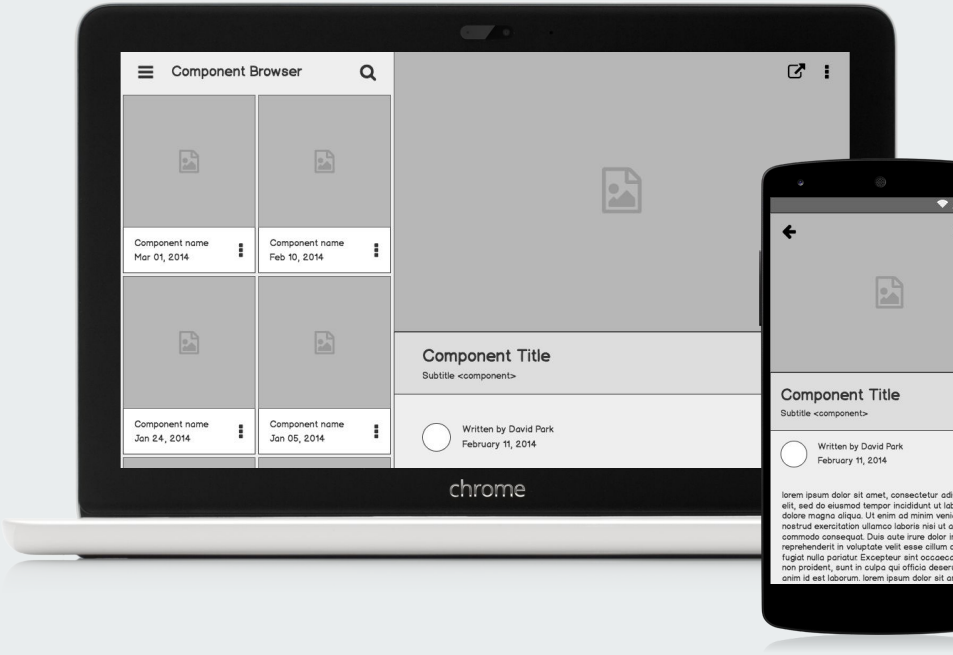
Quantum Machine Learning

An Introduction to
Quantum - Powered AI



Presenter: Aakansha Vijay,

Apurva Kacholiya,

Gunjan Bhojwani,

Hargun Yashkumar Ashwinbhai,

Sachin Kukkar,

Twinkle Gupta

Institute: SKIT & PU, Jaipur, Rajasthan

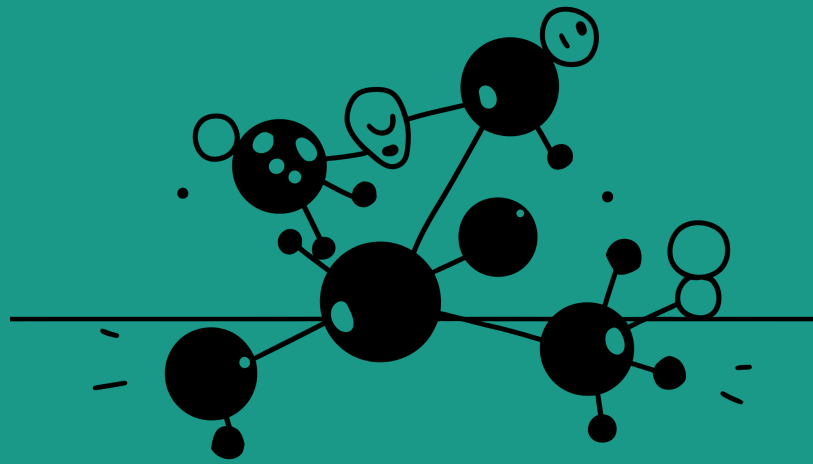
Presented to: Dr. Prashant Verma

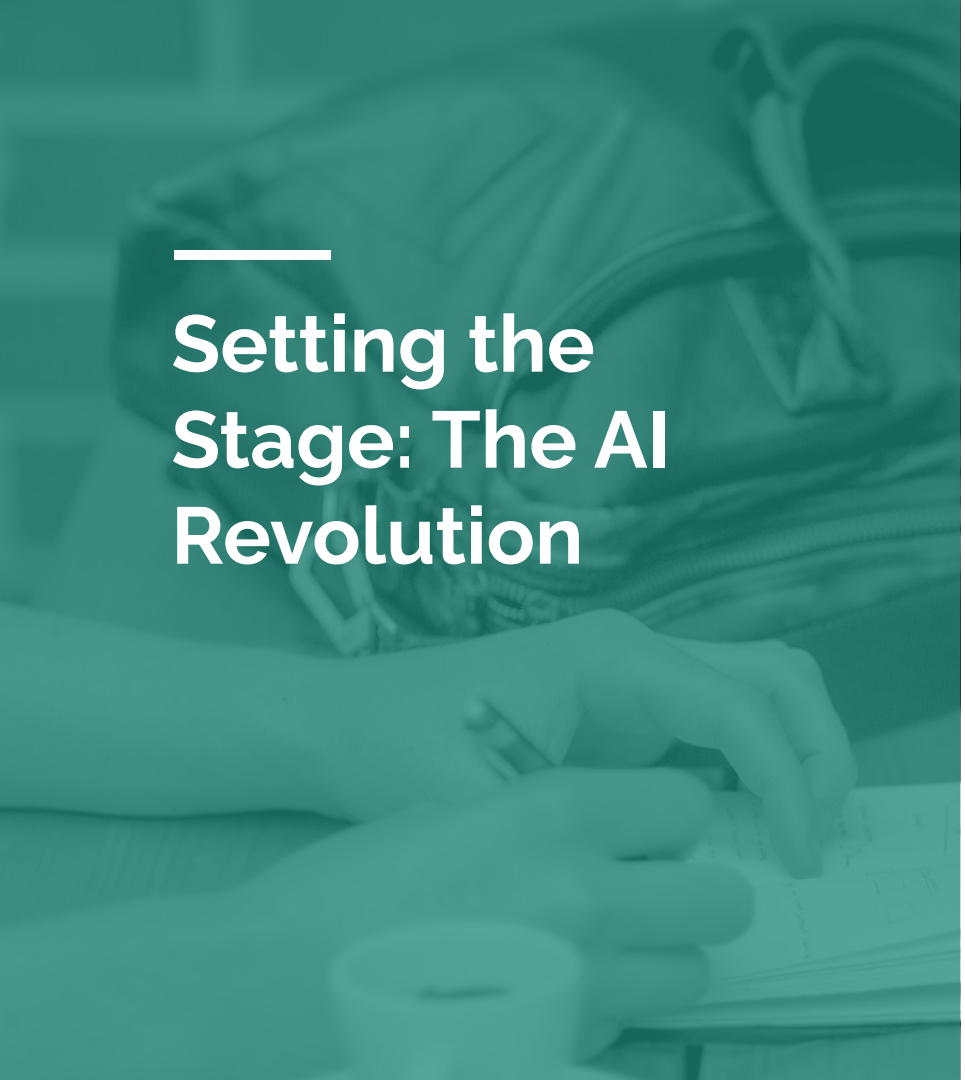
Date: 09 June 2025

TOPICS

- What is Quantum Machine Learning?
- Quantum Computing Fundamentals
- Classical Machine Learning Primer
- Why combine Quantum and ML
- Quantum Data Encoding
- QUantum Kernels
- Variational Quantum Circuits
- Quantum Neural Networks
- Key QML Algorithms
- Challenges in QML
- QML libraries and Tools
- QML Applications
- Case Study : QSVM on Iris Dataset
- Future of QML
- References
- Q&A

What is Quantum Machine Learning?



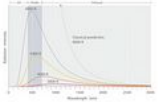


Setting the Stage: The AI Revolution

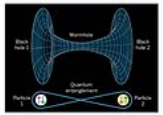
- The 21st century has been significantly shaped by **Artificial Intelligence (AI)** and **Machine Learning (ML)**, driving advancements in every sector from healthcare to finance.
- We've seen unprecedented capabilities in areas like image recognition, natural language processing, and complex decision-making systems.
- However, as data scales and problems become more intricate, classical computing models face **fundamental limits** in processing power, memory, and energy efficiency.
- This increasing demand for more powerful, efficient, and sophisticated AI necessitates exploring **new computational paradigms**.

Theoretical Foundations

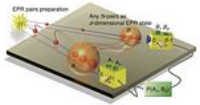
1900 — Planck's Quantum Hypothesis



1935 — The EPR Paradox



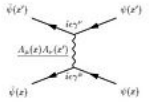
1964 — Bell's Inequality



1970 — Birth of Quantum Information Theory

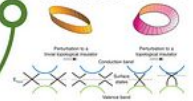
1980 — First Conference on Physics and Computation

1981 — Feynman's Quantum Computer Proposal



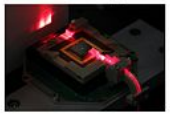
Emergence

1982 — Discovery of Topological Quantum order



Development

2000 — First Trap Ion Quantum Computer

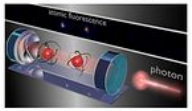


1996 — DiVincenzo Criteria For Quantum Computer

1994 — Grover's Algorithm

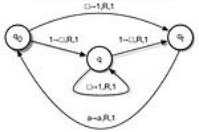
1994 — Shor's Algorithm

1985 — Deutsch's Universal Quantum Computer



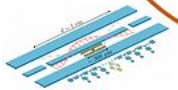
1984 — Quantum Cryptography (BB84 Protocol) By IBM

Benioff's Quantum Turing Machine



Race

2004 — Circuit QED Demo.



Ongoing Advancements

2007 — The Transmon Superconducting Qubit

2007 — D-Wave One Quantum Annealer

2013 — Rigetti Computing

2016 — Microsoft Station Q

2019 — Google Quantum Supremacy

2020 — IBM Quantum Roadmap

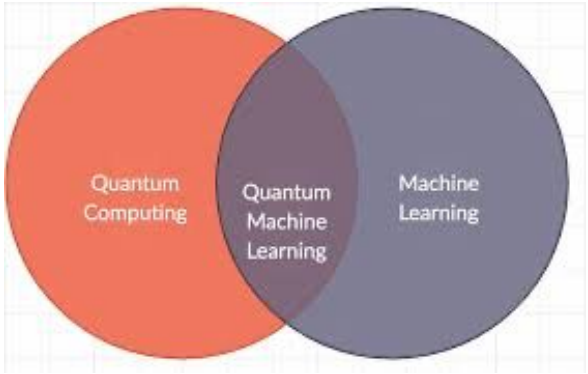
2021 — Company Booming

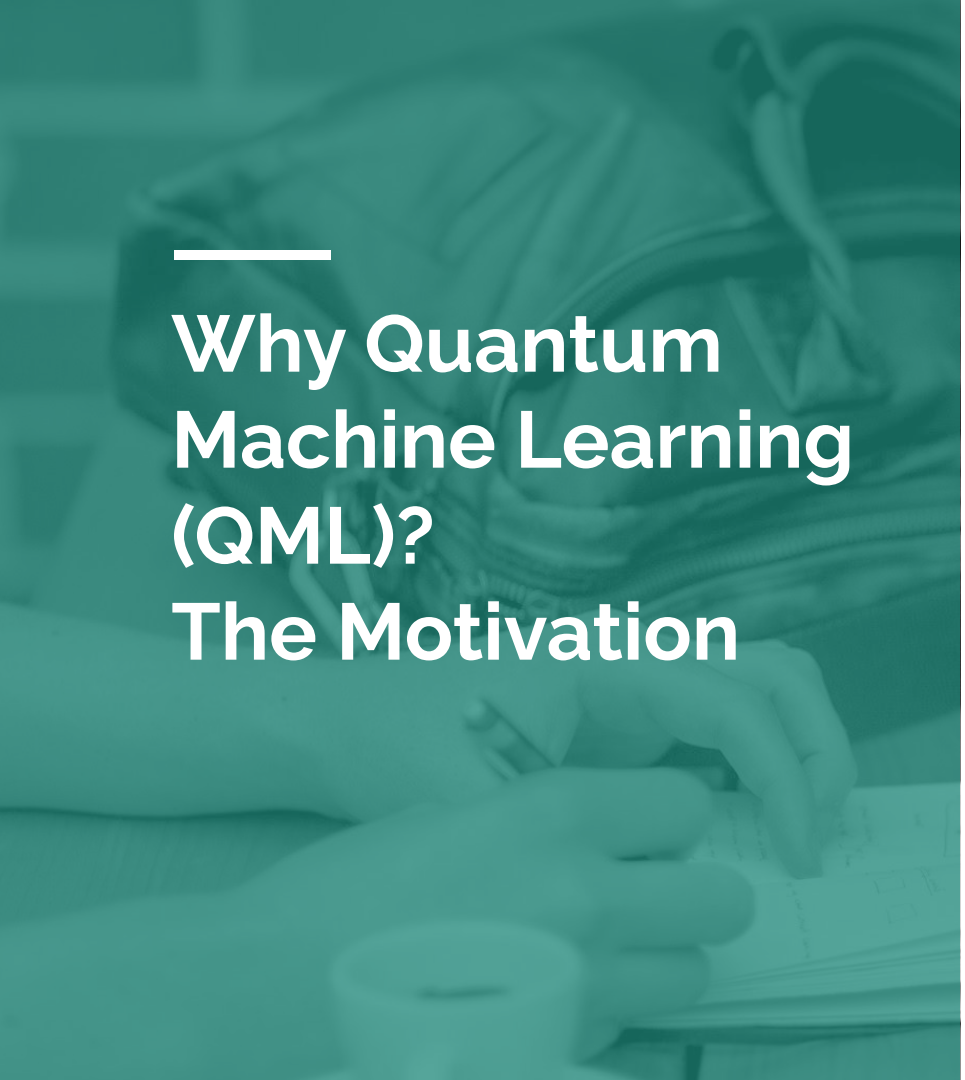
IONQ Honeywell PsiQuantum XANADU

2022 — Quantumpedia's Founding

Defining Quantum Machine Learning (QML)

- QML is an interdisciplinary field that merges quantum computing principles with classical machine learning algorithms.
- It explores how quantum phenomena (superposition, entanglement, interference) can be leveraged to enhance ML tasks.
- Not just classical ML *on* quantum computers, but ML inspired by or enhanced by quantum mechanics.

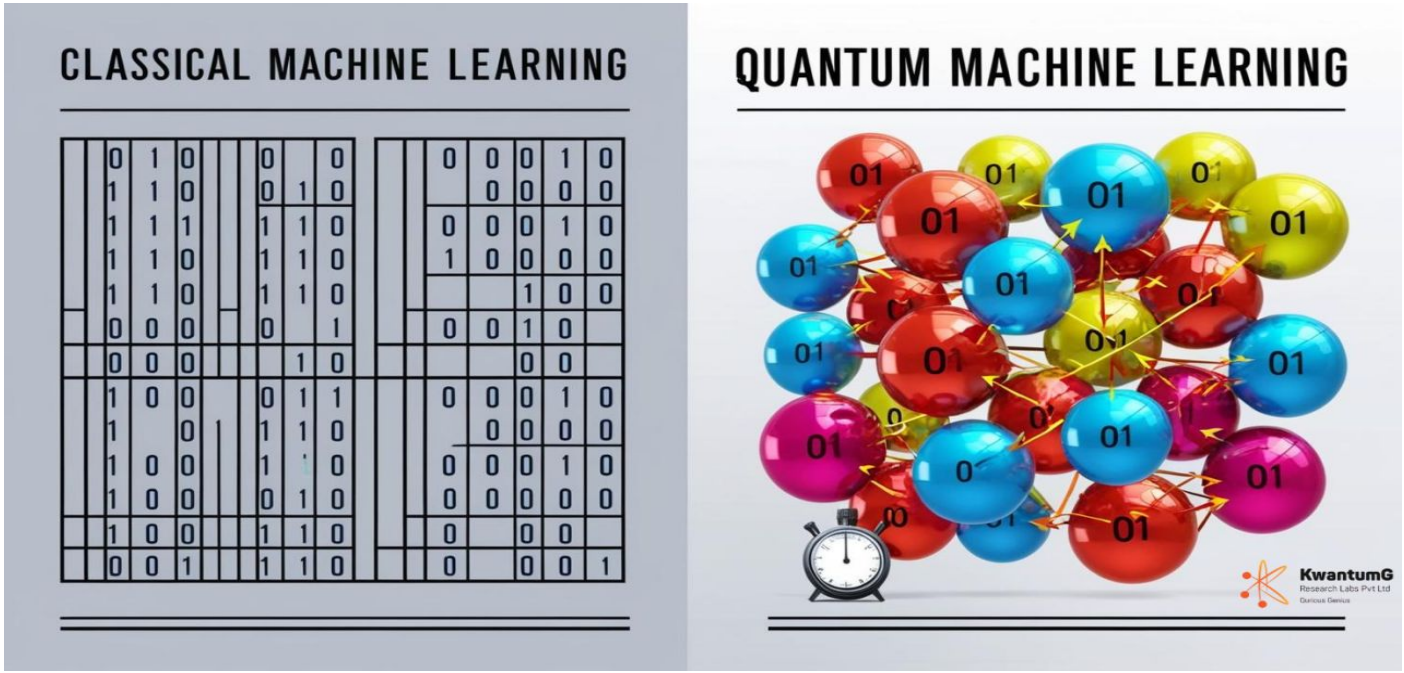




Why Quantum Machine Learning (QML)? The Motivation

- **Overcoming Classical Limitations:** Addressing challenges where classical algorithms struggle (e.g., large datasets, high-dimensional feature spaces, complex optimization problems).
- **Exploiting Quantum Properties:** Using superposition for massive parallel computation and entanglement for capturing complex data correlations.
- **Potential for Speedups:** Algorithmic advantages for specific subroutines (e.g., linear algebra, optimization).
- **Enhanced Data Representation:** Quantum states can represent exponentially more information than classical bits, leading to richer data encoding.

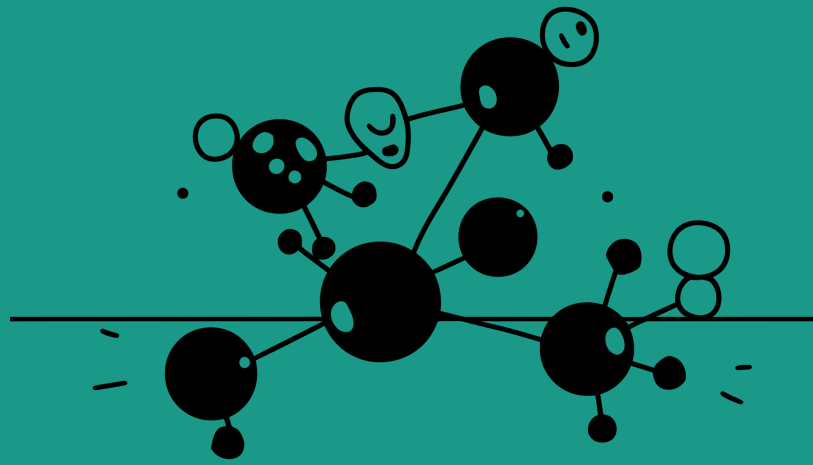
QML vs Classical ML: A Conceptual Pipeline Comparison



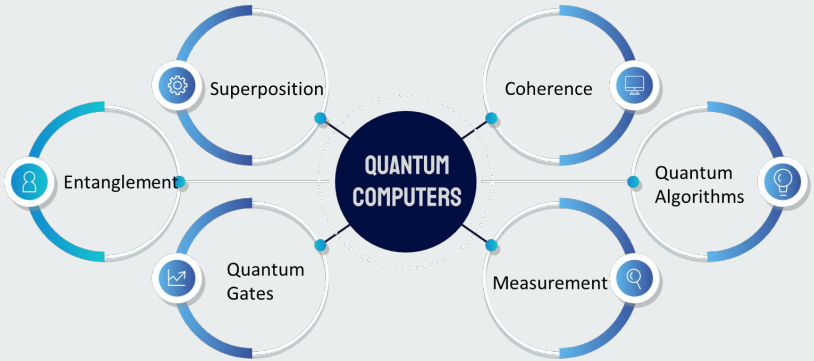
Quantum VS Classical Machine Learning

Advantages	Classical Machine Learning	Quantum Machine Learning
Processing Capability	Limited by Moore's Law and classical hardware constraints	Leverages quantum phenomena for exponential processing power
Data Handling	Struggles with high-dimensional, large-scale data	Efficiently processes large-scale and complex datasets
Algorithm Efficiency	Algorithms may have polynomial or exponential time complexity	Potential for polynomial-time solutions to classically hard problems
Model Accuracy	May be limited by computational feasibility	Enhanced accuracy through better data processing and algorithm design
Scalability	Hardware and processing limitations impact scalability	Promises greater scalability with future quantum hardware advancements

Quantum Computing Fundamentals



A Glimpse into Quantum Mechanics for Computing



- Quantum computing is not just faster classical computing; it leverages the **strange and counter-intuitive laws of quantum mechanics** to perform calculations.
- At the subatomic level, particles behave differently than macroscopic objects. They can exist in multiple places at once, be instantaneously linked, and their properties are probabilistic until measured.
- These principles – **superposition, entanglement, and interference** – are the bedrock upon which quantum computers are built, enabling them to solve problems classically intractable.
- Understanding these fundamental concepts is crucial for appreciating how QML operates.

The Basics of Quantum Mechanics

Wave-particle duality

Uncertainty principle

Superposition

Entanglement

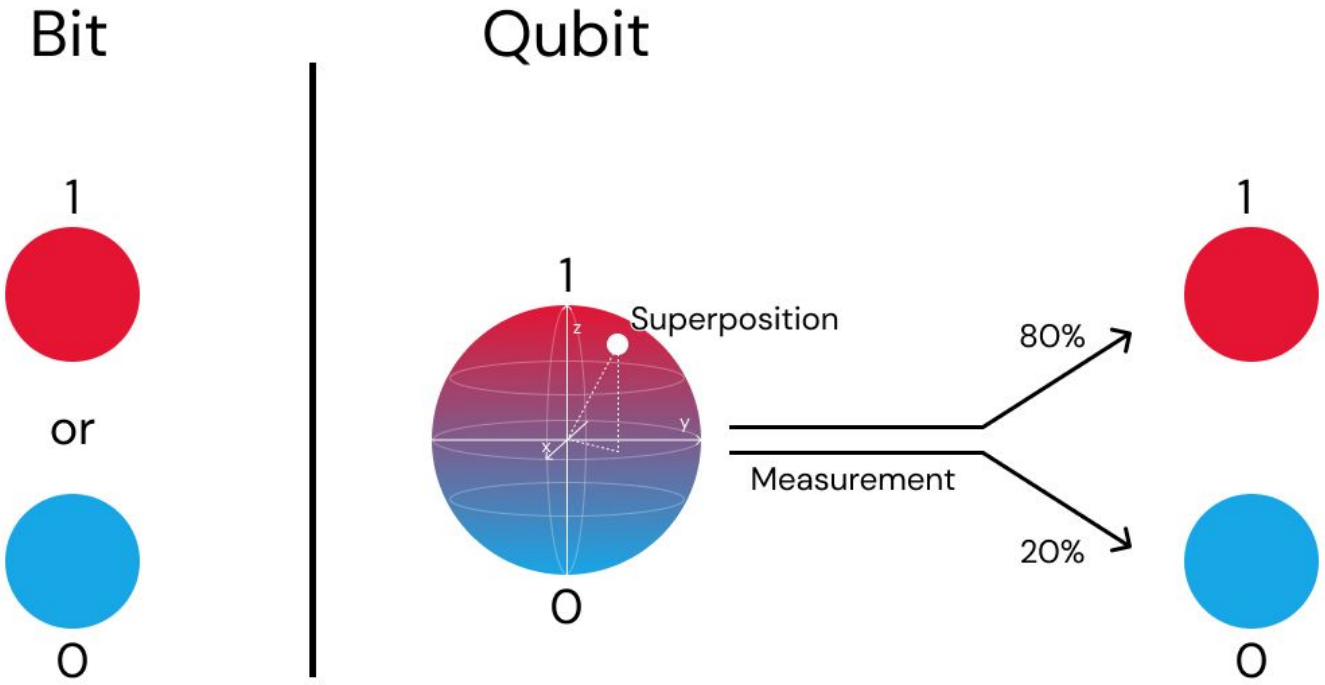
Quantum computing

Quantum cryptography

Qubits: The Basic Unit of Quantum Information

- **Definition:** It is the fundamental building block of quantum information, analogous to a classical bit.
- **Classical Bit:** Can only exist in one of two definite states: 0 or 1. (Think of a light switch: either ON or OFF).
- **Qubit:** Can be 0, 1, or a **superposition of both simultaneously**. This means it can exist in a linear combination of $|0\rangle$ and $|1\rangle$.
- The state of a qubit is described by a complex probability amplitude for each basis state. When measured, the qubit collapses to either 0 or 1 with a certain probability.
- This ability to hold multiple states at once is a cornerstone of quantum parallelism.

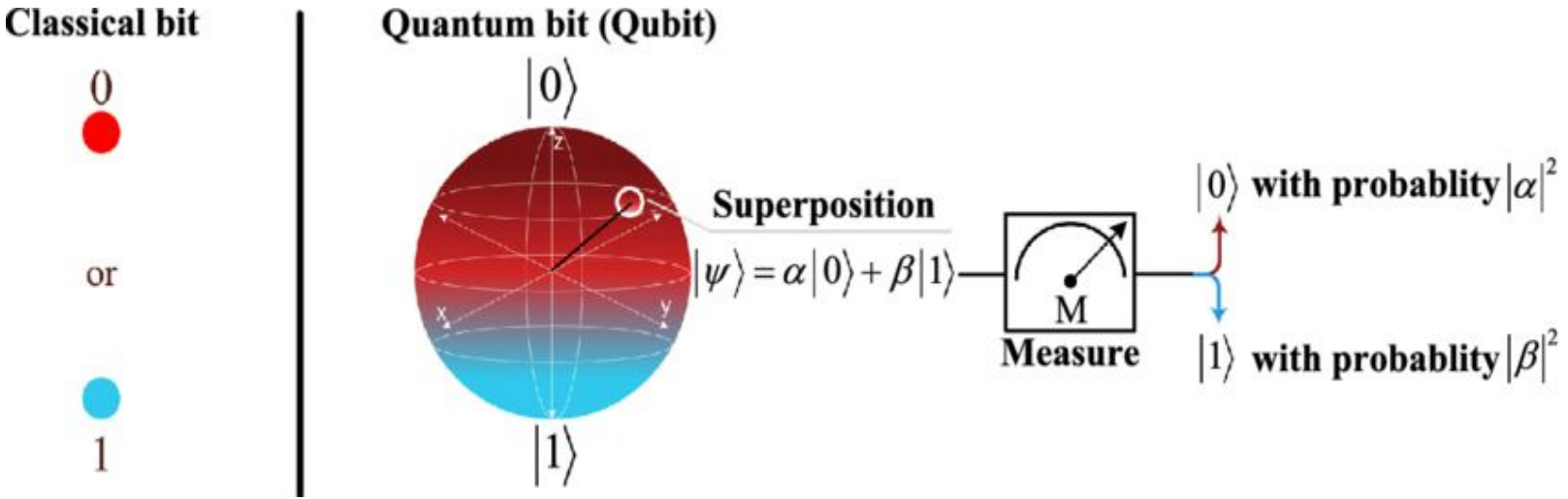
Qubit vs Classical Bit



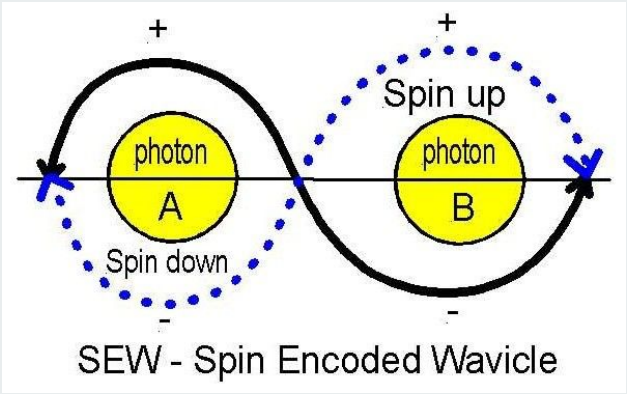
Superposition: The Power of Being Everywhere at Once

- **Elaboration:** A qubit in superposition doesn't just represent 0 *or* 1; it represents a probabilistic combination of **all possible states simultaneously**.
- Mathematically, a qubit state is often written as: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex amplitudes, and the sum of their squared magnitudes ($|\alpha|^2 + |\beta|^2$) must equal 1 (representing the total probability).
- This means a quantum computer with *n* qubits in superposition can effectively process 2^n possibilities in parallel. This **exponential increase in computational space** is a key source of quantum advantage.
- Upon measurement, this superposition collapses to a single definite classical state (0 or 1) based on the probabilities determined by α and β.

Superposition in Qubit



Entanglement: Spooky Action at a Distance

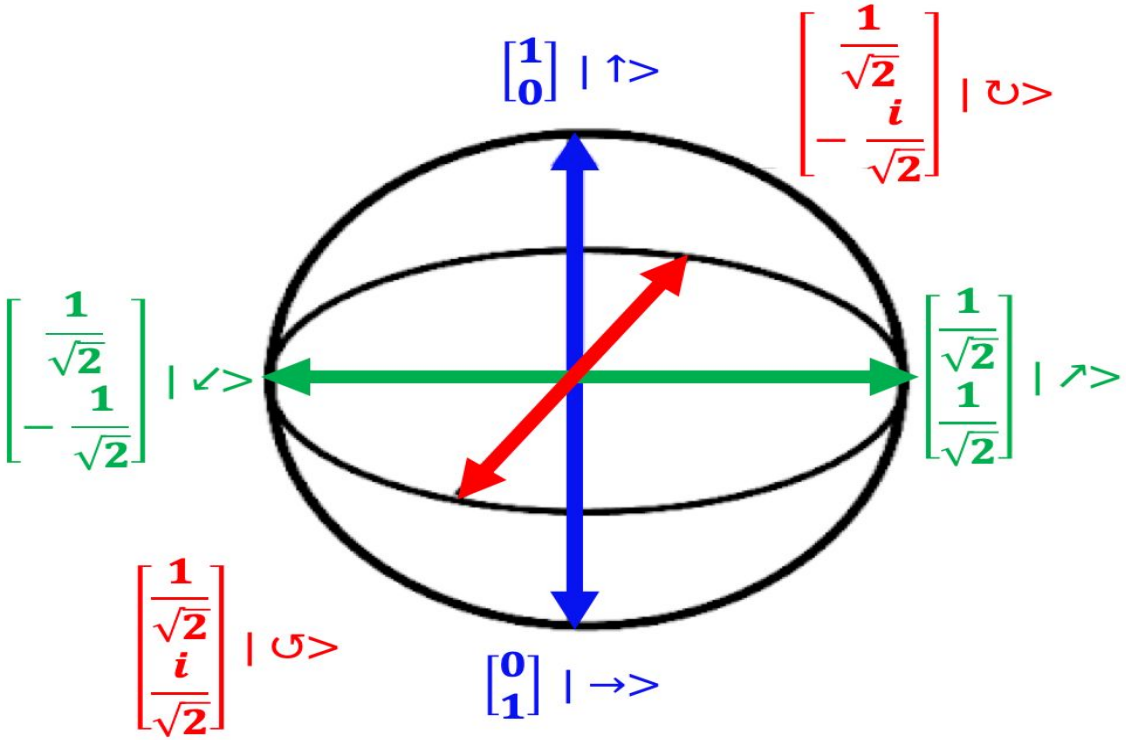


- **Definition:** Entanglement is a profound quantum phenomenon where **two or more qubits become inextricably linked**, forming a single, shared quantum state.
- When qubits are entangled, the state of one qubit **instantaneously influences the state of the others**, regardless of the physical distance between them.
- This "spooky action at a distance" (Einstein's term) creates **non-classical correlations** that are far stronger than any classical correlation.
- **Example:** If two qubits are entangled such that they are always in the same state (both 0 or both 1), and you measure one to be 0, you instantly know the other is also 0, without ever measuring it.

The Bloch Sphere: Visualizing a Single Qubit

- **Purpose:** The Bloch Sphere is a **geometric representation of the pure state space of a single qubit**. It provides an intuitive way to visualize a qubit's state and how quantum operations transform it.
- Any point on the **surface** of the sphere represents a valid pure state of a single qubit, including superpositions.
- The **poles** of the sphere represent the basis states: the North Pole is $|0\rangle$ and the South Pole is $|1\rangle$.
- Rotations on the Bloch Sphere correspond to applying quantum gates. For example, a Hadamard gate rotates $|0\rangle$ to a superposition state on the equator.

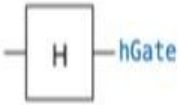
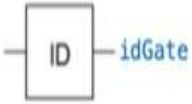
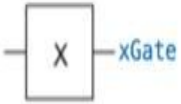
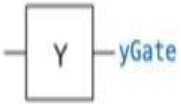
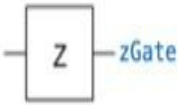
Bloch Sphere

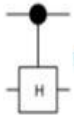
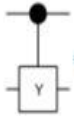


Quantum Gates: The Building Blocks of Circuits

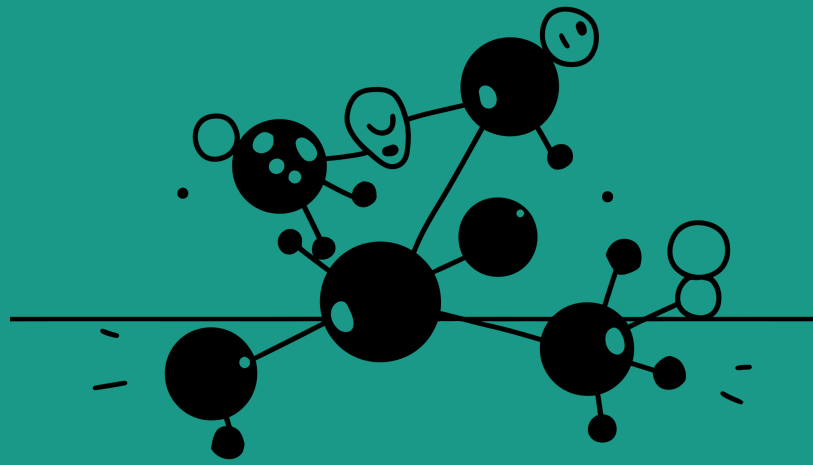
- A quantum gate is a very simple computing device that performs quantum operation on qubits. Quantum gates are one of the essential parts of a quantum computer and are the building blocks of all quantum algorithms.
- Quantum gates are mathematically represented as transformation matrices which operate on inputs to give outputs.
- Two types:
    - Single Qubit Gates
    - Multiple Qubit Gates

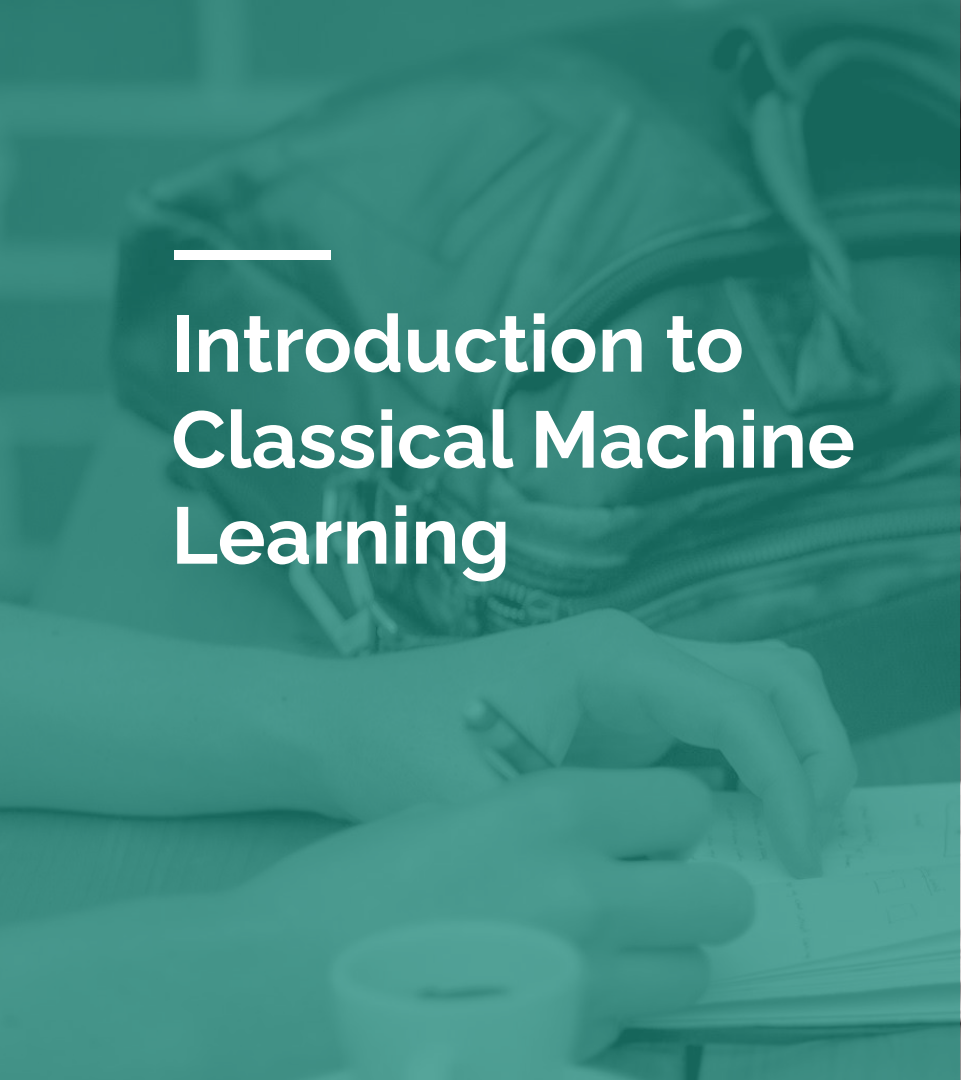
Basic Quantum Gates

Creation Function	Gate Name	No. of Qubits	Matrix Representation	Properties
	Hadamard gate	1	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$	• Traceless • Involutory
	Identity gate	1	$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$	• Identity • Involutory
	Pauli X gate	1	$\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$	• Traceless • Involutory • Pauli group
	Pauli Y gate	1	$\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}$	• Traceless • Involutory • Pauli group
	Pauli Z gate	1	$\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$	• Traceless • Involutory • Pauli group

Creation Function	Gate Name	No. of Qubits	Matrix Representation	Properties
 chGate	Controlled Hadamard gate	2	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$	• Involutory
 cxGate or cnotGate	Controlled X or CNOT gate	2	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	• Involutory
 cyGate	Controlled Y gate	2	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{bmatrix}$	• Involutory
 czGate	Controlled Z gate	2	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$	• Involutory • Swapping control and target qubits does not change gate operation
 swapGate	Swap gate	2	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	• Involutory • Swapping the two target qubits does not change gate operation

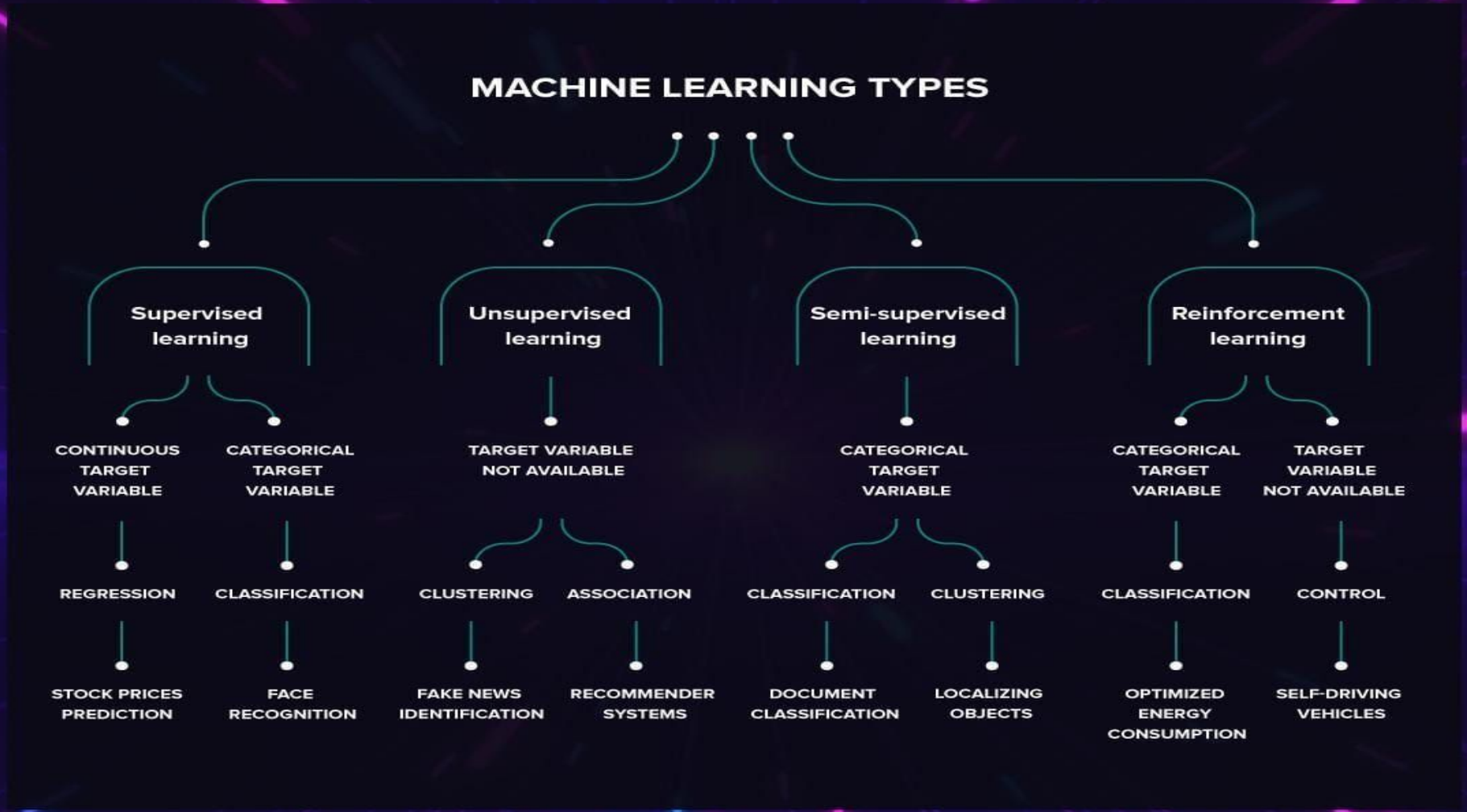
Classical Machine Learning Primer



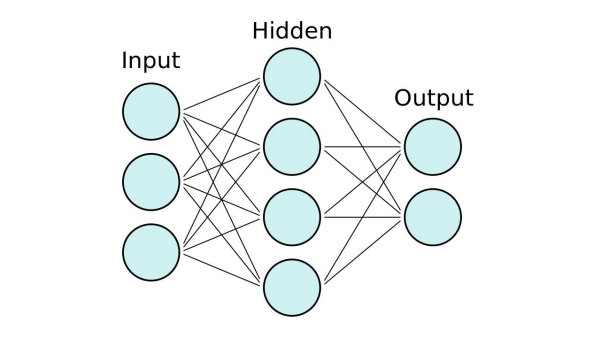


Introduction to Classical Machine Learning

- Machine Learning (ML) is a subset of AI that focuses on enabling computer systems to **"learn" from data without being explicitly programmed** for every task.
- Instead of hard-coded rules, ML algorithms build models based on example data, allowing them to make predictions or decisions.
- ML has transformed countless industries and daily lives, powering everything from personalized recommendations to medical diagnostics and autonomous vehicles.
- It typically involves **identifying patterns, making predictions, or taking actions** based on insights derived from vast amounts of data.


MACHINE LEARNING TYPES
Supervised learning
Unsupervised learning
Semi-supervised learning
Reinforcement learning
CONTINUOUS TARGET VARIABLE
CATEGORICAL TARGET VARIABLE
TARGET VARIABLE NOT AVAILABLE
CATEGORICAL TARGET VARIABLE
CATEGORICAL TARGET VARIABLE
TARGET VARIABLE NOT AVAILABLE
REGRESSION
CLASSIFICATION
CLUSTERING
ASSOCIATION
CLASSIFICATION
CLUSTERING
CLASSIFICATION
CONTROL
STOCK PRICES PREDICTION
FACE RECOGNITION
FAKE NEWS IDENTIFICATION
RECOMMENDER SYSTEMS
DOCUMENT CLASSIFICATION
LOCALIZING OBJECTS
OPTIMIZED ENERGY CONSUMPTION
SELF-DRIVING VEHICLES

Neural Networks: The Powerhouse of Deep Learning



Structure: Input layer, hidden layers, output layer.

Neurons: Basic units performing computations (weighted sum + activation function).

Deep Learning: NNs with multiple hidden layers, capable of learning hierarchical features.

Applications: Image recognition, natural language processing, speech recognition.

Training ML Models: Cost Functions and Optimization

Cost Function (Loss Function):

A mathematical function that **quantifies the "error" or "cost" of a model's predictions** compared to the actual target values.

The goal of training an ML model is to **minimize this cost function**. A lower cost indicates a better-performing model.

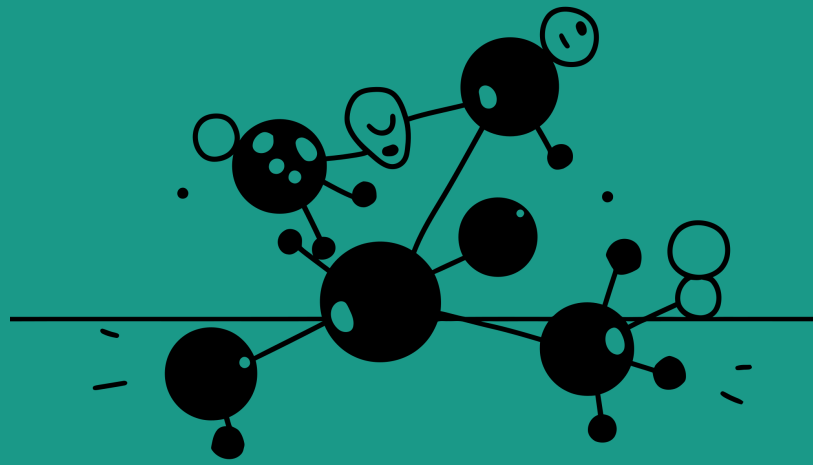
Examples: Mean Squared Error (for regression), Cross-Entropy Loss (for classification).

Optimization:

The iterative process of **adjusting the model's internal parameters** (e.g., weights and biases in a neural network) to minimize the cost function.

This involves finding the optimal set of parameters that make the model's predictions as close as possible to the true values.

Why Combine Quantum and ML?



The Rationale for a Quantum-Classical Symbiosis

NISQ Limitations: Current quantum devices are noisy, have few qubits, and short coherence times.

Purely Quantum ≠ Practical: Deep or large-scale quantum algorithms are not yet feasible.

Hybrid Advantage:

- **Classical:** Handles control logic, data processing, optimization.
- **Quantum:** Performs high-dimensional mapping and complex subroutines efficiently.

Bridge Strategy: Hybrid QML lets us explore quantum benefits today while hardware improves.

Hybrid Classical-Quantum Algorithms: The Feedback Loop

Core Idea: A classical optimizer iteratively tunes a parameterized quantum circuit.

Steps:

1. Initialize quantum circuit parameters (classical)
2. Execute quantum circuit (quantum)
   Measure output, compute cost (classical)
3. Update parameters to minimize loss (classical)
4. Repeat until convergence

Examples: VQE, QAOA, VQC — hybrid algorithms demonstrating this loop

Potential Advantages: Speed-ups in Linear Algebra Subroutines

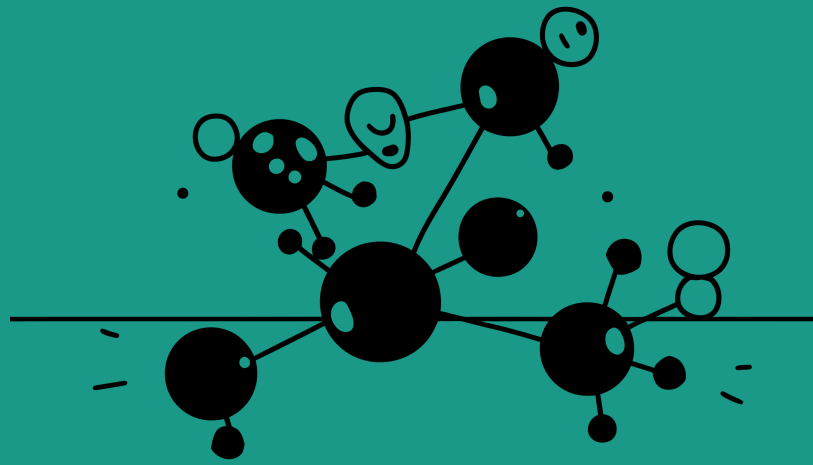
Speed-ups in Linear Algebra Subroutines

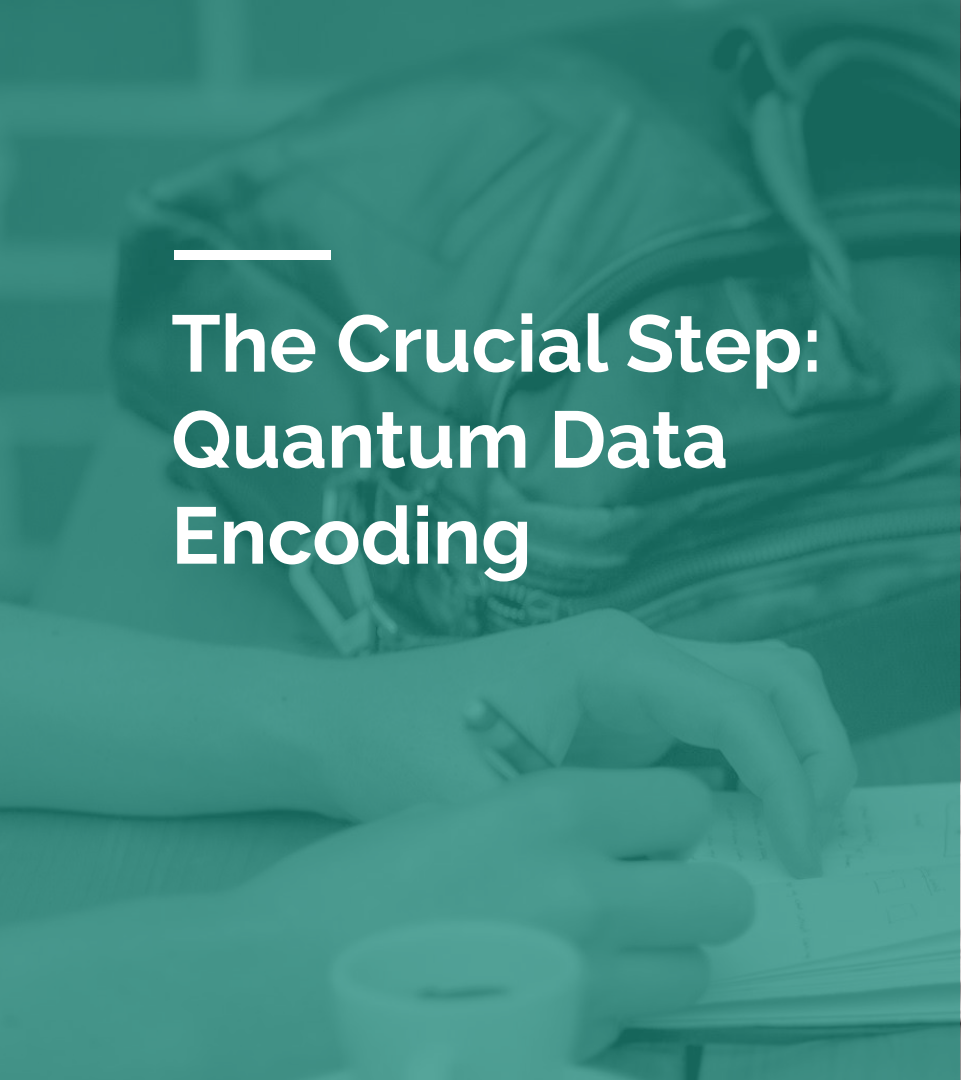
- Many ML algorithms heavily rely on linear algebra (e.g., matrix inversion, eigenvalue decomposition, solving linear equations).
- Quantum algorithms like HHL (Harrow, Hassidim, Lloyd) algorithm can solve linear equations exponentially faster than classical counterparts for certain sparse matrices.

Enhanced Expressivity and Parameter Efficiency

- **Expressivity:** Quantum circuits can explore a much larger Hilbert space than classical models with a comparable number of parameters, allowing for more complex data correlations to be captured.
- **Parameter Efficiency:** Potentially, a quantum model might achieve similar performance to a classical model with significantly fewer trainable parameters, making it more efficient for certain tasks.

Quantum Data Encoding





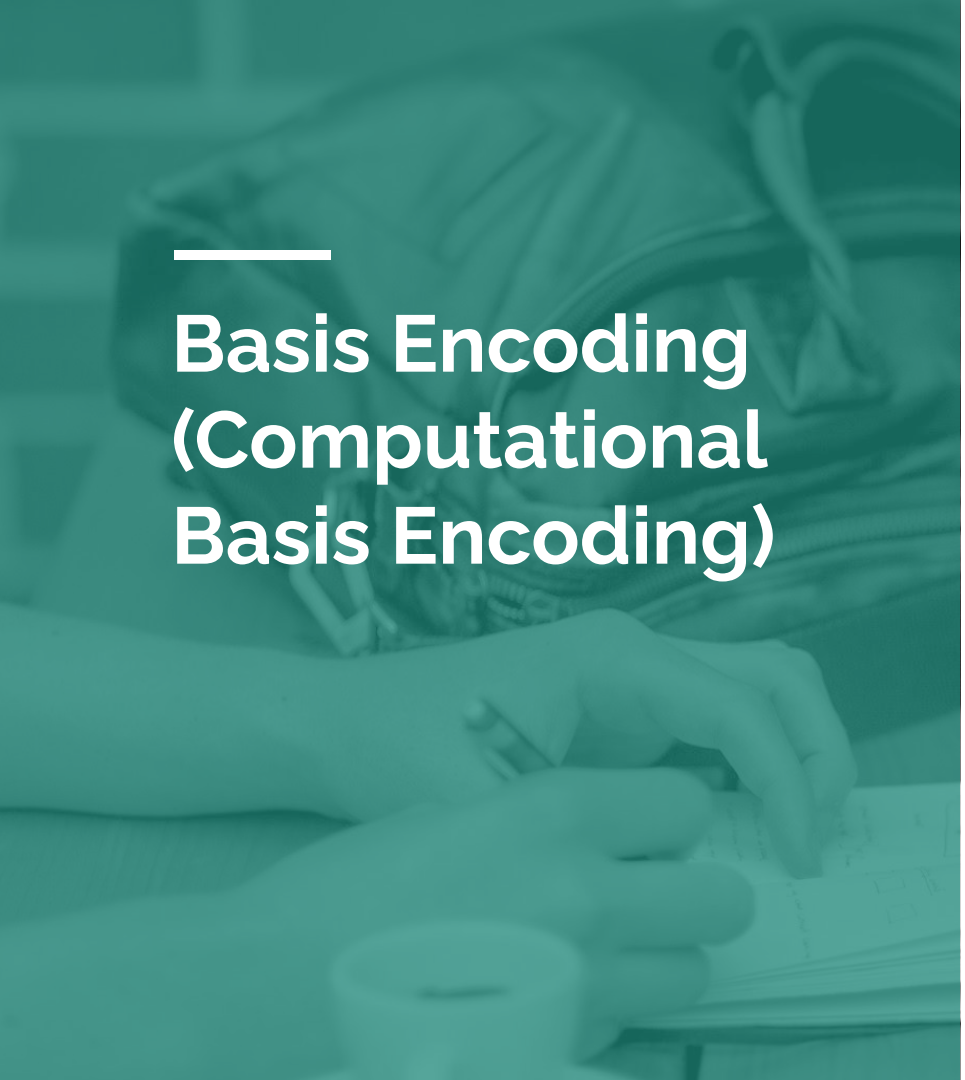
The Crucial Step: Quantum Data Encoding

The Bridge: Classical data, which is typically stored as binary bits, cannot be directly fed into a quantum computer. It must first be **translated or "encoded" into a quantum state**.

Challenge:. It transforms classical information into quantum information, represented by the amplitudes or phases of qubits.

Impact on Performance: The choice of encoding method is paramount. It directly influences:

- The **number of qubits** required.
- The **complexity of the quantum circuit** needed to prepare the state.
- The overall **performance and expressivity** of the QML algorithm.
- The potential for achieving a **quantum advantage**.



Basis Encoding (Computational Basis Encoding)

Method: Represents classical binary data directly into the computational basis states of qubits. Each classical bit corresponds to one qubit.

Example: Classical binary 101 becomes |101> .

Pros: Simple, intuitive, requires a number of qubits equal to the number of classical bits.

Cons: Does not fully utilize quantum properties like superposition or entanglement for data representation itself. Can be qubit-intensive for large classical inputs.

Amplitude Encoding

Method: Encodes classical real numbers into the amplitudes of an n-qubit quantum state.

Example: To encode 4 classical values, you need only 2 qubits.

Pros: Exponential compression of data (N qubits can represent 2N values), utilizes superposition effectively. Highly efficient for high-dimensional vectors.

Cons: Difficult to prepare arbitrary amplitude states on current hardware; requires complex quantum circuits for arbitrary data. Can be lossy if not all information is retained.

Angle Encoding (Parameter Encoding / Feature Mapping)

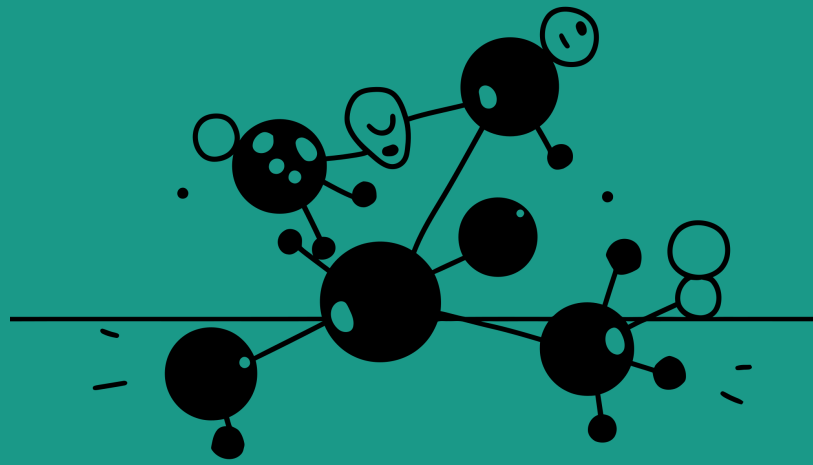
Method: Encodes classical data points by rotating qubits by angles proportional to the data values.

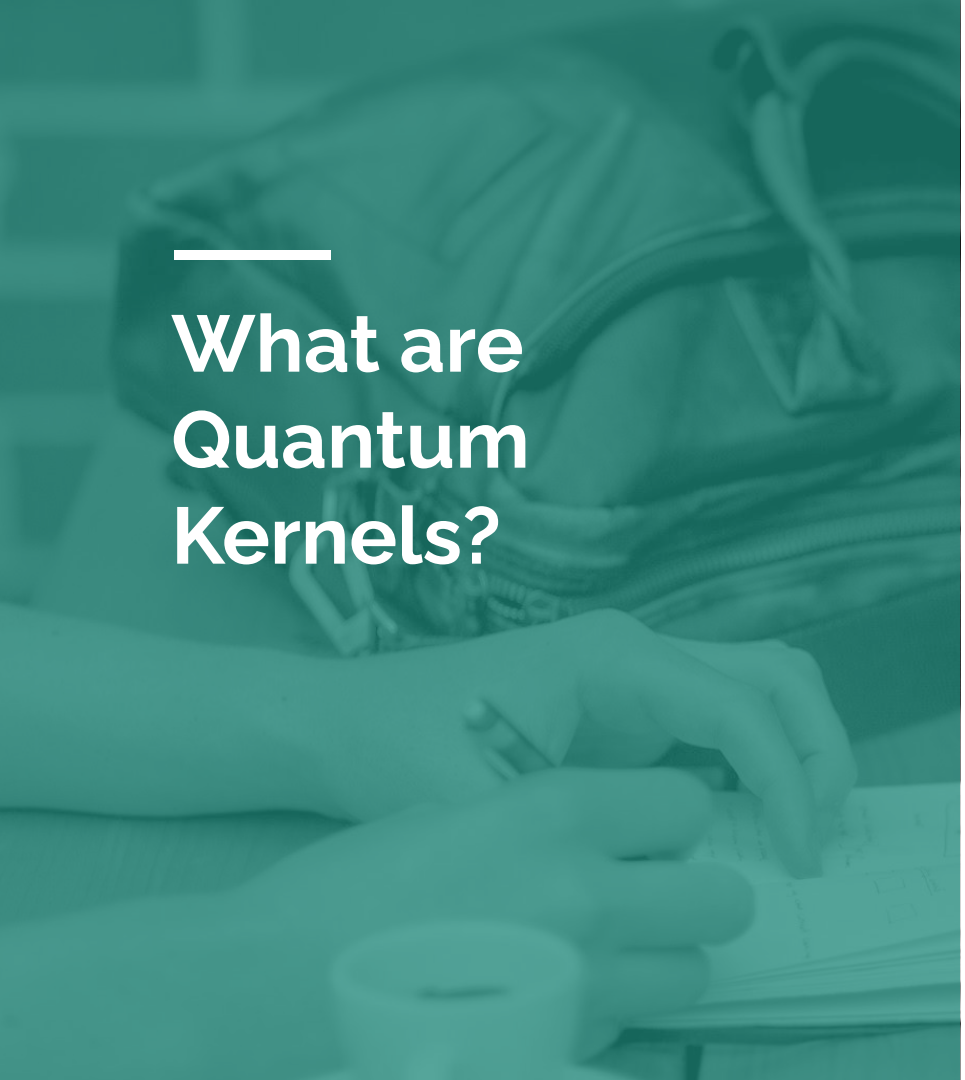
Example: A classical feature might be encoded by applying a Ry(x) or Rz(x) gate to a qubit.

Pros: Flexible, allows for non-linear feature maps, commonly used in Variational Quantum Classifiers (VQC) as it generates quantum feature spaces.

Cons: Can be challenging to design optimal feature maps that lead to quantum advantage; the mapping might not always be easy to visualize.

Quantum Kernels





What are Quantum Kernels?

A Quantum Kernel uses a Quantum Computer to calculate the similarity between data points after encoding them into quantum states.

Quantum kernels form a foundational component of quantum-enhanced machine learning algorithms, particularly in classification problems. Leveraging quantum mechanical principles, these kernels enable efficient similarity computation in high-dimensional Hilbert spaces, potentially offering computational advantages over classical counterparts.

Inner Product of Quantum States

Mathematically: Let ϕ be a feature map over the input Ω, encoding our data into quantum information. A quantum kernel is a kernel given by the inner product (in Dirac notation), defined on our space of quantum states, between two quantum states:

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

This represents the squared fidelity or overlap between the two quantum states and quantifies their similarity in the quantum feature space.

QSVM idea: Compute kernel via quantum circuits

- The Quantum Support Vector Machine (QSVM) algorithm incorporates quantum kernels into the classical SVM framework.

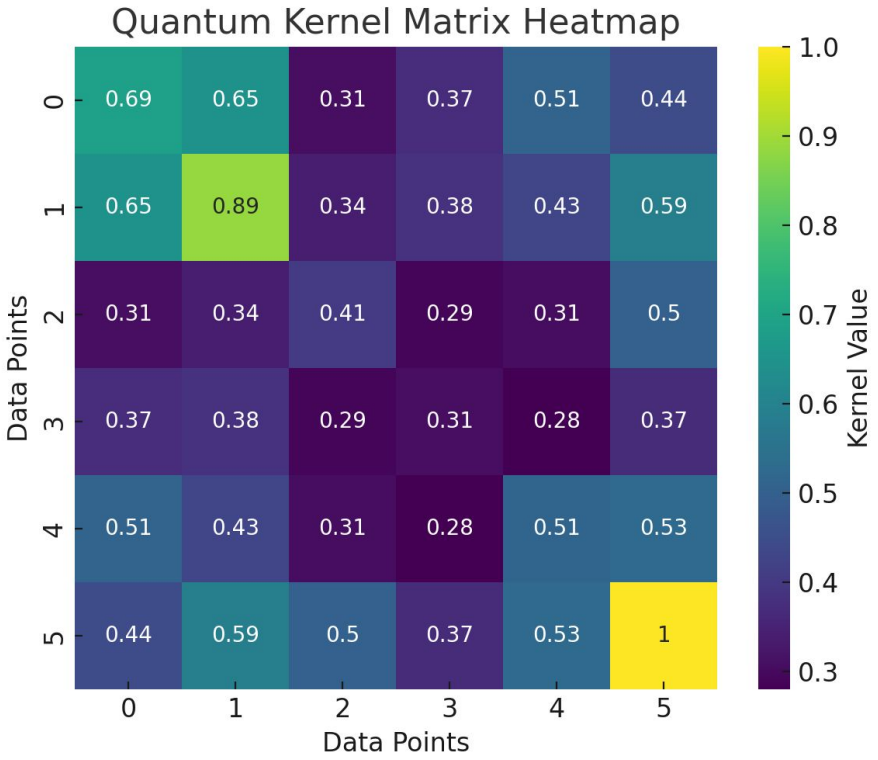
- The quantum computer is tasked with evaluating the kernel matrix K, where each element is computed via quantum circuits that measure the inner products of data-encoded quantum states.

- The classification boundary is then determined through a classical SVM solver using the quantum kernel matrix.

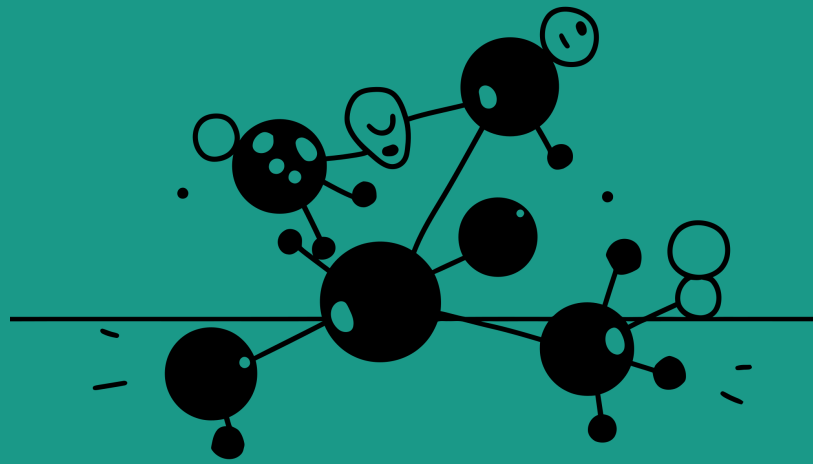
Use in classification Problems

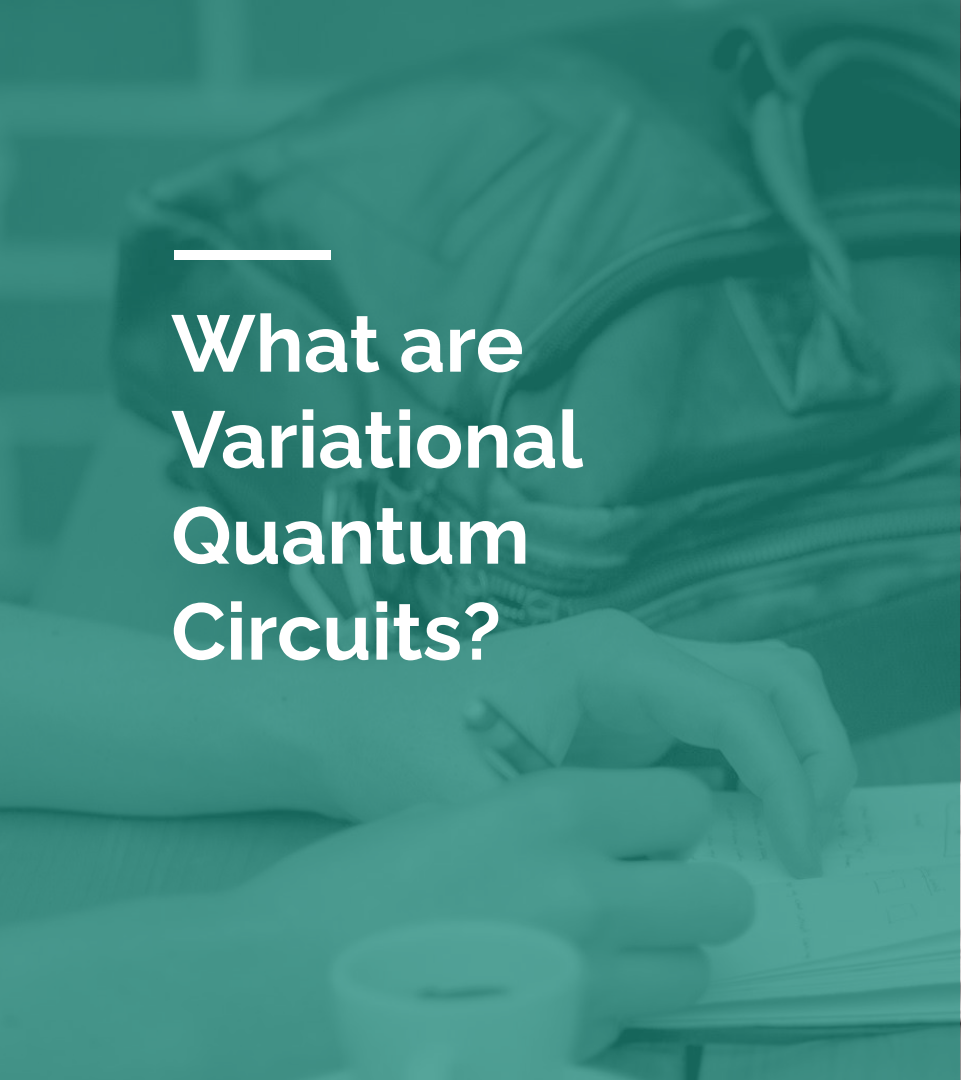
- Quantum kernels have been applied to supervised learning tasks, especially binary classification, where the structure of the data may be non-linearly separable in classical space.
- By mapping data into a quantum Hilbert space, QSVM can discover decision boundaries that are otherwise difficult to achieve with standard kernels.
- Experimental implementations on NISQ devices have demonstrated promise in domains such as finance, material science, and biology.

Visual: kernel matrix heatmap



Variational Quantum Circuits (VQC)





What are Variational Quantum Circuits?

Variational Quantum Circuits (VQCs) are hybrid quantum-classical models at the heart of variational algorithms such as the Variational Quantum Eigensolver (VQE) and Quantum Neural Networks (QNNs).

These circuits are composed of **parameterized quantum gates** whose parameters are optimized through a classical feedback loop.

Parametrized gates: Ry(theta), Rz(phi)

- **Ry(θ)**: Rotation around the Y-axis by angle θ
- **Rz(φ)**: Rotation around the Z-axis by angle φ

They can be applied as:

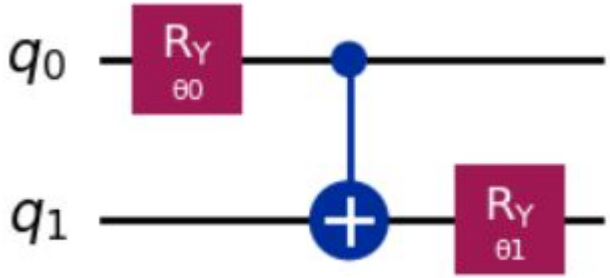
```
qc.ry(theta, qubit)
qc.rz(phi, qubit)
```

Optimization Loop

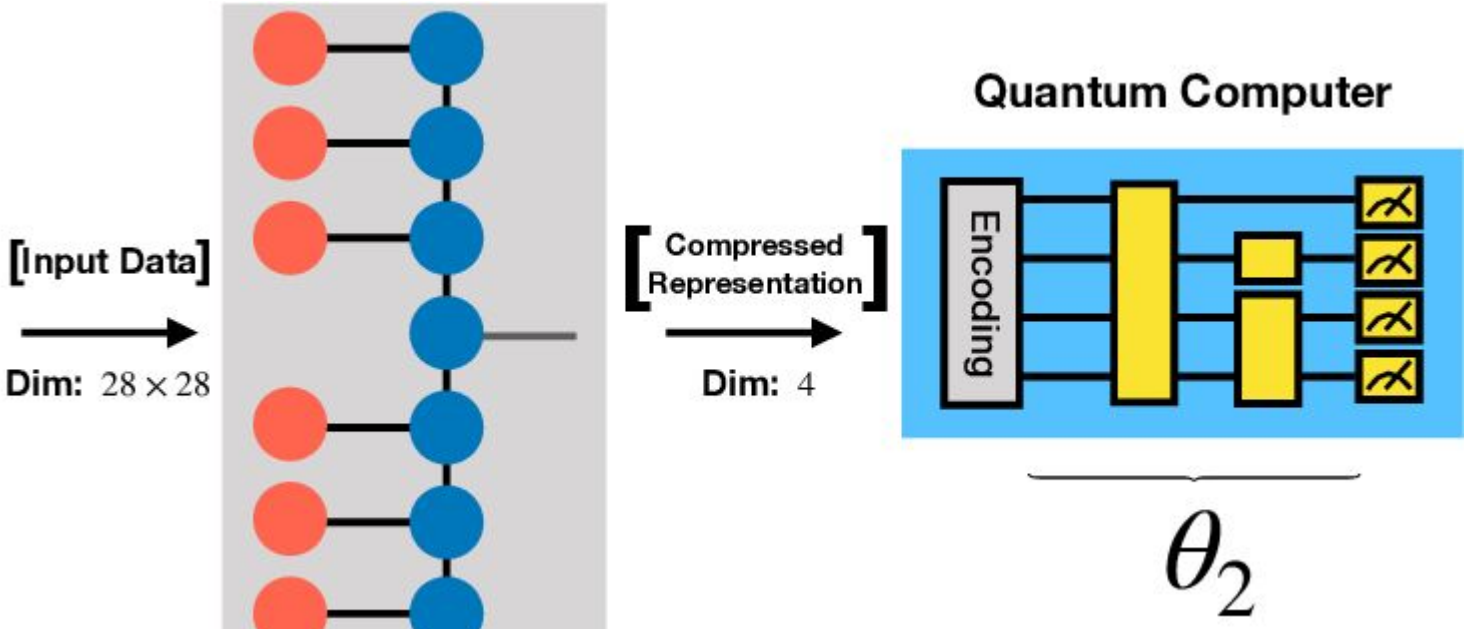
- Initialize parameters θ, φ randomly.
- Construct the quantum circuit with these parameters.
- Run the circuit on a simulator or real device to get measurement results.
- Calculate a cost function from the results.
- Use a classical optimizer (e.g., COBYLA, SPSA, Adam) to update parameters to minimize the cost.
- Repeat until convergence.

```
from qiskit import QuantumCircuit
from qiskit.circuit import Parameter
theta = [Parameter('θ0'), Parameter('θ1')]
# Create variational quantum circuit
qc = QuantumCircuit(2)
qc.ry(theta[0], 0)
qc.cx(0, 1)
qc.ry(theta[1], 1)
qc.draw('mpl')
```

Output:

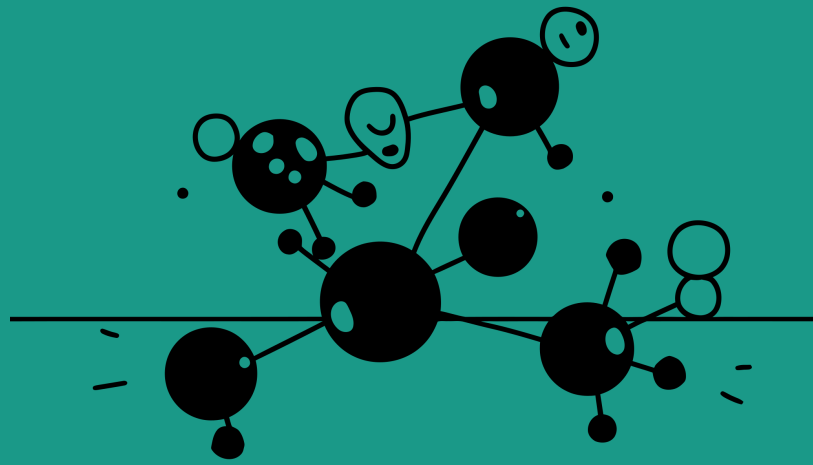


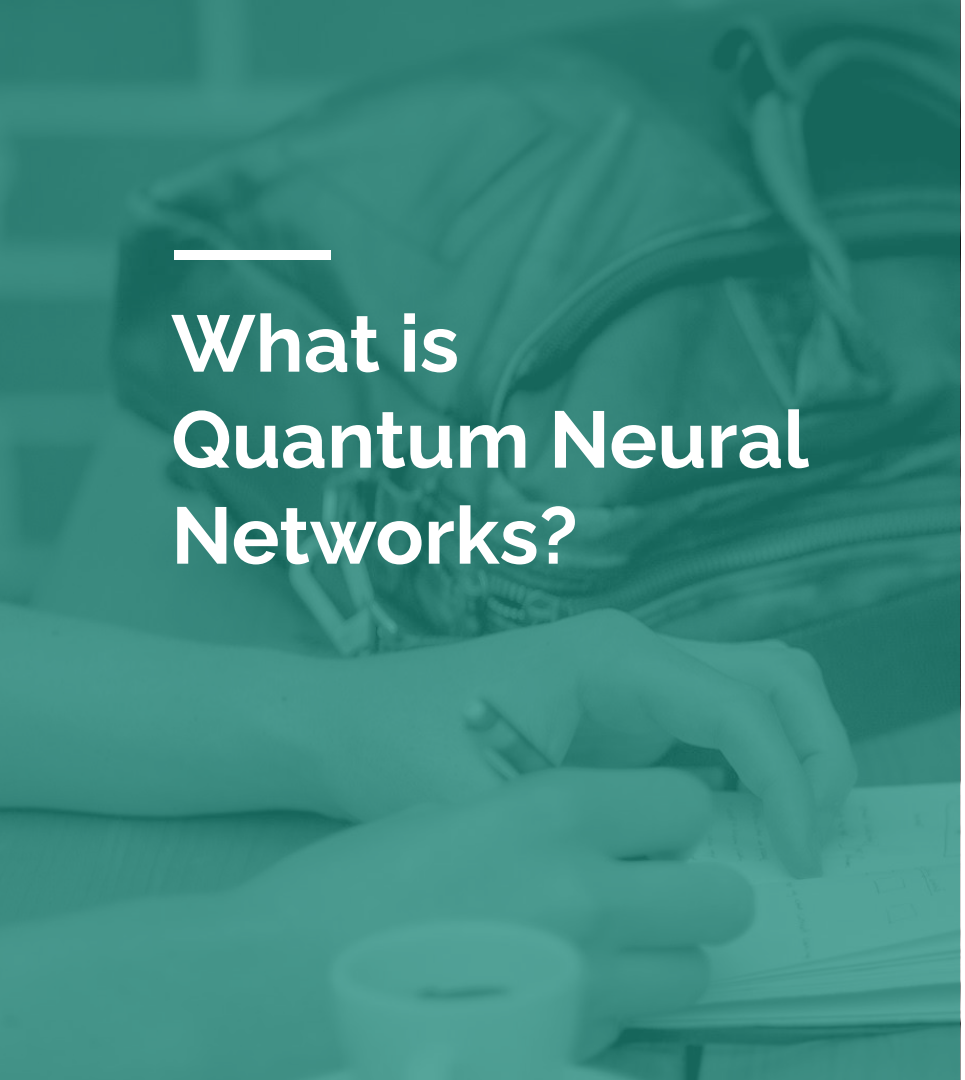
VQC Architecture



Hybrid TN-VQC Architecture. The tunable parameters labeled with θ 1 and θ 2 are the parameters for the MPS and VQC respectively.

Quantum Neural Networks





What is Quantum Neural Networks?

- Quantum Neural Networks (QNNs) represent a groundbreaking convergence of quantum computing and artificial intelligence.
- They promise to revolutionize machine learning by harnessing the unique properties of quantum mechanics.
- Following slides explores the core concepts, challenges, and future potential of QNNs.

Layered Parameterized Quantum Circuits (PQCs)

A Layered Parameterized Quantum Circuit (PQC) is a quantum circuit made up of gates that have tunable (trainable) parameters, arranged in layers—similar to layers in a neural network

Explanation :

- **Parameterized Quantum Gates:**
  - Some quantum gates, like rotation gates (e.g., Ry(θ), Rz(ϕ)), can take real-valued parameters.
  - These parameters are like the weights in classical machine learning models.
- **Layered Architecture:**
  - PQCs are built in repeated layers:
    1. Each layer has rotation gates to modify individual qubits.
    2. Followed by entangling gates (like CNOT) to create interaction between qubits.
- **Training the Circuit:**
  - The parameters are adjusted using classical optimization algorithms (e.g., gradient descent).
  - The goal is to minimize a cost function based on the quantum circuit's output.
- **Use in Quantum Neural Networks (QNNs):**
  - PQCs form the core of QNNs — they process input quantum data and are trained to solve problems like classification or pattern recognition.

Quantum Perceptron Analogy

The Quantum Perceptron is the quantum counterpart of the classical perceptron, used in neural networks. Instead of using real-valued inputs and weights, it uses qubits and parameterized quantum gates.

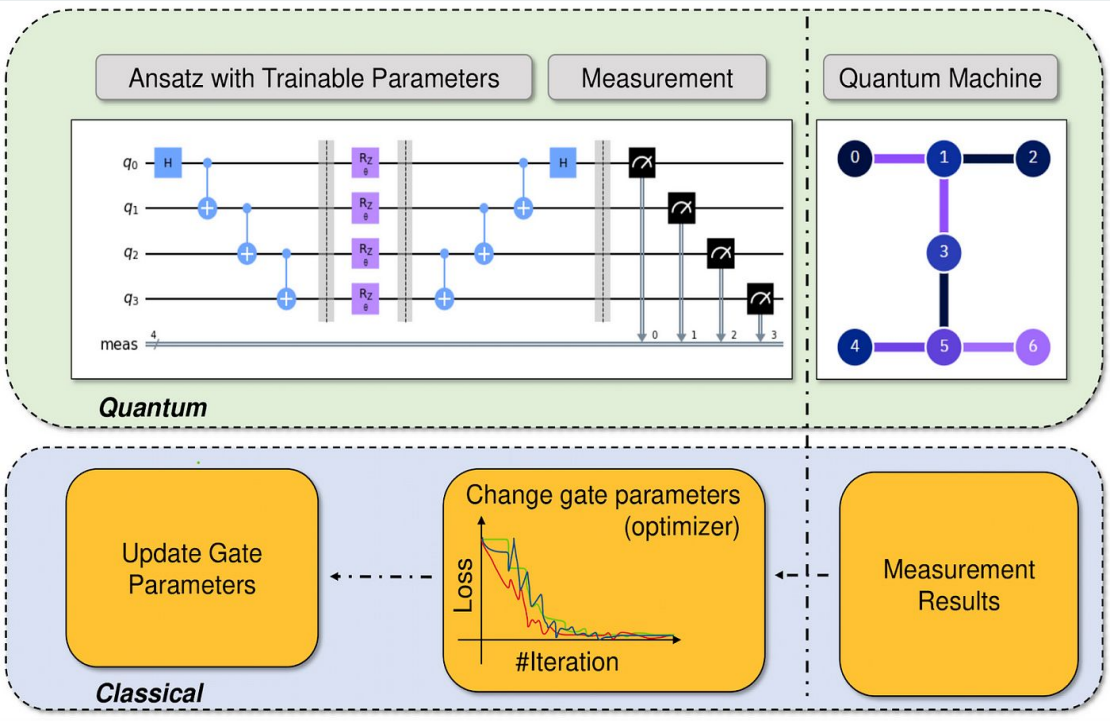
- Inputs are encoded as quantum states.
- Parameterized gates (like Ry(θ)) play the role of adjustable weights.
- Entanglement allows complex interactions between qubits.
- Measurement gives a binary output (0 or 1), like an activation function.

Challenge – Barren Plateaus

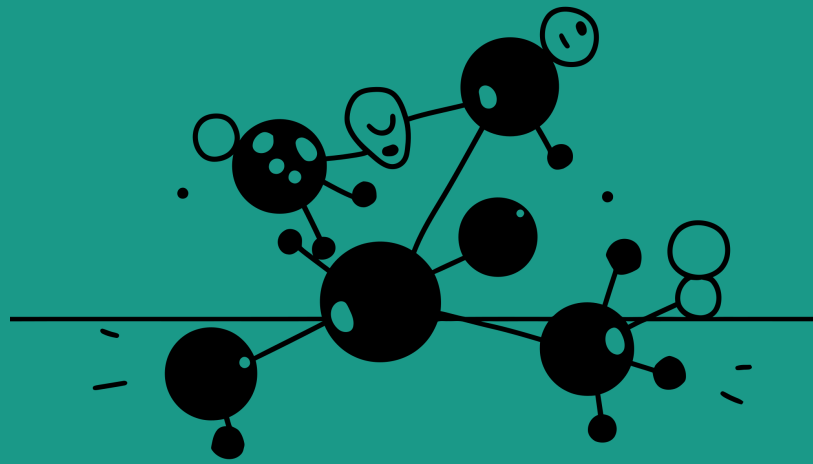
Barren plateaus are a major challenge in training quantum neural networks.

- They occur when the gradient of the cost function becomes extremely small as the number of qubits or circuit depth increases.
- This makes it very hard for optimizers to learn, since parameter updates become ineffective.
- It's similar to getting stuck in a flat region of a loss landscape in classical deep learning.
- More common in deep or random quantum circuits.
- Solutions include using shallower circuits, smart initialization, or layer-wise training.
- In short: barren plateaus slow down or stop learning in quantum machine learning models.

Visual: VQC with multiple layers



Key QML Algorithms



What is QML Algorithm?

Leveraging Quantum

QML uses quantum effects for efficiency.

Faster Problem Solving

Potential for speedups over classical methods.

Enhanced Capabilities

Exploits superposition and entanglement.

Quantum Versions

Classical ML algorithms adapted for quantum.

QSVM – Quantum Support Vector Machine

QSVM applies quantum computing principles to enhance classical SVM algorithms, particularly for classification tasks.

- **Quantum Kernel Estimation:**
  - QSVM leverages quantum states to compute kernel functions efficiently.
  - Uses quantum computers to perform feature space transformations, enabling better separation of non-linear data.
- **Advantages:**
  - Offers exponential speed-up in calculating kernel matrices.
  - Handles high-dimensional data transformations efficiently.
- **Applications:**
  - Image recognition, bioinformatics, and financial modeling.

QSVM – Quantum Support Vector Machine

Steps in QSVM Implementation:

- Data Encoding: Map classical data into quantum states.
- Quantum Kernel Calculation: Use quantum circuits to compute inner products of quantum states.
- Classification: Train the SVM using quantum-enhanced kernel values.
- Prediction: Apply the model to classify unseen data.

QkNN – Quantum k-Nearest Neighbors

QkNN enhances the classical kNN algorithm by exploiting quantum mechanics for faster distance computations and improved nearest-neighbor searches.

- **Features and Principles:**
  - **Quantum Distance Calculation:**
    1. Uses quantum algorithms like Grover's search for faster distance computations.
    2. Reduces the time complexity of finding the nearest neighbors.
  - **Quantum Parallelism:**
    1. Evaluates distances to multiple points simultaneously using quantum superposition.
  - **Advantages:**
    1. Exponentially faster for large datasets.
    2. Reduces computational bottlenecks in high-dimensional data spaces.

QkNN – Quantum k-Nearest Neighbors

Steps in QkNN Implementation:

- Quantum Data Encoding: Encode data points as quantum states.
- Distance Computation: Use quantum algorithms to calculate distances in parallel.
- Neighbor Selection: Identify the k-nearest neighbors using quantum searches.
- Class Prediction: Use majority voting among the neighbors for classification.

QPCA – Quantum Principal Component Analysis

QPCA is the quantum version of PCA, a dimensionality reduction technique that uses quantum algorithms for enhanced computational efficiency.

- **Quantum State Preparation:** Encodes the covariance matrix as a quantum state.
- **Eigenvalue and Eigenvector Estimation:** Uses quantum phase estimation to extract principal components.
- **Advantages**:
  - Exponential speed-up for calculating eigenvalues and eigenvectors.
  - Handles large-scale data efficiently.
- **Steps in QPCA Implementation:**
  - Data Preparation: Represent the data covariance matrix in a quantum state.
  - Quantum Phase Estimation: Compute eigenvalues and eigenvectors of the covariance matrix.
  - Dimensionality Reduction: Select top components based on eigenvalues.
  - Projection: Transform the original data into the reduced-dimensional space.

qGAN – Quantum Generative Adversarial Network

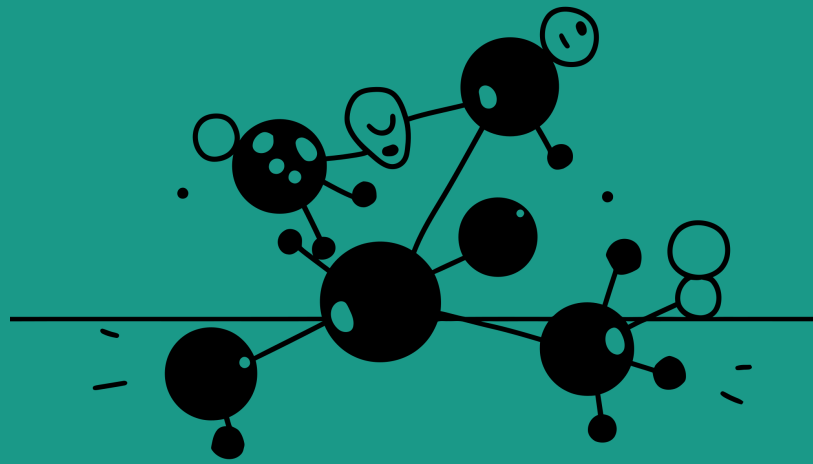
Quantum adaptation of GANs for generating data distributions and patterns.

- **Features and Principles:**
  - Quantum Generator: Employs quantum circuits to create synthetic data. Utilizes quantum superposition and entanglement for complex distributions.
  - Quantum or Classical Discriminator: Compares generated data with real data to iteratively improve the generator.
- **Advantages:**
  - Leverages quantum parallelism for efficient training.
  - Models high-dimensional and intricate data distributions.
- **Steps in qGAN Implementation:**
  - Quantum Generator Design: Construct a quantum circuit encoding random quantum states.
  - Discriminator Setup: Use a classical or quantum discriminator to evaluate generated vs. real data.
  - Training: Alternate updates between generator and discriminator to minimize loss.
  - Data Generation: Generate new data samples using the trained quantum generator.

Table: Classical vs Quantum variants

Algorithm	Classical Version	Quantum Version	Quantum Advantage
SVM	Support Vector Machine	**QSVM**	Efficient kernel computation in high dimensions
k-NN	k-Nearest Neighbors	**QkNN**	Faster similarity search via quantum parallelism
PCA	Principal Component Analysis	**QPCA**	Exponential speed-up in extracting eigenvalues
GAN	Generative Adversarial Network	**qGAN**	Efficient modeling of complex data distributions

Challenges in QML



Noisy Quantum Gates

- **What is it?**
  - Quantum gates are not perfect on real hardware.
  - They introduce errors due to interaction with the environment (decoherence) and imperfect control.
- **Types of Quantum Noise:**
  - Depolarizing Noise: Qubit randomly loses its state with some probability.
  - Dephasing Noise: Loss of quantum phase information.
  - Amplitude Damping: Qubit tends to relax from $|1\rangle$ to $|0\rangle$ state.
  - Gate Errors: Inaccuracies in applying unitary operations.
- **Impact on QML:**
  - Degrades the quality of learning.
  - Affects repeatability and accuracy of measurement outcomes.
  - Makes training unstable or ineffective.

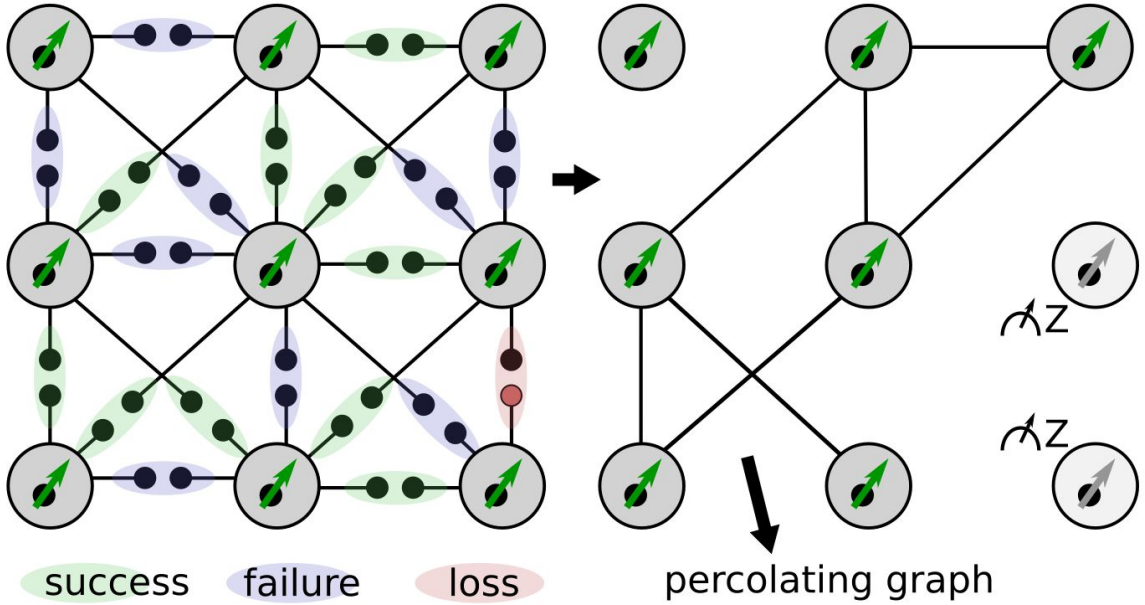
Barren Plateaus

- **Definition:**
  - A phenomenon where the cost function gradients approach zero, making it difficult to optimize quantum circuits.
- **Causes:**
  - Random Initialization: Parameters initialized with near-zero expected gradients.
  - Circuit Complexity: Deep circuits lead to exponentially vanishing gradients.
  - Global Cost Functions: Cost functions based on global quantum state properties are more prone to barren plateaus.
- **Effects:**
  - Slows training progress.
  - Makes optimization computationally infeasible for large systems.
- **Mitigation Strategies:**
  - Localized Cost Functions: Focus on local properties of the system.
  - Layerwise Training: Optimize a few layers at a time.
  - Parameter Initialization: Start with parameters that avoid flat regions.

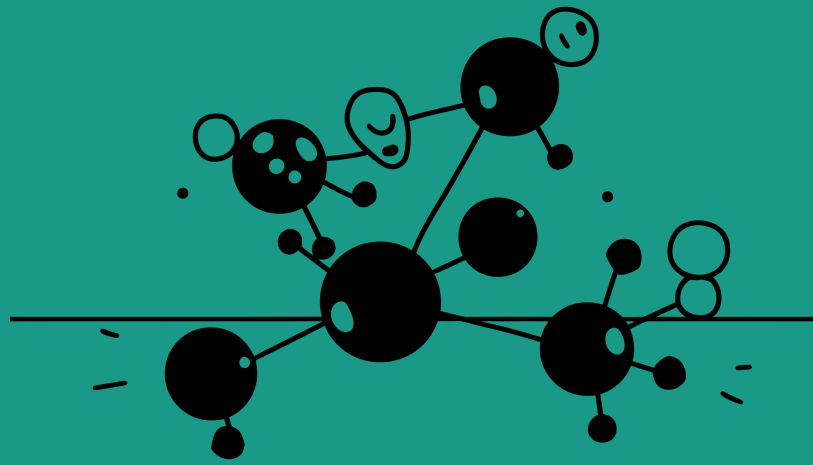
Data Encoding Bottleneck

- **Definition:**
  - Occurs when encoding classical data into quantum states becomes inefficient or limits the quantum model's capacity.
- **Causes:**
  - High Dimensionality: Encoding large datasets requires many qubits.
  - Complex Encoding: Deep circuits introduce noise and errors.
  - Limited Circuit Depth: Shallow circuits fail to capture complex data patterns.
- **Effects:**
  - Reduces quantum model expressivity.
  - Limits the effectiveness of quantum machine learning.
- **Mitigation Strategies:**
  - Efficient Encoding: Use hybrid methods like feature map optimization.
  - Dimension Reduction: Preprocess with techniques like PCA before quantum encoding.
  - Data Reuse: Recycle encoded states for multiple computations.

Loss surface comparison:



QML Libraries and Tools



1.

Prominent QML Libraries in 2025

Key players in QML include **Qiskit ML, PennyLane, TensorFlow, Quantum+, Amazon Braket ML** and **QMLTorch.**
Each offers unique features for quantum data processing

2.

Understanding the Ecosystem Map

An ecosystem map will illustrate the **interconnections** between various libraries and their **hardware support,** Showcasing the collaborative landscape of QML tools.

Exploring Key Features of Qiskit

01 Quantum Circuits

Easily create and manipulate **quantum circuits** to leverage their power in various machine learning tasks.

02 Hybrid Quantum-Classical Algorithms

Combining **quantum** and **classical resources** leads to improved performance and results in machine learning applications.

03 Data Handling

Efficient tools for importing and managing datasets, making it easier to work with large amounts of data in quantum models.

04 Community Contributions

The **open-source community** enhances Qiskit Machine Learning, providing additional tools and features for users globally.

Quantum Circuit for Classification

using qiskit for machine learning applications

Quantum Circuit Creation

This code snippet demonstrates how to create a simple **quantum circuit** for **classification** using Qiskit Machine Learning. It highlights the initial setup required for building quantum models.

Integration with Classical Models

The integration of **quantum circuits** with classical models shows the **versatility** of Qiskit in practical **machine learning** scenarios, enabling hybrid approaches for better performance.

Qiskit Machine Learning

Qiskit Machine Learning provides tools and techniques for implementing **quantum algorithms** in various machine learning tasks, showcasing its relevance in the field.

Practical Applications

This tutorial illustrates the **practical applications** of quantum circuits in machine learning, emphasizing their potential in solving real-world classification problems.

PennyLane Overview

- **Quantum hardware compatibility**

  PennyLane supports various **quantum hardware**, allowing users to run their algorithms on different platforms seamlessly.

- **Automatic differentiation**

  This library offers **automatic differentiation** for quantum circuits, enabling efficient optimization of quantum algorithms.

- **Integration with machine learning frameworks**

  PennyLane easily integrates with popular **machine learning libraries** such as PyTorch and TensorFlow, enhancing its usability.

- **Quantum neural networks**

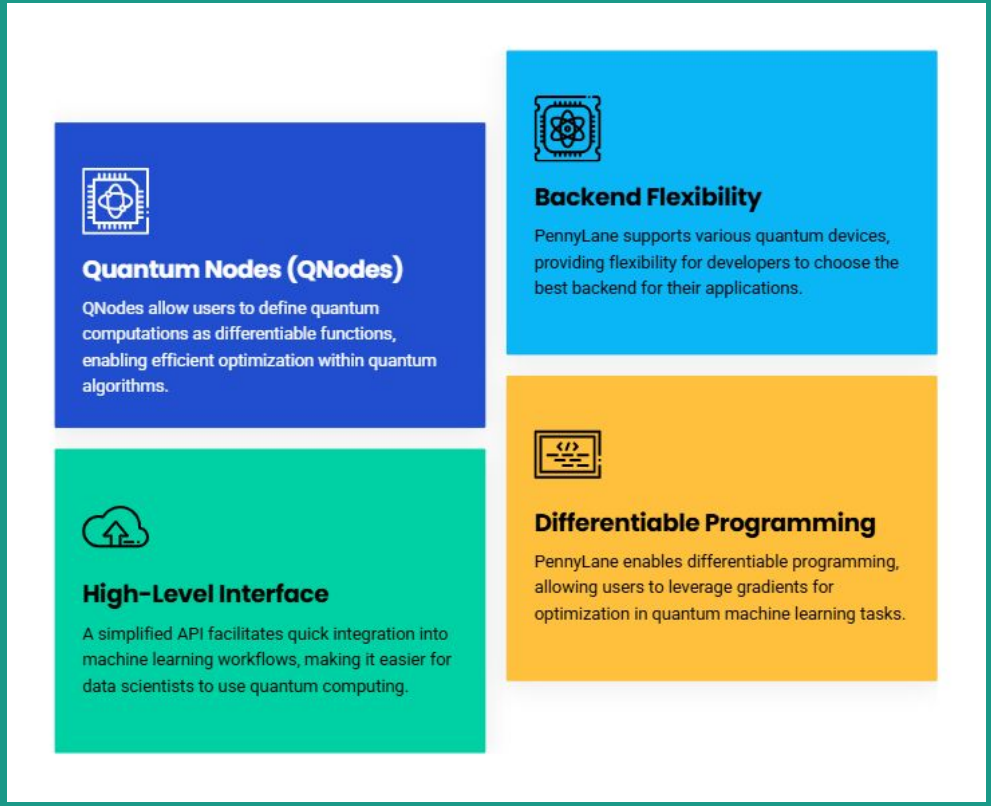
  Utilize **quantum neural networks** for advanced machine learning tasks, leveraging quantum properties for improved performance.

- **Quantum-enhanced feature extraction**

  PennyLane facilitates **quantum-enhanced feature extraction**, improving the performance of classical machine learning models.

PennyLane : Core Functionalities Overview

Discover the essential features of PennyLane that enhance quantum computing and machine learning integration



TensorFlow Quantum Overview

- **Extension of TensorFlow for quantum**

  TensorFlow Quantum (TFQ) enhances TensorFlow by enabling **quantum computing** integration into machine learning workflows, facilitating advanced modeling techniques.

- **Native quantum circuit support**

  TFQ offers **native support** for **quantum circuits**, allowing users to seamlessly incorporate quantum algorithms into their machine learning projects.

- **Hybrid quantum-classical models**

  The framework provides tools for creating **hybrid models** that combine both **quantum** and **classical** computing techniques for enhanced performance.

- **Quantum reinforcement learning**

  TFQ can be applied in **quantum reinforcement learning** scenarios, helping to solve complex decision-making problems using quantum techniques.

- **Circuit learning for datasets**

  Utilizing **quantum circuit learning** allows for the analysis and understanding of **complex datasets**, paving the way for innovative data-driven solutions.

Exploring Key Features of TensorFlow Quantum

Seamless Integration

TensorFlow Quantum allows for easy and efficient integration with existing TensorFlow models, promoting a smooth transition into quantum computing applications.

Quantum Circuit Layer

It features a dedicated **Quantum Circuit Layer** that enables users to directly work with quantum circuits, facilitating the design and execution of quantum algorithms.

Advanced Optimization Tools

The library includes **advanced optimization** tools specifically designed for enhancing the performance of quantum algorithms, improving their effectiveness and efficiency.

Support for Quantum ML

TensorFlow Quantum supports various **quantum machine learning** workflows, making it a versatile tool for researchers and developers in the quantum space.

Comparative Analysis of QML Libraries

Exploring features and usability of Qiskit, PennyLane, and TensorFlow Quantum

PennyLane

1. **Supported Hardware**: Various quantum devices
2. **Supported Hardware**: Various quantum d
3. **Ease of Use**: High

Qiskit

1. **Primary Use**: Quantum Circuits
2. **Supported Hardware**: IBM Q hardware
3. **Ease of Use**: Moderate

Synergies of Quantum Machine Learning Libraries

Qiskit Capabilities

Qiskit provides robust quantum circuit capabilities, enabling complex quantum computations and simulations.

PennyLane Integration

PennyLane enables seamless integration with classical ML frameworks, allowing hybrid quantum-classical applications.

TensorFlow Quantum Support

TensorFlow Quantum offers strong support for deep learning architectures, facilitating advanced model training.

Collaborative Potential

Leveraging strengths from each can lead to enhanced **QML applications**, fostering innovation in quantum computing.

QML Applications Overview

Exploring Quantum-Machine Learning Synergy

01

Quantum Bits (Qubits)

Qubits can represent multiple states simultaneously, enabling **parallel processing** for more efficient computation.

02

Concept of Quantum Supremacy

Quantum Supremacy refers to the capability of quantum computers to perform calculations that surpass classical computers, marking a significant advancement in technology.

03

Diverse Industrial Applications

Applications Across Industries include fields like financial modeling and drug discovery, showcasing the potential of quantum computing to transform various sectors.

Quantum Finance Innovations

Transforming Financial Services with QML

Advanced Risk Analysis	Quantum algorithms can process vast datasets to predict market trends and **risks** more accurately, enhancing decision-making processes for financial institutions.
Optimized Portfolio Management	Utilizing enhanced models for **optimizing** asset allocation, QML enables better investment strategies to improve returns.
Enhanced Fraud Detection	QML facilitates **faster** identification of fraudulent activities through advanced pattern recognition, significantly reducing potential losses.
IBM's Case Study	IBM's Quantum Financial Services initiative showcases practical applications of QML in finance, demonstrating its potential impact on the industry.

Quantum-Level Chemistry Innovations

01 **Advanced Molecular Simulation**

QML allows for complex simulations that classical computers cannot handle, enhancing our understanding of chemical processes.

02 **Accelerated Drug Discovery**

Predictive modeling of molecular interactions speeds up the development of new pharmaceuticals, making drug discovery more efficient.

03 **Innovative Material Science**

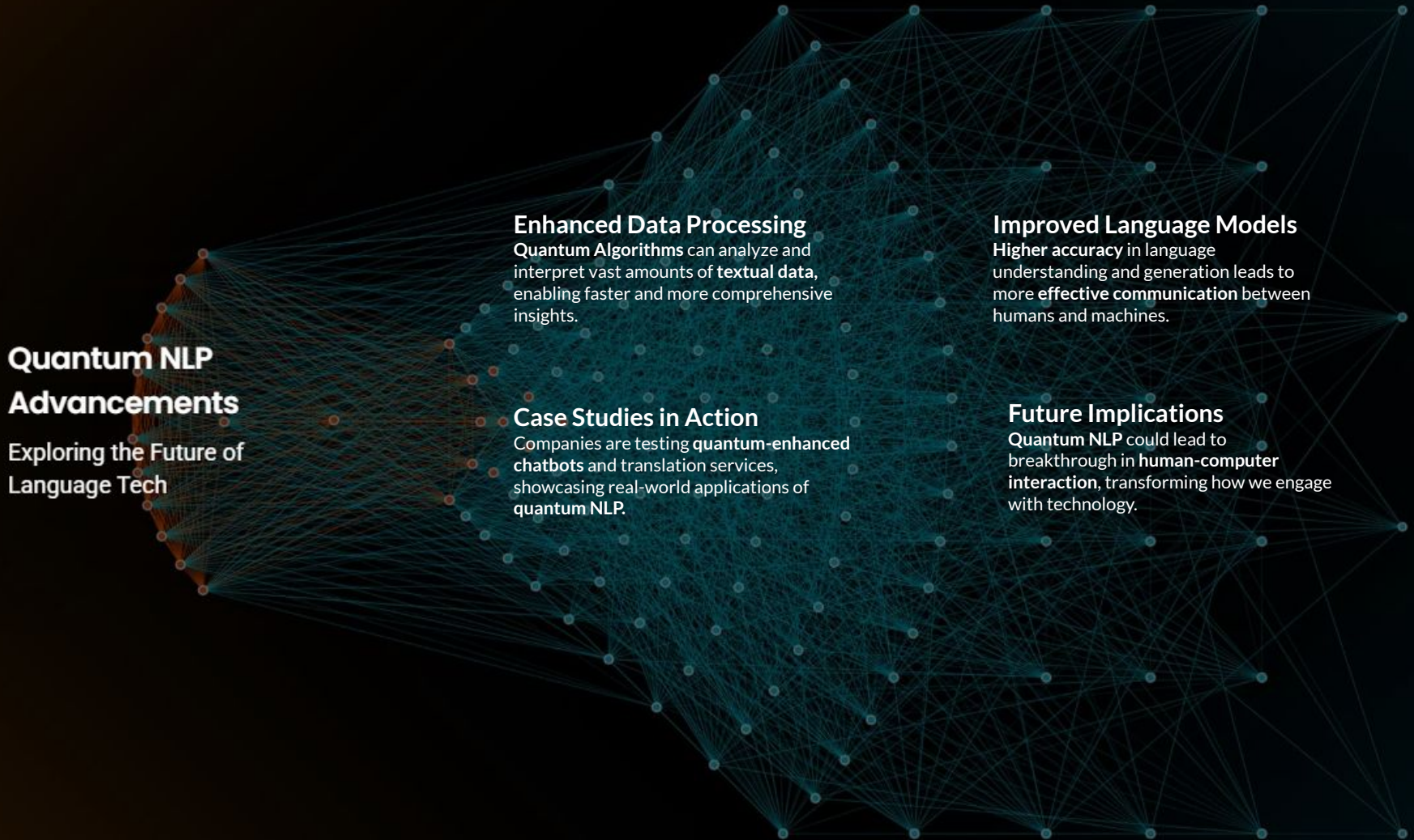
Designing new materials with specific properties at the molecular level is revolutionized by QML techniques, leading to novel applications.

04 **Google's Chemical Reaction Simulations**

Google's initiatives in simulating chemical reactions highlight the transformative potential of QML in pharmaceuticals and beyond.

Quantum NLP Advancements

Exploring the Future of Language Tech



Enhanced Data Processing

Quantum Algorithms can analyze and interpret vast amounts of **textual data,** enabling faster and more comprehensive insights.

Improved Language Models

Higher accuracy in language understanding and generation leads to more **effective communication** between humans and machines.

Case Studies in Action

Companies are testing **quantum-enhanced chatbots** and translation services, showcasing real-world applications of **quantum NLP.**

Future Implications

Quantum NLP could lead to breakthrough in **human-computer interaction**, transforming how we engage with technology.

IBM’s Quantum Computing Innovations

IBM Quantum Experience		A cloud-based platform enabling users to explore and experiment with **quantum algorithms** effectively.
Real-world Applications		IBM partners with various industries, leveraging **quantum solutions** to tackle complex problems and enhance performance.
Research Contributions		Ongoing advancements in **qubit technology** and **error correction** methods are pushing the boundaries of quantum computing.
Future Plans		IBM aims to expand its **quantum capabilities** and resources, making technology more accessible for broader applications.

Google Quantum AI Innovations

- **Sycamore Processor Supremacy**

  The **Sycamore Processor** achieved **quantum supremacy** in a specific computational task, showcasing Google's leadership in quantum technology.

- **Open Source Quantum Software**

  Development of tools like **Cirq** allows users to easily engage in **quantum programming**, promoting accessibility and innovation in the field.

- **Collaborative Research Efforts**

  Google's partnerships with **academia and industry** are crucial for advancing **quantum applications** and fostering innovation across sectors.

- **Future of Quantum Applications**

  Google's vision includes leveraging **quantum technology** for practical uses in various sectors, aiming for transformative impacts on industry practices.

Challenges in Quantum Implementation

Exploring current and future obstacles in QML applications

Scalability Issues

Developing scalable quantum systems that can effectively tackle real-world problems is a major challenge.

High Error Rates

Reducing error rates in quantum computations is essential for reliable performance in applications.

Seamless Integration

Integrating quantum solutions with existing classical systems without friction is crucial for adoption.

Talent Shortage

The demand for skilled professionals in quantum computing outpaces the current supply, hindering progress.

Future Insights on QML Applications

Exploring advancements, adoption, and ethical discussions in QML.

Continuous advancements in tech

Expect ongoing improvements in **quantum technology**, enhancing performance and capabilities.

Wider adoption across industries

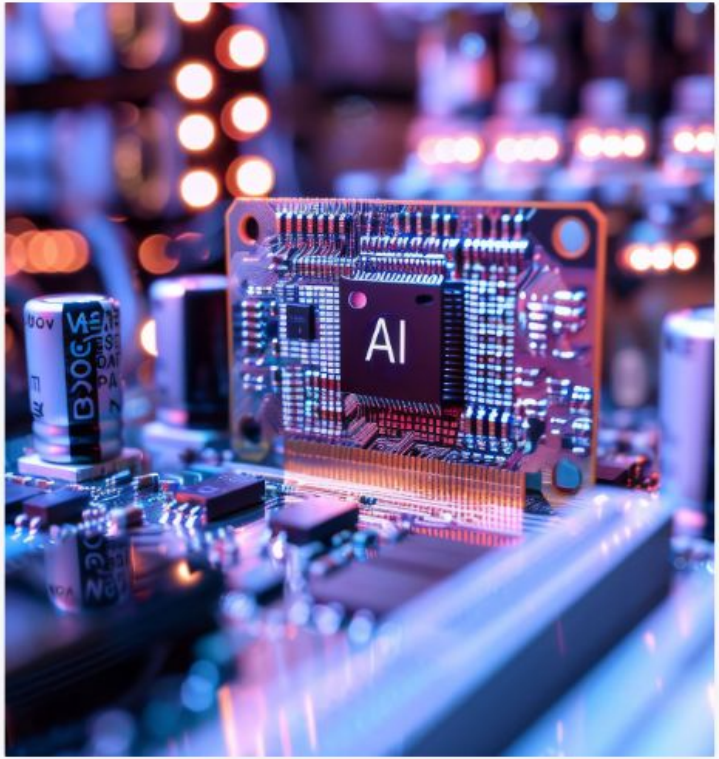
Various sectors will increasingly implement **quantum machine learning** to boost efficiency and innovation.

Collaborative efforts for breakthroughs

Interdisciplinary partnerships will facilitate significant **breakthroughs** in quantum applications and research.

Ethical discussions ongoing

There will be a continuous focus on the **ethical implications** of quantum technologies in society.



Case Study - QSVM on Iris Dataset

Outline

Dataset: IRIS

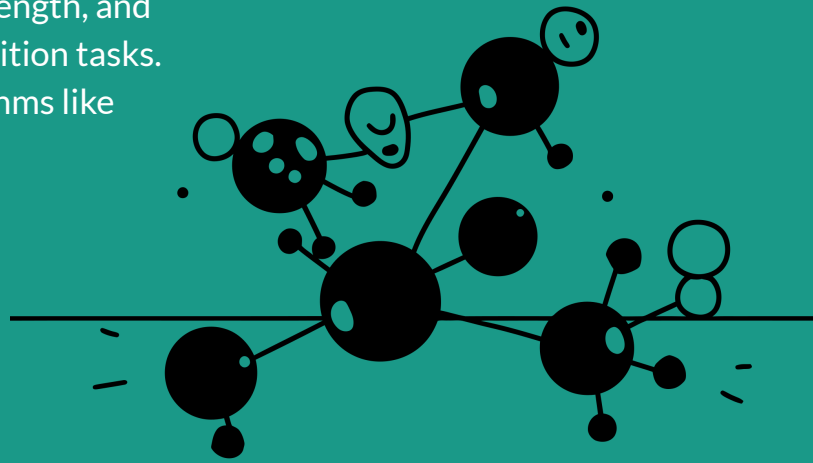
Encoding

Classifier : QSVM

Accuracy Comparison

IRIS DATASET :

The **Iris dataset** is a widely used dataset in machine learning, containing 150 samples from three Iris flower species: Setosa, Versicolor, and Virginica. Each sample includes four features — sepal length, sepal width, petal length, and petal width. It's mainly used for classification and pattern recognition tasks. The dataset is simple, well-balanced, and ideal for testing algorithms like SVM and QSVM.



```
[ ] data=pd.read_csv('/content/Iris.csv')
```

```
[ ] data.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Encoding: Angle Encoding

Angle Encoding (also called *rotation encoding*) is a method used to encode classical data into quantum states by applying rotation gates to qubits. It's one of the simplest and most commonly used data encoding strategies in quantum machine learning.

Code Snippet

```
[ ] from qiskit import QuantumCircuit
    import numpy as np
```

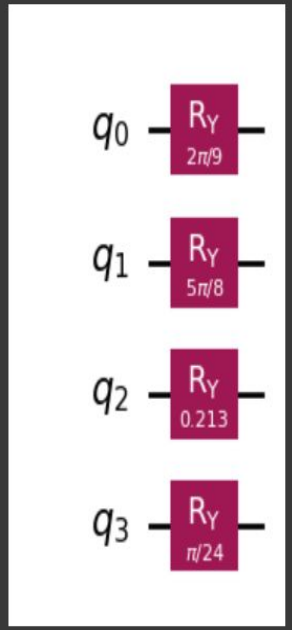
```
[ ] scaler = MinMaxScaler(feature_range=(0, np.pi))
    data_normalized = scaler.fit_transform(data[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']])
```

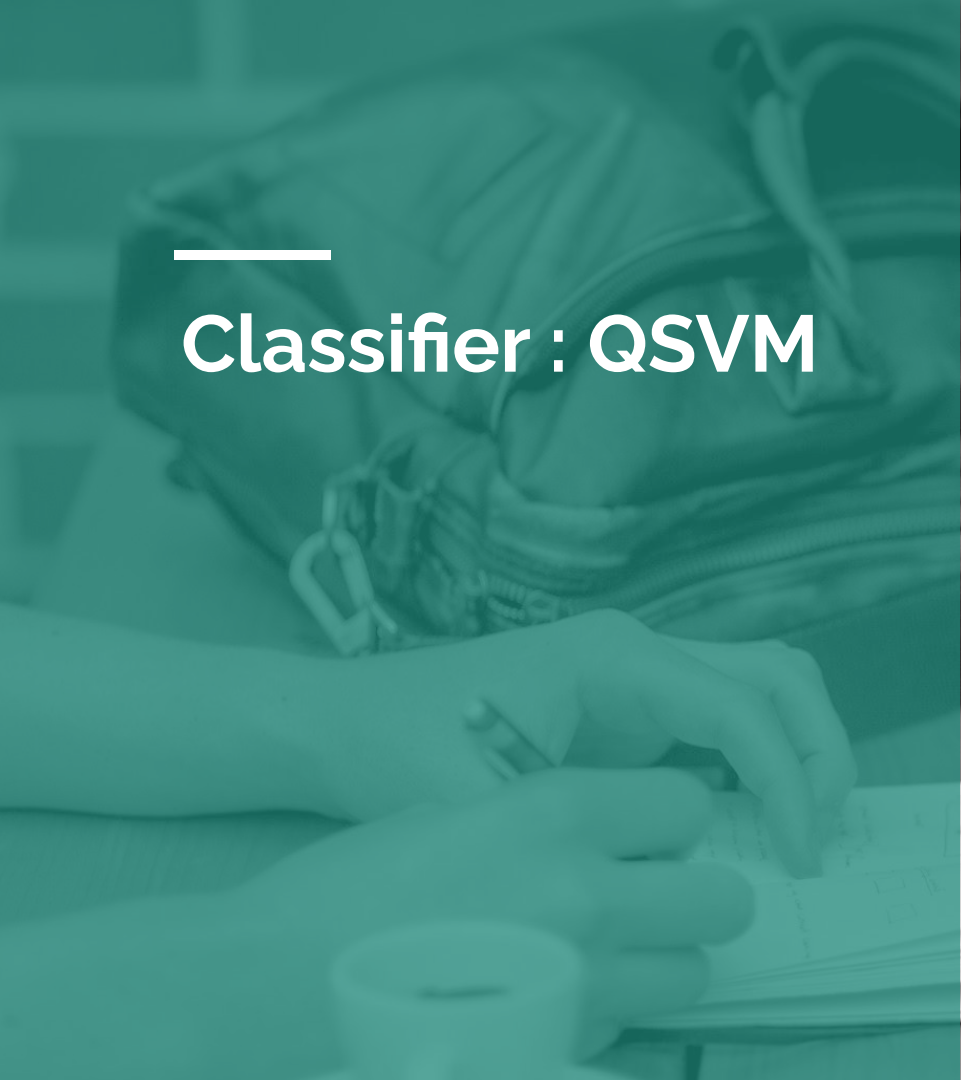
```
[ ] def angle_encoding(data_point):
        """
        Create a quantum circuit with angle encoding (RY) for a single data point.
        Each feature is encoded into one qubit.
        """
        num_qubits = len(data_point)
        qc = QuantumCircuit(num_qubits)

        for i, angle in enumerate(data_point):
            qc.ry(angle, i)

        return qc
```

```
[ ] quantum_circuits = [angle_encoding(sample) for sample in data_normalized]
```

```
[ ] from IPython.display import display
```

```
qc_sample = angle_encoding(data_normalized[0])
qc_sample.draw('mpl')
```





Classifier : QSVM

A **QSVM** is the **quantum version of a classical SVM (Support Vector Machine)**, used for classification tasks in quantum machine learning. It uses quantum kernels to map data into high-dimensional Hilbert spaces via quantum circuits.

Code Snippets

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from qiskit_aer import Aer
from qiskit.providers.basicaer import BasicAer
from qiskit.utils import algorithm_globals
from qiskit.circuit.library import ZZFeatureMap
from qiskit_machine_learning.kernels import FidelityQuantumKernel
from qiskit_machine_learning.algorithms.classifiers import QSVC
```

```
[9] backend = Aer.get_backend('qasm_simulator')
    iris = load_iris()
```

```
[10] X, Y = iris.data, iris.target
```

```
[11] print(X.shape)
     print(len(set(Y)))
```

```
(150, 4)
3
```

```
[12] X = X[Y != 2]
     Y = Y[Y != 2]
```

```
[14] from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import MinMaxScaler
```

```
[19] from qiskit_machine_learning.kernels import FidelityQuantumKernel
     from qiskit_machine_learning.algorithms.classifiers import QSVC
     from qiskit.utils import algorithm_globals
```

```
[15] scaler = MinMaxScaler(feature_range=(0, np.pi))
     X_scaled = scaler.fit_transform(X)
```

```
[16] train_x, test_x, train_y, test_y = train_test_split(X_scaled, Y, test_size=0.2, random_state=42)
```

```
[17] num_features = X.shape[1]
```

```
[18] feature_map = ZZFeatureMap(feature_dimension=num_features, reps=3, entanglement='full')
```

```
[20] algorithm_globals.random_seed = 42
```

```
<ipython-input-20-261ddb4c96d1>:1: DeprecationWarning: The property ``qiskit.utils.algorithm_globals.QiskitAlgorithmGlobals.random_seed`` is deprecate
  algorithm_globals.random_seed = 42
```

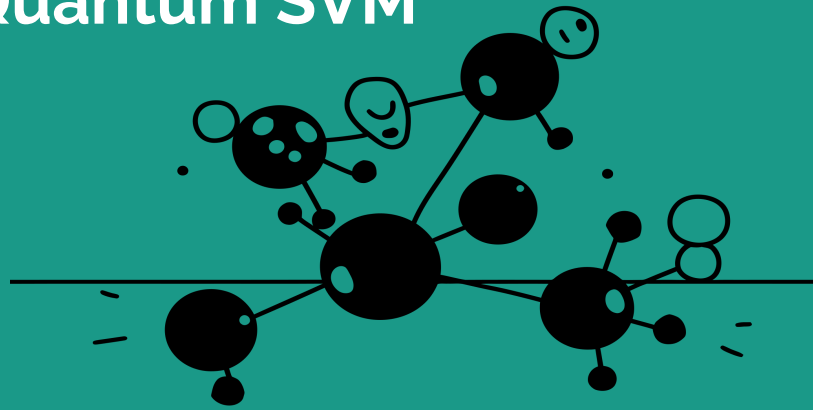
Final Output

```
[21] quantum_kernel = FidelityQuantumKernel(feature_map=feature_map)
     qsvc = QSVC(quantum_kernel=quantum_kernel)
     qsvc.fit(train_x, train_y)
     print("Accuracy:", qsvc.score(test_x, test_y))
```

```
Accuracy: 0.55
```

Accuracy comparison:

Between Classical SVM and Quantum SVM



```
[23] from sklearn.svm import SVC
```

```
[31] import matplotlib.pyplot as plt
```

```
[26] svc = SVC(kernel='rbf')  # You can try 'linear', 'poly', etc.
     svc.fit(train_x, train_y)
     svc_accuracy = svc.score(test_x,test_y)
```

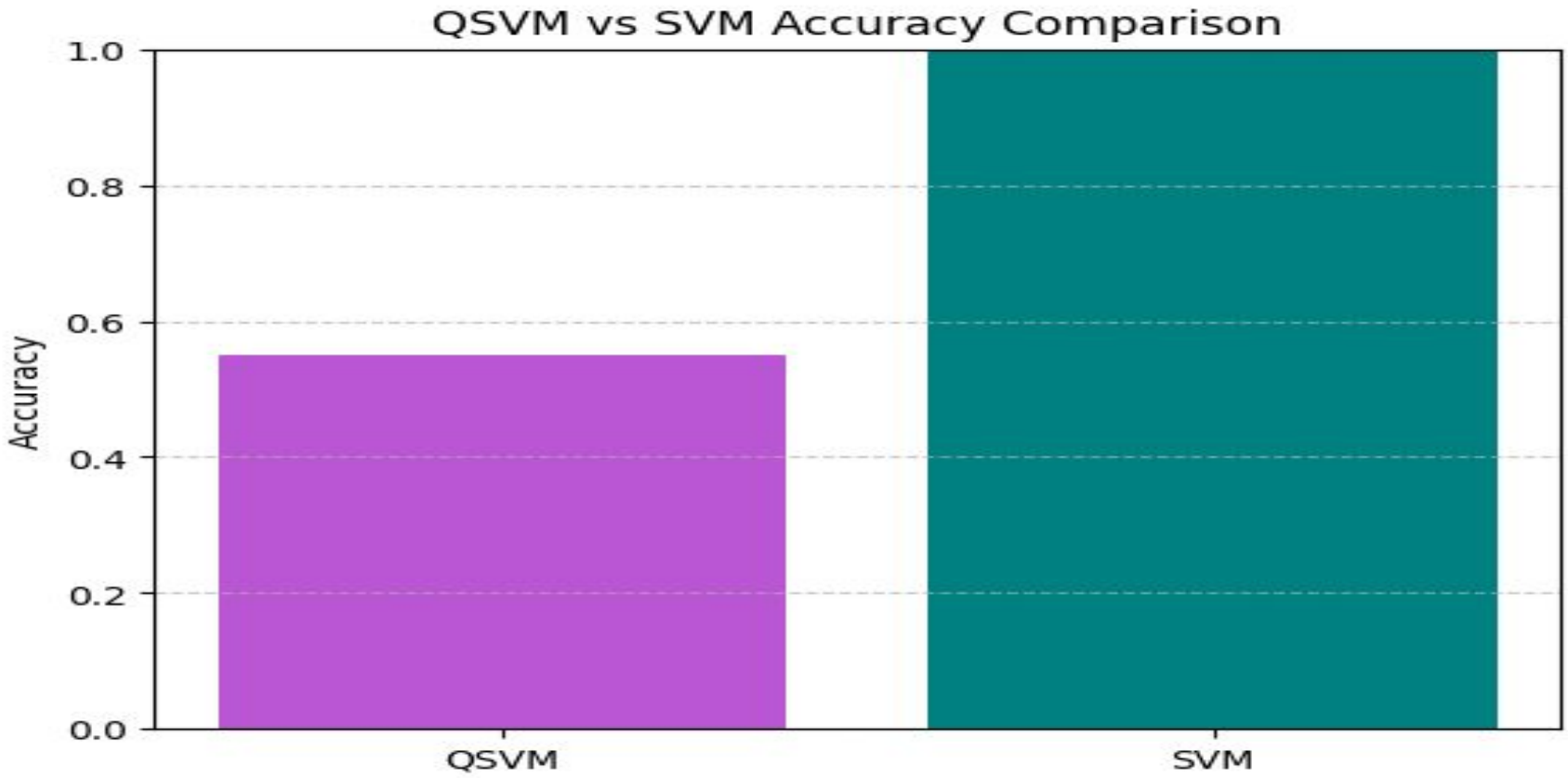
```
[28] print(f"QSVM Accuracy: {qsvc.score(test_x, test_y):.2f}")
     print(f"Classical SVM Accuracy: {svc_accuracy:.2f}")
```

```
QSVM Accuracy: 0.55
Classical SVM Accuracy: 1.00
```

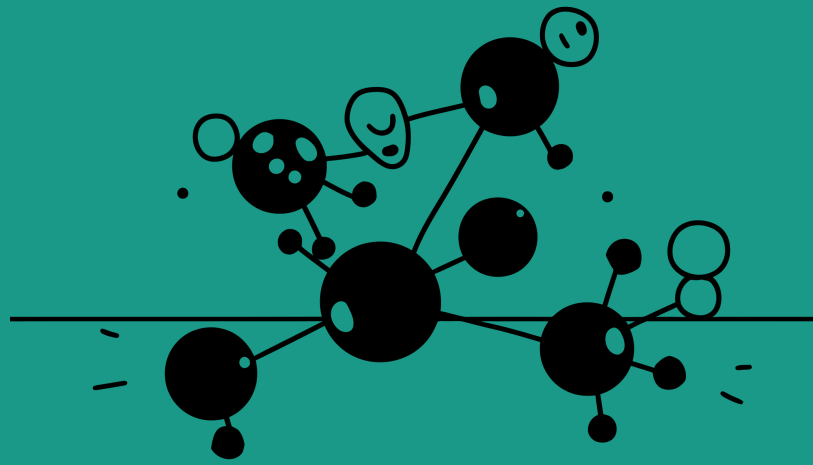
```
[29] labels = ['QSVM', 'SVM']
     accuracies = [qsvc.score(test_x, test_y), svc_accuracy]
```

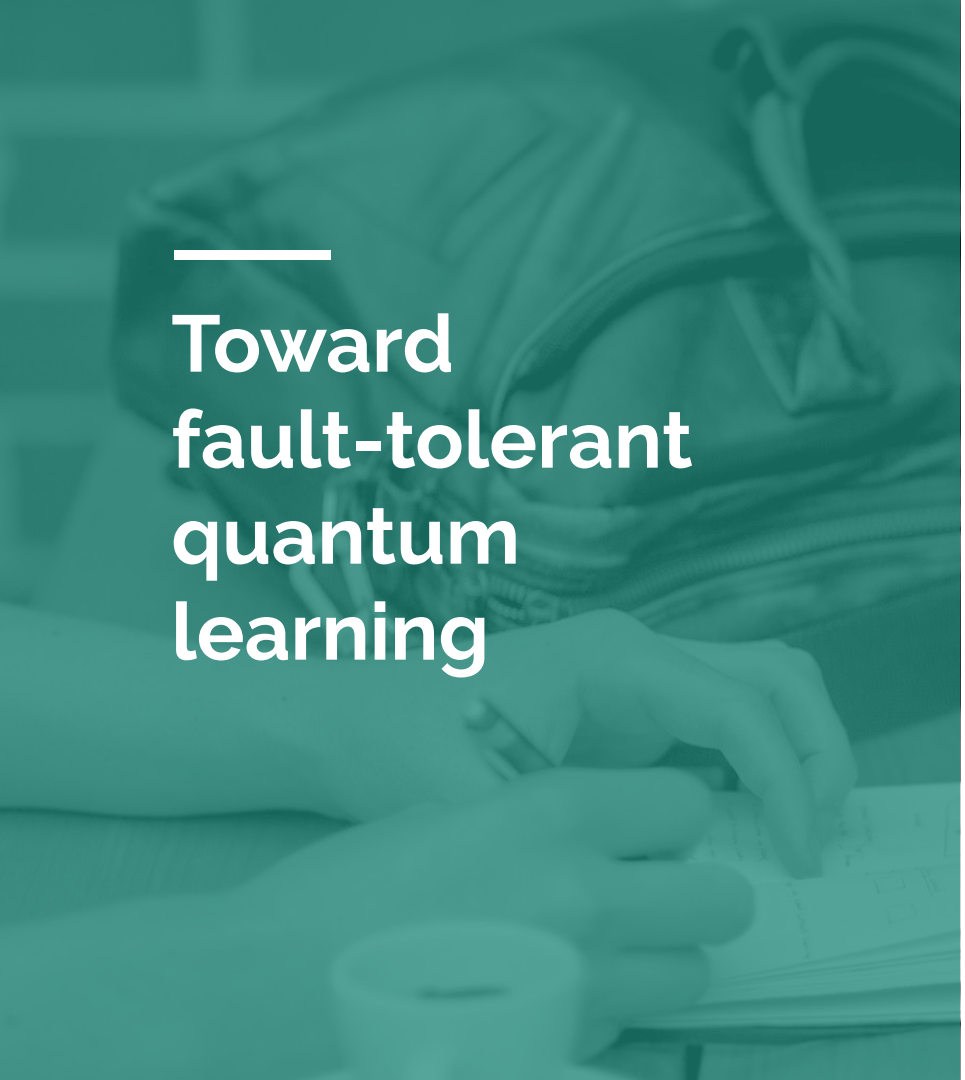
```
[32] plt.bar(labels, accuracies, color=['mediumorchid', 'teal'])
     plt.ylim(0, 1)
     plt.ylabel('Accuracy')
     plt.title('QSVM vs SVM Accuracy Comparison')
     plt.grid(axis='y', linestyle='--', alpha=0.7)
     plt.show()
```

Result Plot


QSVM vs SVM Accuracy Comparison
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
QSVM
SVM

Future of QML





Toward fault-tolerant quantum learning

"Toward Fault-Tolerant Quantum Learning" refers to developing quantum machine learning (QML) algorithms and systems that remain accurate and reliable even in the presence of noise and errors, which are common in current quantum computers.

It focuses on designing quantum learning algorithms that are robust against **errors** using techniques like quantum error correction, noise-resilient circuits, and fault-tolerant architectures, so they can work effectively on near-term and future quantum devices.

Research areas: explainability, scaling

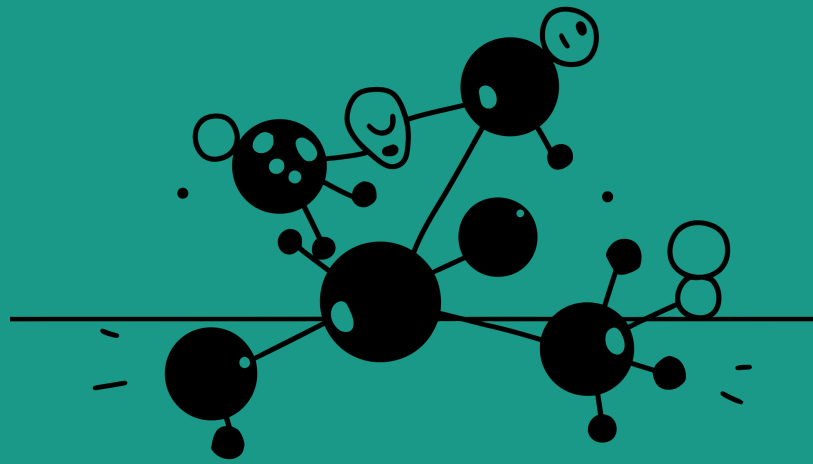
Explainability:

Focuses on making quantum algorithms, especially quantum machine learning, understandable **to** humans—explaining how and why a quantum model reaches a decision or prediction.

Scaling:

Involves developing hardware and algorithms that can grow to handle larger, more complex problems, including increasing qubit count, reducing error rates, and improving quantum circuit efficiency to make quantum computing practical at scale.

Visual: Roadmap For QML(2025 - 2030)



Focus Areas

Performance & Optimization | Developer Tools | Architecture & Language | Platform & Compatibility

2025

Qt 6.9

April 2025

- ✓ Enhanced QML Language Server
- ✓ Improved JavaScript debugging
- ✓ Better type inference
- ✓ Performance optimizations

Qt 6.10

October 2025

- ✓ Advanced QML compiler optimizations
- ✓ New animation framework
- ✓ Improved memory management
- ✓ Better mobile platform support

2026

Qt 6.11

April 2026

- ✓ Strong typing improvements
- ✓ Enhanced C++ to QML integration
- ✓ New Qt Quick 3D features
- ✓ Improved accessibility

Qt 6.12 LTS

October 2026

- ✓ Long-term support release
- ✓ Mature QML compilation pipeline
- ✓ Comprehensive documentation
- ✓ Enterprise-ready features

🚀 Mid-Point Milestone

By 2026, QML will have mature compilation to native code, significantly improved performance, and comprehensive tooling support. The framework will be enterprise-ready with robust type systems and excellent developer experience.

2027

Qt 7.0

April 2027

- ✓ Major architectural overhaul
- ✓ New QML syntax enhancements
- ✓ Native compilation by default
- ✓ Breaking changes for modernization

Qt 7.1

October 2027

- ✓ WebAssembly optimization
- ✓ Advanced state management
- ✓ Improved cross-platform consistency
- ✓ New component library

2028

Qt 7.2

April 2028

- ✓ AI-assisted development tools
- ✓ Advanced rendering pipeline

Qt 7.3

October 2028

- ✓ Immersive UI components
- ✓ Advanced gesture recognition

- ✓ Advanced rendering pipeline
- ✓ Reactive programming patterns
- ✓ Cloud-native features

- ✓ Advanced gesture recognition
- ✓ Real-time collaboration tools
- ✓ Performance monitoring integration

2029

Qt 7.4

April 2029

- ✓ Quantum computing integration
- ✓ Next-gen AR/VR support
- ✓ Autonomous system interfaces
- ✓ Edge computing optimization

Qt 7.5 LTS

October 2029

- ✓ Long-term support release
- ✓ Stable ecosystem
- ✓ Comprehensive security features
- ✓ Industry 4.0 readiness

2030

Qt 7.6

April 2030

- ✓ Neural UI adaptation
- ✓ Sustainable computing features
- ✓ Universal accessibility
- ✓ Interplanetary communication ready

Qt 8.0 Preview

October 2030

- ✓ Next-generation preview
- ✓ Quantum-native architecture
- ✓ Consciousness-aware interfaces
- ✓ Paradigm shift preparation

🎯 Vision 2030

By 2030, QML will have evolved into a truly universal UI framework capable of creating interfaces for any device, platform, or computing paradigm. It will seamlessly integrate with AI, quantum computing, and emerging technologies while maintaining its core philosophy of declarative, efficient UI development.

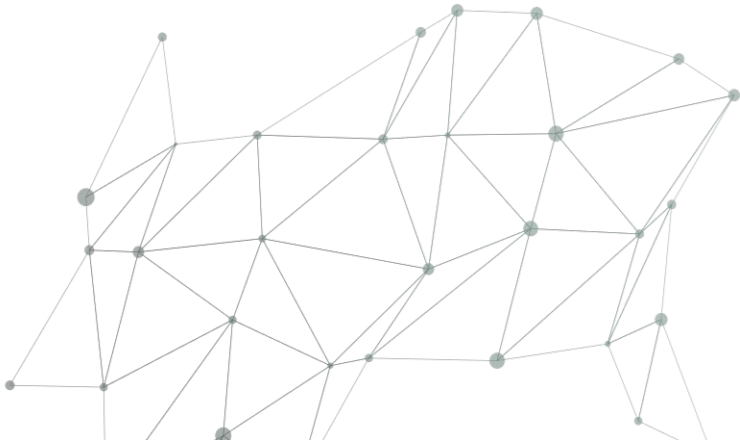
References

Peter Wittek, "Quantum Machine Learning"

Schuld & Petruccione, "Supervised Learning with Quantum Computers"

IBM Qiskit Tutorials

arXiv : Top papers on Quantum Computing



Thank You!!

Questions?