Workshop on Computation in Experimental Particle Physics

The prospect of quantum machine learning algorithms in High Energy Physics

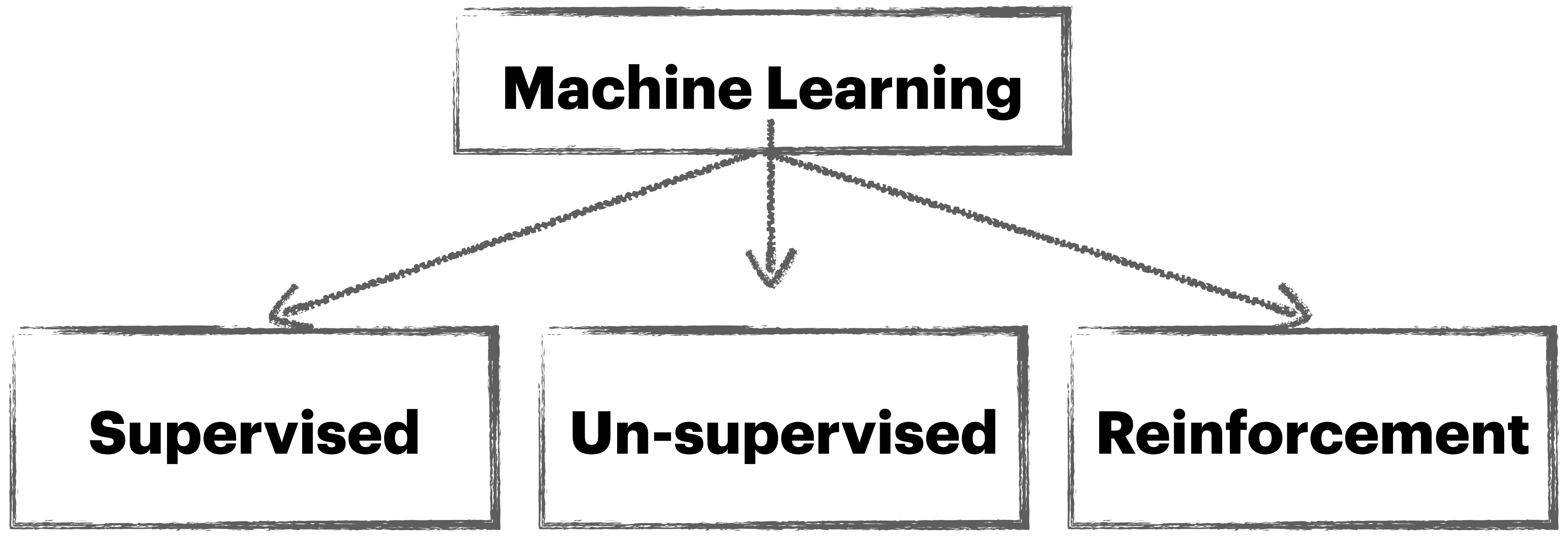
Abdualazem Fadol Mohammed

July 18, 2023

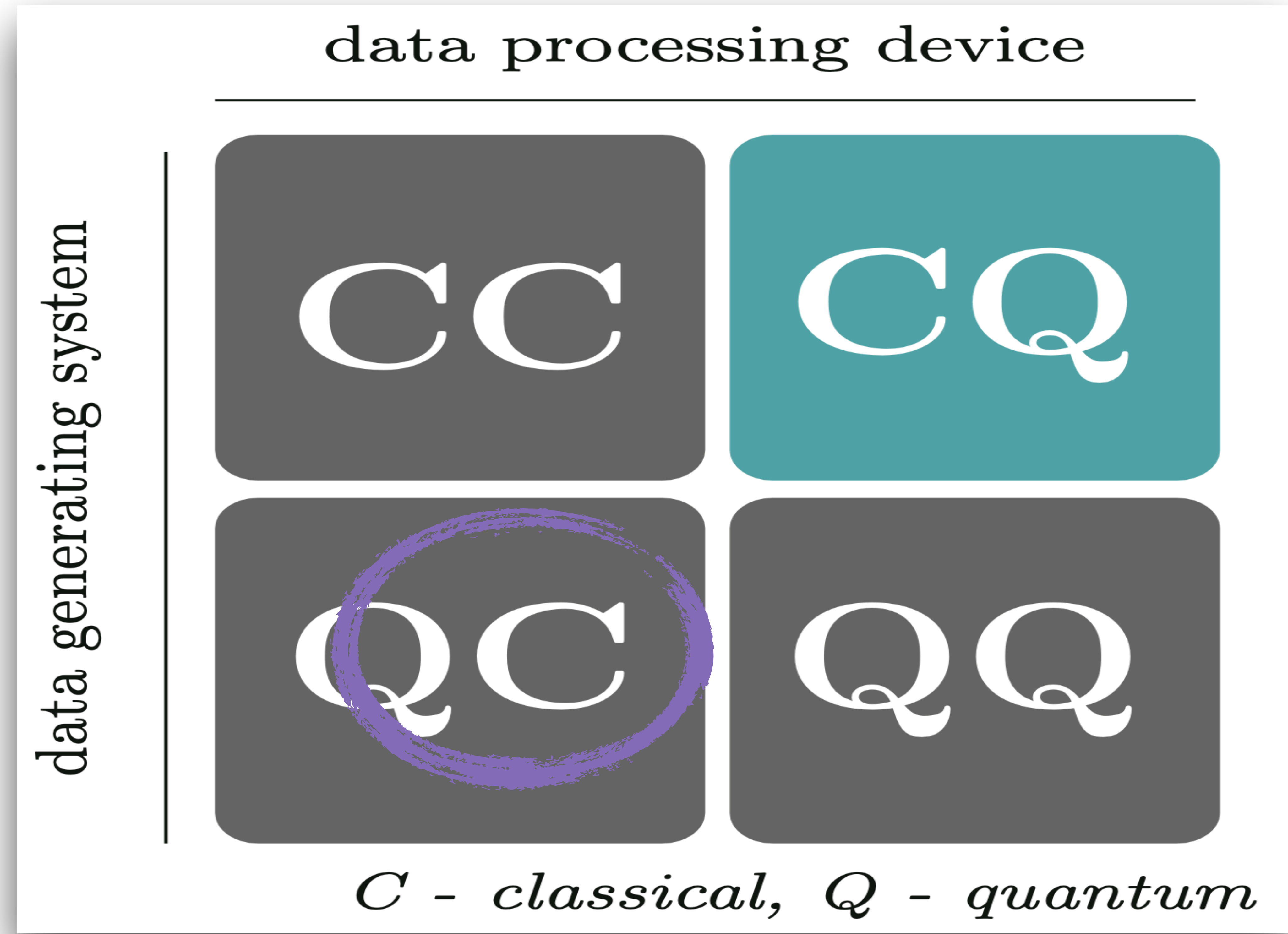
Introduction

Maria Schuld, and Francesco Petruccione. Vol. 17. Berlin: Springer, 2018

- ☐ Classical Machine learning: seeks to find patterns in data.



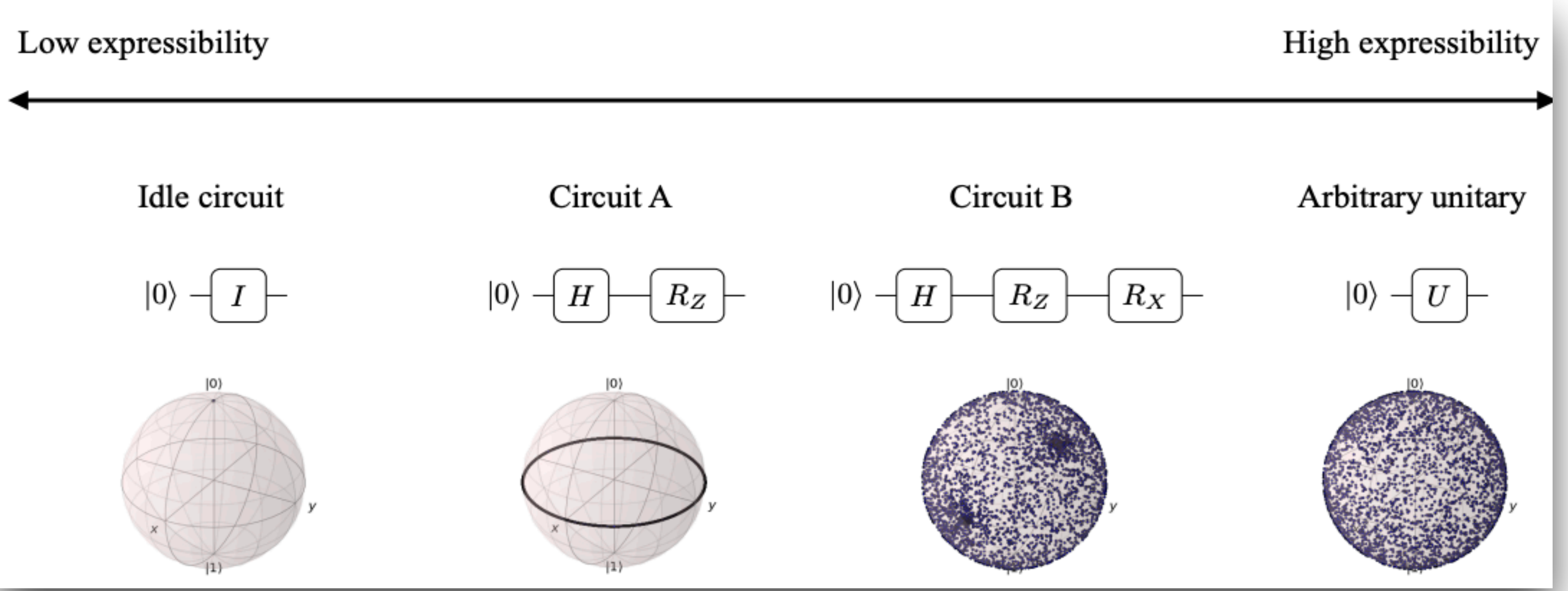
- ☐ Quantum Machine Learning: Based on the following approaches:
  - Weather data is classical (C) or quantum (Q), and
  - an algorithm runs in a classical (C) or quantum (Q) computer
- ☐ Quantum algorithms—an equivalent to classical algorithms:
  - Grover search and amplitude amplification
  - Hybrid Training for Variational Algorithms



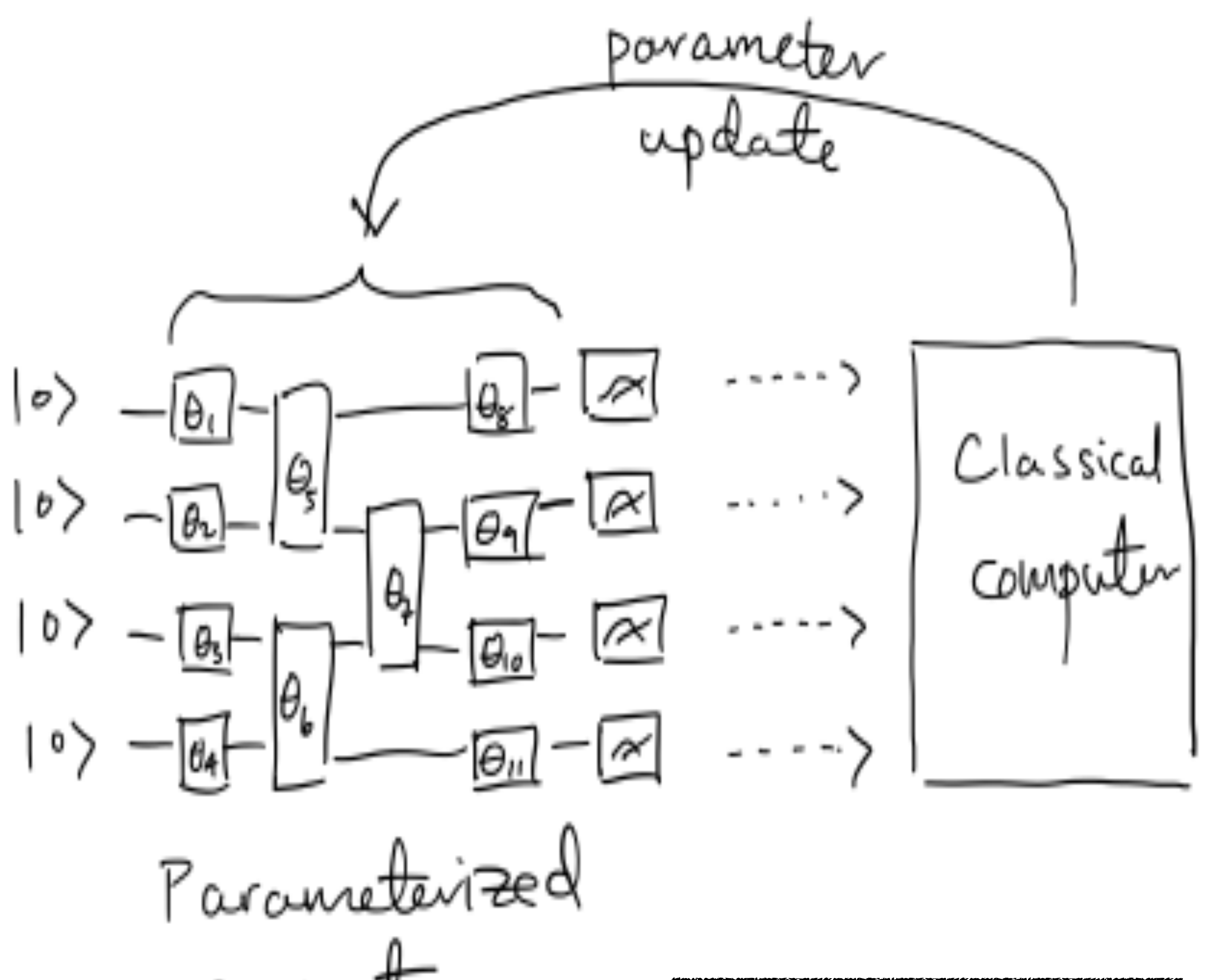
- ☐ Examples of the QC (no viable hardware):
  - QPCA
  - QSVM
  - QClustering

Introduction

- ☐ **Parametrised Quantum Circuits:** useful for short-term quantum devices that can be defined using tunable parameters.
- ☐ Keeping in mind certain properties:
  - Expressibility



  - Entangling capability: the Meyer-Wallach measure
  - Hardware efficiency: qubits connectivity, fidelities ...



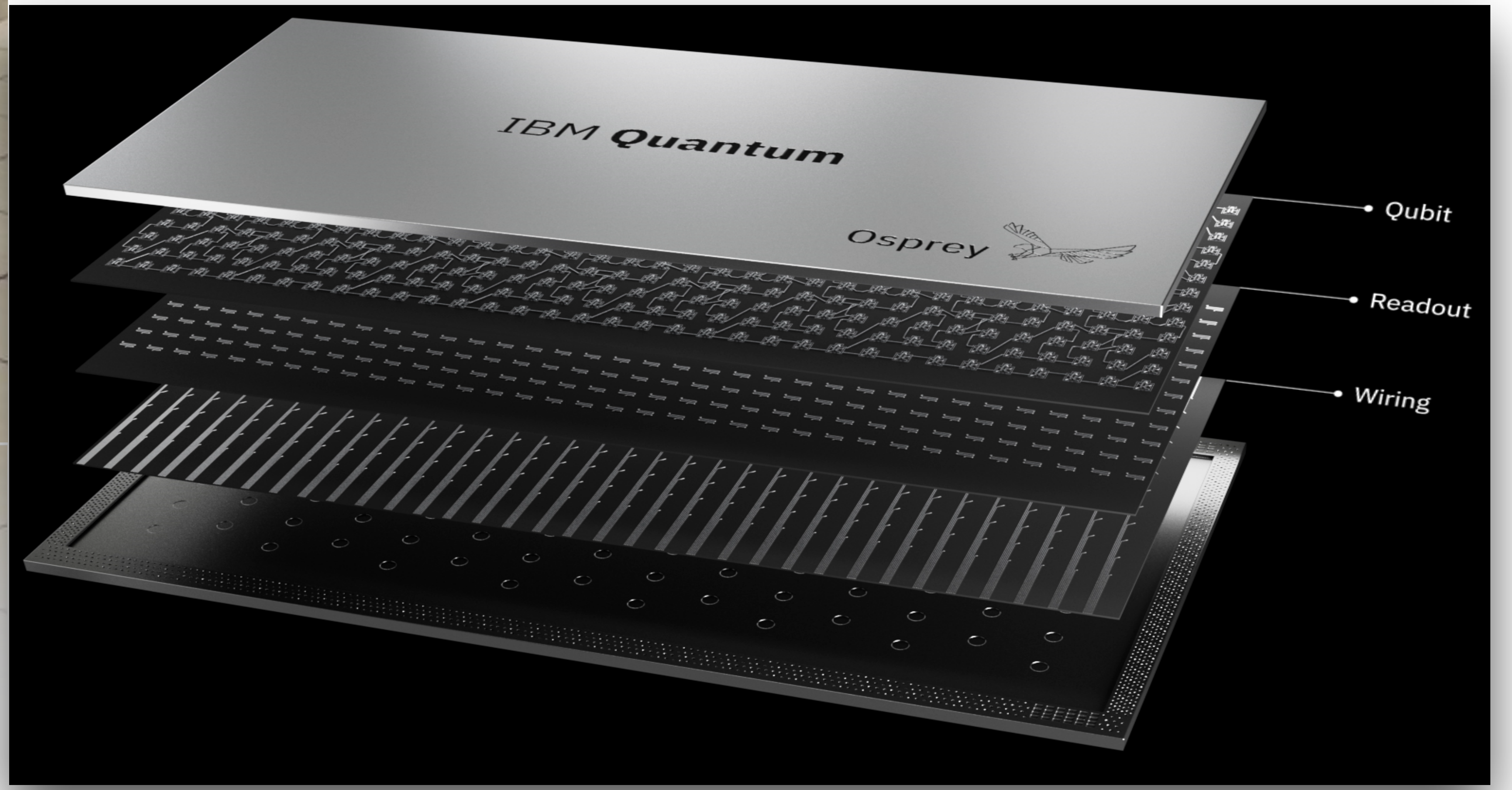
By Hanna Sim

- ☐ For general information about Quantum Computing idea, see Li Teng's talk.

IBM quantum computer

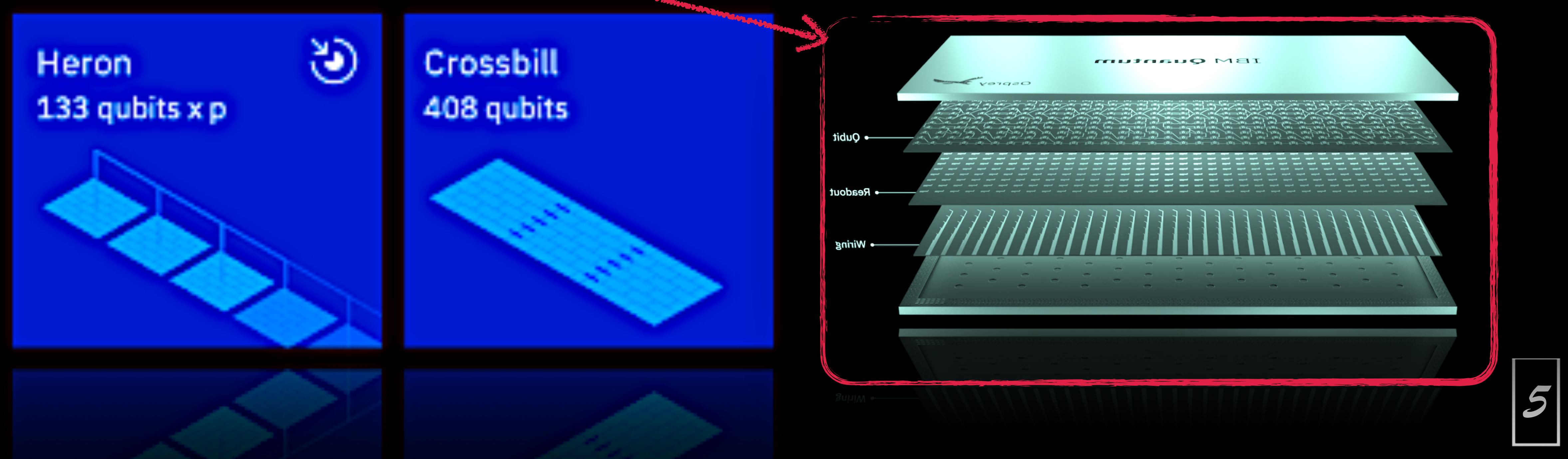
Credited to Thomas Prior for TIME

- ☐ Taking quantum computing out of the lab:
  - NY Computing Data Centre.
  - It provides over 20 computers.
  - Scales the processor's availability.
  - It provides over 20 comput



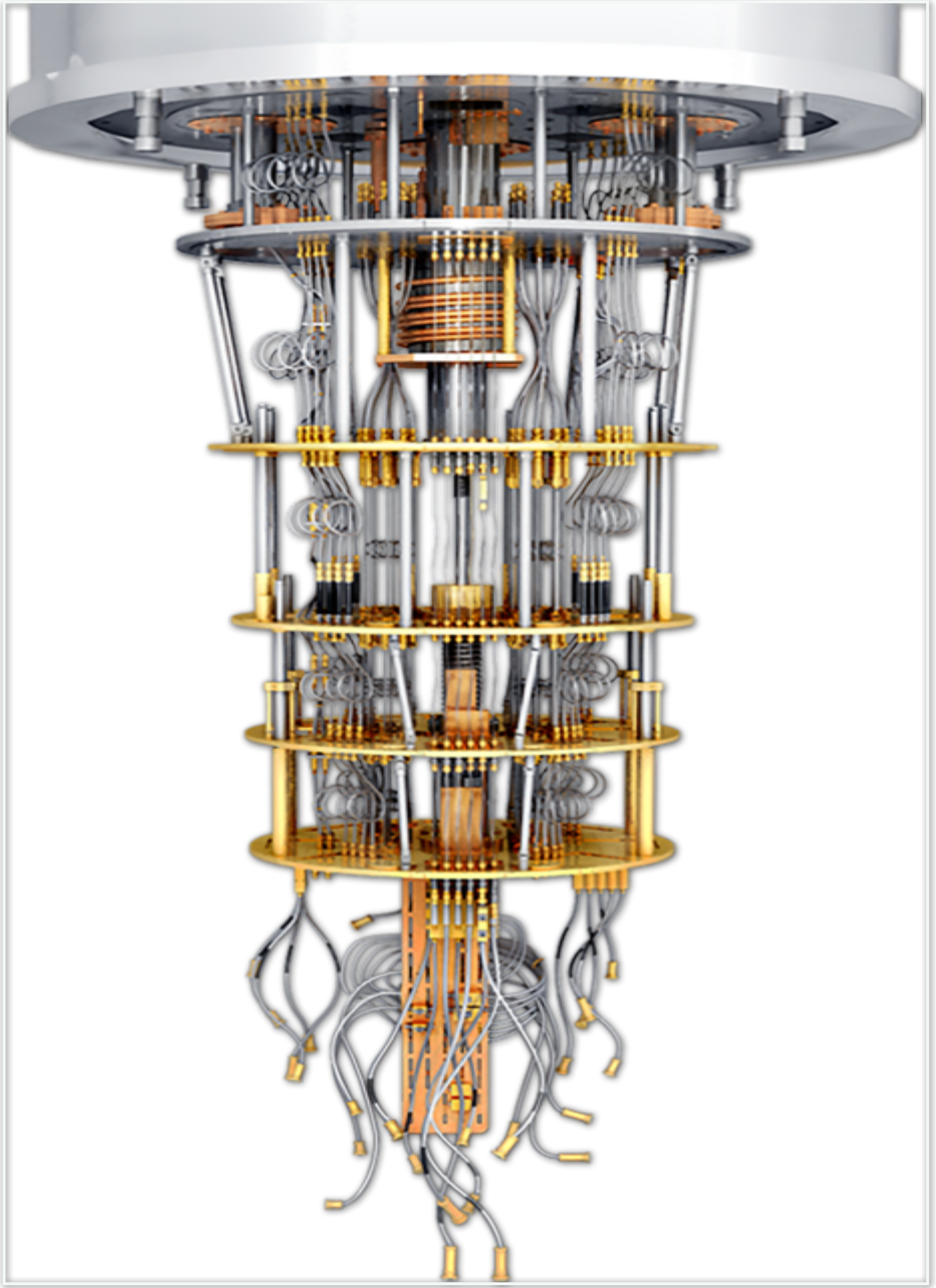
- ☐ IBM provides up to 7 qubits for free with an opportunity to apply for a researcher account with more qubits.

IBM quantum computer roadmap



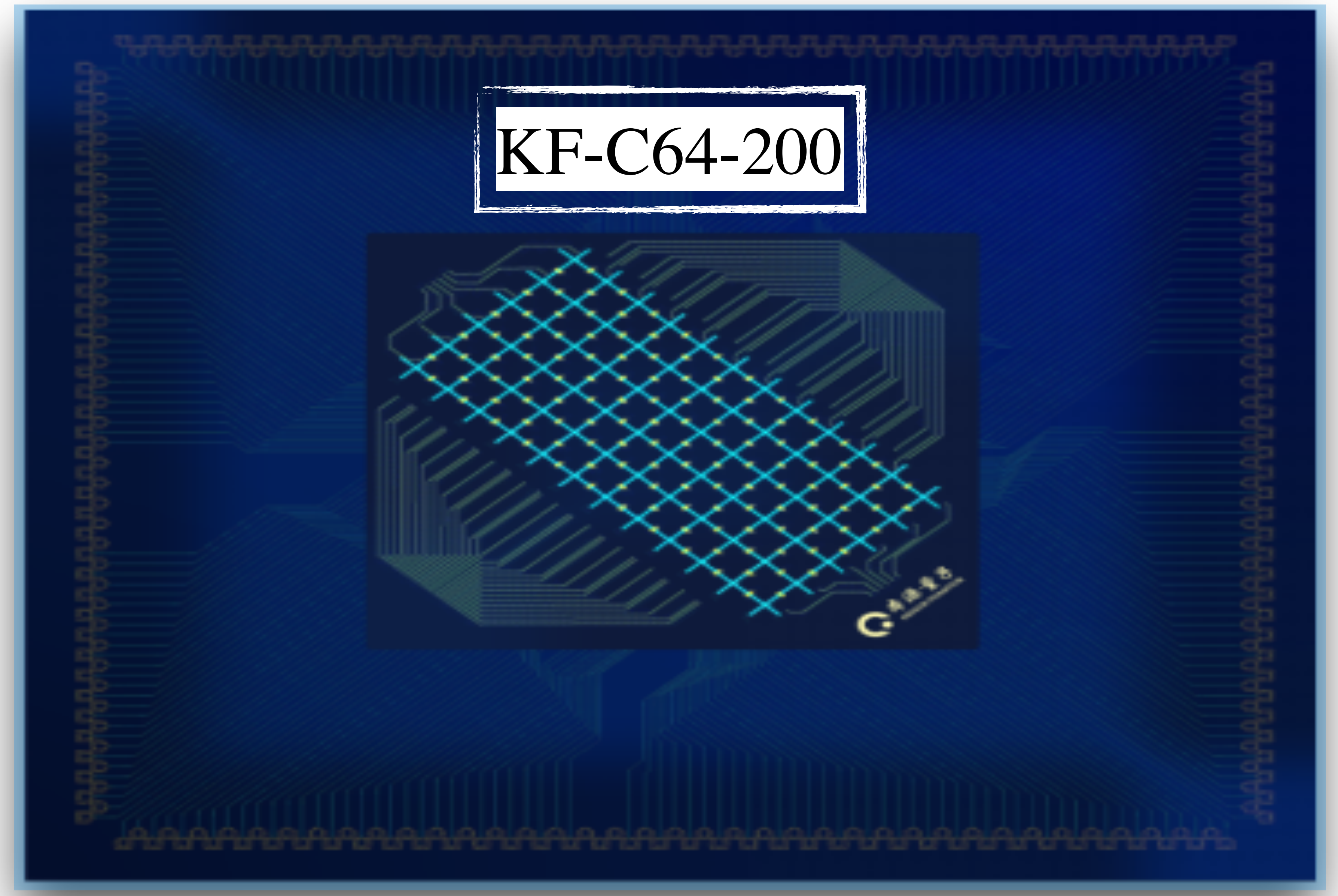
- IBM has ambitious pursuits:
  - 433-qubit IBM Quantum Osprey
  - three times larger than the Eagle processor
  - going up to 10k-100k qubits

Origin quantum computer



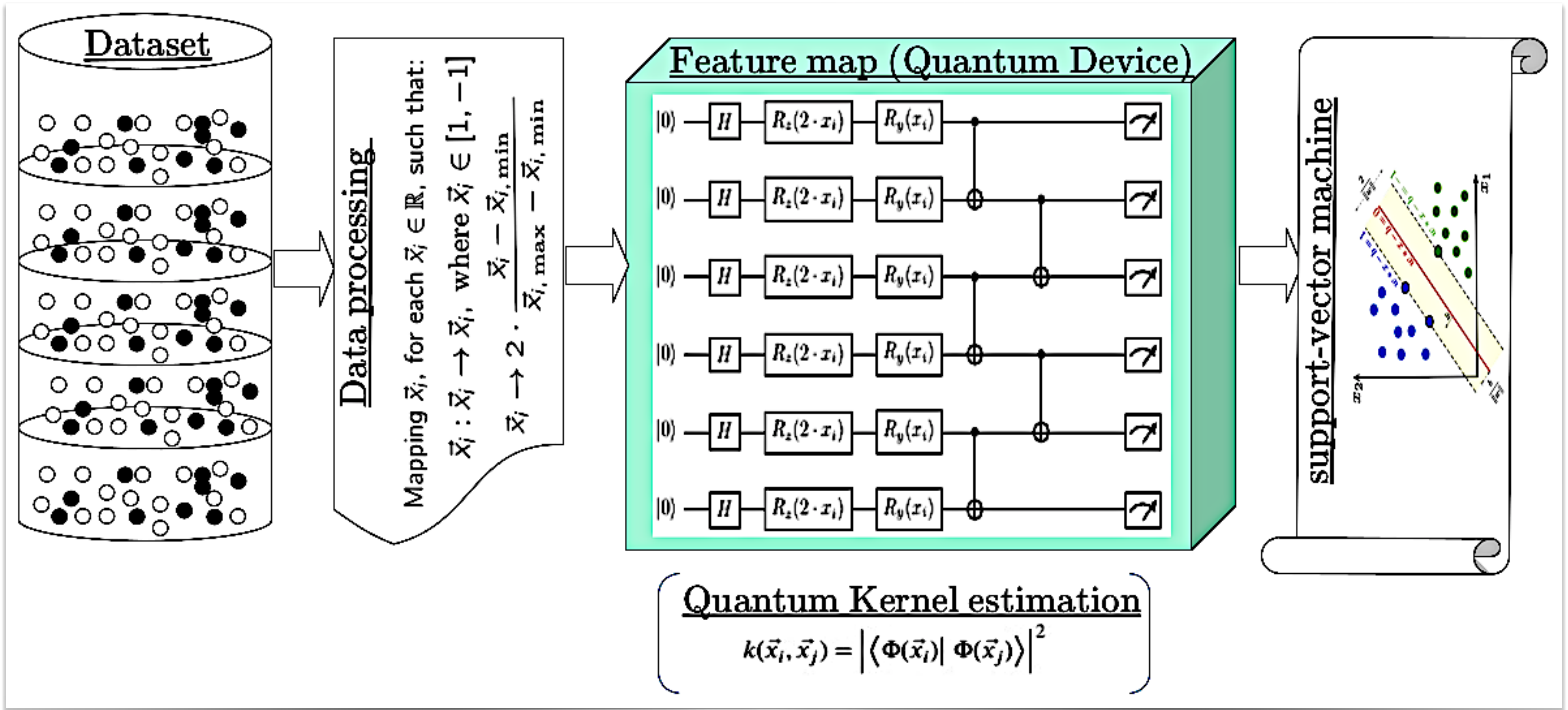
Taken from Origin web page

- ☐ Origin launched 64-qubit QPU:
  - Single-qubit gate fidelity > 99.9 % .
  - Double-qubit gate fidelities > 98 % .
  - Readout fidelities > 96 % .



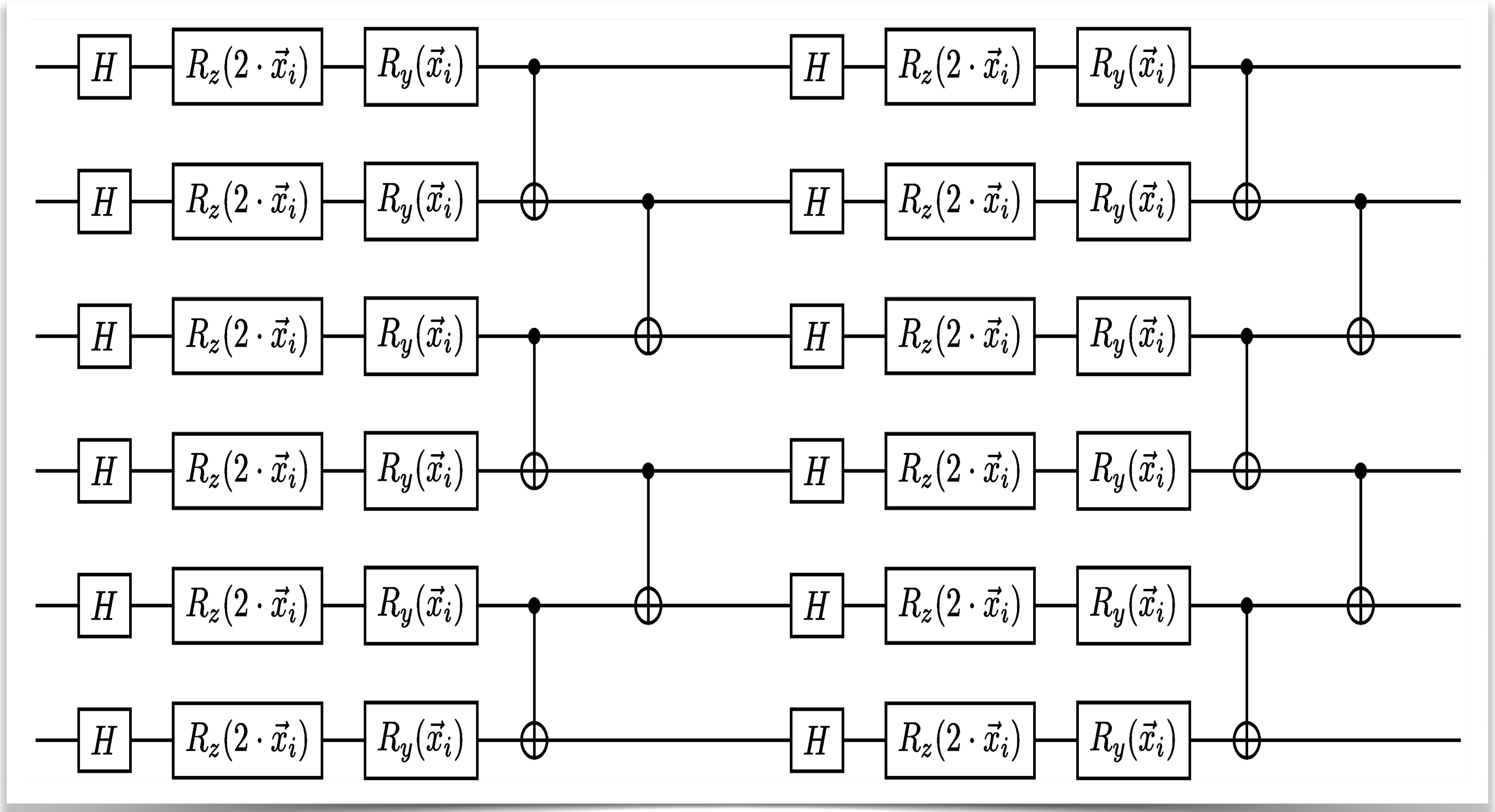
☐ Origin quantum computer provides up to 6 qubits for free. However, another hardware called Quafu provides up to 136 qubits. The Beijing Academy of Quantum Information Sciences maintains it.

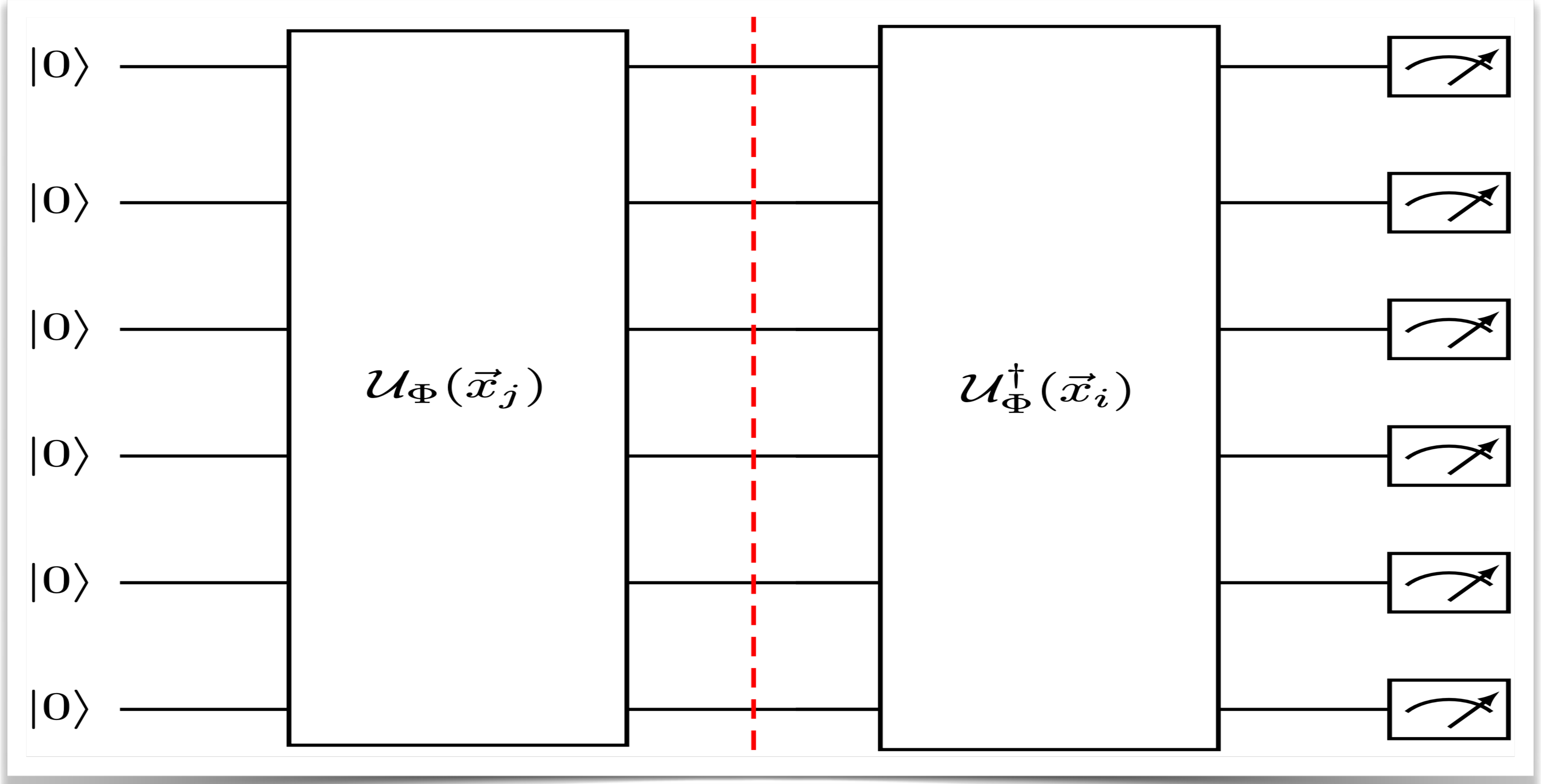
Data encoding and processing



Feature map and quantum kernel estimation

- ☐ The quantum feature map dictates the kernel:
  - Single-qubit Hadamard gate
  - Single-qubit rotation gates $R_z(x)$ and $R_y(x)$
  - Two-qubit CNOT entangling gates
  - Two identical layers (depth)





- ☐ The quantum support vector kernel estimation:

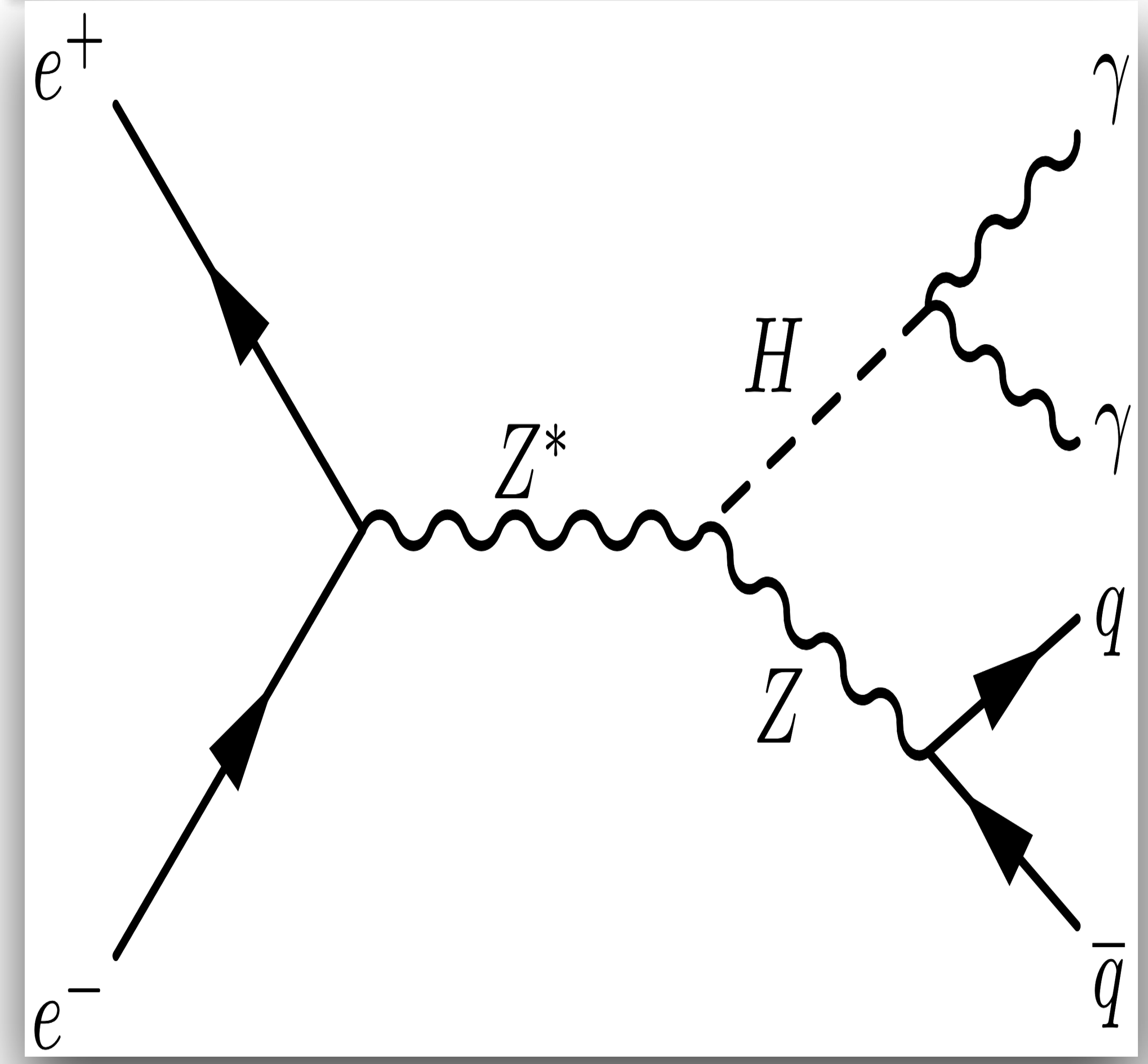
$$k(\vec{x}_j, \vec{x}_j) = \left| \left\langle 0^{\otimes N} \right| \; U^{\dagger}_{\phi(\vec{x}_i)} U_{\phi(\vec{x}_j)} \left| 0^{\otimes N} \right\rangle \right|^2$$

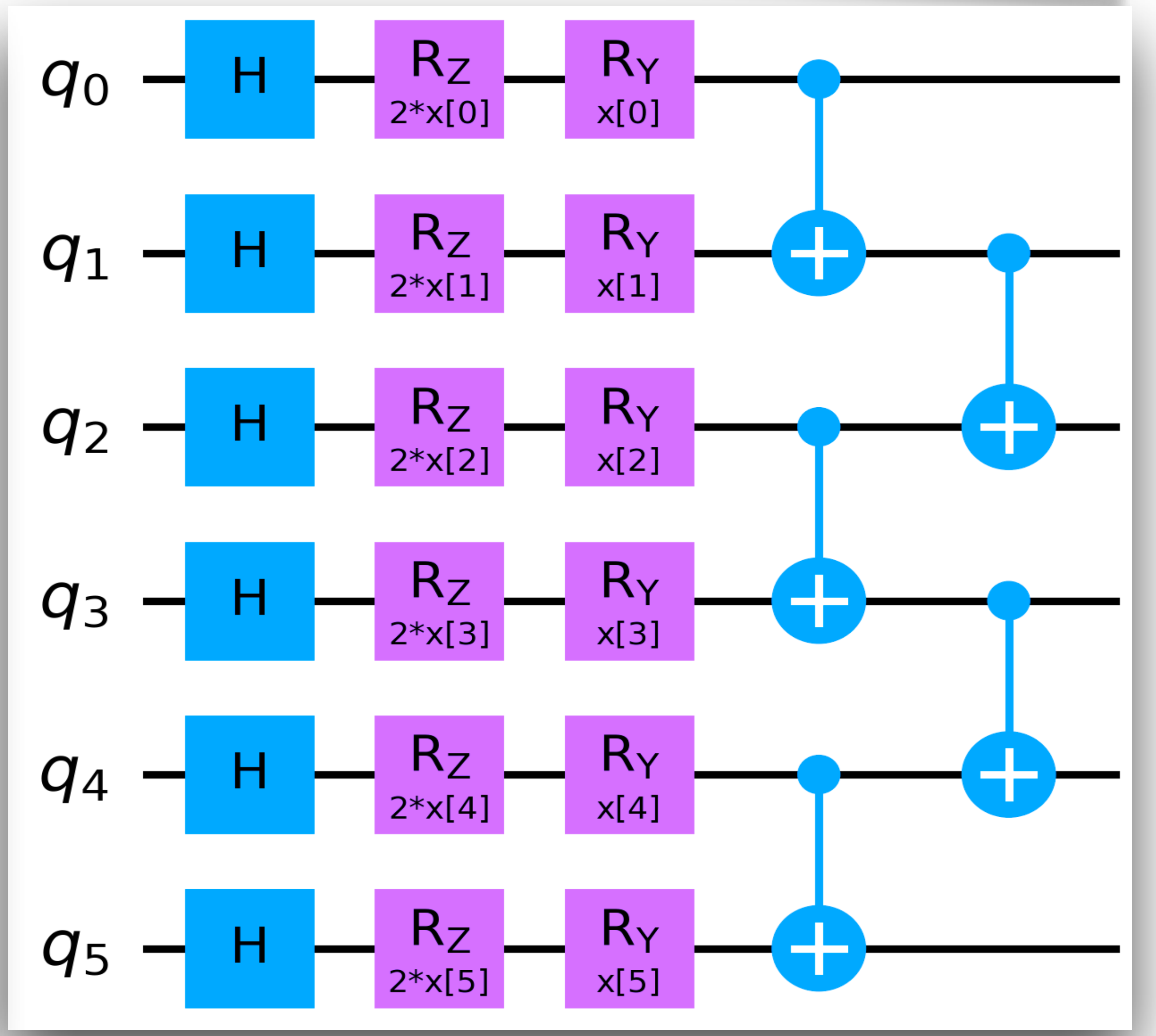
- ☐ $N \equiv$ Six qubits are mapped to six variables.
- ☐ The expectation of each data point w.r.t the rest.

Feature map optimisation

- ☐ The following feature map form was found to work best for the $e^+e^- \to ZH \to \gamma\gamma q\bar{q}$ signal.
- ☐ Five thousand events were used with different rotation combinations $R_y(\vec{x})$ and $R_z(\vec{x})$.
- ☐ The current entanglements were the best among others, such as full entanglement.
- ☐ The quantum circuit is repeated twice to achieve better entanglements between qubits.
- ☐ The area under the curve (AUC) decides the best rotation and entanglements.

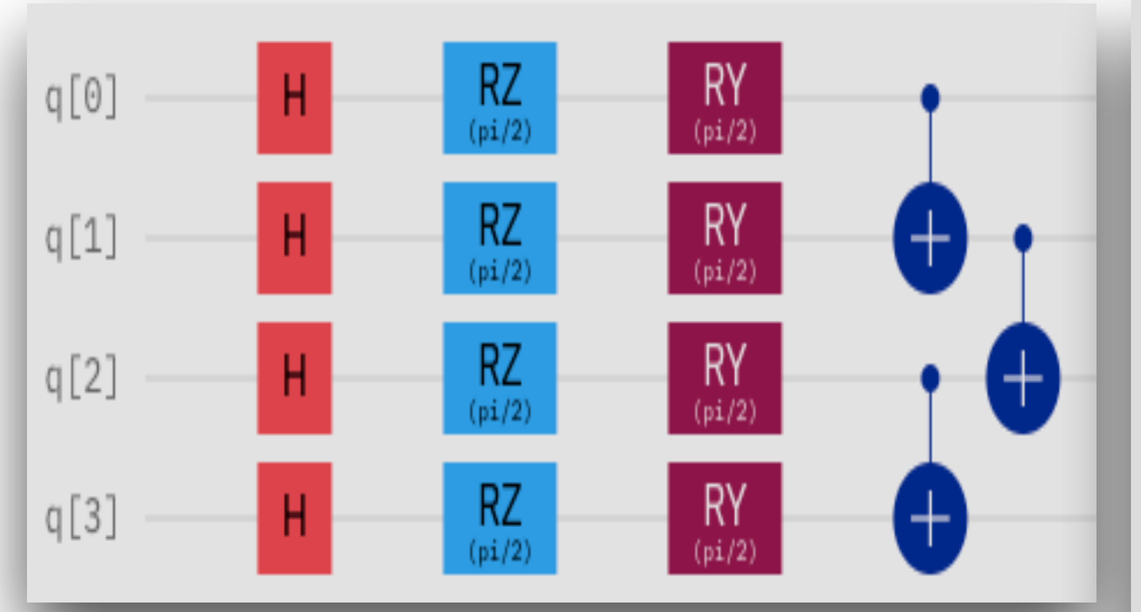
Rotation	Depth	Events	Best AUC	Variation
$R_z(2\cdot\vec{x}_i)+R_y(\vec{x}_i)$	2	5000	0.935	0.009
$R_z(\vec{x}_i)+R_y(\vec{x}_i)$			0.933	0.015
$R_y(\vec{x}_i)+R_x(\vec{x}_i)$			0.932	0.015
$R_z(\vec{x}_i)+R_z(\vec{x}_i)$			0.932	0.014
$R_y(\vec{x}_i)$			0.928	0.008
$R_z(\vec{x}_i)$			0.928	0.008

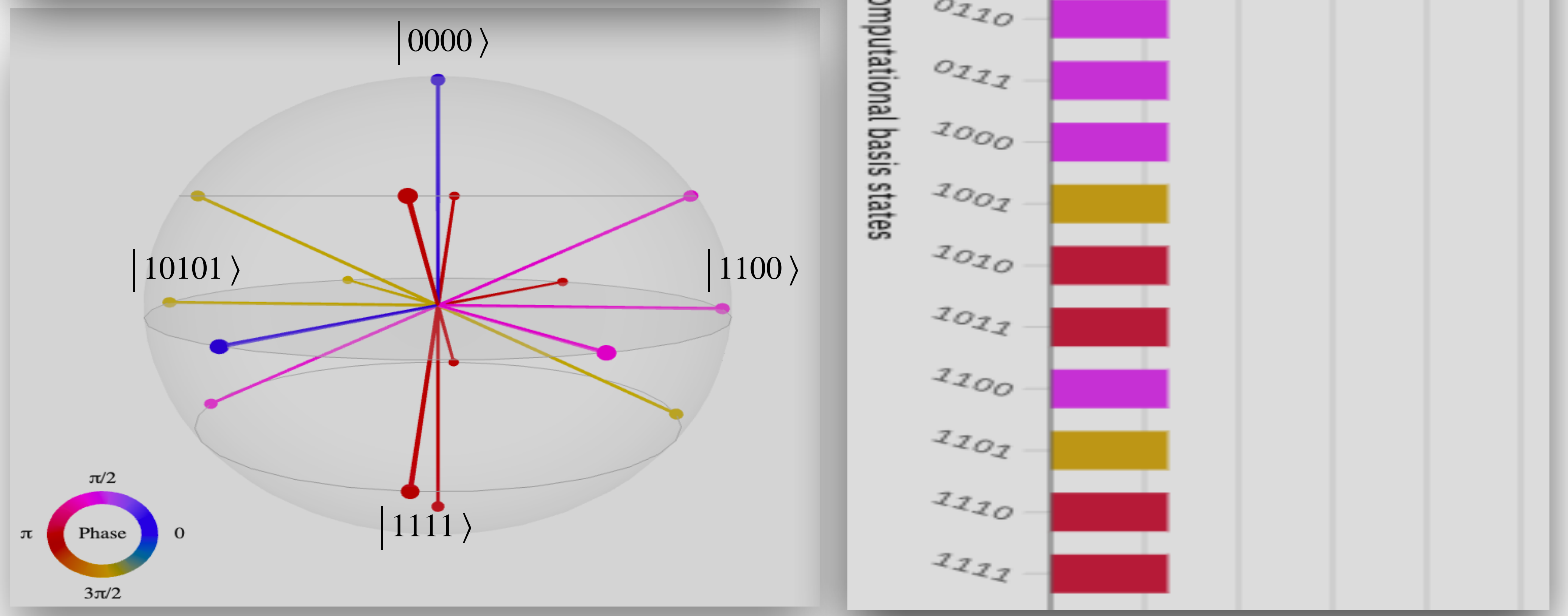




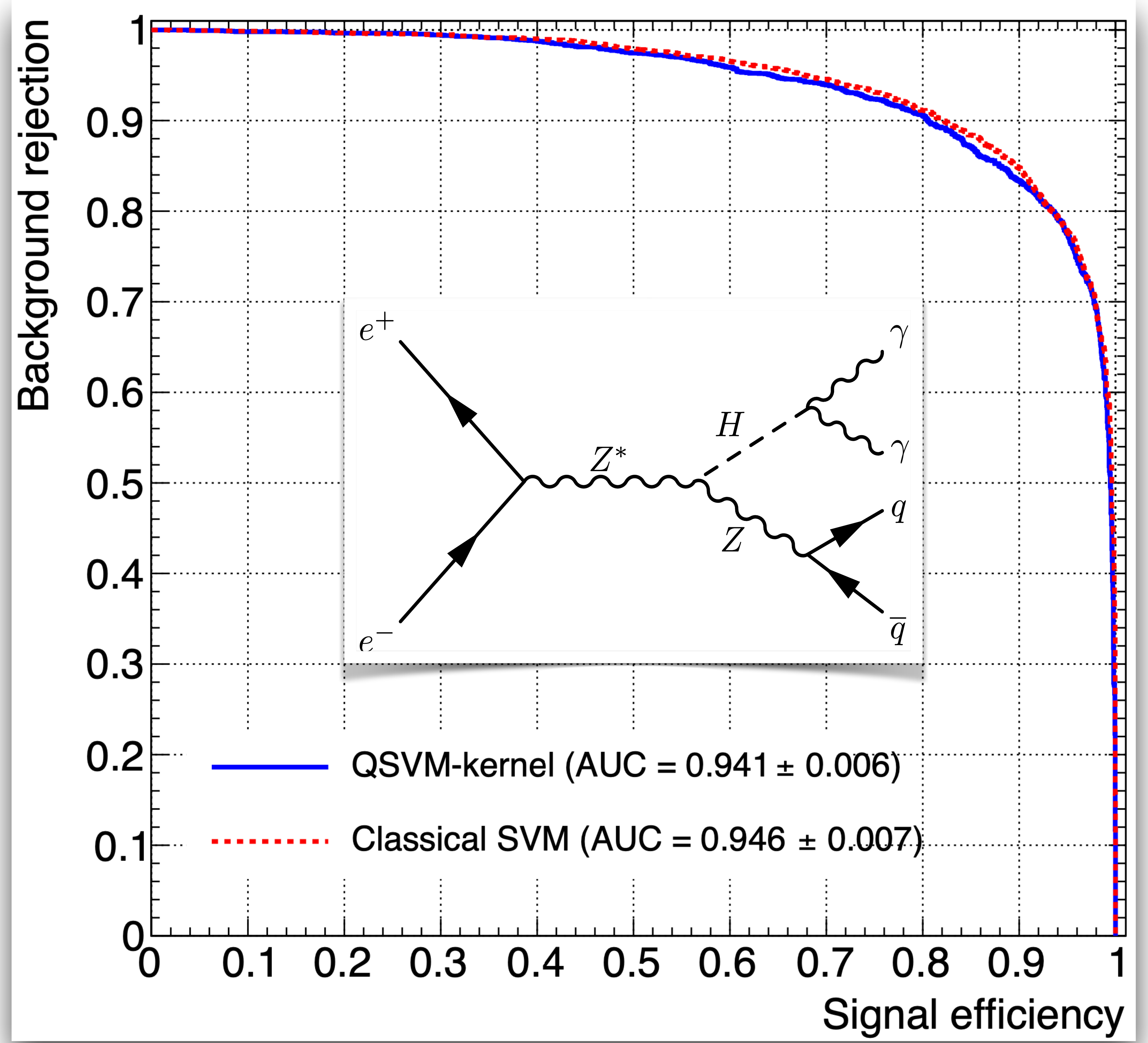
The performance of the quantum simulator

- ☐ The quantum simulator with the following:
  - Statevector Simulator developed by the **Qiskit software package**

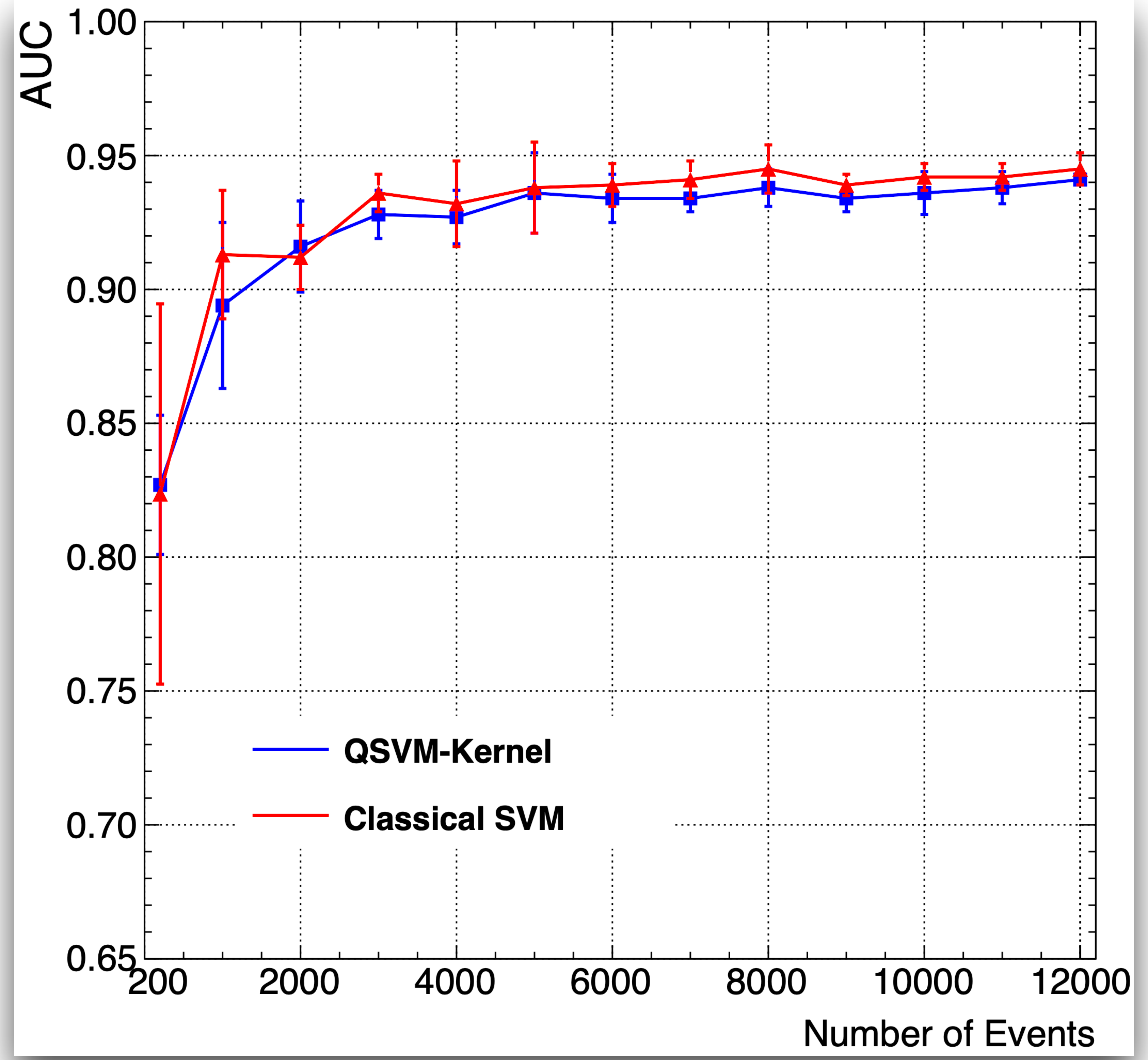


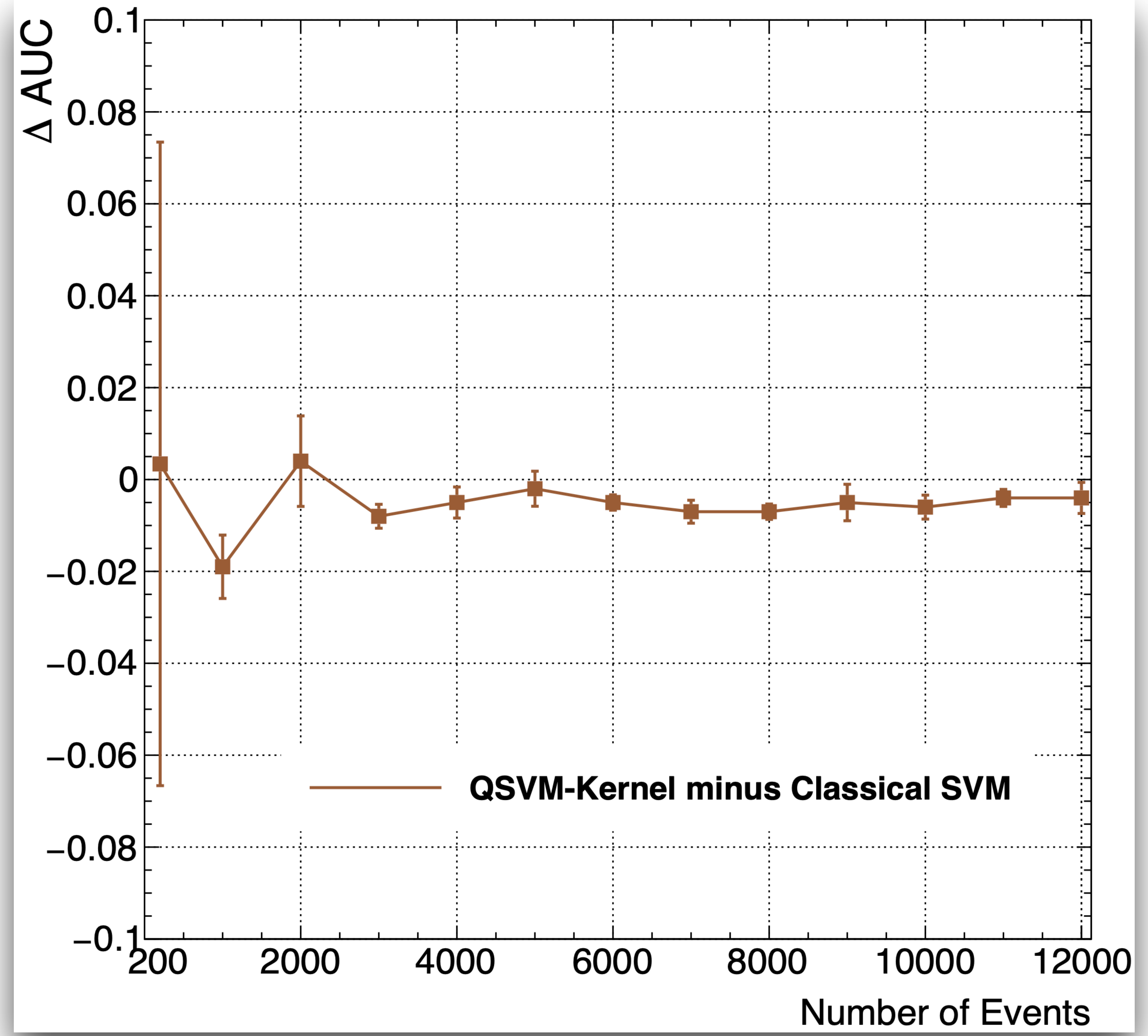


  - Six quantum bits or simply qubits
- ☐ A total of 12000 events were used.



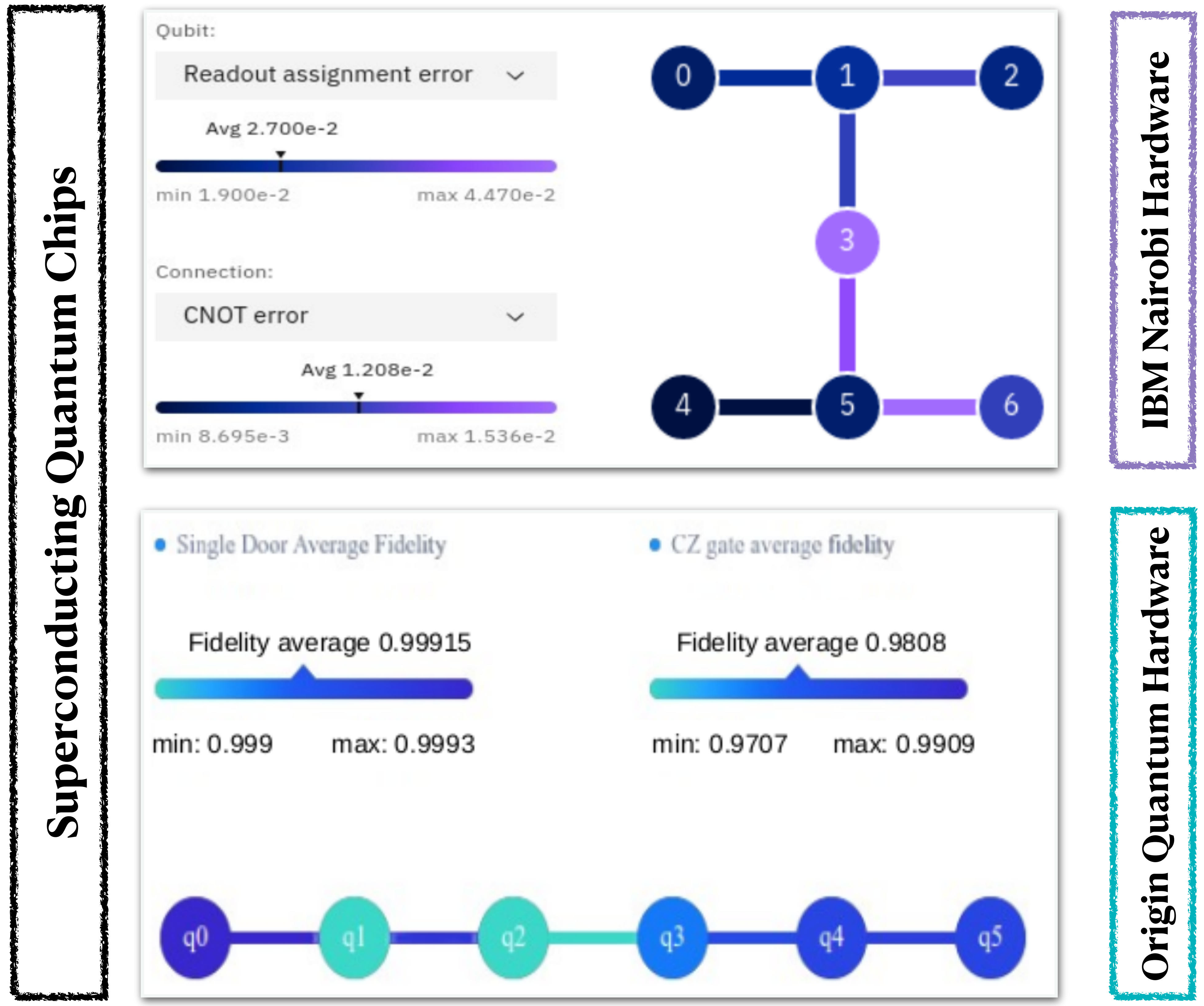
AUC versus the number of events

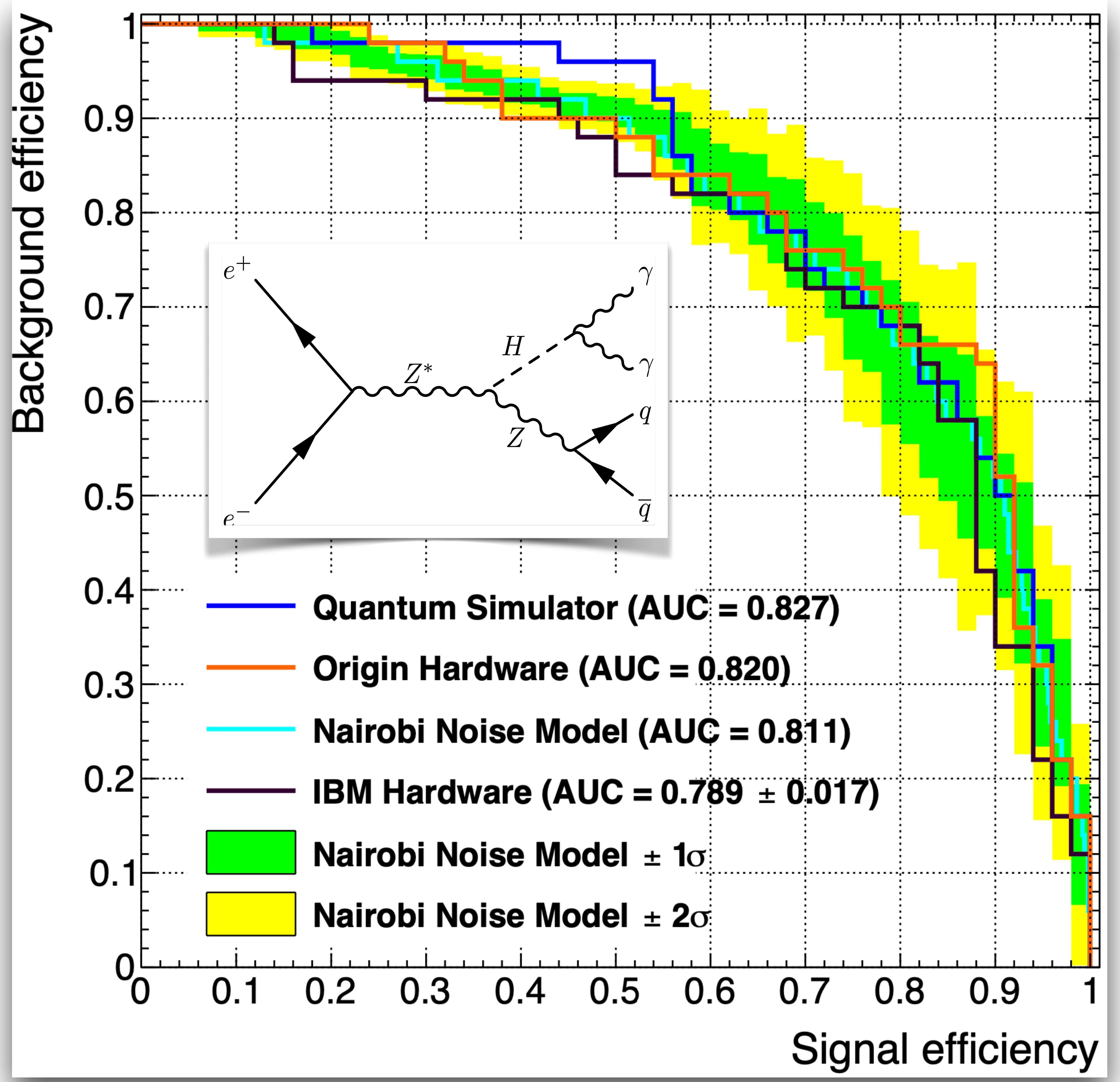




☐ Comparison between the QSVM-kernel and classical SVM classifiers with different numbers of events.

The performance of quantum computers

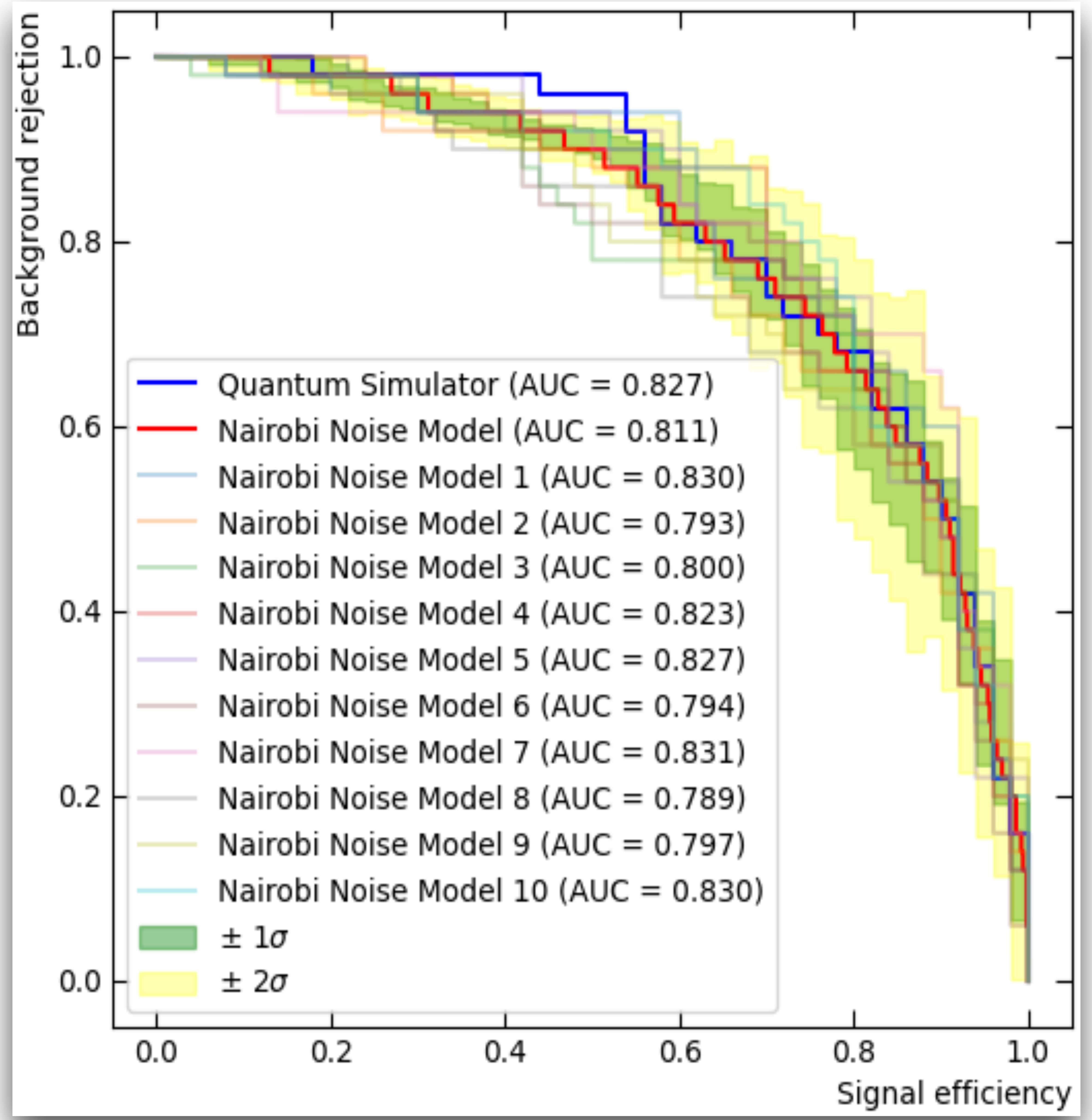




- ☐ Six qubits were used for both quantum hardware.
- ☐ 100 events were used for the training and testing.
- ☐ Comparable performance is observed between the IBM and Origin quantum hardware.

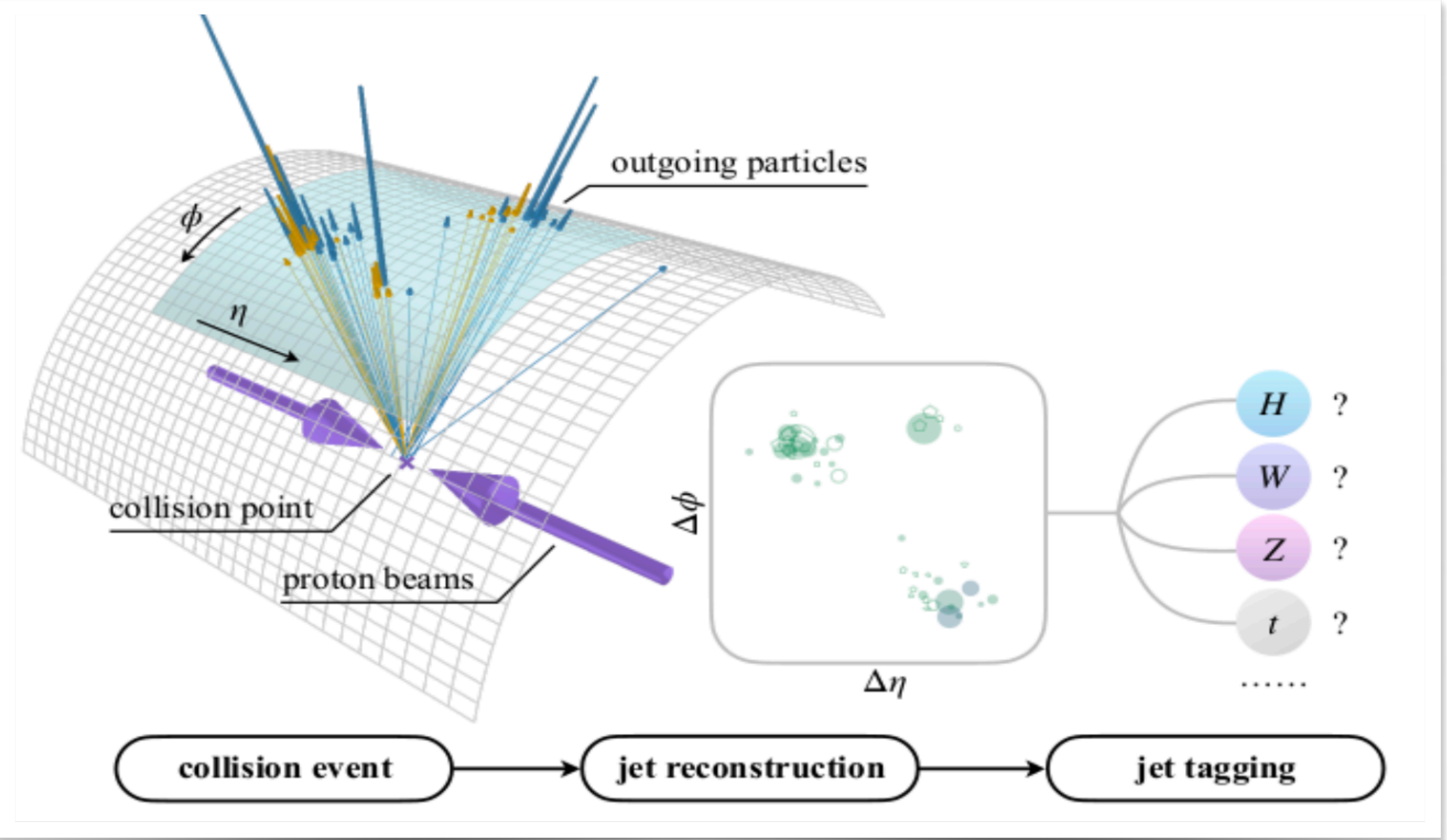
Noise modelling in the IBM Nairobi

- ☐ Noise in quantum computers: Quantum computers are susceptible to noise.
  - An electromagnetic signal coming from a WiFi
  - A disturbance in the Earth's magnetic field
- ☐ The model used automatically generates a simplified noise model for a real device.
- ☐ It takes into account the following:
  - The gate error probability of each basis gate
  - The gate length of each basis gate
  - T_1 and T_2 relaxation time constant
  - The readout error probability
- ☐ The estimated noise in the IBM Nairobi computer is 0.017.

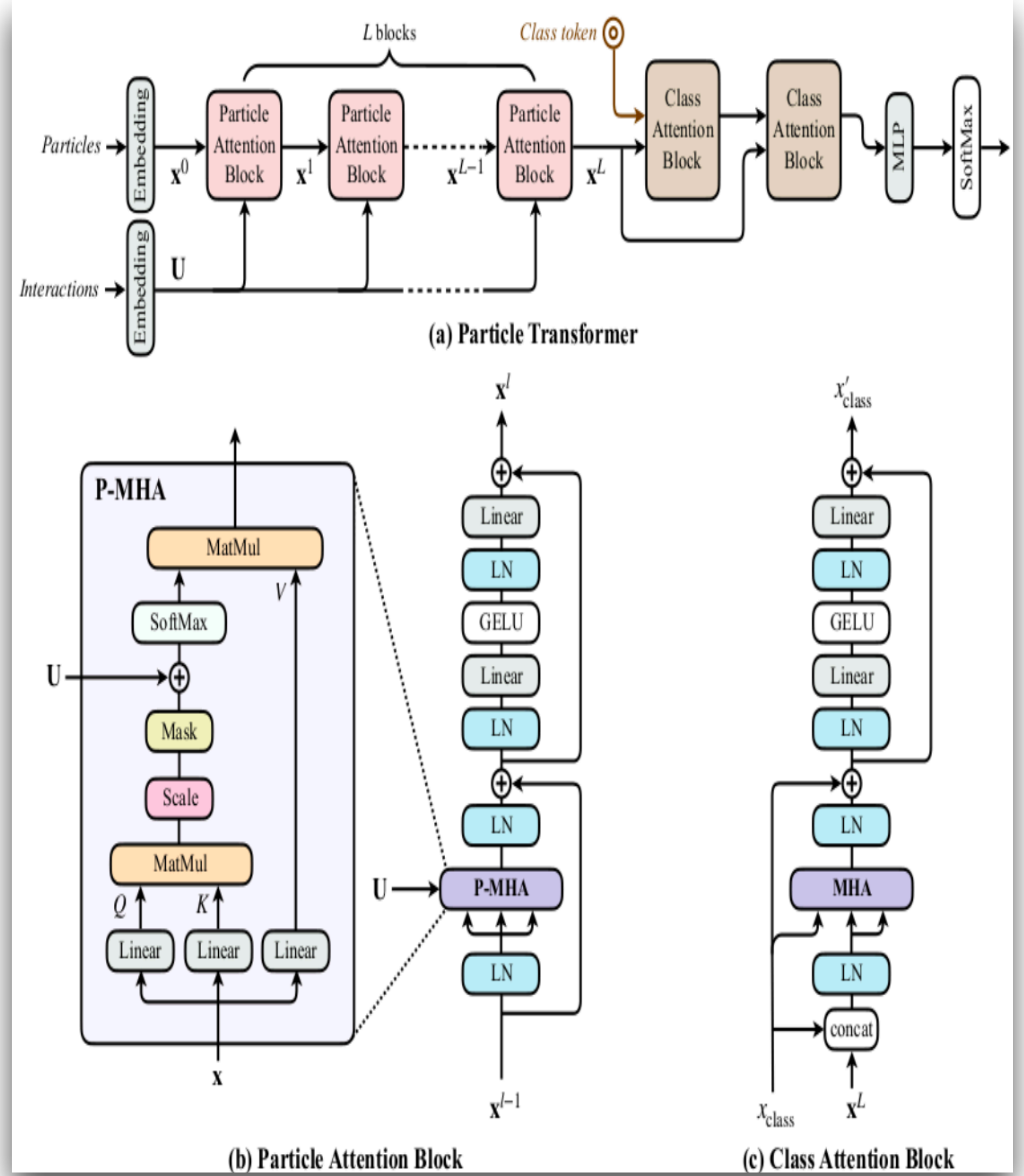


The Particle Transformer

- ☐ Jet tagging classification in particle physics

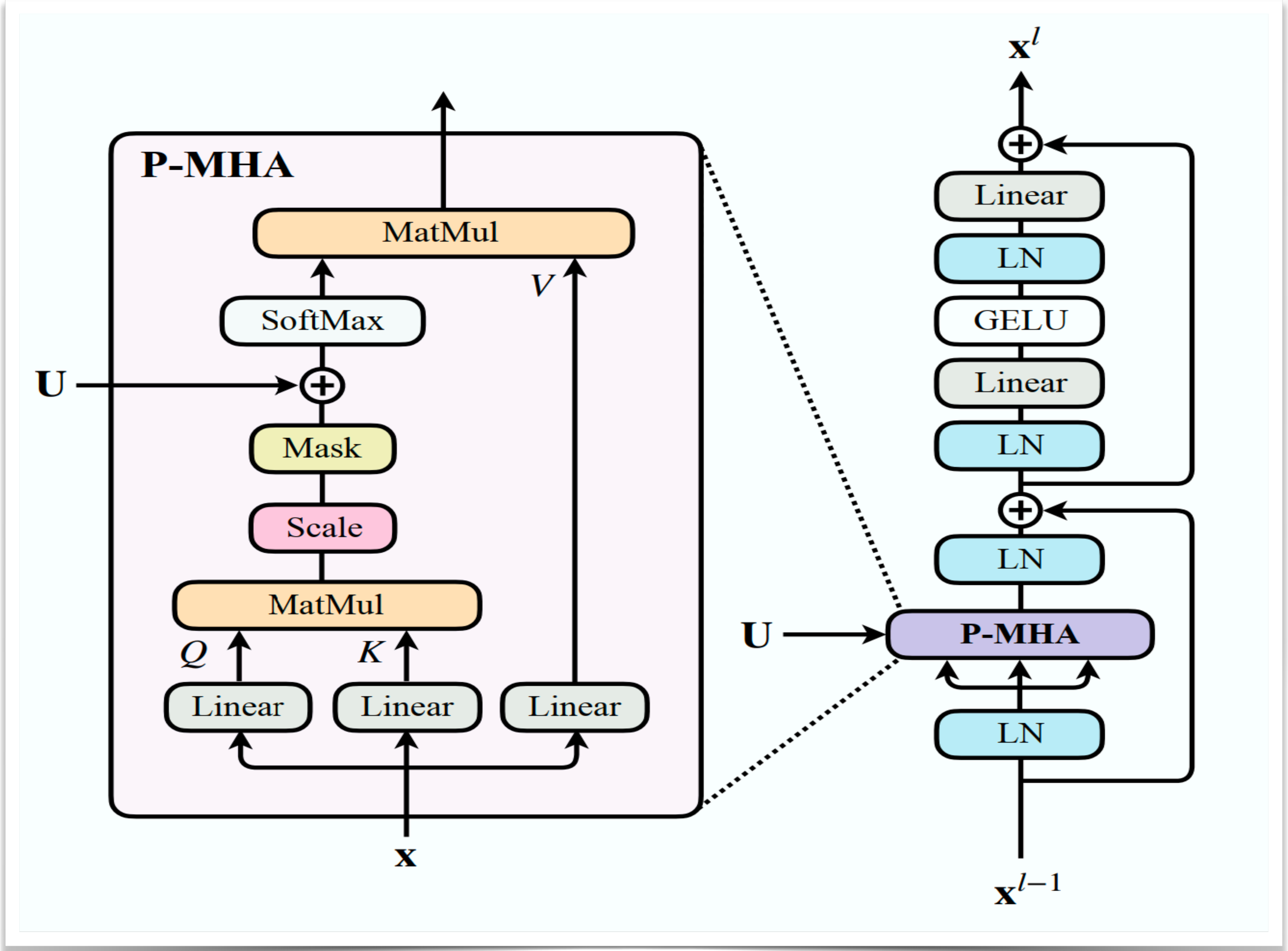


- ☐ Particle: a list of features for each particle
- ☐ Interactions: features involving a pair of particles
- ☐ Passing through a series of "attention" to MLP
- ☐ ArXiv: 2202.03772: Particle Transformer

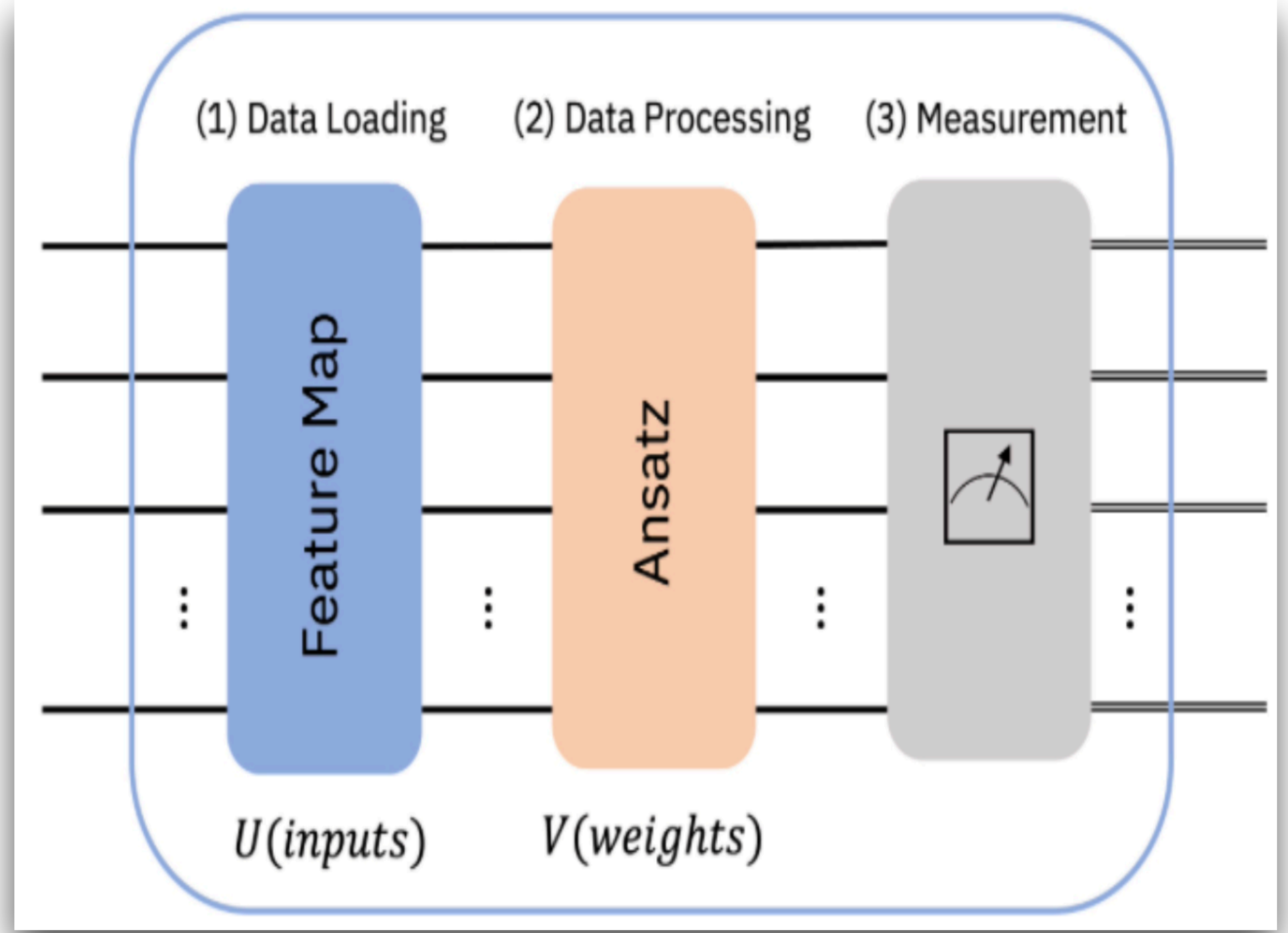


The Quantum Particle Transformer

- ☐ Particle transformer
  - Multihead-Attention based on PyTorch
  - The idea is to replace this part with a quantum



Quantum Neural Network



- ☐ The trainable parameters are added using the Anzatz with a feature-map that acts as an encoder.

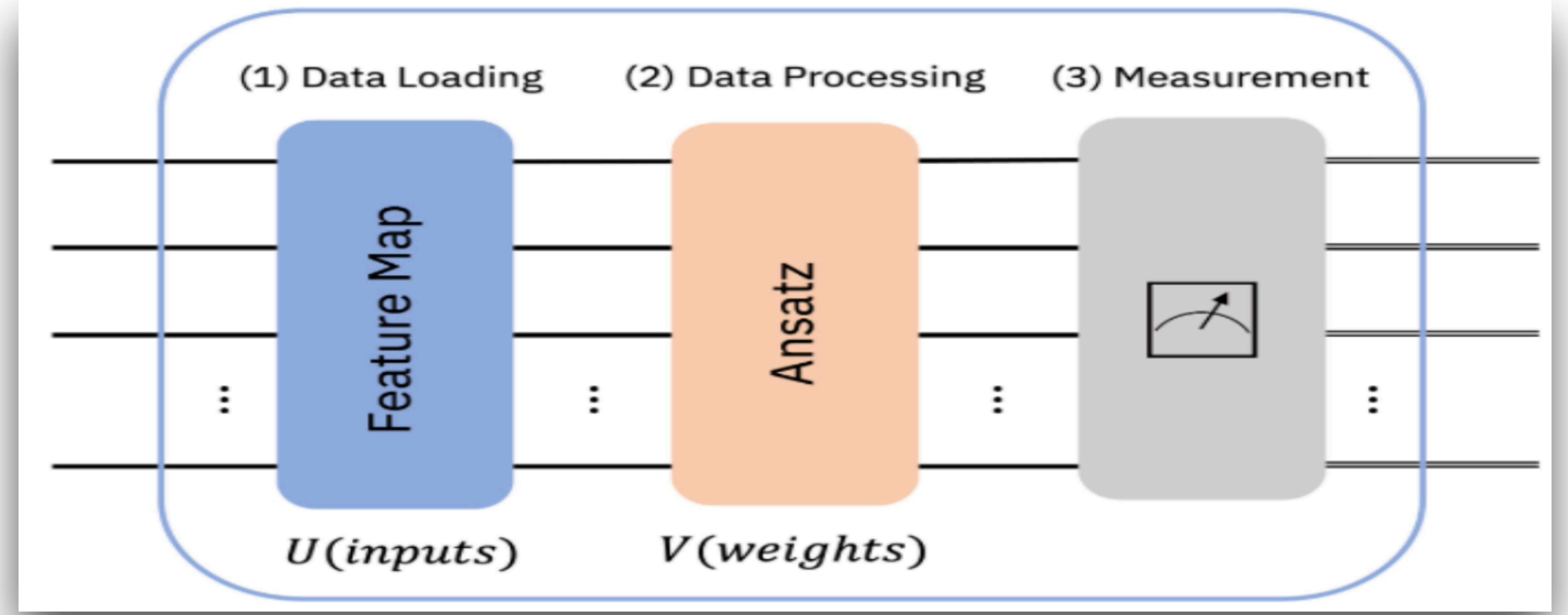
The Quantum Particle Transformer

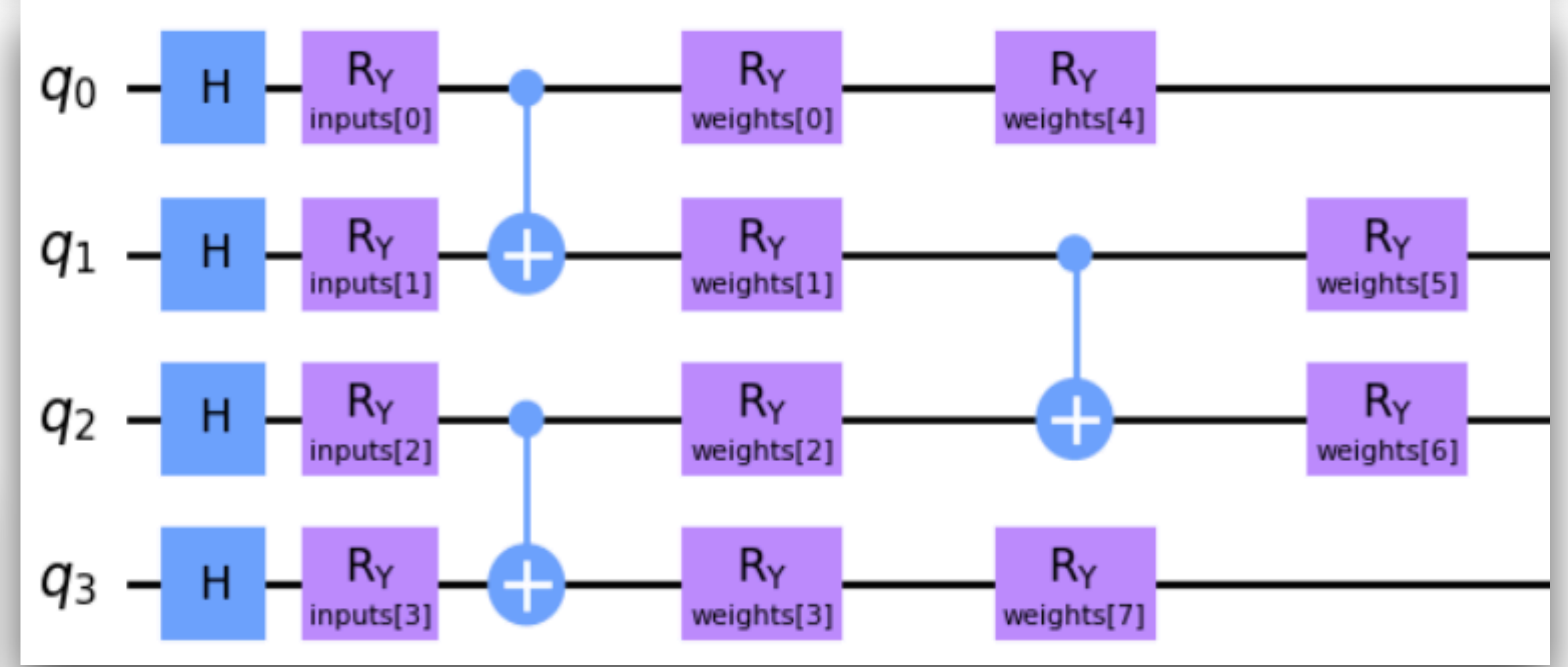
- ☐ The implementation of the quantum self-attention

```python
class QuantumSelfAttention(nn.Module):
    def __init__(self, embed_size, heads):
        super(QuantumSelfAttention, self).__init__()
        self.embed_size = embed_size
        self.heads = heads
        self.head_dim = embed_size // heads

        assert (self.head_dim * heads == embed_size), "Embed size needs to be divided by heads"

        self.values = QuantumLinearLayer(self.embed_size, self.head_dim)
        self.keys   = QuantumLinearLayer(self.embed_size, self.head_dim)
        self.queries  = QuantumLinearLayer(self.embed_size, self.head_dim)
        self.fc_out = QuantumLinearLayer(heads * self.head_dim, embed_size)

        #print(values.shape)
        #print(keys.shape)
        #print(queries.shape)

    def forward(self, values, keys, queries, mask):
        N = queries.shape[0]
        value_len, key_len, query_len = values.shape[1], keys.shape[1], queries.shape[1]

        values = self.values(values)
        keys = self.keys(keys)
        queries = self.query(queries)

        values = values.reshape(N, value_len, self.heads, self.head_dim)
        keys = keys.reshape(N, key_len, self.heads, self.head_dim)
        queries = queries.reshape(N, query_len, self.heads, self.head_dim)

        energy = torch.einsum("nqhd,nkhd->nhqk", [queries, keys])

        #if mask is not None:
        #    energy = energy.masked_fill(mask == 0, float("-1e20"))

        attention = torch.softmax(energy / (self.head_dim ** 0.5), dim=3)

        out = torch.einsum("nhql,nlhd->nqhd", [attention, values]).reshape(N, query_len, self.heads * self.head_dim)
        out = self.fc_out(out)

        return out
```

```python
def forward(self, x):
    x /= torch.sqrt(torch.sum(x**2))
    #qnn = QNNetwork(n_qubits)
    qnn_weights = algorithm_globals.random.random(self.qnn.num_weights)
    qnn_forward_batched = self.qnn.forward([x,x], qnn_weights)

    print(f"\nShape: {qnn_forward_batched.shape}")

    return qnn_forward_batched
```

Quantum Neural Network





- ☐ The trainable parameters are added using the Anzatz with a feature-map that acts as an encoder.

Summary

- Provided a quick overview of the basic idea behind Quantum Machine Learning; and
- showed, as an example, the support-vector machine:
  - A similar performance between classical and quantum was obtained.
  - Study the noise effect with a simplified model.
- Particle transformer is a bit complicated with all the self-attention added to it.
- We constructed a quantum self-attention based on a quantum neural network.