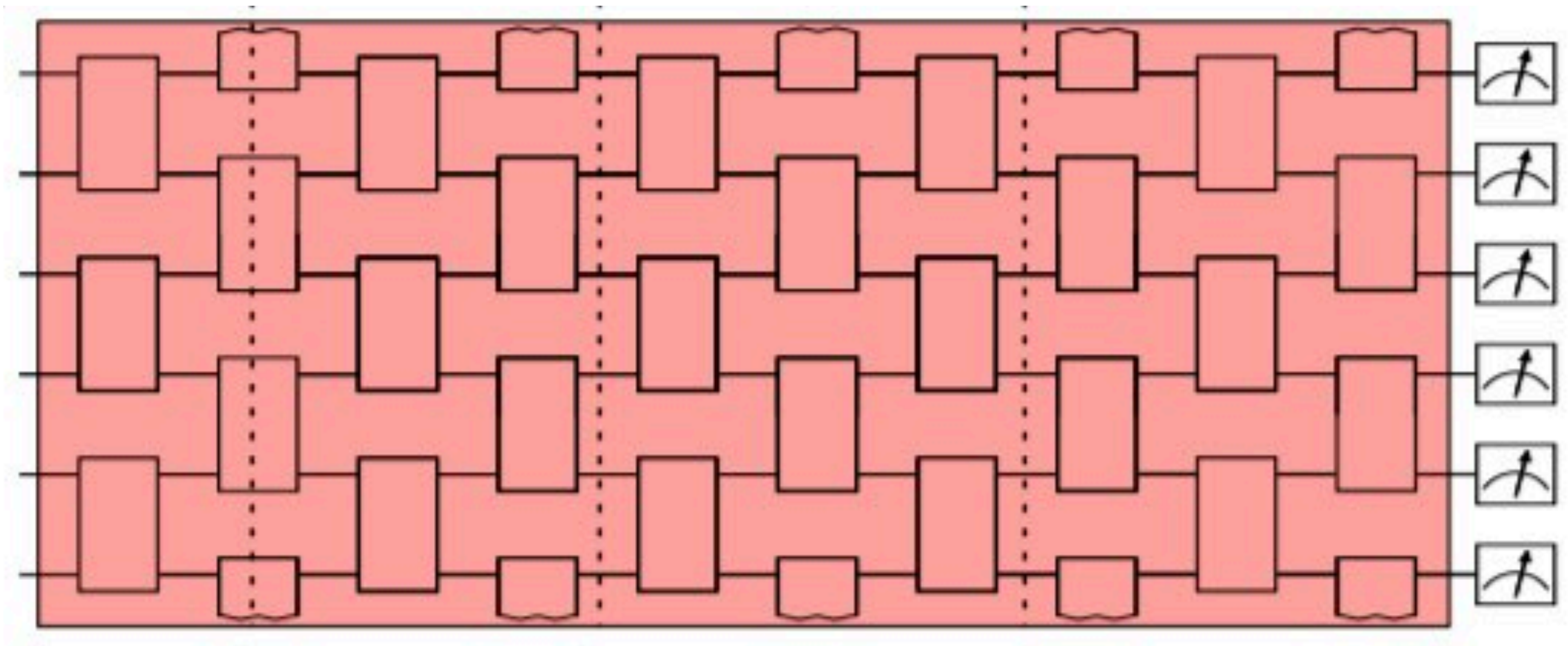
- variational circuits are affected by the presence of large regions in the loss landscape where the variance of the gradient is almost 0 (flat loss landscape)
- a circuit initialised in one of these areas will be untrainable using any gradient-based algorithm
- it can be shown that the gradient's variance exponentially decrease with the # of qubits

$$\langle \partial_\theta L \rangle \simeq 0$$

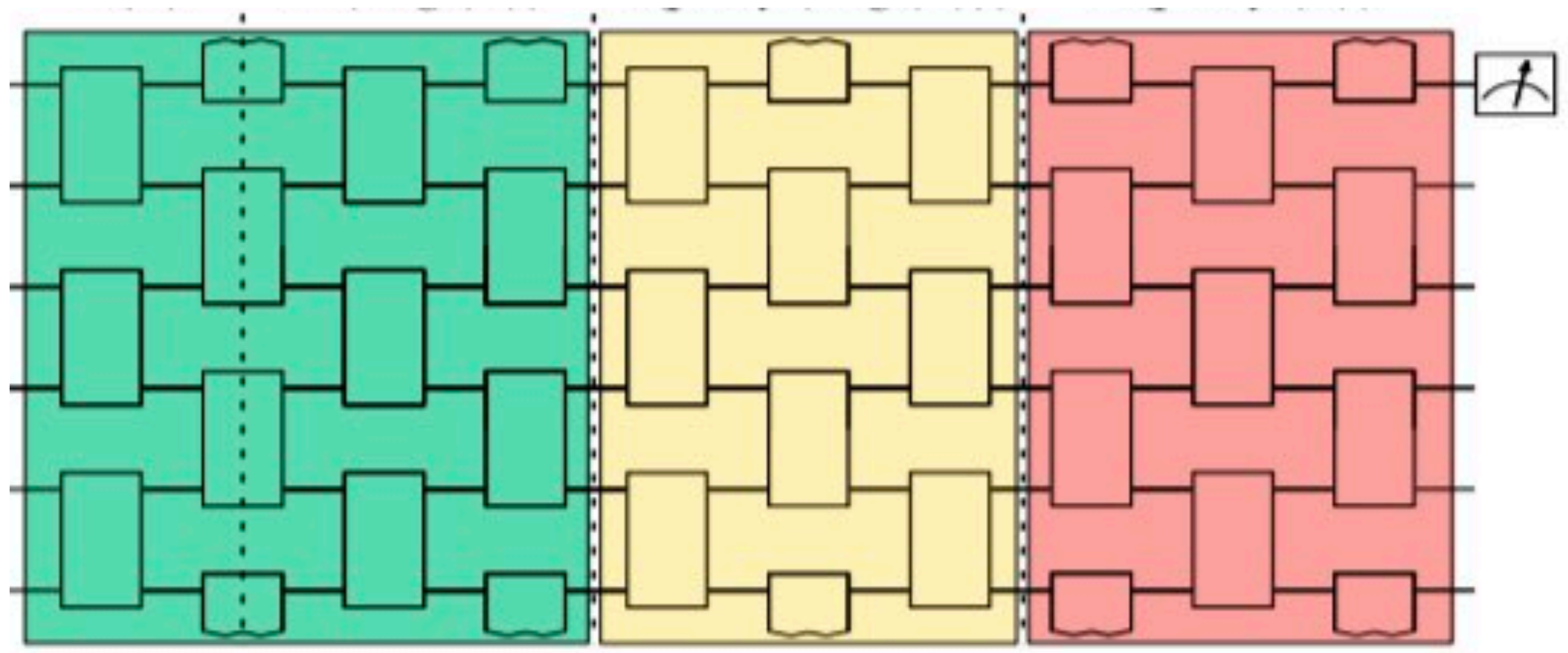
$$\text{Var}[\partial_\theta L] \sim 2^{-n}$$

J.R. Mc Clean et al., Nat. Comm.

- a possible mitigation strategy: use local cost functions that only have information from part of the circuit coupled with not too-deep circuits and not too much entanglement:



global loss



local loss

$$\text{Var}[\partial_\theta L] \gtrsim \text{poly(n)}^{-1}$$

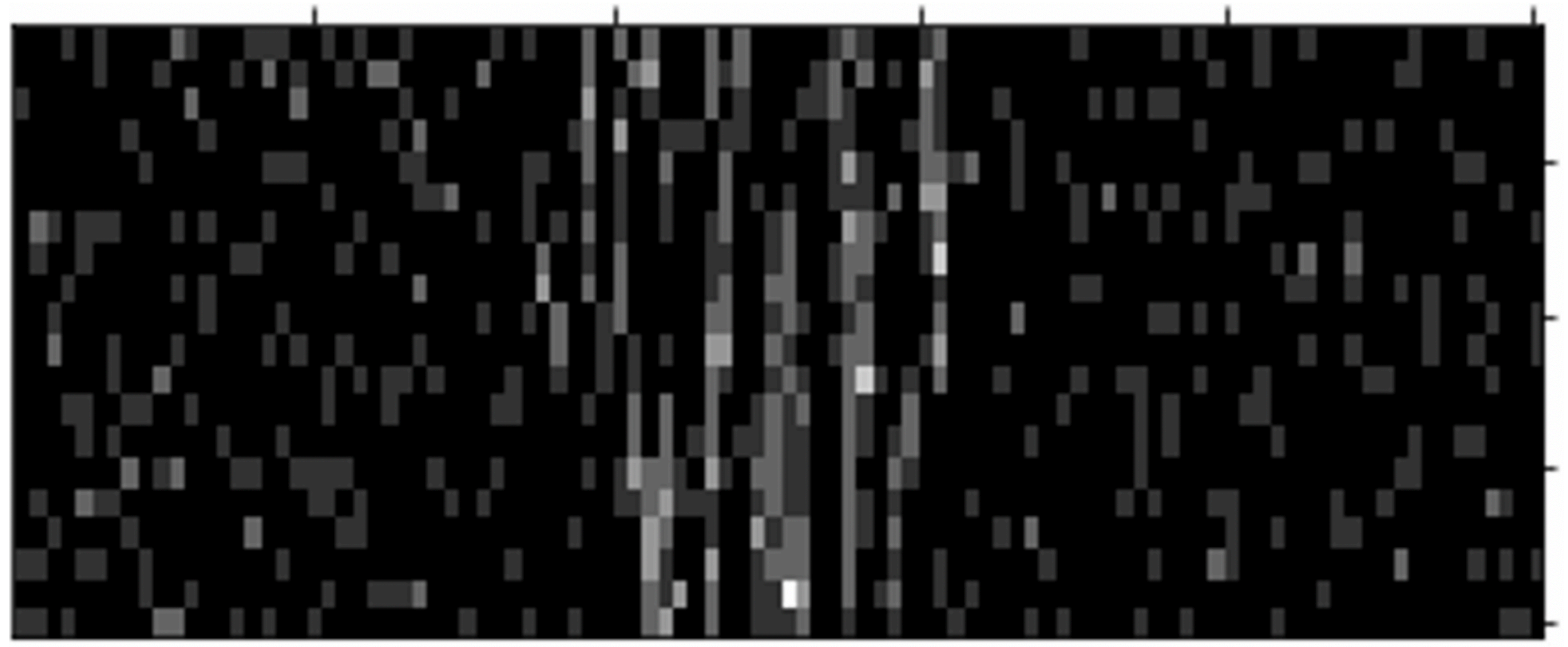
Cerezo et al: arXiv:2001.00550

EXAMPLE OF A QNN: ANOMALY DETECTION WITH A QUANTUM-AE

- a Quantum-AE able to identify highly displaced decays using the ATLAS muon spectrometer information

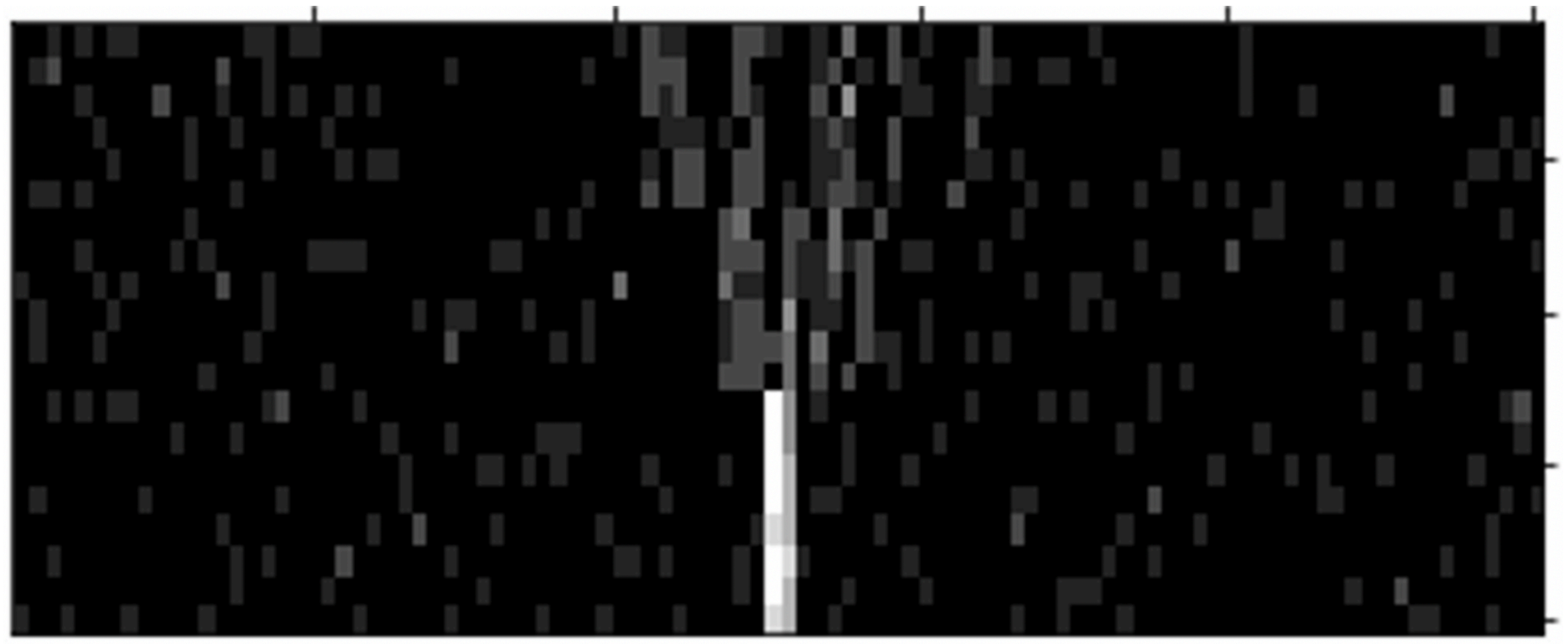
NORMAL event

“image” representation of a prompt decay in multi-muons

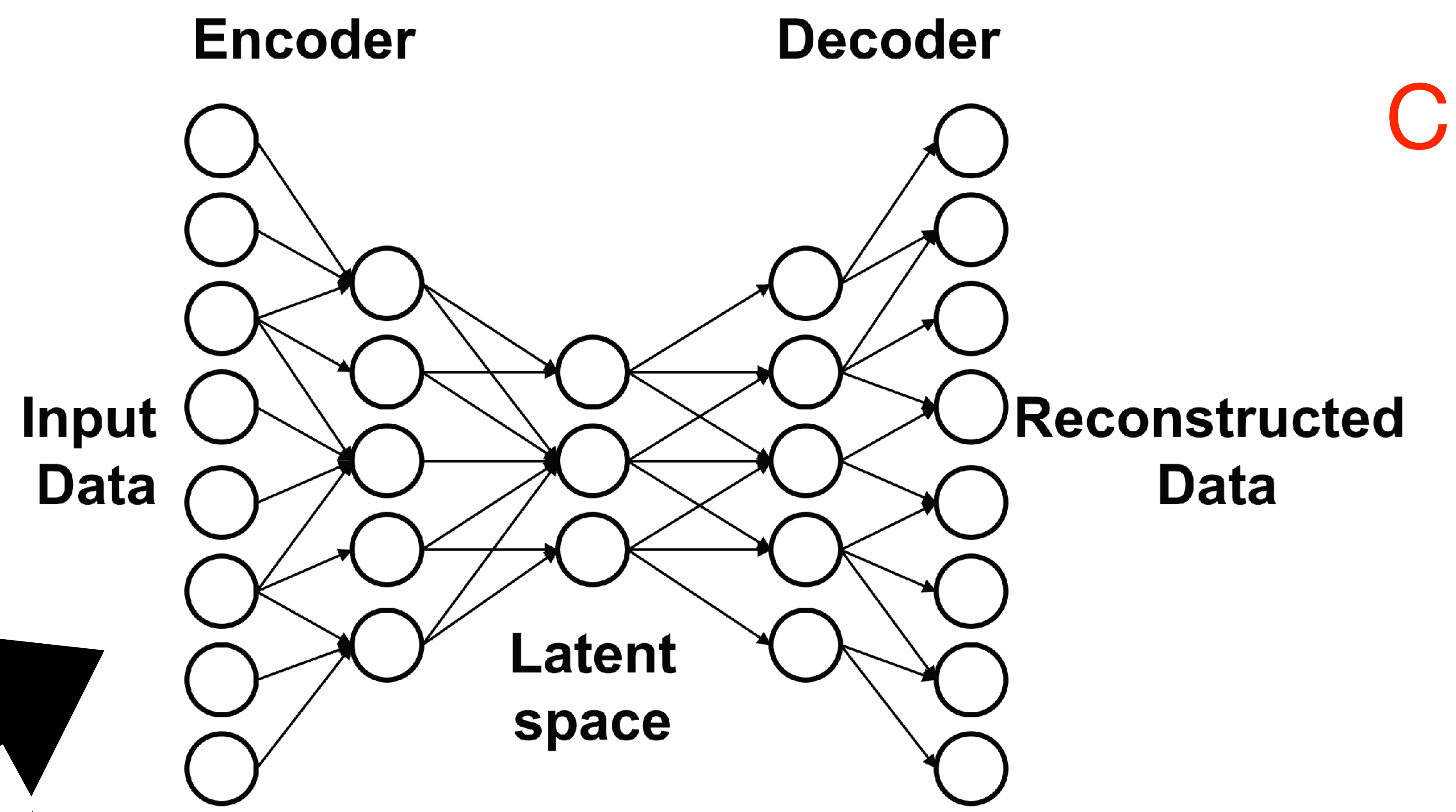


ANOMALOUS event

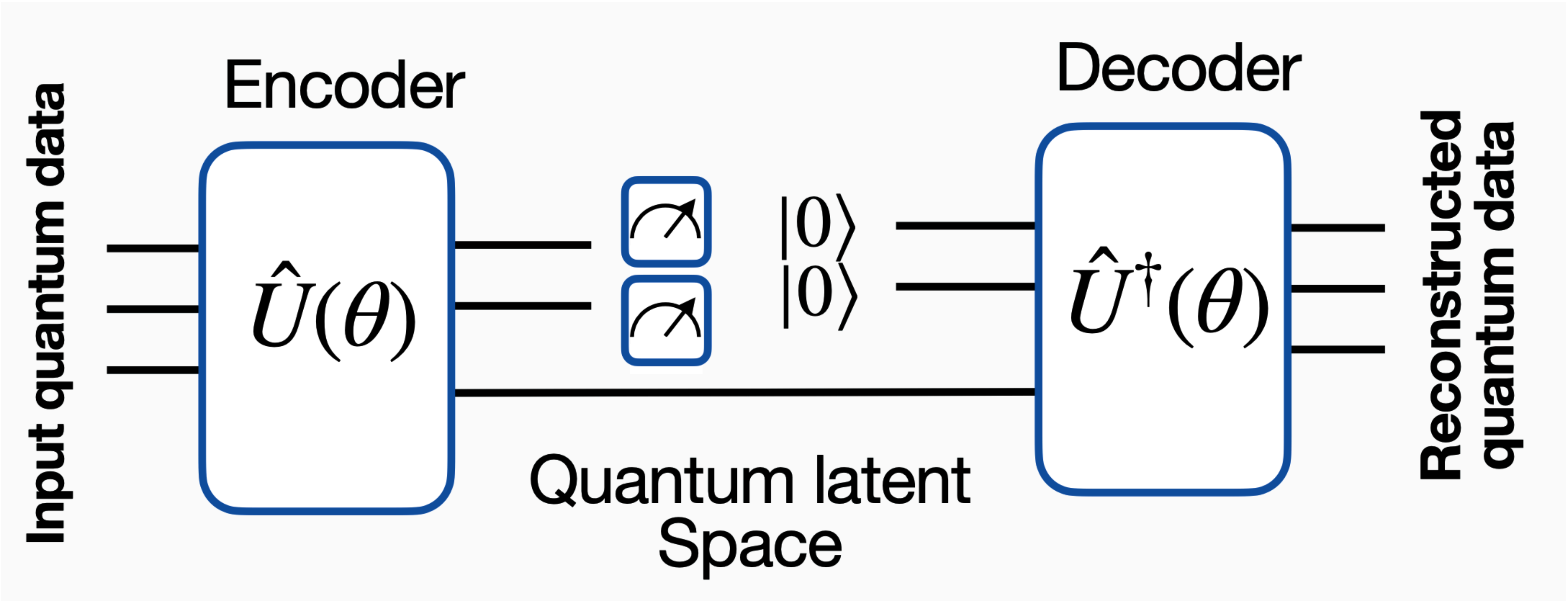
“image” representation of a highly displaced decay in multi-muons



Classical AE



Quantum-AE



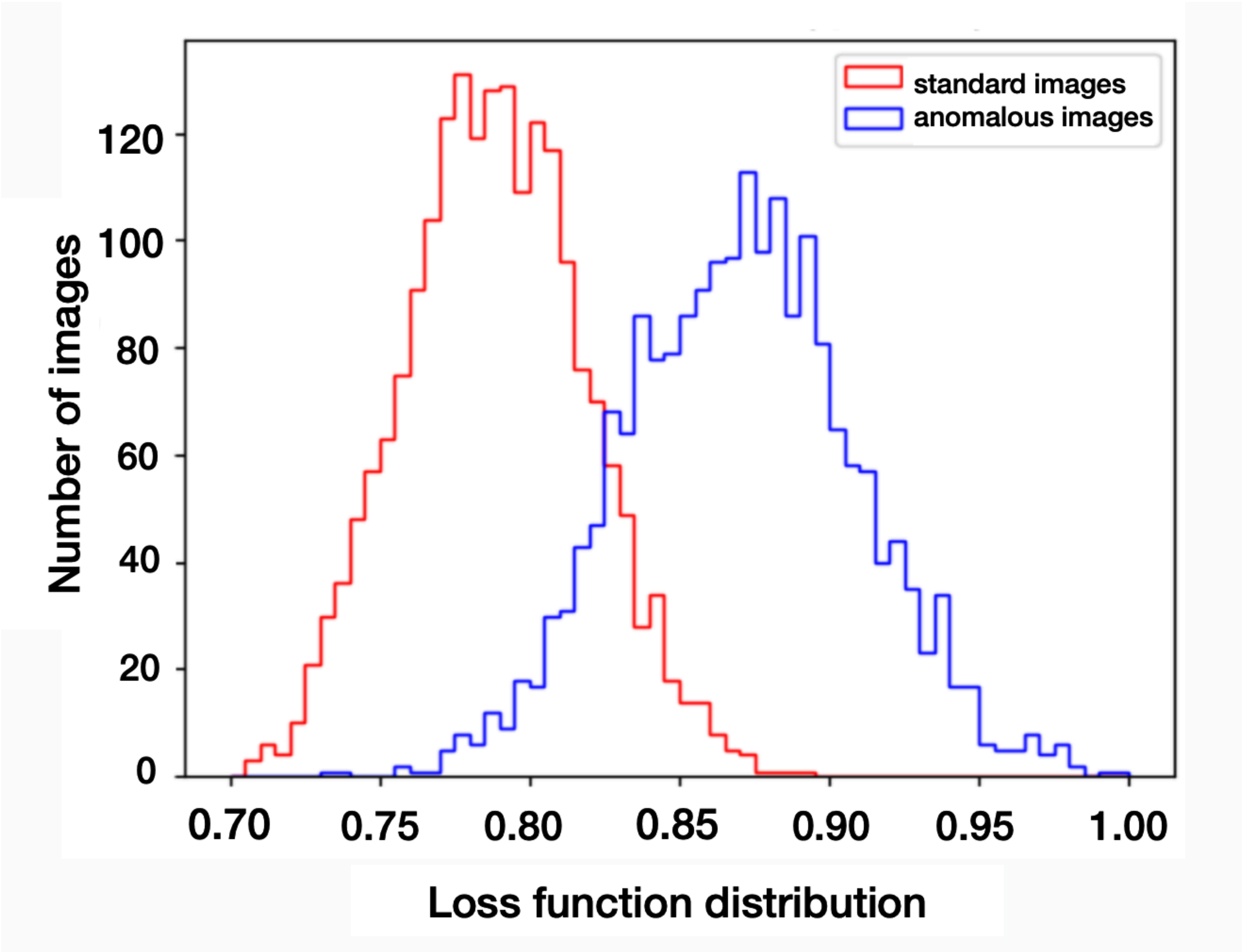
S.Bordoni et al, Particles 2023, 1, 1–15

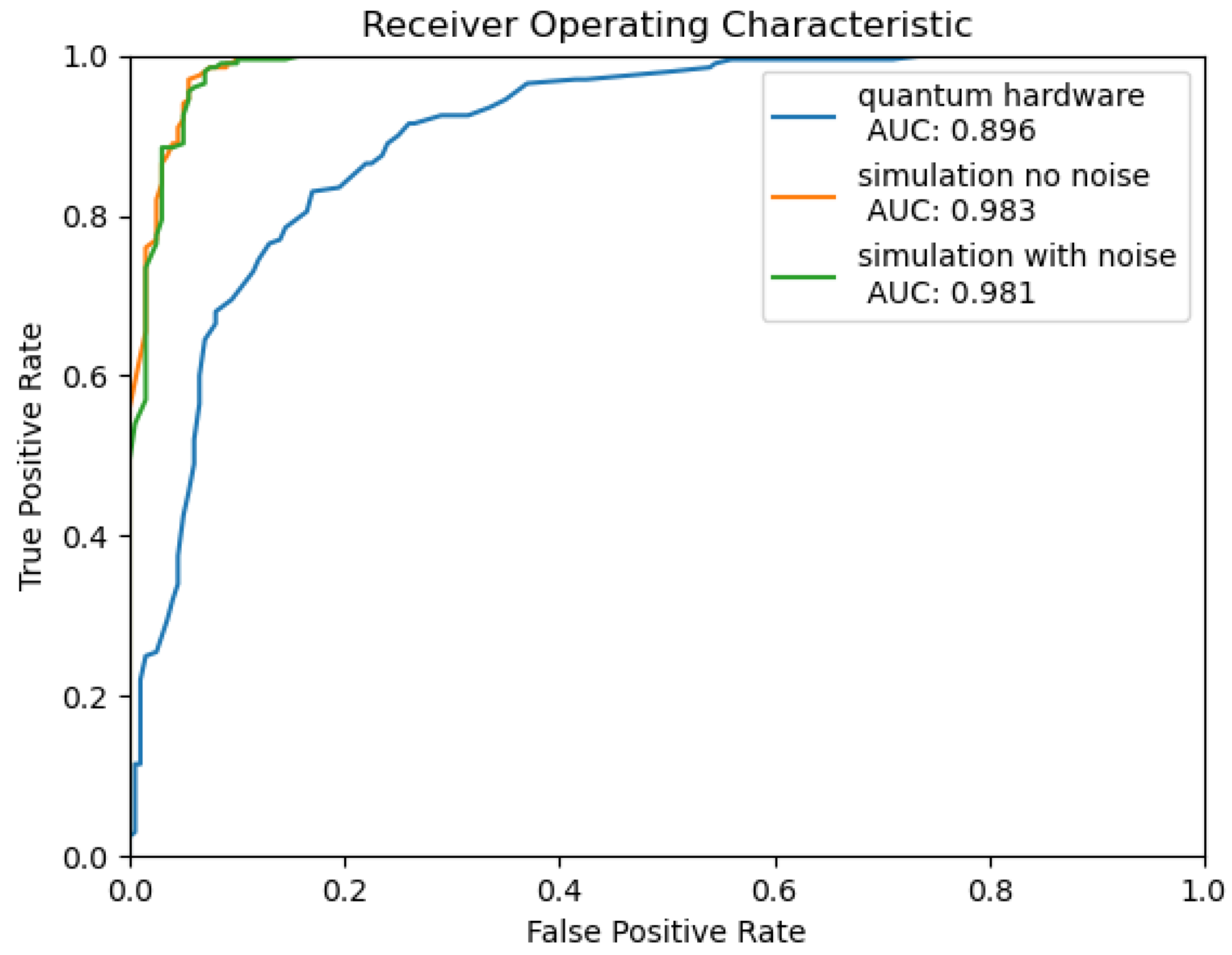
ANOMALY DETECTION WITH QUANTUM-AE

$U(\theta)$

```
q0:      ─RX─RY─RZ─o──────────X─RX─RY─RZ─o──────────X─RX─RY─RZ─o──────────X─RX─RY─  ...
q1:      ─RX─RY─RZ─X─o────────│─RX─RY─RZ─X─o────────│─RX─RY─RZ─X─o────────│─RX─RY─  ...
q2:      ─RX─RY─RZ───X─o──────│─RX─RY─RZ───X─o──────│─RX─RY─RZ───X─o──────│─RX─RY─  ...
q3:      ─RX─RY─RZ─────X─o────│─RX─RY─RZ─────X─o────│─RX─RY─RZ─────X─o────│─RX─RY─  ...
q4:      ─RX─RY─RZ───────X─o──│─RX─RY─RZ───────X─o──│─RX─RY─RZ───────X─o──│─RX─RY─  ...
q5:      ─RX─RY─RZ─────────X─o─RX─RY─RZ─────────X─o─RX─RY─RZ─────────X─o─RX─RY─  ...

q0: ...  RZ─o──────────X─RX─RY─RZ─o──────────X─RX─RY─RZ─o──────────X─RX─RY─RZ─
q1: ...  RZ─X─o────────│─RX─RY─RZ─X─o────────│─RX─RY─RZ─X─o────────│─RX─RY─RZ─
q2: ...  RZ───X─o──────│─RX─RY─RZ───X─o──────│─RX─RY─RZ───X─o──────│─RX─RY─RZ─
q3: ...  RZ─────X─o────│─RX─RY─RZ─────X─o────│─RX─RY─RZ─────X─o────│──────────
q4: ...  RZ───────X─o──│─RX─RY─RZ───────X─o──│─RX─RY─RZ───────X─o──│──────────
q5: ...  RZ─────────X─o─RX─RY─RZ─────────X─o─RX─RY─RZ─────────X─o──────────
```

parametric quantum circuit ansatz





description of the quantum noise and quantum error correction

a crucial issues still to be solved ...

S.Bordoni et al, Particles 2023, 1, 1–15

DEVELOPING IN PRACTICE A QML MODEL

- several software frameworks and libraries available:
  - Qiskit (IBM)
  - Cirq (Google)
  - Ocean (D-Wave)
  - Pennylane (Xanadu)
  - Qibo (Open source project)
  - Q# (Microsoft)
  - Forest (Rigetti)
  - …

- allow to define quantum circuits, simulate them, optimise for implementation on quantum hw, …
- Provide a python library for differentiable programming of quantum computers, making possible to train a quantum circuit in a similar way as an artificial neural network …

PENNYLANE FRAMEWORK

- supports execution and training of quantum programs on various backends, making ML frameworks like numpy, pytorch, tensor flow, "quantum-aware"
- allows to run quantum circuits on different simulators or on real hardware devices without making any changes
- can interface with external libraries & quantum hardware (Qiskit, Forest, Cirq. Strawberry Fields)

```
import pennylane as qml
from pennylane import numpy as np

# create a quantum device
dev1 = qml.device("default.qubit", wires=1)

@qml.qnode(dev1)
def circuit(phi1, phi2):
  # a quantum node
  qml.RX(phi1, wires=0)
  qml.RY(phi2, wires=0)
  return qml.expval(qml.PauliZ(0))

def cost(x, y):
  # classical processing
  return np.sin(np.abs(circuit(x, y))) - 1

# calculate the gradient
dcost = qml.grad(cost, argnum=[0, 1])
```

a parametric quantum circuit in pennylane

define the quantum device (quantum node), with 1 wire (eg 1 qubit)

define the circuit:

- a unitary transformation: $U = RY(\phi_2)RX(\phi_1)$
- a measurement: expectation value (qml.expval) over the observable PauliZ (σ_z) operator on wire 0

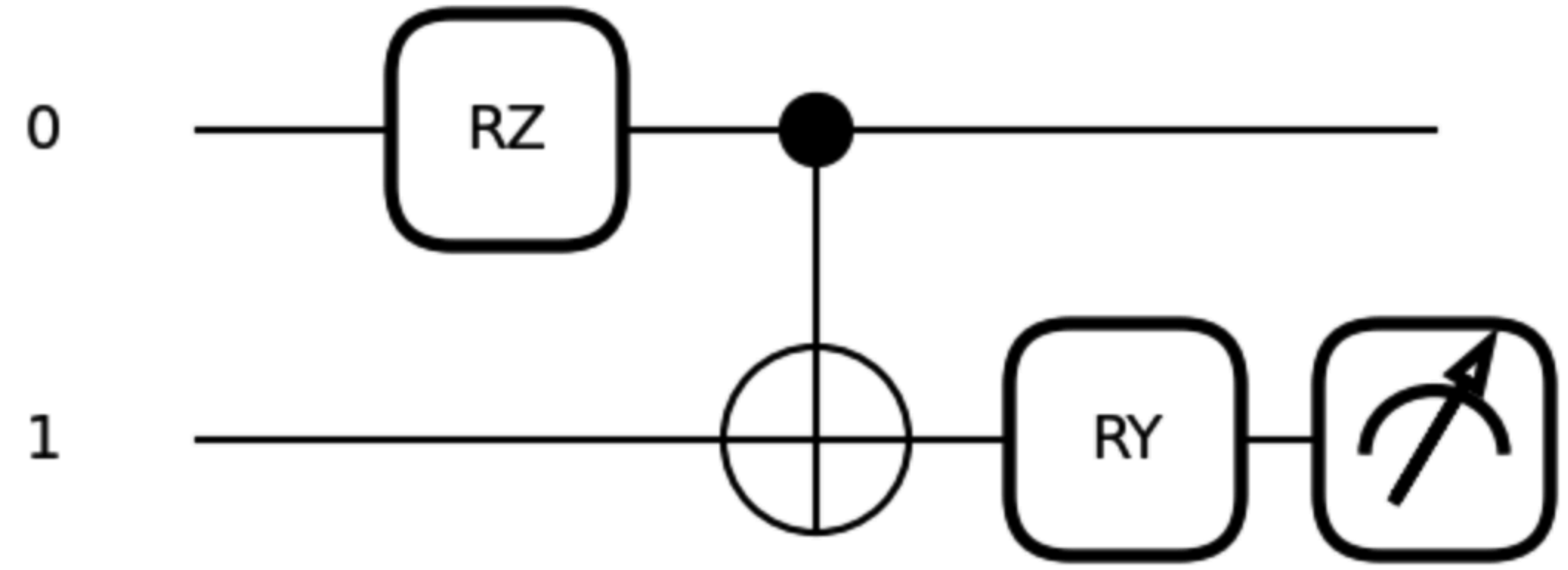
define a loss function (a classical one)

compute the gradient of the cost function

- another example with 2 qubits

```
[1]  !pip install pennylane
```

```
[3]  import pennylane as qml

     dev = qml.device('default.qubit', wires=2)

     @qml.qnode(dev, interface='torch')
     def circuit(x):
         qml.RZ(x, wires=0)
         qml.CNOT(wires=[0,1])
         qml.RY(x, wires=1)
         return qml.expval(qml.PauliZ(1))

     result = circuit(0.543)

     import matplotlib.pyplot as plt

     qml.drawer.use_style("black_white")
     fig, ax = qml.draw_mpl(circuit)(0.543)
     plt.show()
```



TRAIN VARIATIONAL QUANTUM CIRCUITS LIKE ANN

```python
import torch
from torch.autograd import Variable

data = torch.tensor([(0., 0.), (0.1, 0.1), (0.2, 0.2)])


def model(phi, x=None):
    return x*phi
```

```python
def loss(a, b):
    return torch.abs(a - b) ** 2

def av_loss(phi):
    c = 0
    for x, y in data:
        c += loss(model(phi, x=x), y)
    return c

phi_ = Variable(torch.tensor(0.1), requires_grad=True)
opt = torch.optim.Adam([phi_], lr=0.02)

for i in range(5):
    l = av_loss(phi_)
    l.backward()
    opt.step()
```

```python
from pennylane import *
import torch
from torch.autograd import Variable

data = [(0., 0.), (0.1, 0.1), (0.2, 0.2)]

dev = device('default.qubit', wires=2)

@qnode(dev, interface='torch')
def circuit(phi, x=None):
    templates.AngleEmbedding(features=[x], wires=[0])
    templates.BasicEntanglerLayers(weights=phi, wires=[0, 1])
    return expval(PauliZ(wires=[1]))
```

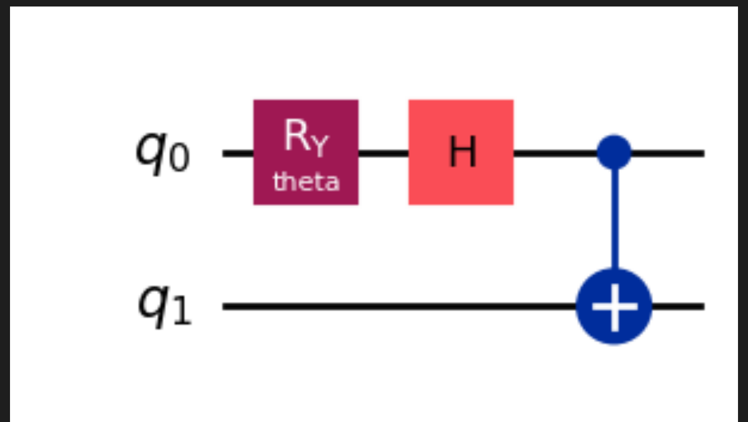
```python
def loss(a, b):
    return torch.abs(a - b) ** 2

def av_loss(phi):
    c = 0
    for x, y in data:
        c += loss(circuit(phi, x=x), y)
    return c

phi_ = Variable(torch.tensor([[0.1, 0.2],[-0.5, 0.1]]), requires_grad=True)
opt = torch.optim.Adam([phi_], lr=0.02)

for i in range(5):
    l = av_loss(phi_)
    l.backward()
    opt.step()
```

QISKIT [1]

- Is a generic term referring to a collection of software developed by IBM for executing programs on quantum computers
  - Qiskit SDK: open-source SDK for working with quantum computers at the level of circuits, operators, and primitives
  - Qiskit Runtime Service: cloud-based service for executing quantum computations on IBM Quantum hardware
  - Qiskit Aer: high-performance quantum computing simulators with realistic noise models
- WARNING: Qiskit is updated frequently; the versions are quite different from each other!
  - E.g., the ML library has been deprecated for about one year

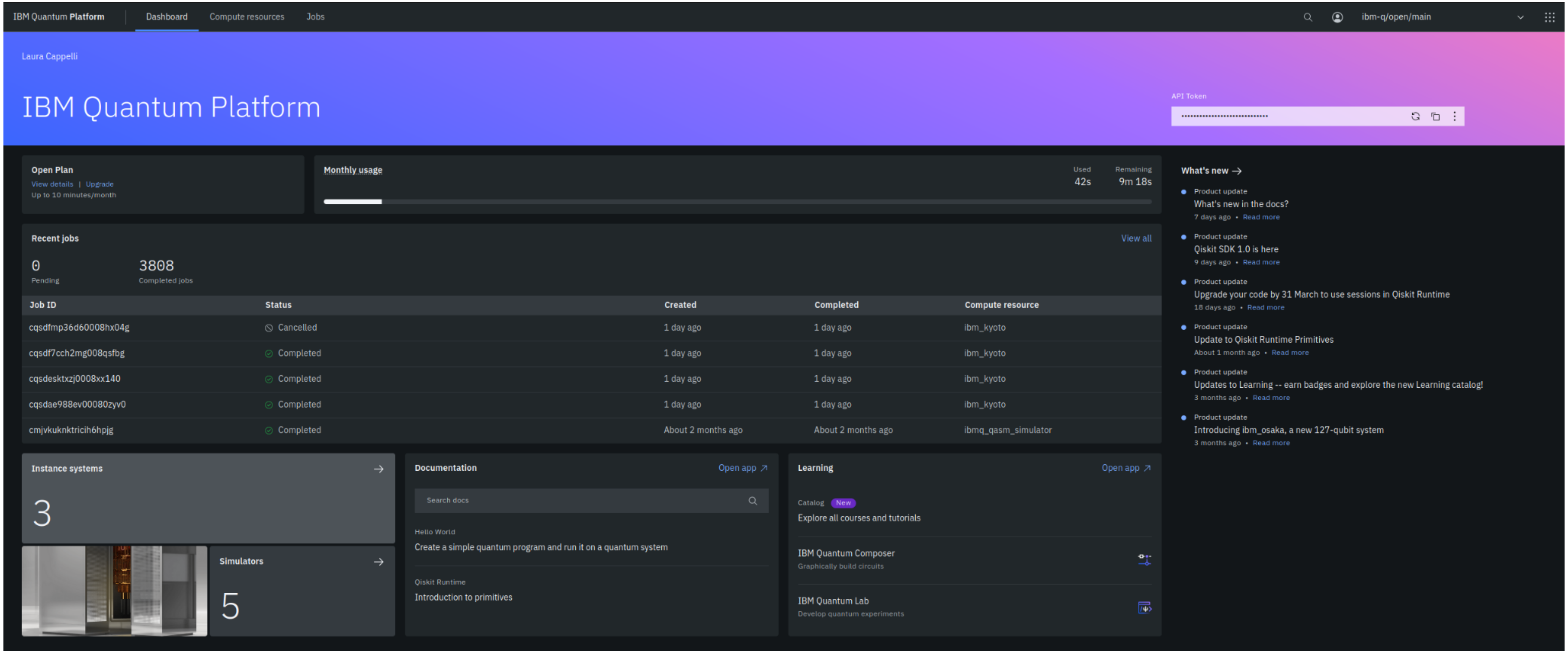
a parametric quantum circuit in Qiskit

```python
from qiskit import QuantumCircuit
from qiskit.circuit import Parameter
from qiskit.quantum_info import SparsePauliOp
from qiskit.primitives import StatevectorEstimator
import numpy as np

# circuit definition
qc = QuantumCircuit(2)
qc.ry(Parameter('theta'), 0)
qc.h(0)
qc.cx(0,1)
qc.draw("mpl", style="iqp")

# observable(s) whose expected values you want to compute
observable = SparsePauliOp(["ZI"])

# value(s) for the circuit parameter(s)
parameter_values = [[0], [np.pi/6], [np.pi/2]]

# select a backend
estimator = StatevectorEstimator()

# run the circuit
job = estimator.run([(qc, observable, parameter_values)])
result = job.result()
print(f" > Expectation value: {result[0].data.evs}")
```



QISKIT [2]

- IBM provides an open plan of 10 minutes/month use of its quantum hardware
  - You can use the API Token to submit jobs to remote hardware
  - Queue time can be long...



Finanziato dall'Unione europea NextGenerationEU

Ministero dell'Università e della Ricerca

Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA

ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

This activity is partially supported by ICSC – Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing, funded by European Union – NextGenerationEU