Quantum Computing
A Practical Perspective

Marco Venere

marco.venere@polimi.it

November 28th, 2024
Politecnico di Milano

Agenda

- November 7th → Theory Recap on Quantum Computing
- November 20th → Initial Setup and First Experiments
- November 21st → Grover's Algorithm
- November 25th → Combinatorial Optimization
- November 28th → VQE, QNN, QMC
- December 3rd → Quantum Error Correction & Mitigation – Projects Presentation

(it may be subject to variations)

Variational Quantum Eigensolver

VQE is a well-known quantum framework that can be used in several contexts:

1. Chemistry
2. Physical simulations
3. Optimization problems

VQE: Underlying Reasons

The Variational Quantum Eigensolver is a quantum algorithm used to find the lowest energy state of a quantum chemical system.

Indeed, it computes an upper bound for the ground-state energy of a Hamiltonian, which is used to compute the energetic properties of molecules and materials.

Drug discovery, material science, and chemical engineering require computation of correlations between electrons and other particles that demand an exponential number of parameters to process.

Since physical systems to simulate follow quantum mechanics rule, encoding them in qubits is natural and allows for an exponential speedup!

VQE: Overall Algorithm

Main steps:

1. Initialize a qubit register
2. Apply an *Ansatz* (parameterized quantum circuit) to model a quantum system that we want to simulate
3. After applying the circuit, the state of the qubits models a wavefunction
4. We can measure qubits to estimate the energy
5. A classical optimizer modifies parameters to minimize the energy

This and following slides based on: Tilly, Jules, et al. "The variational quantum eigensolver: a review of methods and best practices." *Physics Reports* 986 (2022): 1-128.

VQE: Hamiltonian

Assumption: the energy of a quantum systems is given by the Hamiltonian operator $\hat{H}$. Since energy is quantized, if we measure the energy of $|\psi\rangle$, we get a discrete number E. This number is the outcome of the measurement. Just like any other measurements, after applying $\hat{H}|\psi\rangle$, the state will collapse to one of the basis states.

The basis we are considering is made of the **eigenstates** of $\hat{H}$. For each of these states, we have a value of energy: these are the **eigenvalues** of $\hat{H}$.

VQE: Mathematical Formulation

Let's make it a little more formal:

Given a wavefunction $|\psi\rangle$ and a Hamiltonian $\widehat{H}$, the ground state energy associated with this Hamiltonian, E_0, is bounded by:

$$E_0 \leq \frac{\langle\psi|\widehat{H}|\psi\rangle}{\langle\psi|\psi\rangle}$$

What we need to is to find a parameterization of $|\psi\rangle$ that minimizes the expectation value of the Hamiltonian. When we find such a minimum, we find also the upper bound for the ground state energy, and we can potentially approach the real ground state energy with an arbitrary degree of precision.

Mathematically: we aim to find an approximation of the **eigenvector** $|\psi\rangle$ of the Hermitian operator $\widehat{H}$ corresponding to the lowest **eigenvalue**, E_0.

VQE: From Maths To QCs

To solve this problem on a quantum computer, we define an *Ansatz* that can be implemented on a quantum device as a series of quantum gates. These must be parameterized unitary operations:

$$|\psi\rangle = U(\theta)|0\rangle^{\otimes n}$$

Given this formulation, the VQE optimization problem can be expressed as:

$$E_{VQE} = \min_{\theta} \left\langle 0 \left| U^{\dagger}(\theta) \hat{H} U(\theta) \right| 0 \right\rangle$$

This is the cost function of VQE.

VQE: From Maths To QCs

$$E_{VQE} = \min_{\theta} \left\langle 0 \left| U^{\dagger}(\theta) \hat{H} U(\theta) \right| 0 \right\rangle$$

Practically, we can define the Hamiltonian as the tensor product of Pauli operators (X,Y,Z,I):

$\hat{P}_a \in \{I, X, Y, Z\}^{\otimes n}$, where N is the number of qubits: $\hat{H} = \sum_a^{\mathcal{P}} w_a \hat{P}_a$

where $\mathcal{P}$ is the number of Pauli strings in the Hamiltonian.

The function to minimize becomes:

expectation value (with QCs)

sum to minimize (with CPUs)

$$E_{VQE} = \min_{\theta} \sum_{a}^{\mathcal{P}} w_a \left\langle 0 \left| U^{\dagger}(\theta) \hat{P}_a U(\theta) \right| 0 \right\rangle$$

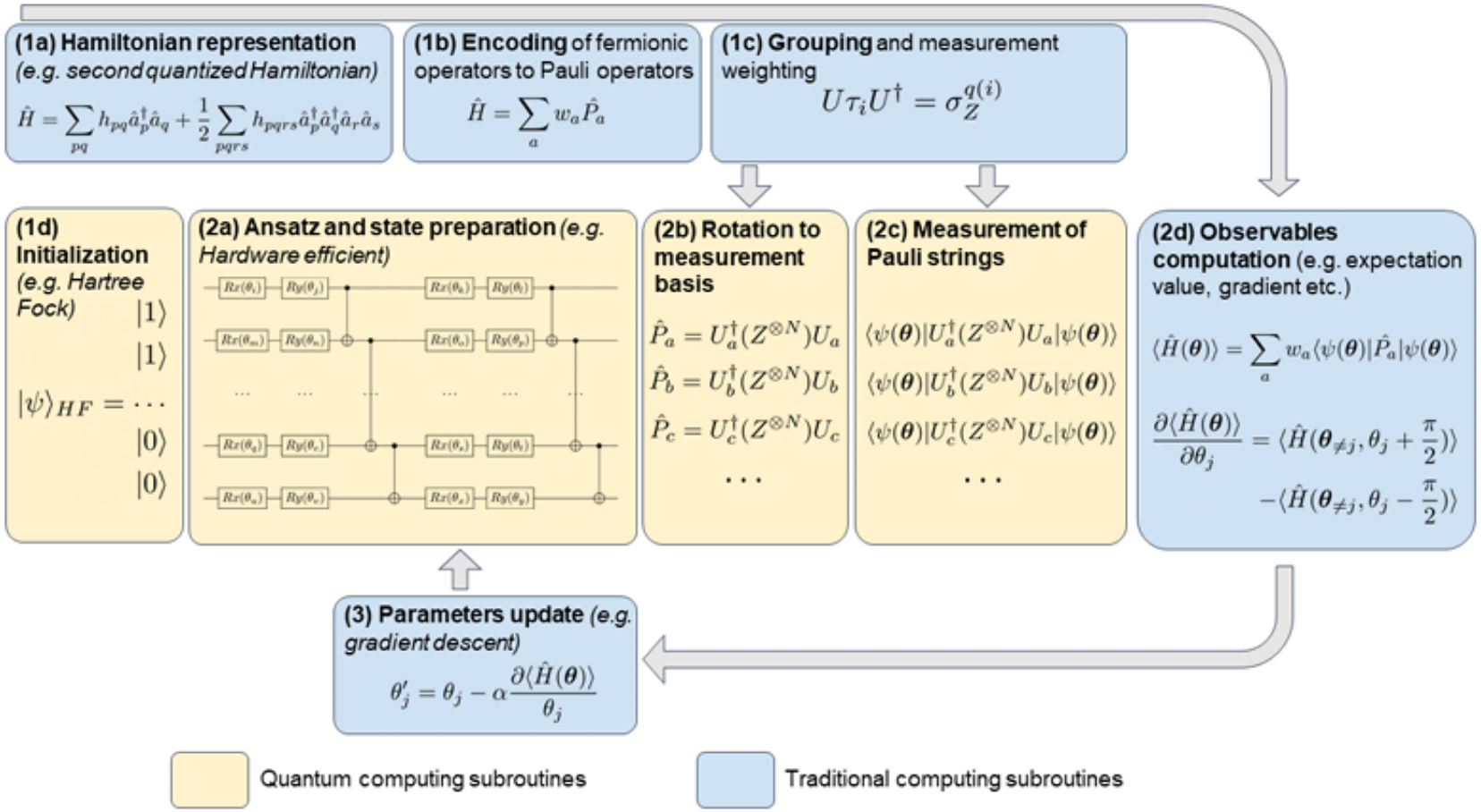
VQE: From Maths To QCs

Parameters are therefore updated to minimize this cost function.

Just like QAOA, we use an optimizer, e.g., via gradient descent, to perform this process. We specify accuracy thresholds, maximum number of iterations, and further hyperparameters that can be tuned to increase accuracy.

Furthermore, the encoding of the physical system consists of way more phases than QAOA, based on the physical properties of the system.

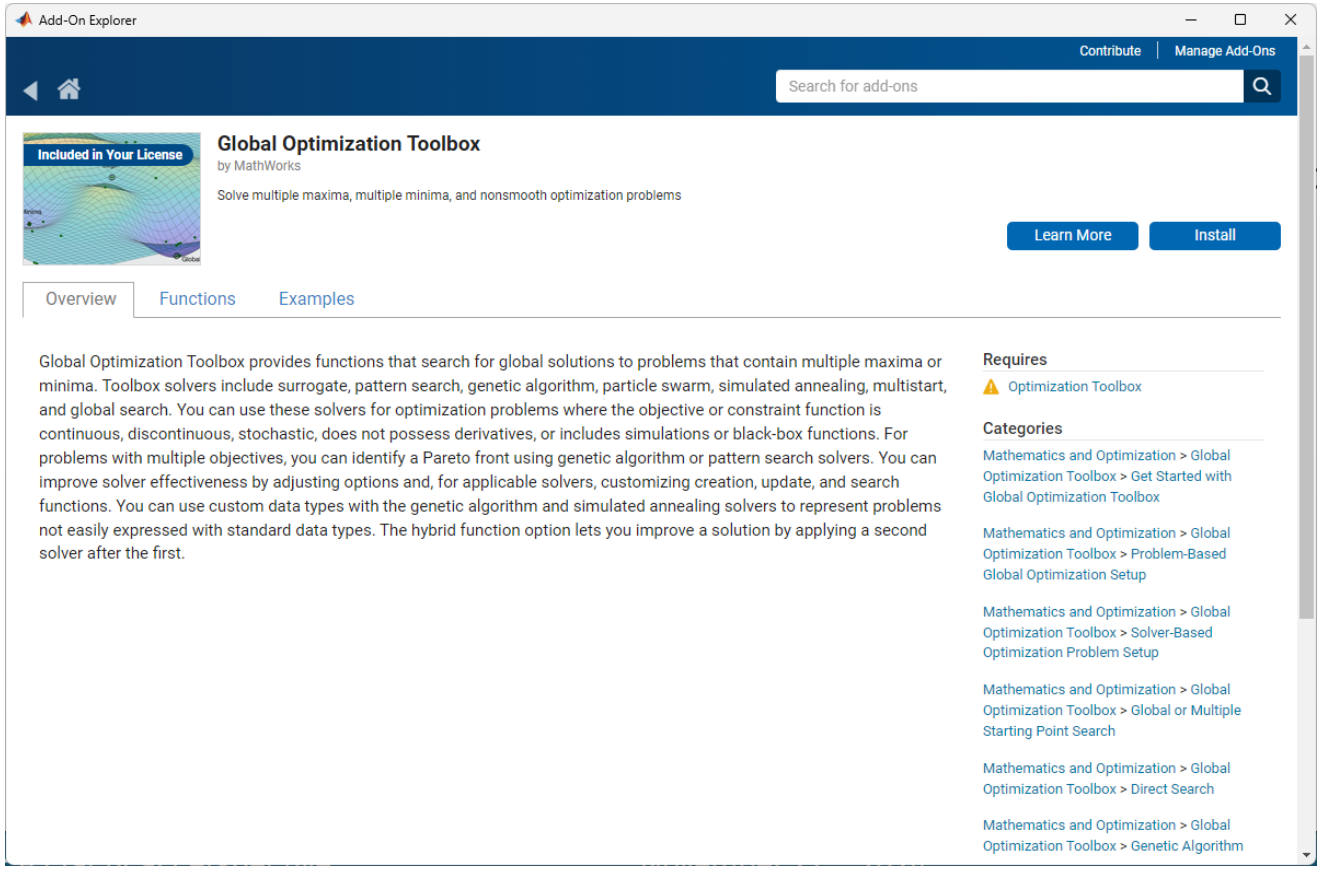
VQE: From Maths To QCs



Tilly, Jules, et al. "The variational quantum eigensolver: a review of methods and best practices." *Physics Reports* 986 (2022): 1-128.

Let’s Experiment With MATLAB

We will now use MATLAB to run some physics experiments! We need some add-ons!



Quantum Machine Learning

Another important use case for Quantum Computing is Machine Learning!

Both supervised and unsupervised learning can be performed with the help of quantum computers.

Most of the approaches are hybrid classical/quantum: they define an *Ansatz* circuit and a classical optimizer will explore the parameter space to find the optimal parameters for the circuit.

The *Ansatz* is therefore used as the **model** of the QML algorithm.

Machine Learning: Basics

Given a dataset $D = \{d_1, d_2, d_3, d_4, d_5, \dots\}$, we want to learn a function $f(x)$ that receives elements of the dataset as input and outputs a quantity.

In supervised learning, we learn to associate inputs with outputs:

1. regression: we learn a function with continuous output (e.g., given the picture of a dog, predict its height);
2. Classification: we learn a discrete function (e.g., given the picture of a dog, predict its breed).

In unsupervised learning, instead, we learn hidden properties of data (no output given in the dataset) and cluster data points in groups, based on these properties.

Machine Learning Algorithms

Supervised learning:

- Linear Regression
- Logistic Regression
- Naïve Bayes
- Decision Trees
- Support-Vector Machines
- K-Nearest Neighbor
- Neural Networks
- Similarity Learning

Machine Learning Algorithms

Unsupervised learning:

- Hierarchical Clustering
- K-Means
- DBSCAN
- Anomaly Detection with Isolation Forest
- Variational Autoencoders

QML Algorithms

A generic quantum machine learning algorithm works in the following way:

1. We encode the dataset into qubits
2. We apply an *Ansatz* (parameterized quantum circuit)
3. We measure qubits
4. We update the parameters of the *Ansatz* accordingly.

This and following slides based on: David Peral-García, Juan Cruz-Benito, Francisco José García-Peñalvo, Systematic literature review: Quantum machine learning and its applications, Computer Science Review, Volume 51, 2024, 100619, ISSN 1574-0137, https://doi.org/10.1016/j.cosrev.2024.100619.

Quantum Boltzmann Machines (QBM)

QBM defines the following Hamiltonian model:

$$H_\theta = \sum_{i=0}^{p-1} \theta_i h_i$$

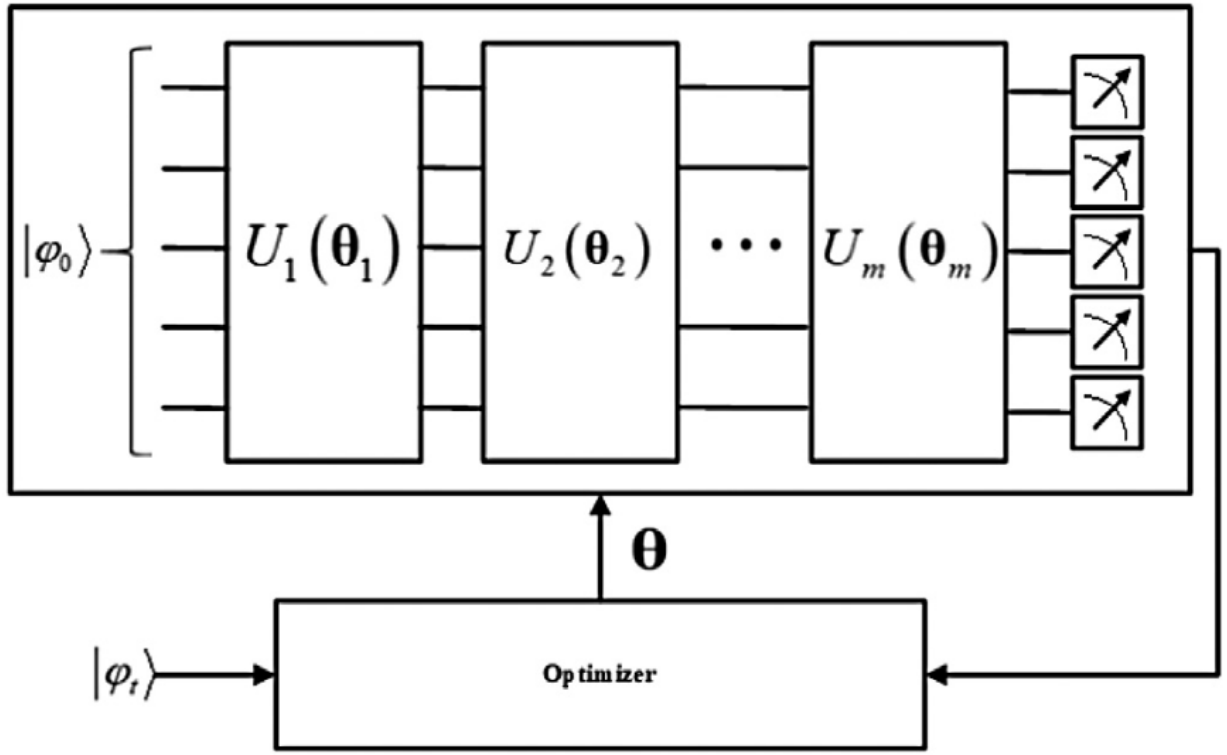
where $\theta \in \mathbb{R}^p$ and $h_i = \bigotimes_{j=0}^{n-1} \sigma_{j,i}$, for $\sigma_{j,i} \in \{I, X, Y, Z\}$ acting on qubit j.

QBMs are typically represented by an Ising Model.

The qubits used by this model may belong to two different subsets: visible qubits (determining the output of the model) and hidden qubits (for internal state).

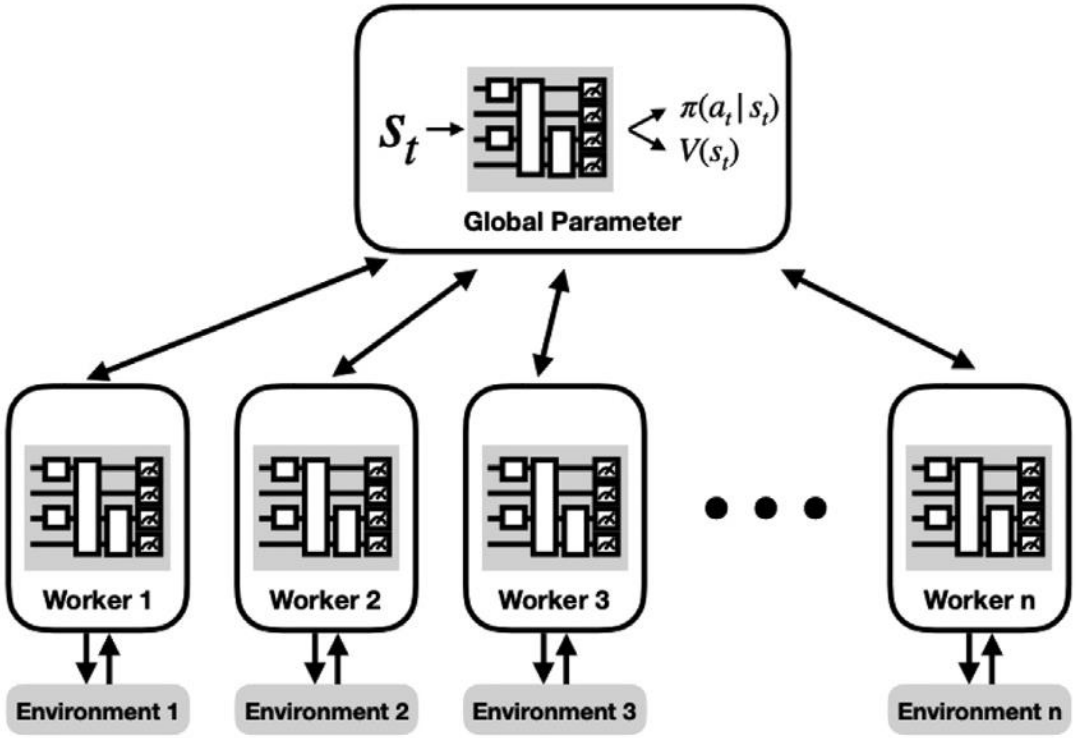
Variable-Depth Variational Quantum Circuits

vVQC is a model where the depth of the quantum circuit is automatically changed so as to increase the score of a metric.



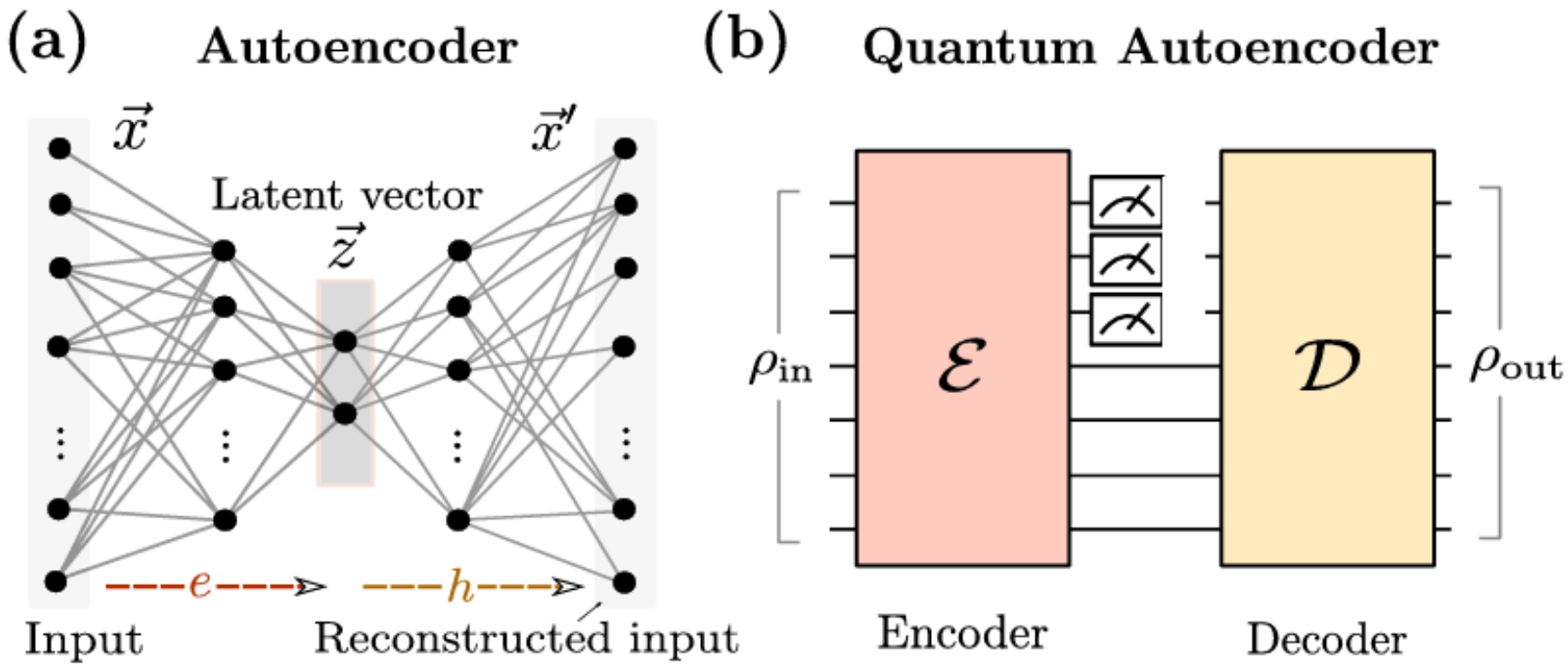
Quantum Asynchronous Advantage Actor-Critic

Some works also present distributed learning, where multiple workers model some parameters, and then global parameters are synchronized.



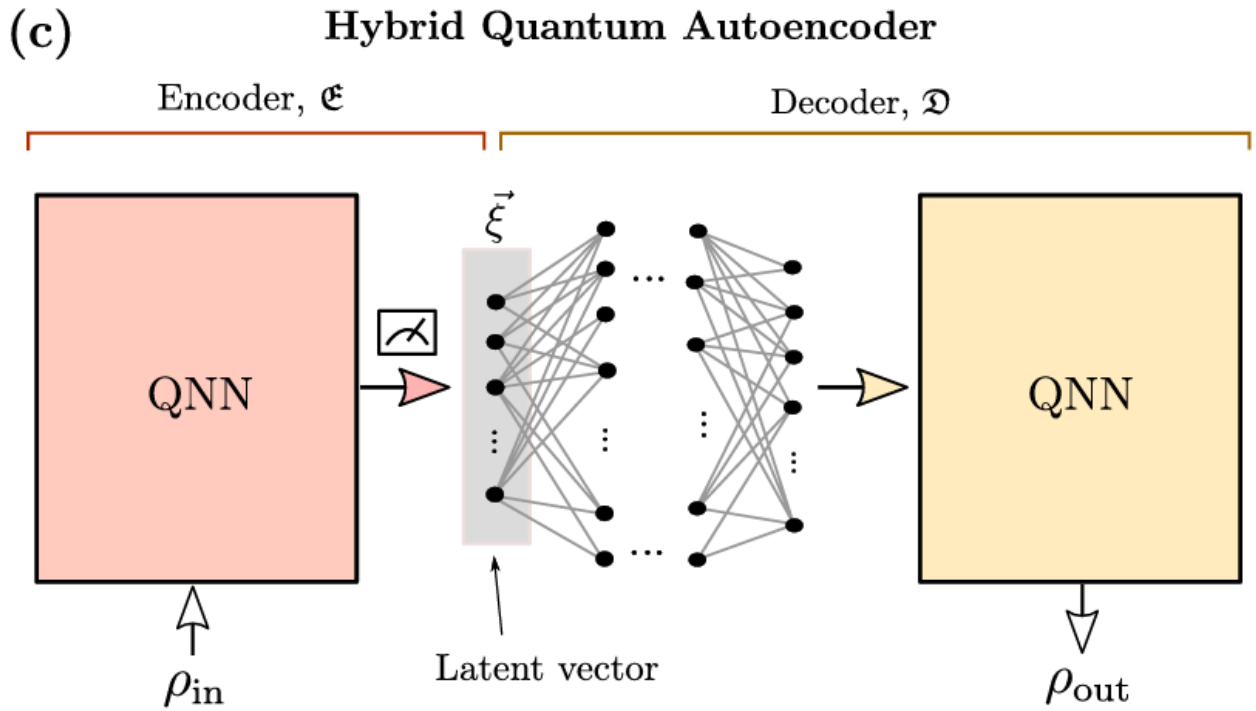
Quantum Autoencoders

The latent representation learnt by the decoder is obtained by measuring a subset of the qubits and leaving the representation on the other qubits.



Hybrid Quantum Autoencoders

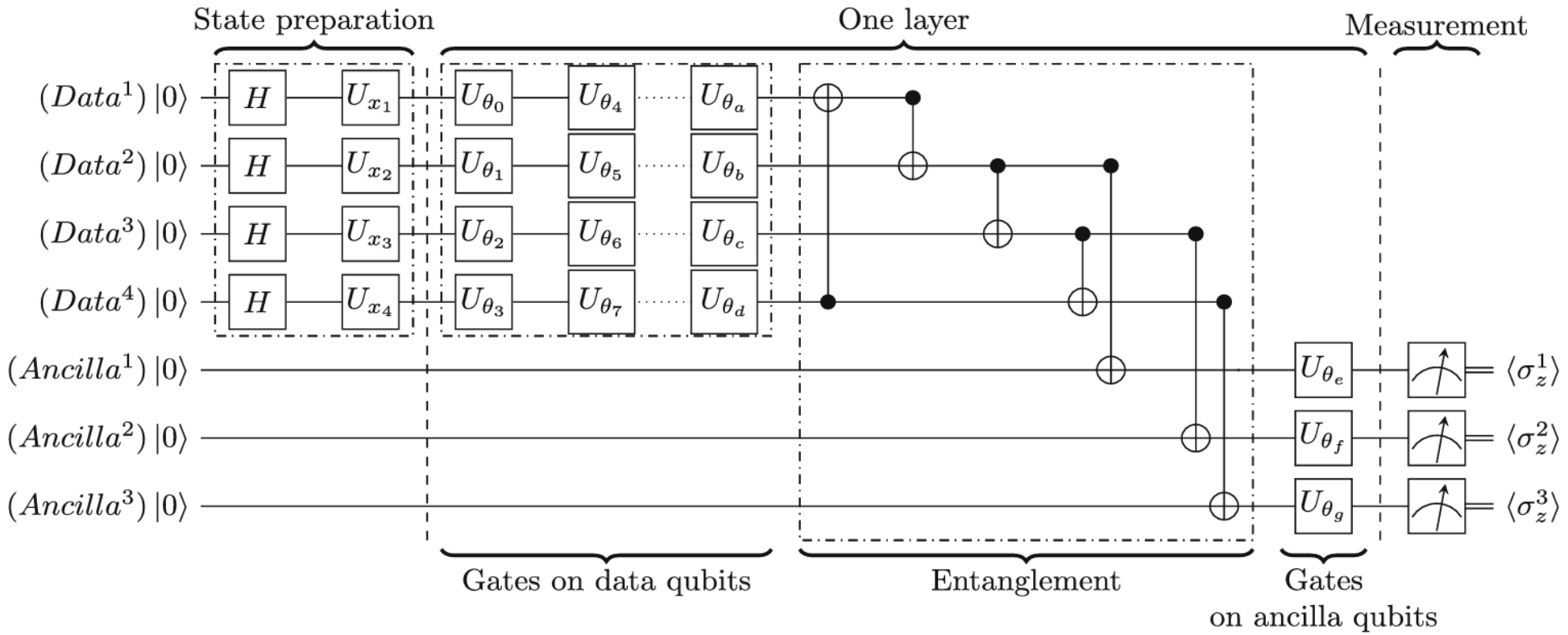
This model incorporates classical neural networks with parameterized quantum circuits. The quantum encoder is measured to produce a classical latent vector. The decoder uses a classical neural network + quantum neural network to decode.



Quantum Multiclass Classifier

Data is initialized via state preparation of qubits. Then, multiple rotational gates are applied in layers.

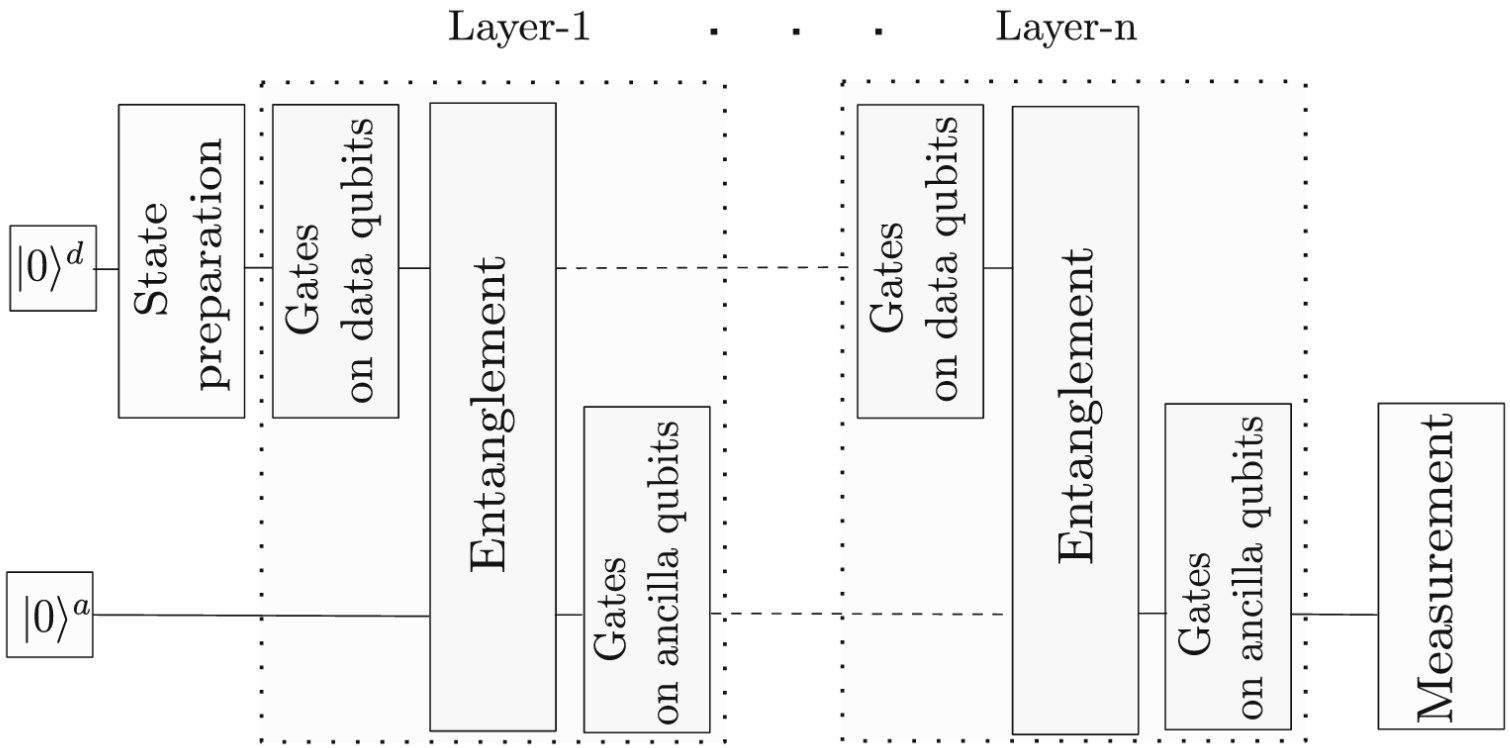
The final features to learn are encoded in ancilla qubits.



Quantum Multiclass Classifier

Data is initialized via state preparation of qubits. Then, multiple rotational gates are applied in layers.

The final features to learn are encoded in ancilla qubits.



Support Vector Machines (QSVM-kernel)

The quantum support vector machine QSVM and the quantum kernel estimator (QSVM-kernel) exploit the quantum state space as a feature space to compute kernel inputs efficiently. Data is mapped to quantum states in the following way:

$$|\Phi(\vec{x})\rangle = \Gamma_{\Phi(\vec{x})}|0^{\otimes n}\rangle$$

We have a 2^N dimensional feature space, with N qubits.

The circuit consists of two layers:

$$\Gamma_{\Phi(\vec{x})} = U_{\Phi(\vec{x})}H^{\otimes N}U_{\Phi(\vec{x})}H^{\otimes N}$$

where $U_{\Phi(\vec{x})}$ is a unitary operator that encodes the classical input data. Data allows finding a separation hyperplane among the input entries, to perform classification.

SWAP Test

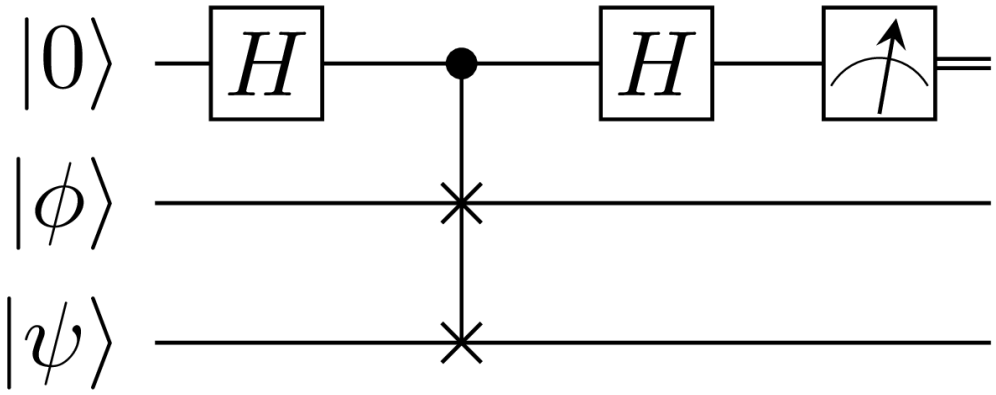
In the SWAP test, it is possible to see how much two qubit states differ from each other:

$$|0,\phi,\psi\rangle \rightarrow \frac{1}{\sqrt{2}}(|0,\phi,\psi\rangle + |1,\phi,\psi\rangle)$$

$$\rightarrow \frac{1}{\sqrt{2}}(|0,\phi,\psi\rangle + |1,\psi,\phi\rangle)$$

$$\rightarrow \frac{1}{2}(|0,\phi,\psi\rangle + |1,\phi,\psi\rangle + |0,\psi,\phi\rangle - |1,\psi,\phi\rangle)$$

$$= \frac{1}{2}|0\rangle(|\phi,\psi\rangle + |\psi,\phi\rangle) + \frac{1}{2}|1\rangle(|\phi,\psi\rangle - |\psi,\phi\rangle)$$

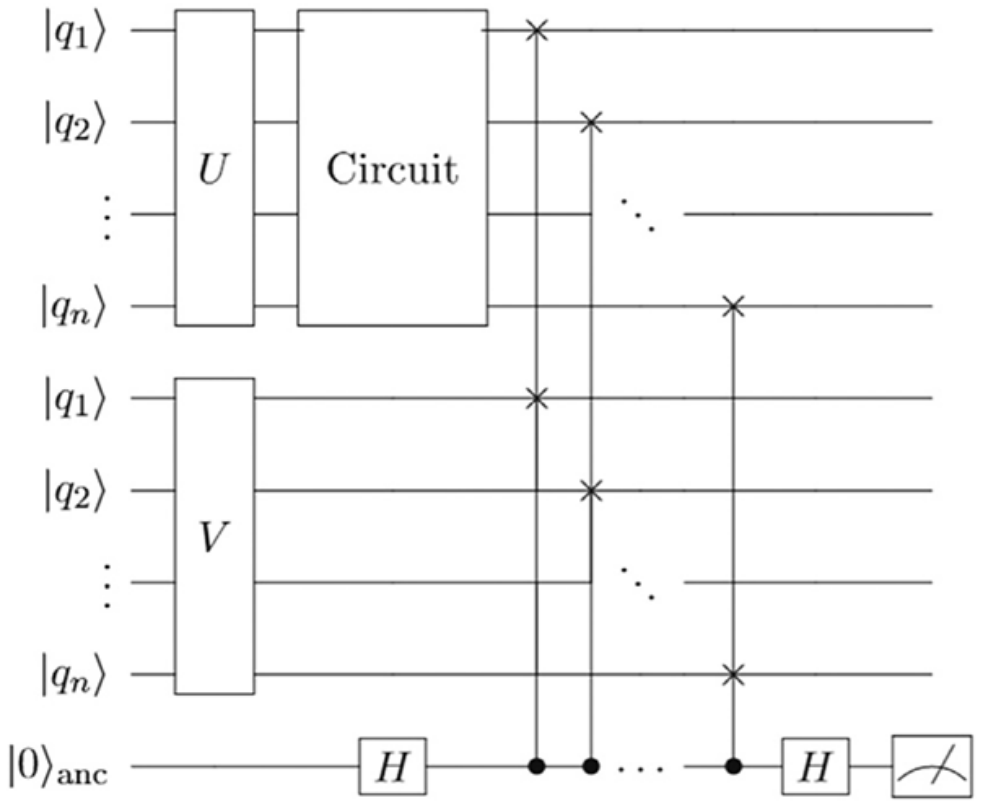


The probability to measure 0 on the first qubit is: $P(M=0) = \frac{1}{2}(\langle\phi|\langle\psi| + \langle\psi|\langle\phi|)\frac{1}{2}(|\phi\rangle|\psi\rangle + |\psi\rangle|\phi\rangle) =$

$= \frac{1}{2} + \frac{1}{2}|\langle\psi|\phi\rangle|^2$. The more measurements we do, the more we can estimate similarity!

Quantum k-Nearest Neighbor (Qk-NN)

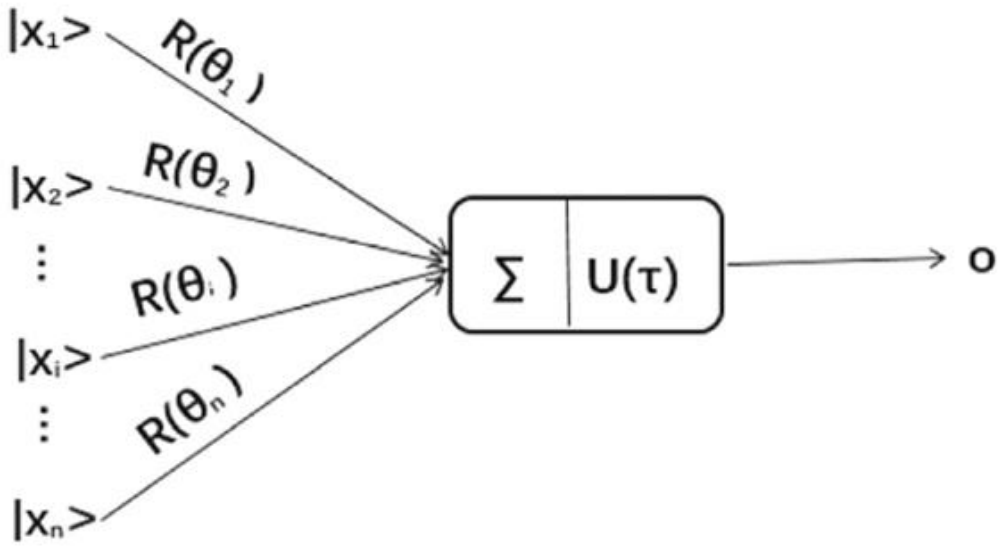
By exploiting SWAP tests, we can encode data and test their similarity with their k nearest neighbors.



Quantum Neural Networks (QNN)

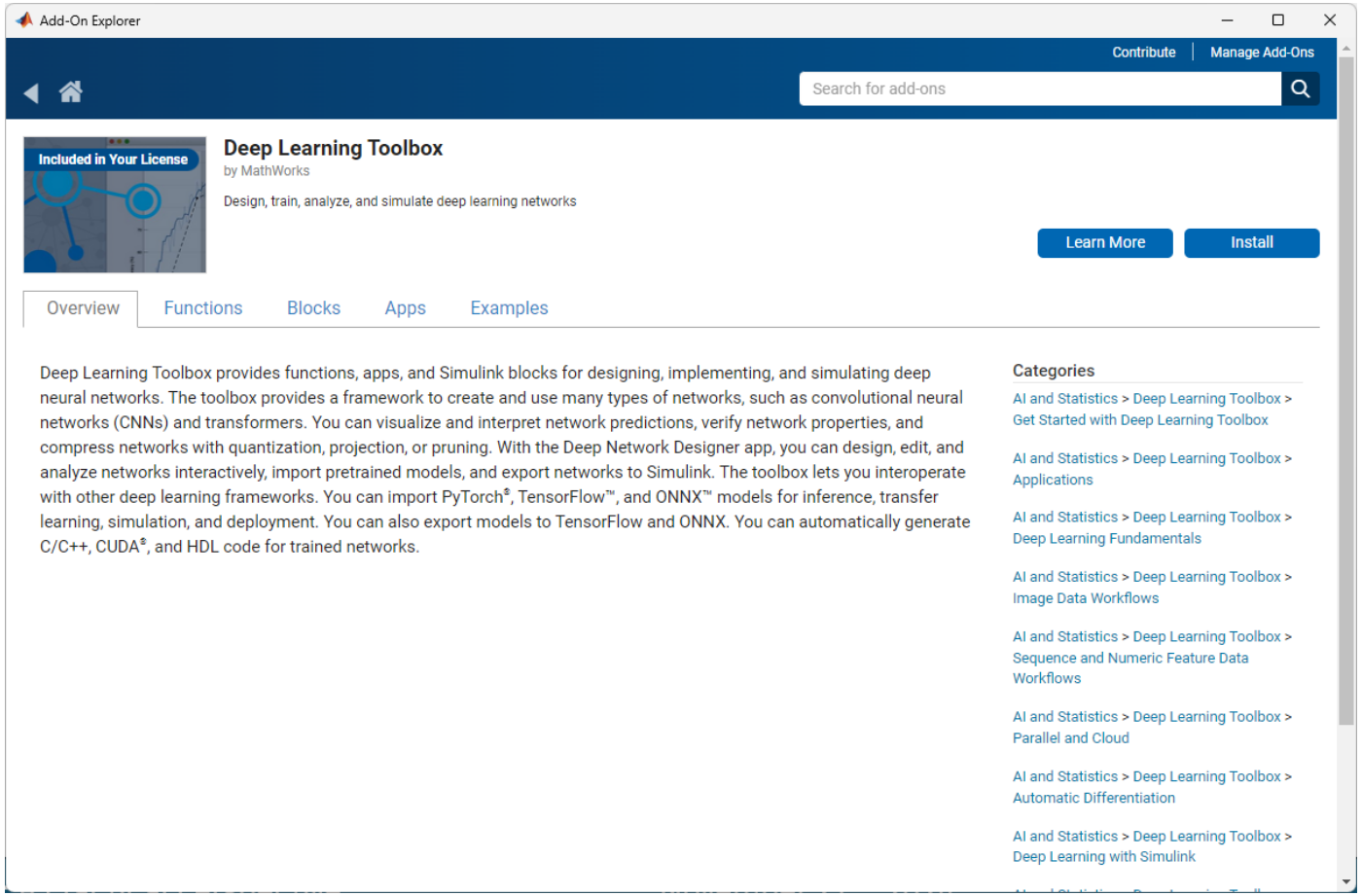
The basic component of a quantum neural network is the quantum neuron. It receives the input state $|x_i\rangle$ and outputs a real number o. To do so, it exploits specific rotation gates.

Data are encoded in the amplitudes of qubits: N qubits can encode 2^N elements. Multiple layers, made of neurons, are trained. Quantum layers and classical layers may alternate.

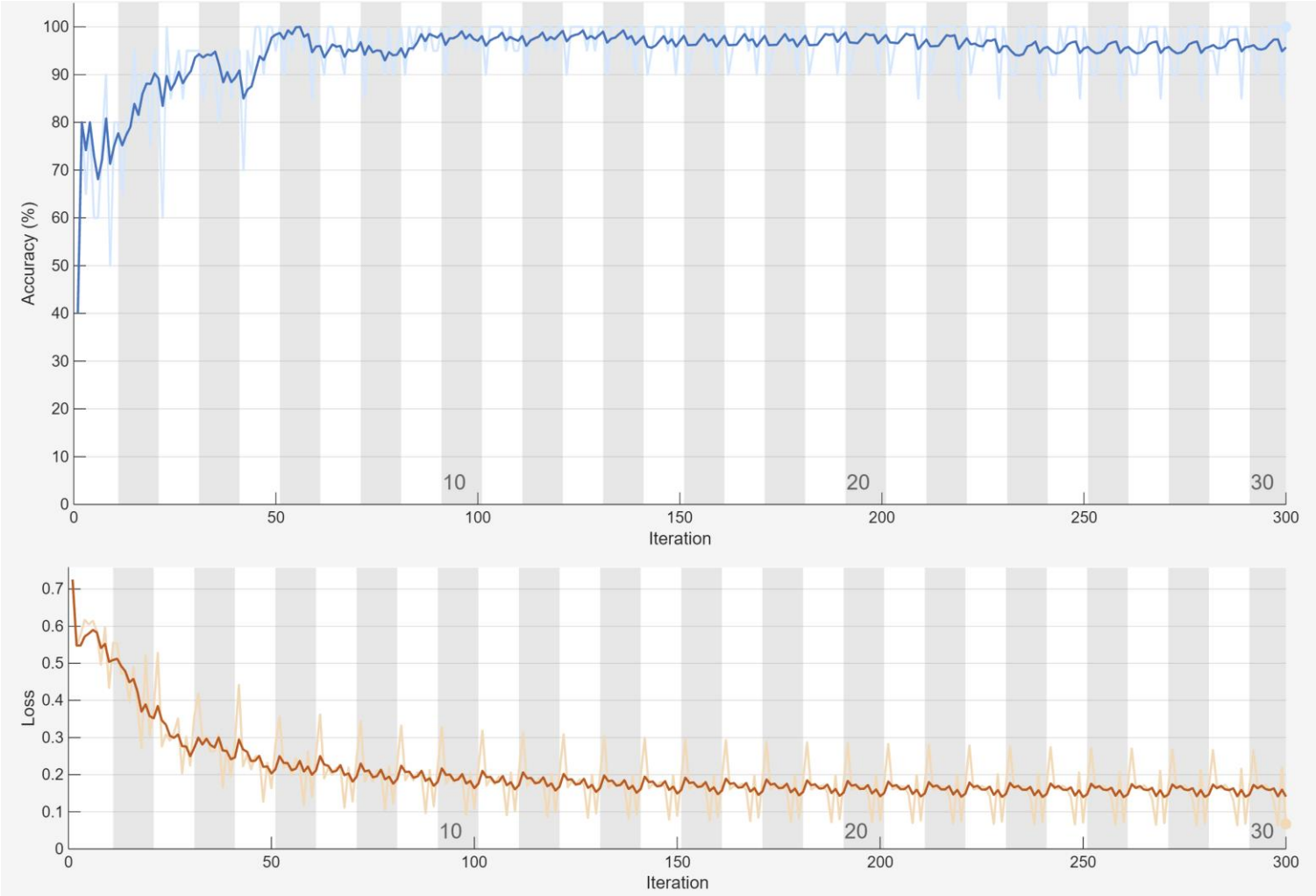


Let’s Experiment With MATLAB

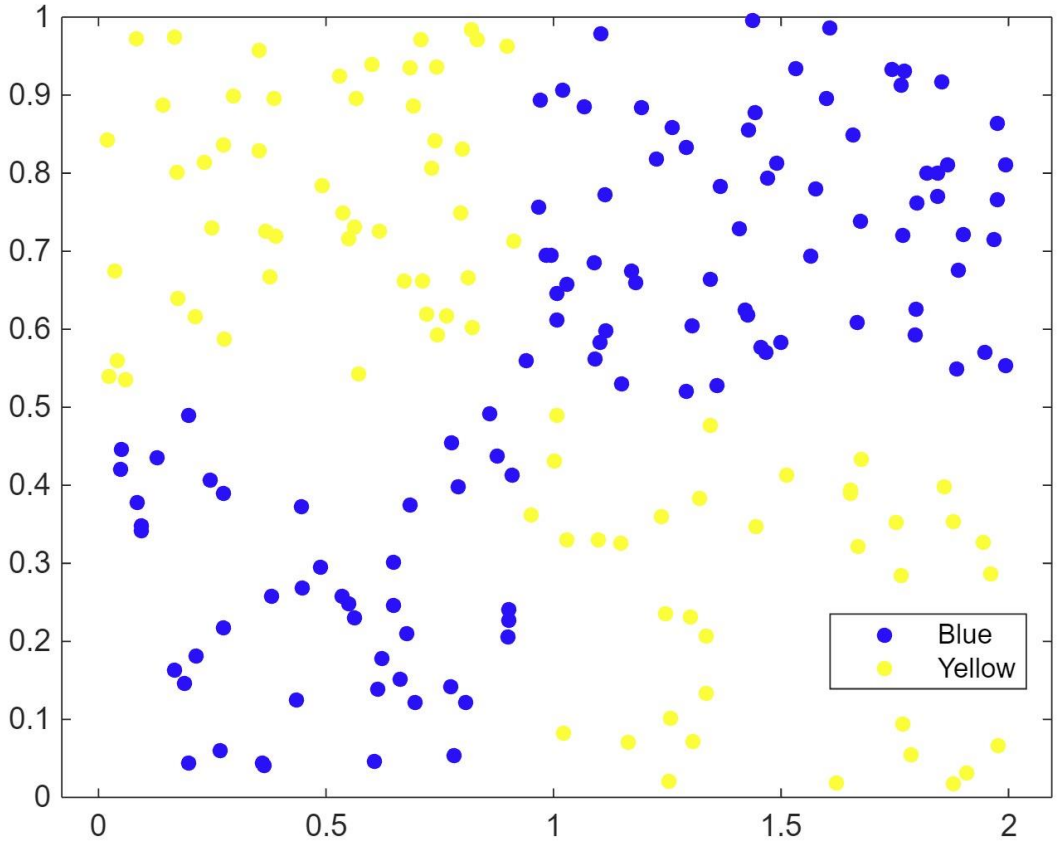
We can now use MATLAB to create our own Quantum Neural Network!

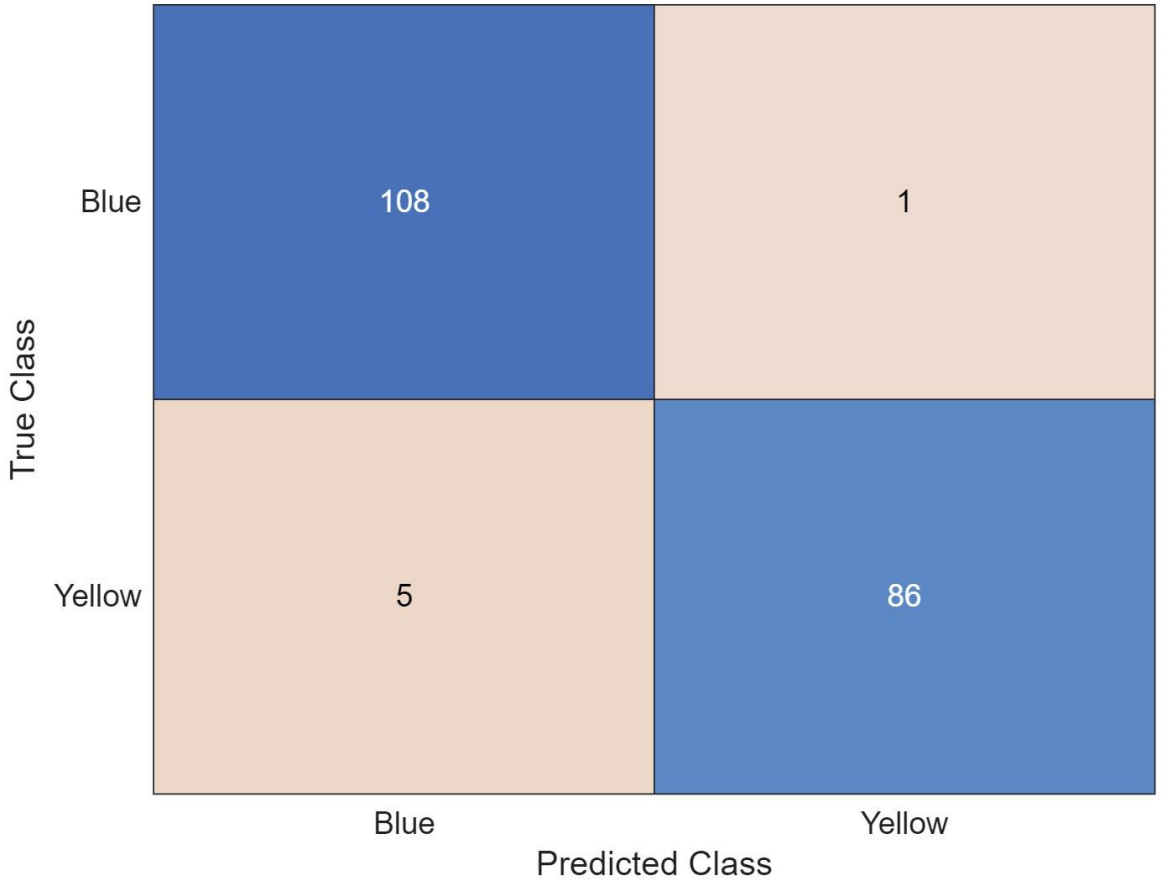


Let’s Experiment With MATLAB



Let’s Experiment With MATLAB





Quantum Monte Carlo Simulation

Another interesting use case for quantum computing is Monte Carlo simulation. In particular, this allows simulating complex systems, such as the quantum many-body system.

Indeed, many physical simulation problems, finance, optimization, etc., require an exponential number of parameters to control, becoming therefore impossible to solve even for today's parallel hardware.

Quantum Monte Carlo can solve (in some cases, exactly) many of these problems.

State of the Art provides several different variations for this algorithm.

Quantum Monte Carlo Simulation

General Idea: we have a random variable, and we want to estimate its mean. Steps to follow:

1. Encode the probability distribution of the random variable in some qubits
    - To do so, we may use quantum variational circuits, or exact non-parametric rotation gates, depending on the problem.
2. Encode the value of the random variable onto a value qubit
3. Use amplitude estimation to estimate the probability of the value qubit being in a specific state
    - Multiple methods to use, e.g.: iterative amplitude estimation or quantum phase estimation (QPE). QPE can be easily implemented as the inverse of the Quantum Fourier Transform.

Let’s Experiment With MATLAB

We will now use MATLAB to perform a Quantum Monte Carlo Simulation.

Thank you for your attention!

Quantum Computing
A Practical Perspective

Marco Venere

marco.venere@polimi.it

November 28th, 2024
Politecnico di Milano