Riding the Quantum Wave: From Earth To Intelligence

How quantum computing bridges physics-based modeling and AI?

SULAIMAN UREIGA
AI Eng & Researcher

AMNAH SAMARIN
HPC Developer & Computational Geophysicist

SULAIMAN UREIGA

AI Eng & Researcher

AMNAH SAMARIN

HPC Developer & Computational Geophysicist

Agenda

Quantum Computing

Overview of the current state and core principles.

Quantum Application In Geophysics

End-to-end pipeline for 1D wave equation simulation, including circuit design, state preparation, theoretical analysis, and experiments.

Quantum Application In AI

Overview of AI/ML, introduction to quantum machine learning and QNNs, and discussion of applications and limitations.

Quantum Computing

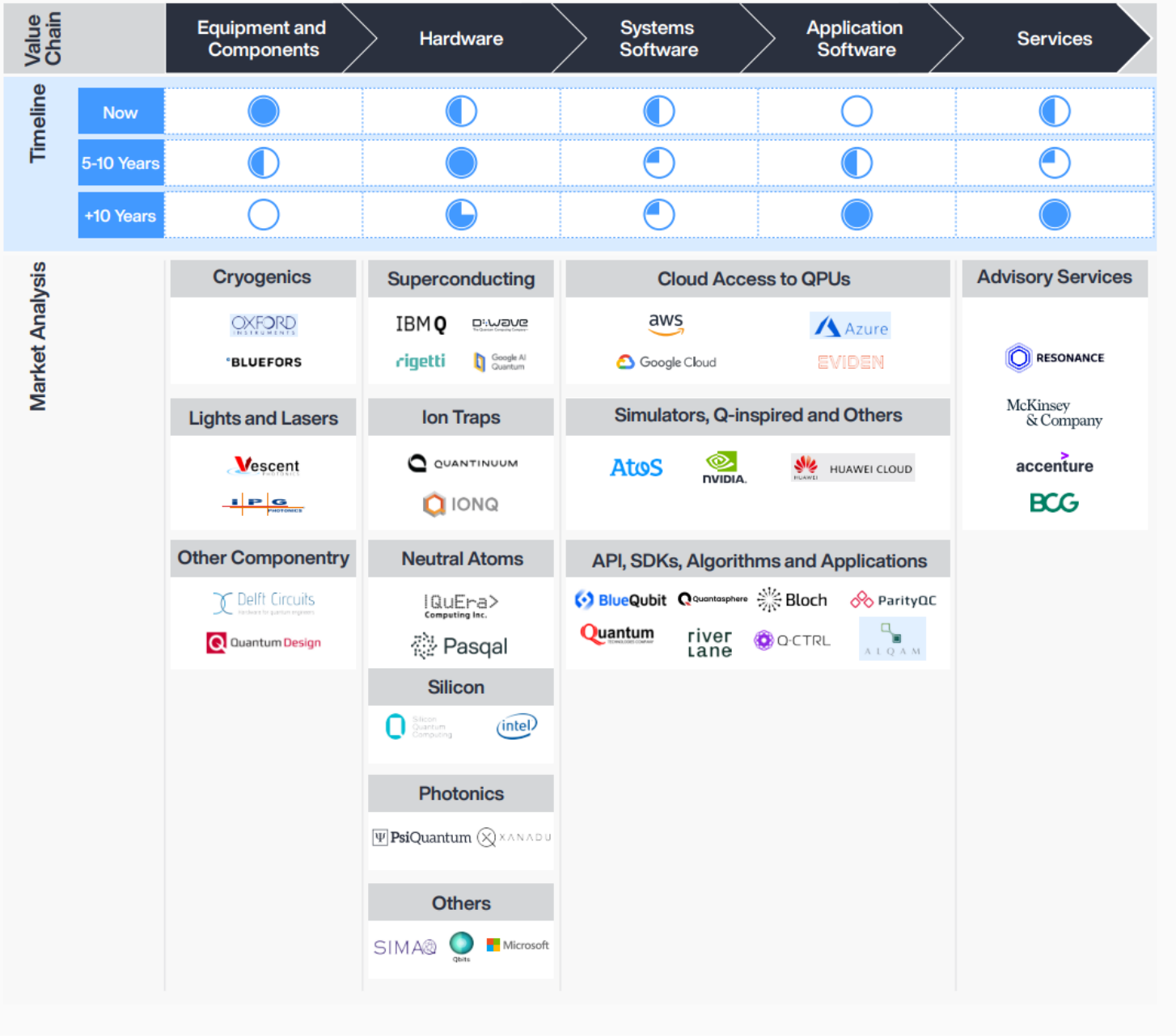
- GLOBAL OVERVIEW
- BASIC PRINCIPLES

Global Overview

- Where are we right now?
- Do we currently have a quantum advantage?

- Current State of Quantum Computing
  - No clear quantum advantage (yet)
  - Hardware and software are still fragmented
  - Rapid ecosystem growth (hardware, cloud, SDKs)

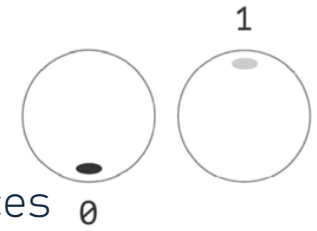
The goal is to be quantum-ready when an advantage emerges



No market value captured · A Limited market value captured · Some market value captured
Significant market value captured · Full market value captured

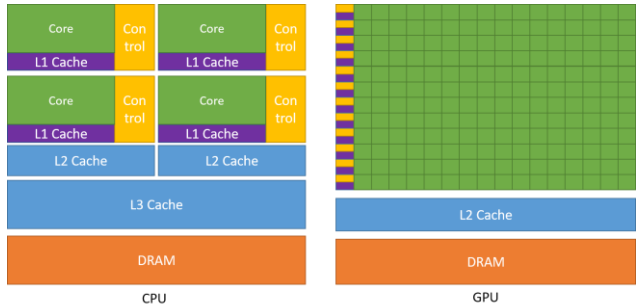
Source: Towards Saudi Arabia's Quantum-Enabled Future Report by C4IR

Classical Computing

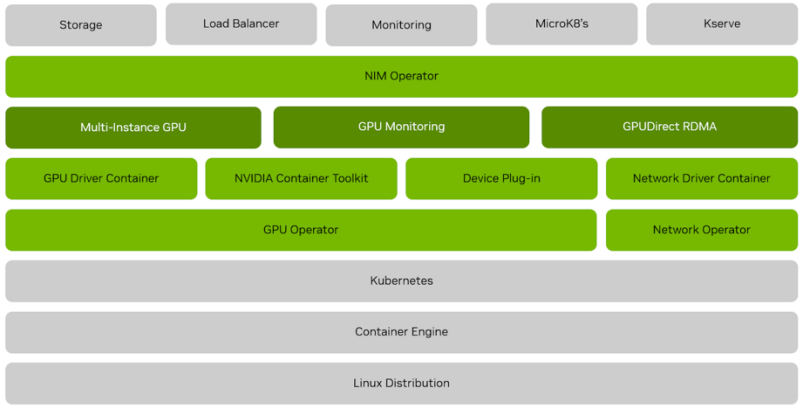


Fully developed market stack for all services

Hardware



NVIDIA fully supported software stack



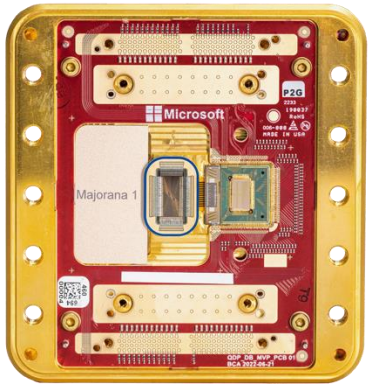
Source

Quantum Computing

Research-level development for all services

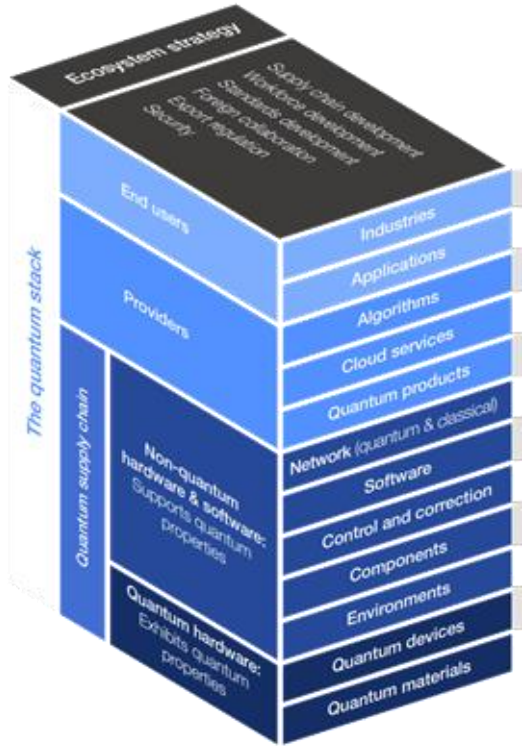
Hardware

The main platforms—superconducting, trapped ions, photonic, neutral atoms, and semiconductor



Microsoft, Majorana 1

Vision of Quantum Stack



WEF Quantum Economy Blueprint 2024

Classical computing is optimized and scalable; quantum computing is powerful but still experimental.

Basic Principles

Qubit

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Normalization:

$$|\alpha|^2 + |\beta|^2 = 1$$

N-qubits system:

$$|\psi\rangle = \sum_i \alpha_i |i\rangle, i \in \{0, \ldots 2^n - 1\}$$

A quantum system represents 2^n states simultaneously.

Gates

- Quantum gates are unitary operations
$$U^\dagger U = I$$
- They manipulate probability amplitudes

Core Gates:

- Pauli-X Gate → flips state
- Pauli-Z → changes phase
- Hadamard → creates superposition
- CNOT → creates entanglement

Circuits

Quantum computation follows three steps:

1) Initialization
$$|\psi_0\rangle = |0\rangle^{\otimes n}$$
2) Evolve
$$|\psi_f\rangle = U|\psi_0\rangle$$
3) Measure
$$P(i) = |\langle i|\psi_f\rangle|^2$$

Measurement collapses the quantum state

State Preparation / Data Encoding

A mapping from classical data → quantum state

$$|\psi(x)\rangle = U(x)|0\rangle$$

Variational Quantum Algorithms (VQAs)

A hybrid approach to solve optimization problems on near-term quantum devices

- Hybrid quantum–classical algorithms for optimizing parameterized quantum circuits
- Train quantum circuits similar to neural networks
- Core paradigm for Noisy Intermediate-Scale Quantum (NISQ)era quantum computing

Key Idea (Variational Principle)

$$E(\theta) = \langle \psi(\theta) \mid H \mid \psi(\theta) \rangle$$

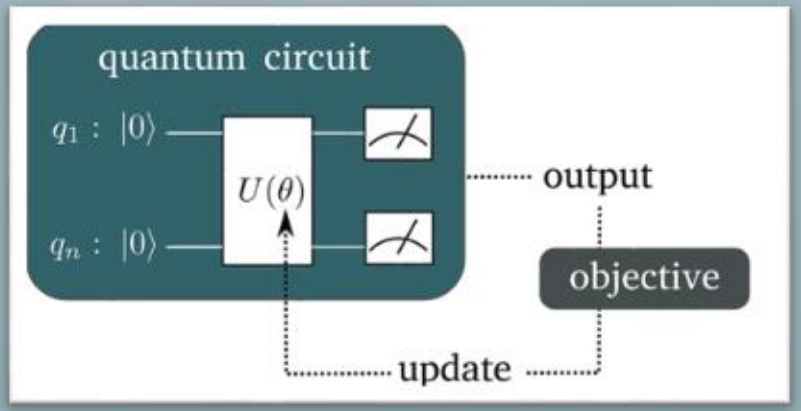
Find parameters θ that minimize the cost function

$$E_0 \approx \min_{\theta} E(\theta)$$

- $E(\theta)$:cost function (energy to minimize)
- θ: trainable parameters of the quantum circuit
- E_0 :ground-state energy
- H: Hamiltonian describing the system
- $\mid \psi(\theta) \rangle$:parameterized quantum state

Optimization Loop

1. Initialize state
   $\psi(\theta) = \boldsymbol{U}(\theta) \mid 0 \rangle$
2. Evaluate cost
   $E(\theta) = \langle \psi(\theta) \mid H \mid \psi(\theta) \rangle$
3. Update parameters (classical optimizer)
   $\theta \rightarrow \theta'$
4. Repeat until convergence



Now that we understand the basics,
Let's move to a real application !

Quantum application in Geophysics

- QUANTUM SIMULATION OF THE 1D WAVE EQUATION FULL PIPELINE MAP
- 1D WAVE SIMULATION QUANTUM CIRCUIT
- PARAMETRIZED QUANTUM CIRCUIT FOR STATE PREPARATION
- THEORETICAL INSIGHT
- EXPERIMENTS

Motivation

We explore how NISQ quantum computers can simulate the 1D wave equation using hardware-efficient algorithms. This work builds on:

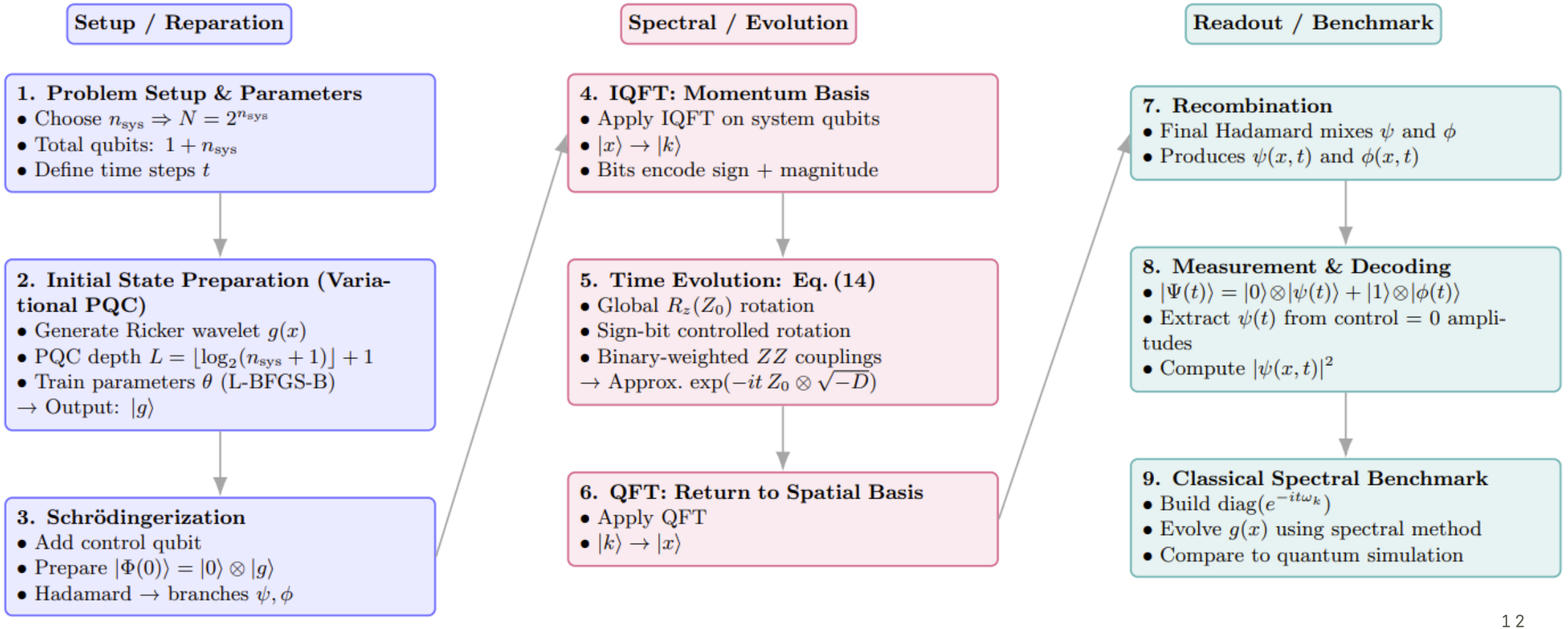
- Quantum Algorithm for Simulating the Wave Equation (Phys. Rev. A, 2019), which introduces Schrödingerization and the QFT-based spectral formulation of PDEs.
- Noisy Intermediate-Scale Quantum Simulation of the 1D Wave Equation (Wright et al., 2024), which demonstrates a hardware-efficient approximation (Eq. 14) suitable for real NISQ hardware—this is the algorithm reproduced and studied in the notebook.

Why it matters:

- Explore the possibility of faster simulations
- New physics modeling paradigms

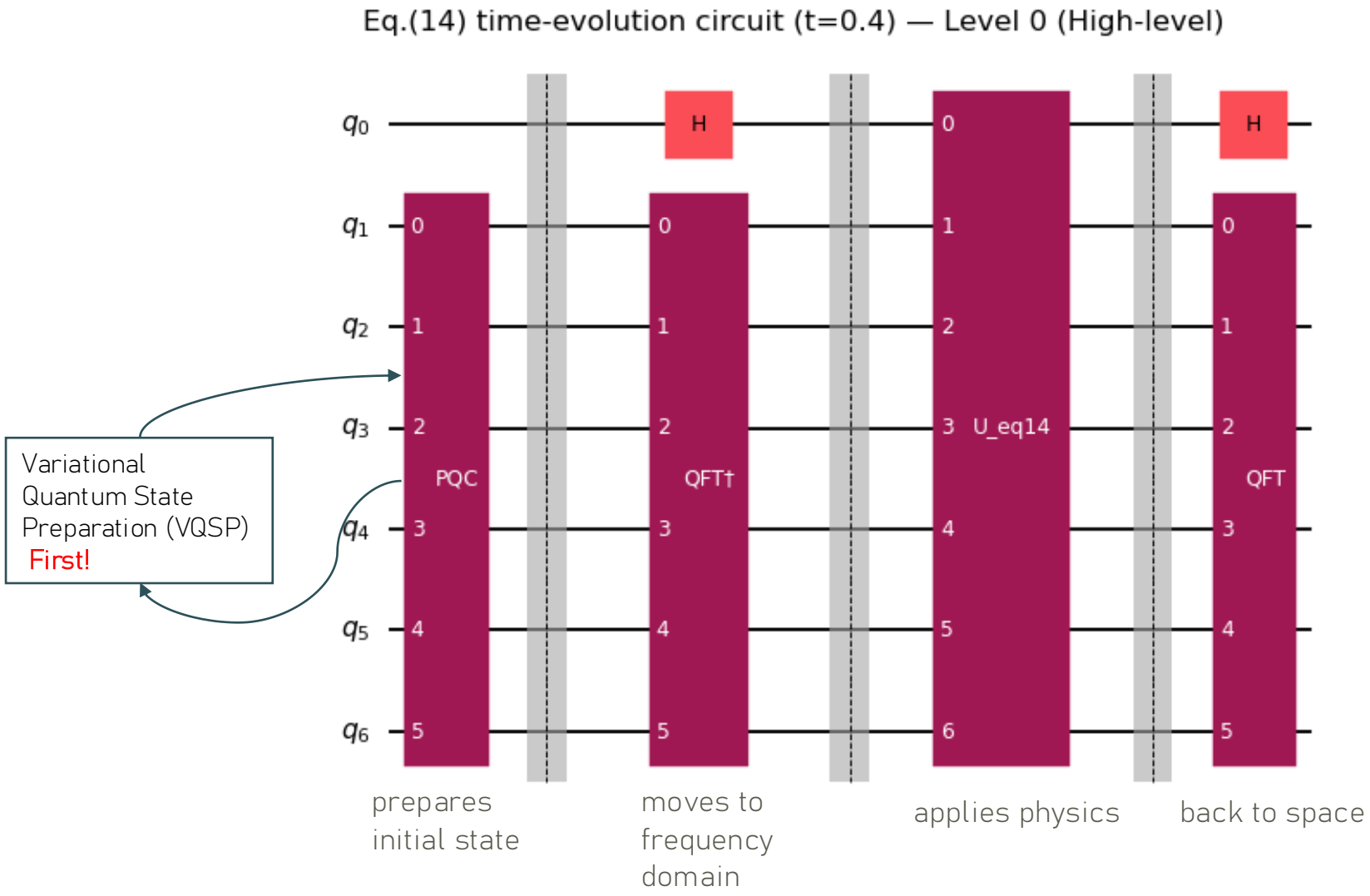
Quantum Simulation of the 1D Wave Equation – Full Pipeline

The pipeline transforms a classical PDE into a quantum-evolvable system using Schrödingerization and spectral methods.



1D Wave Equation Quantum Circuit

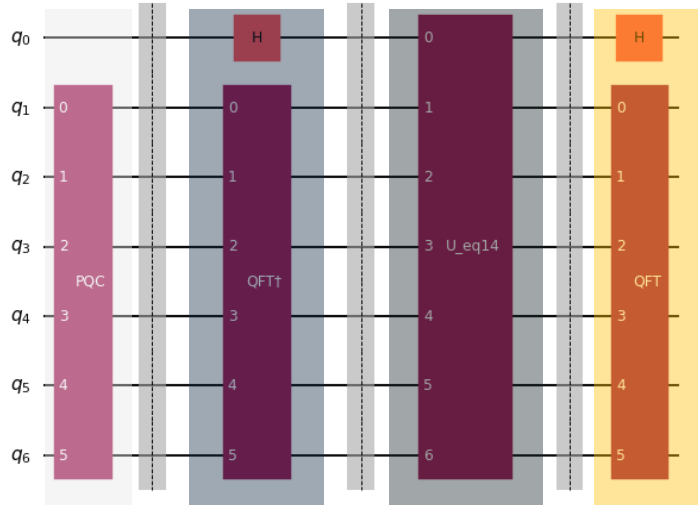
$$|\Phi(t)\rangle = (H \otimes \mathrm{QFT})\, U_{\mathrm{eq14}}(t)\, (H \otimes \mathrm{QFT}^{\dagger})\, |\Phi(0)\rangle.$$



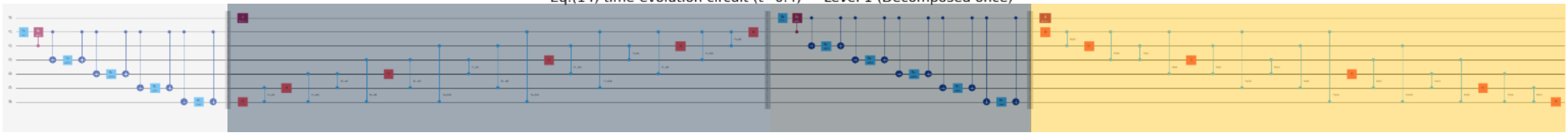
Output:

$$|\Psi(t)\rangle = |0\rangle \otimes |\psi(t)\rangle + |1\rangle \otimes |\phi(t)\rangle.$$

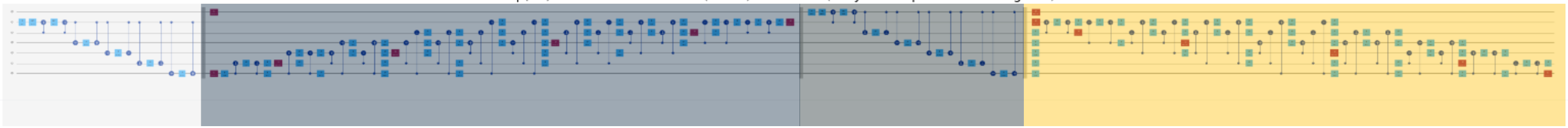
Eq.(14) time-evolution circuit (t=0.4) — Level 0 (High-level)



Eq.(14) time-evolution circuit (t=0.4) — Level 1 (Decomposed once)



Eq.(14) time-evolution circuit (t=0.4) — Level 2 (Fully decomposed to basis gates)



Variational Quantum State Preparation (VQSP)

Using parametrized quantum circuit

1. Define Spatial Grid & Target Wavefield $|g\rangle$
- Choose number of system qubits n_{sys}
  $\Rightarrow N = 2^{n_{\text{sys}}}$ spatial grid points
- Build Ricker wavelet $g(x_j)$ for $j = 0, \ldots, N-1$
- Normalize $\sum_j |g_j|^2 = 1$
- Store as `Target_Ricker` $\rightarrow$ target state $|g\rangle$

2. Choose PQC Ansatz: Brickwall Circuit $U(\theta)$
- n_{sys}-qubit nearest-neighbor brickwall
  Blocks $(0,1)$, $(2,3)$, ... then shifted
- Depth $L = \lfloor \log_2(n_{\text{sys}} + 1) \rfloor + 1$
- Each 2-qubit block:
  - Pre-rotations (Rz–Ry–Rz)
  - 3-CNOT entangling layer
  - Post-rotations (Rz–Ry–Rz)
- Angles form the parameter vector θ

3. Forward Pass: Prepare $|\psi(\theta)\rangle$
- Bind parameters θ into PQC
- Simulate $U(\theta)|0\cdots0\rangle$
- Output $\psi(\theta) \in \mathbb{C}^{2^{n_{\text{sys}}}}$

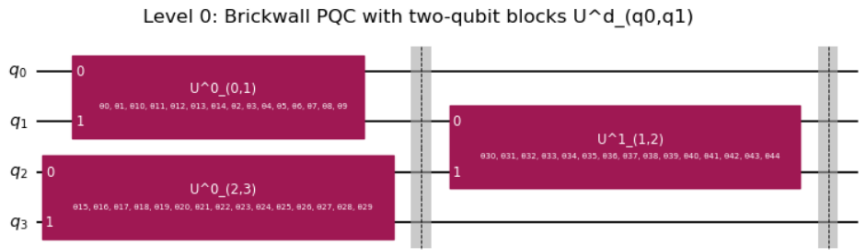
4. Cost Evaluation: Overlap with $|g\rangle$
- Compute $\langle g \mid \psi(\theta)\rangle$
- Cost: $C(\theta) = 1 - \Re[\langle g \mid \psi(\theta)\rangle]$
- Interpretation:
  - $C(\theta) = 0$: perfect match
  - $C(\theta) \approx 1$: poor match

5. Classical Optimizer Loop (L-BFGS-B)
- Initialize parameters θ_0
- L-BFGS-B quasi-Newton updates:
  - Uses cost and gradient history
- Iterate until convergence
- Output optimized parameters θ^*

6. Extract Trained Circuit & Assess Fidelity
- Bind θ^* to PQC $\rightarrow$ `trained_qc_ricker`
- Compute trained state $\psi(\theta^*)$
- Evaluate:
  - overlap $= \langle g \mid \psi(\theta^*)\rangle$
  - Fidelity $= |\langle g \mid \psi(\theta^*)\rangle|^2$
- Compare:
  - $\Re(g_j)$ vs. $\Re(\psi_j)$
  - $\Im(g_j)$ vs. $\Im(\psi_j)$
  - $|g_j|^2$ vs. $|\psi_j|^2$
- Confirms PQC learned Ricker wavelet

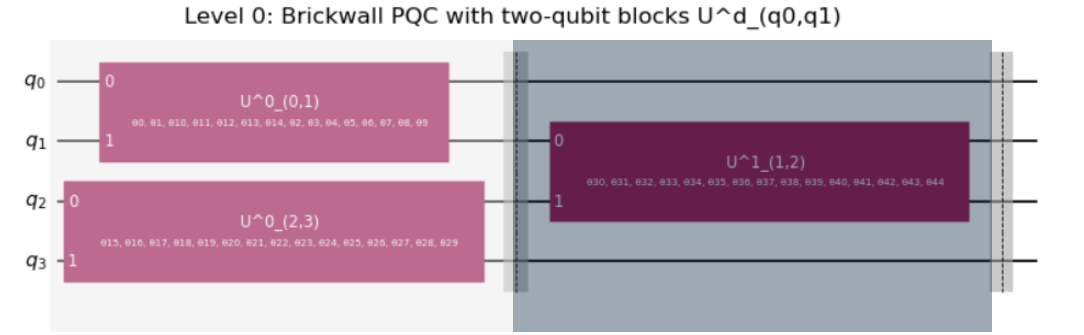
2. Initial State Preparation (Variational PQC)
- Generate Ricker wavelet $g(x)$
- PQC depth $L = \lfloor \log_2(n_{\text{sys}} + 1) \rfloor + 1$
- Train parameters θ (L-BFGS-B)

$\rightarrow$ Output: $|g\rangle$



Variational Quantum State Preparation (VQSP)

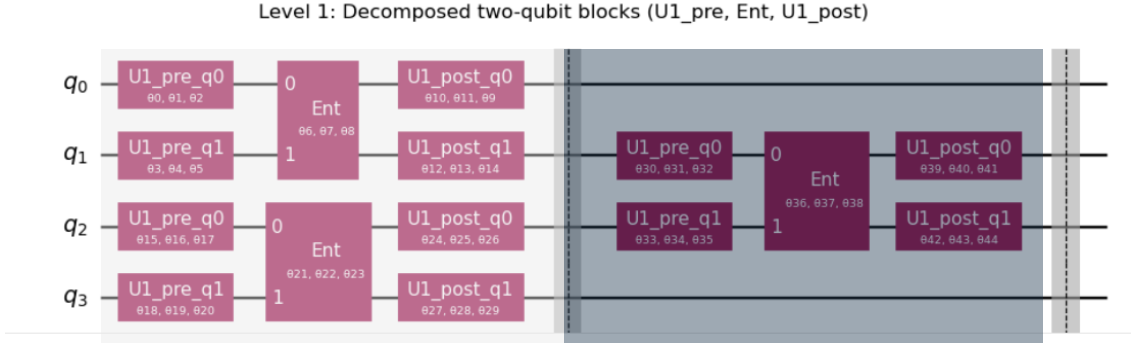
Using parametrized quantum

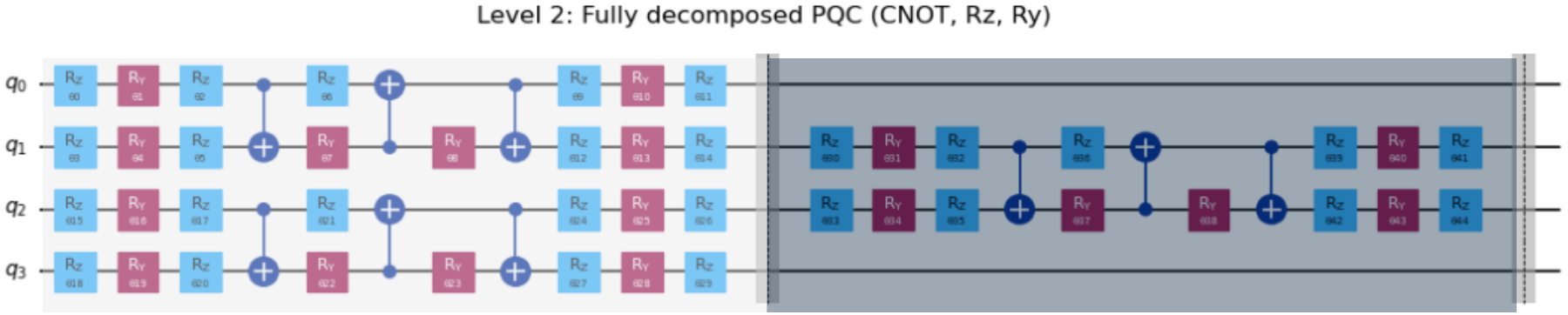


2. Initial State Preparation (Variational PQC)

- Generate Ricker wavelet $g(x)$
- PQC depth $L = \lfloor \log_2(n_{\text{sys}} + 1) \rfloor + 1$
- Train parameters θ (L-BFGS-B)

$\rightarrow$ Output: $|g\rangle$





Theoretical Insight

The goal is to simulate wave dynamics using quantum circuits

Overview of Schrödingerization

From Wave Equation to Quantum Form Wave equation

Efficient Quantum Evolution

Overview of Schrödingerization

A method to convert classical PDEs into a quantum-compatible form.

Quantum systems evolve according to:

$$\frac{d}{dt}|\Psi(t)\rangle = -iH|\Psi(t)\rangle$$

Enables the time evolution

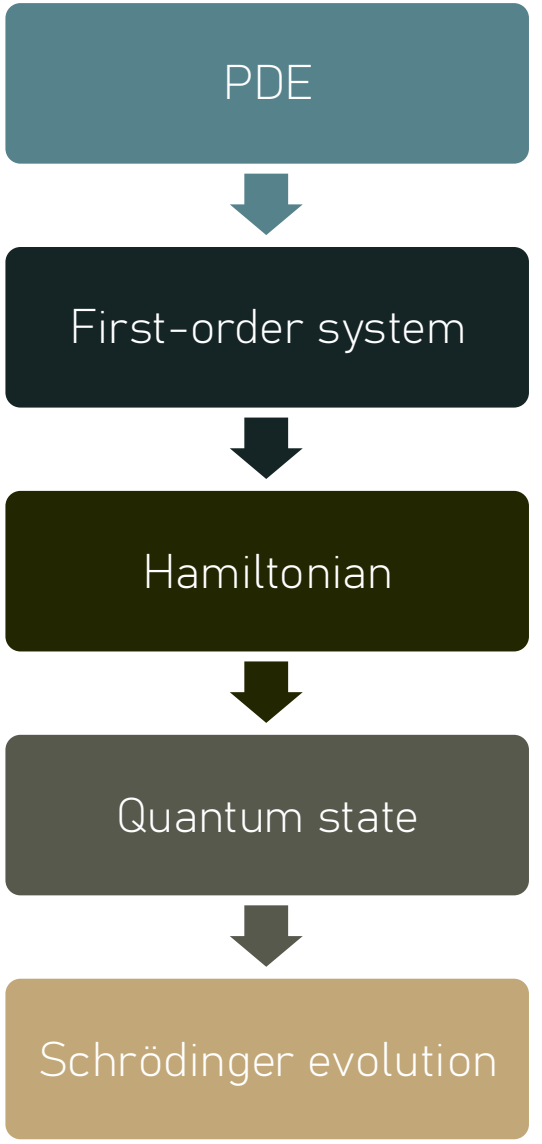
$$|\Psi(t)\rangle = e^{-iHt}|\Psi(0)\rangle$$

Many physical equations are:

- Second-order in time
- Not directly compatible with quantum evolution

Key Idea

- Transform the system into a first-order Schrödinger-like equation
- Introduce auxiliary variables → expand the system



From Wave Equation to Quantum Form Wave equation

1- Classical wave equation:

$$\frac{\partial^2 \psi}{\partial t^2} = \Delta \psi$$

Not directly compatible with quantum evolution (second-order and System is not unitary)

2- Reduce to first-order system:

Define $\phi = \frac{\partial \psi}{\partial t}$

$$\frac{d}{dt}\begin{pmatrix}\psi \\ \phi\end{pmatrix} = \begin{pmatrix}0 & 1 \\ \Delta & 0\end{pmatrix}\begin{pmatrix}\psi \\ \phi\end{pmatrix}$$

3- Make it Schrödinger-like

Remember Schrödinger form $\frac{d}{dt}|\Psi(t)\rangle = -\mathrm{i}H|\Psi(t)\rangle$

$$\frac{d}{dt}\begin{pmatrix}\psi \\ \phi\end{pmatrix} = -iH_w\begin{pmatrix}\psi \\ \phi\end{pmatrix}, \qquad H_w = \mathrm{i}\begin{pmatrix}0 & 1 \\ \Delta & 0\end{pmatrix}$$

4- Time evolution becomes

$$|\Psi(t)\rangle = e^{-iH_w t}|\Psi(0)\rangle$$

where $U(t) \approx e^{-iH_w t}$

5- Embed into a quantum state:

Now map this vector into a quantum system:

$$|\Psi(t)\rangle = |0\rangle|\psi(t)\rangle + |1\rangle|\phi(t)\rangle$$

Embed using a control qubit:

- $(|0\rangle \rightarrow \psi)$ (displacement)
- $(|1\rangle \rightarrow \phi)$ (velocity)

.

Efficient Quantum Evolution 1/2

Hardware efficient approximation to limit the circuit depth.

$$U(t) \approx e^{-it(2^{n-1}-1)\pi Z_0} \left(|0\rangle\langle 0|_1 + e^{itN\pi Z_0}|1\rangle\langle 1|_1\right) \prod_{q=2}^{n} e^{it\,2^{n-q}\pi\, Z_0 Z_q}.$$

Efficient Quantum Evolution

One-Step Quantum Evolution

$$|\Phi(t)\rangle = (H \otimes QFT)\, U(t)\, (H \otimes QFT^*)\, |\Phi(0)\rangle$$

Combines state mixing + spectral transformation + time evolution

Execution Pipeline

1. Initialize state

$$|\Phi(0)\rangle = |0\rangle|\psi(0)\rangle + |1\rangle|\phi(0)\rangle$$

2. Mix components

   (Hadamard on control qubit)⟶ couples displacement (ψ) and velocity (ϕ)

3. Transform to spectral domain (QFT)

   move to momentum basis where dynamics simplify

4. Apply time evolution

   $U(t) \approx e^{-iH_w t}$ ⟶ implemented using hardware-efficient approximation

5. Return to spatial domain (inverse QFT)

6. Unmix components (Hadamard) ⟶ recover ($\psi(t)$), ($\phi(t)$)

Efficient Quantum Evolution 2/2

Spectral Method Insight

After applying Quantum Fourier Transform, the operator becomes diagonal in the Fourier basis.

$$\Delta = QFT\ D\ QFT^*$$

The Laplacian transforms into a diagonal operator

- Each mode $|k\rangle$ evolves independently
- No interaction between different frequency components

Time Evolution Simplifies

$$U(t) = e^{-iHt} \Rightarrow |k\rangle \rightarrow e^{-i\lambda_k t}|k\rangle$$

Each state gets a phase shift only and no amplitude mixing

Why This Is Hardware-Efficient:

Phase shifts implemented via:

- $R_z(\theta)$ gates
- Controlled phase gates

These are native to most quantum devices and low-depth and low-error

Hardware-Efficient Approximation

After QFT, eigenvalues follow:

$$E_k = -4\sin^2\left(\frac{\pi k}{N}\right)$$

Quantum circuits do not natively implement nonlinear functions

($\sin(\cdot)$) requires:

- Function approximation (e.g., Taylor expansion)
- Many gates → deep, noisy circuits

$$\sin\left(\frac{\pi k}{N}\right) \approx \frac{\pi k}{N}$$

Eigenvalues and therefore Evolution becomes a simple function of k

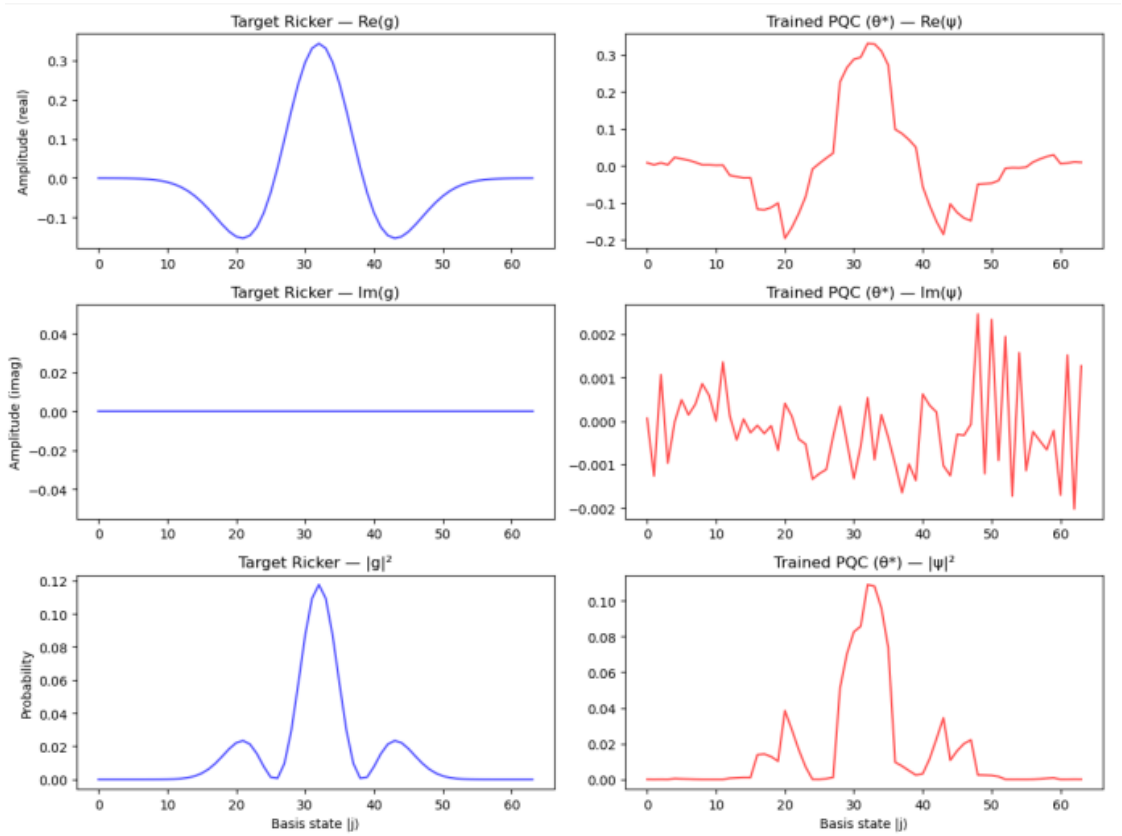
$$E_k \approx \left(\frac{\pi k}{N}\right)^2$$

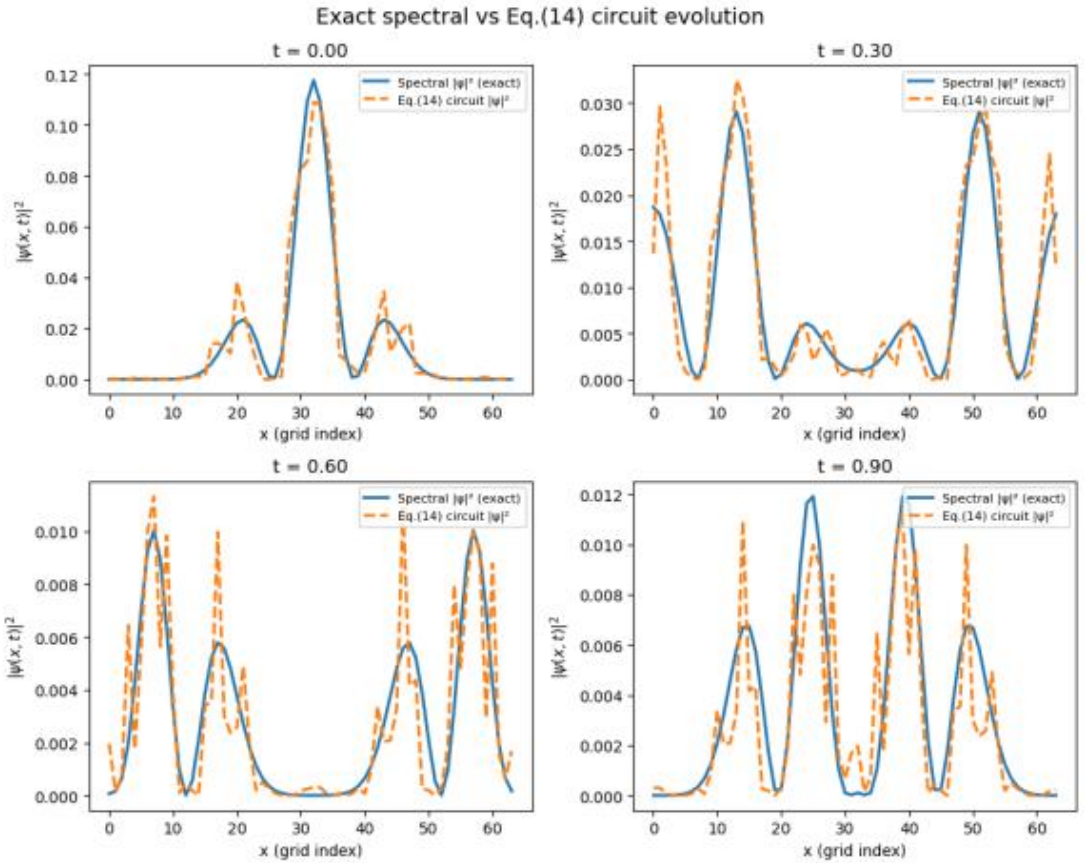
Hardware Implementation

- Linear dependence → decomposes into:
- Single-qubit (R_z) rotations
- Two-qubit ZZ interactions
- No complex function evaluation
- Reduced circuit depth
- Lower noise

Experiment

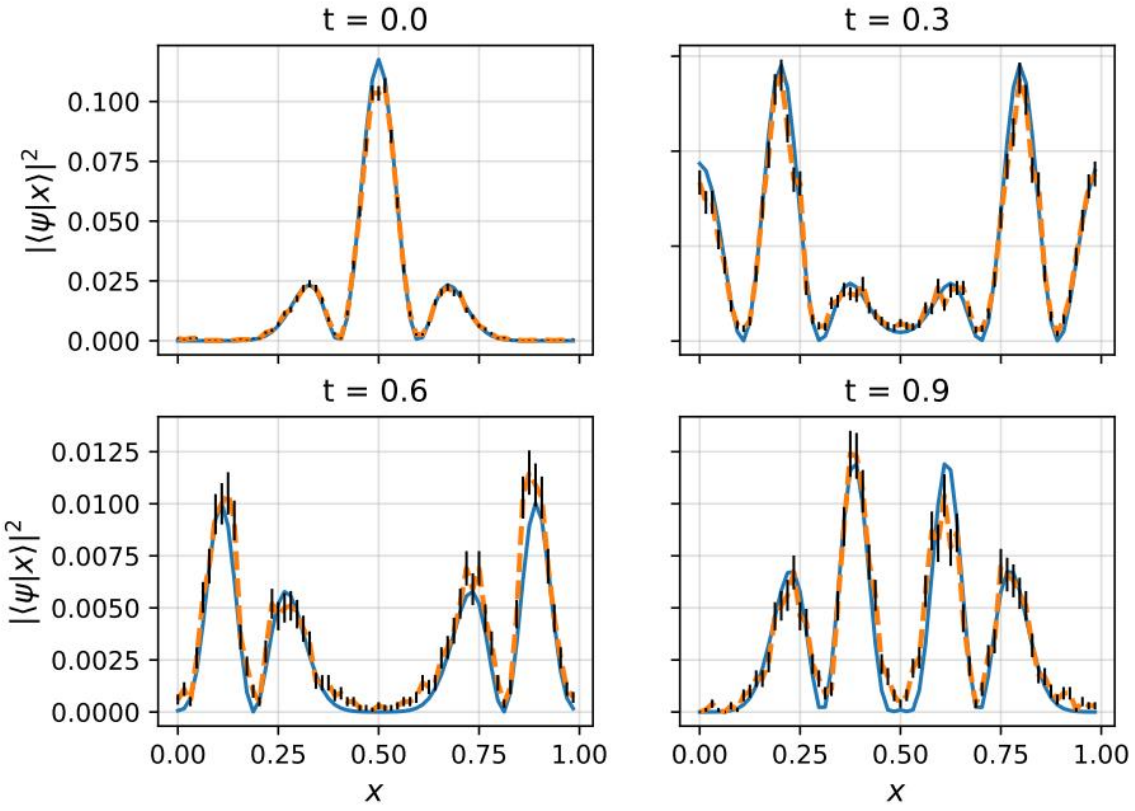
```
n_qubits = 7, n_sys = 6, N = 64, PQC depth = 3
```

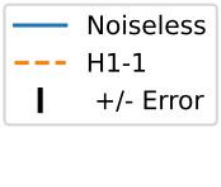




Quantinuum H1-1 quantum computer

(Wright et al., 2024),





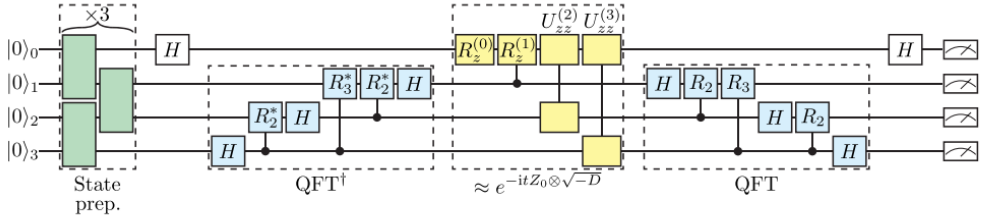


FIG. 1. Circuit for approximately simulating the wave equation (2) on a quantum computer for $n+1=4$ qubits. The circuit can be split into three parts: initial state preparation, time evolution and final state readout via qubit measurements. We define the initial state in terms of a variational brickwall circuit of depth floor $(\log_2(n+1))+1$, and is composed of generic nearest-neighbor two-qubit unitaries. The time evolution consists of a diagonal operator approximating $\exp(-itZ_0\otimes\sqrt{-D})$ preceded by the inverse quantum Fourier transform $\mathrm{QFT}^\dagger$ and proceeded by the QFT. Here H is the Hadamard gate, $R_\kappa = \mathrm{diag}(1, \exp(i2\pi/2^\kappa))$, R_κ^* is the conjugate of R_κ, $R_z^{(\kappa)} = \exp(-i\theta_\kappa Z)$, $U_{zz}^{(\kappa)} = \exp(-i\theta_\kappa Z\otimes Z)$, $\theta_0 = 3\pi t$, $\theta_1 = -8\pi t$, $\theta_2 = -2\pi t$ and $\theta_3 = -\pi t$. These gates are derived from Eq. (14). Note that we label the qubits 0, 1, ..., n from top to bottom.

FIG. 2. Quantinuum system model H1-1 hardware results for simulating a Ricker wavelet according to the wave equation at different times t and $n = 6$ qubits. The exact solution (blue solid line) is shown alongside the approximate solution (orange dashed line) which is sampled from H1-1 with $N_{\mathrm{shots}} = 10,000$ shots. Samples are collected by measuring all qubits. The black error bars represent $\pm\epsilon_{\mathrm{MC}}$, where $\epsilon_{\mathrm{MC}} = \sigma/\sqrt{N_{\mathrm{shots}}}$ is the Monte Carlo sampling error with σ being the standard deviation. Circuits were evaluated on the Quantinuum system model H1-1 [52] on December 1st-6th, 2023. All shots are included in the figure, however we only display the un-normalized state corresponding to the wavefield $\psi(x,t)$. Additional details about the experiments can be found in Appendix C.

Qiskit for VQSP

```
import matplotlib.pyplot as plt
import numpy as np
from qiskit import QuantumCircuit
from qiskit.circuit import Parameter
from qiskit.quantum_info import Statevector
from scipy.optimize import minimize
```

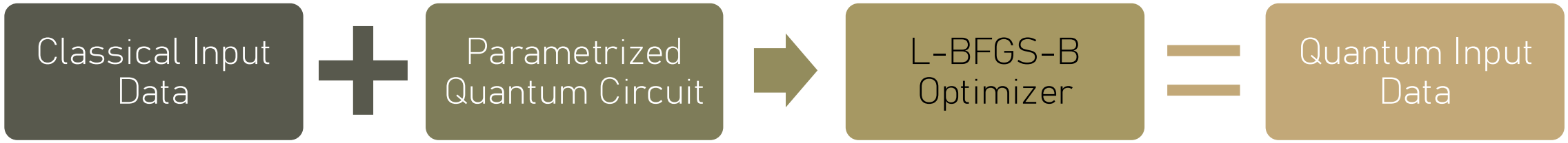
In this notebook, I used a parameterized quantum circuit (PQC) to prepare a target wavefield arising from the 1D acoustic wave equation. The PQC is trained by adjusting its parameters (θ) so that the output state

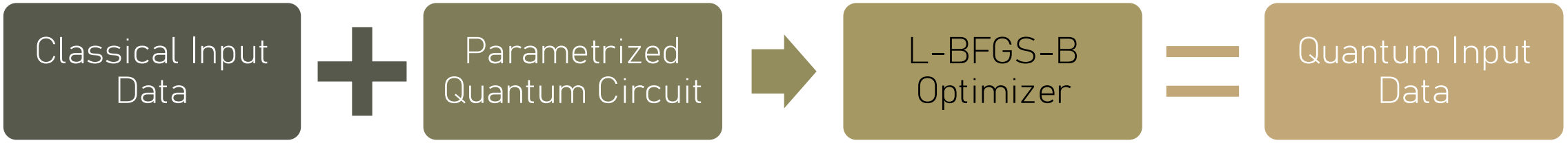
$$\psi(\theta) = U(\theta)\,|0\cdots0\rangle$$

Code can be found here:

https://github.com/AmnahSamarin/1D-Wave-Equation-Quantum-Simulation.git

notebooks/VariationalQuantumStatePreparation.ipynb





the continuous wavelet function need to efficiently encoded into the amplitude distribution over the computational basis.

We want the initial quantum state

$$|\Phi(0)\rangle = \sum_{j=0}^{N-1} \psi_j(0)|j\rangle$$

to encode a spatial function $\psi(x, 0)$ — in this case, a Ricker wavelet. The Ricker wavelet is a localized oscillatory function often used to represent a pulse:

$$\psi(x, 0) = \left(1 - \frac{x^2}{\sigma^2}\right) e^{-x^2/(2\sigma^2)}$$

They discretize this over $N = 2^n$ grid points, map $x_j \to j$, and normalize it to unit norm to be a valid quantum state.

They can't just "load" arbitrary amplitudes $\psi_j(0)$ directly — amplitude encoding is exponentially costly in general. So instead they use a shallow, trainable PQC (like a "quantum neural network") to approximate that Ricker wavelet distribution.

```python
def ricker_target_state(n_qubits, sigma=0.8):
    N = 2 ** n_qubits
    # grid centered around 0 with unit spacing (rescale as you like)
    x = np.linspace(-(N-1)/2, (N-1)/2, N) / (N/4)
    g = (1 - (x**2)/(sigma**2)) * np.exp(-x**2/(2*sigma**2))  # real-valued Ricker
    g = g.astype(np.complex128)
    g /= np.linalg.norm(g)  # normalize to unit vector
    return g
```

Parametrized Quantum Circuit

(a) Brickwall layout

- They use 7 qubits.
- Circuit depth = 3 (three alternating layers of two-qubit blocks).
- Total of 18 two-qubit blocks (each green rectangle).

Each two-qubit block has its own parameter set

$$\boldsymbol{\theta}_m = (\theta_1^{(m)}, \ldots, \theta_9^{(m)})$$

(b) Inside each two-qubit block

Each block is a generic SU(4)-like entangling unit parameterized efficiently.

It's structured as:

$$U^{(m)} = \left[U_1(\theta_1^{(m)}, \theta_2^{(m)}, \theta_3^{(m)}) \otimes U_1(\theta_4^{(m)}, \theta_5^{(m)}, \theta_6^{(m)})\right] \cdot U_{\text{ent}}(\theta_7^{(m)}, \theta_8^{(m)}, \theta_9^{(m)})$$

Where:

$$U_1(\alpha, \beta, \gamma) = R_z(\alpha) R_y(\beta) R_z(\gamma)$$

Entangling part uses:

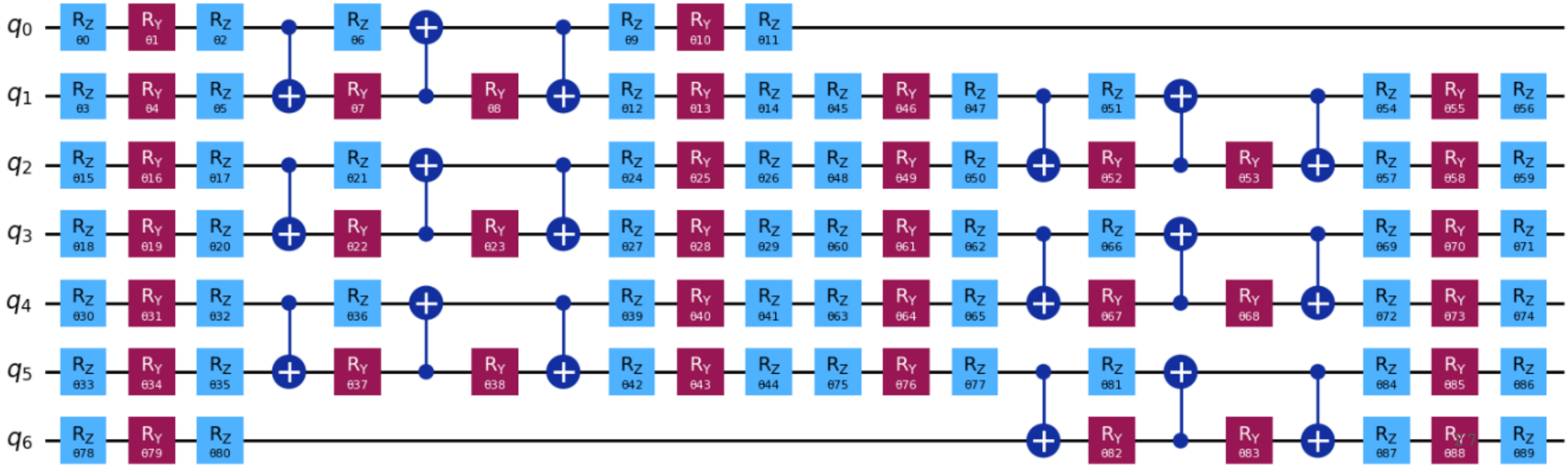
$$U_{\text{ent}} = \text{CNOT}_{1,2}\, R_z(\theta_7)\, R_y(\theta_8)\, \text{CNOT}_{2,1}\, R_y(\theta_9)$$

That's a universal two-qubit entangling gate parameterized by 9 angles.

```python
# --- 1. Generic single-qubit unitary U1(α,β,γ) = Rz(α) Ry(β) Rz(γ)
def U1(qc, q, a, b, c):
    qc.rz(a, q)
    qc.ry(b, q)
    qc.rz(c, q)

# --- 2. Three-CNOT entangling block (matches Fig. 5 b)
#     Sequence: CNOT(0→1) → Rz(q0) → Ry(q1) → CNOT(1→0) → Ry(q1) → CNOT(0→1)
def entangling_block(qc, q0, q1, thetas):
    # thetas = [θ7, θ8, θ9]
    qc.cx(q0, q1)          # first CNOT (q0→q1)
    qc.rz(thetas[0], q0)   # Rz on control
    qc.ry(thetas[1], q1)   # Ry on target
    qc.cx(q1, q0)          # second CNOT (q1→q0)
    qc.ry(thetas[2], q1)   # Ry on target again
    qc.cx(q0, q1)          # third CNOT (q0→q1)

# --- 3. Complete two-qubit block U^(m)
#     2× pre-rotations, 3-CNOT entangler, 2× post-rotations → 15 parameters total
def two_qubit_block(qc, q0, q1, thetas):
    """
    thetas = [θ1 ... θ15]
    """
    # Pre-rotations
    U1(qc, q0, thetas[0], thetas[1], thetas[2])
    U1(qc, q1, thetas[3], thetas[4], thetas[5])

    # Entangling section (3 CNOTs)
    entangling_block(qc, q0, q1, thetas[6:9])

    # Post-rotations
    U1(qc, q0, thetas[9], thetas[10], thetas[11])
    U1(qc, q1, thetas[12], thetas[13], thetas[14])
```

```
# --- 4. Assemble full brickwall PQC (Fig. 5 a)
def brickwall_PQC(n_qubits=4, depth=3):
    qc = QuantumCircuit(n_qubits)
    params = []
    param_index = 0

    for d in range(depth):
        # Alternate coupling pattern per layer: (0,1),(2,3) → (1,2) → repeat
        pairs = [(i, i + 1) for i in range(d % 2, n_qubits - 1, 2)]
        for (q0, q1) in pairs:
            thetas = [Parameter(f"θ{param_index + i}") for i in range(15)]
            params += thetas
            two_qubit_block(qc, q0, q1, thetas)
            param_index += 15
    return qc, params

# --- 5. Example usage
qc, params = brickwall_PQC(n_qubits=7, depth=2)
print(f"Total parameters: {len(params)}")
qc.draw('mpl')
```

```
Total parameters: 90
```



L-BFGS-B Optimizer

Train $U(\theta)$ so that $U(\theta)|0\rangle^{\otimes n} \approx |g\rangle$ by minimizing $C(\theta) = 1 - \mathrm{Re}\langle g|\psi(\theta)$

$$C(\theta) = 1 - \mathrm{Re}\left(\langle g(x)|U(\theta)|0\rangle^{\otimes n}\right)$$

- $U(\theta)$: the parameterized quantum circuit (PQC) that prepares the quantum state.
- $|0\rangle^{\otimes n}$: the initial "all-zeros" state on the quantum computer.
- $|g(x)\rangle$: the target wavefunction (Ricker wavelet) represented as a complex vector.
- $|\psi(\theta)\rangle = U(\theta)|0\rangle^{\otimes n}$: the state produced by the PQC for a given set of parameters.
- $\langle g|\psi\rangle$: the complex inner product (overlap) — tells how close the two states are.
- The cost function $C(\theta)$ is minimized when the real part of this overlap is maximized (i.e., when $|\psi\rangle$ matches $|g\rangle$).
- If the PQC perfectly matches the target ($\langle g|\psi\rangle = 1$),
  $\rightarrow C = 1 - 1 = 0 \rightarrow$ minimum cost.
- If the PQC produces an orthogonal state ($\langle g|\psi\rangle = 0$),
  $\rightarrow C = 1 - 0 = 1 \rightarrow$ maximum cost.

```python
def cost(theta_vec):
    """One time cost calculation"""
    bind_map = {p: float(theta_vec[i]) for i, p in enumerate(params)}
    qc_bound = qc.assign_parameters(bind_map)
    psi = Statevector.from_instruction(qc_bound).data
    overlap = np.vdot(g, psi)          # <g|ψ(ϑ)>
    return 1.0 - np.real(overlap)      # paper's cost function

# build cost
cost = make_cost(qc, params, g)

# optimize with L-BFGS-B (increase maxiter toward 100_000 if you want to
res = minimize(
    cost,
    theta0,
    method="L-BFGS-B",
    options=dict(maxiter=600, ftol=1e-9, gtol=1e-8, maxcor=20, eps=1e-8)
)

print("\nOptimization status:", res.message)
print("Iterations:", res.nit)
print("Final cost:", res.fun)
```

In [66]:

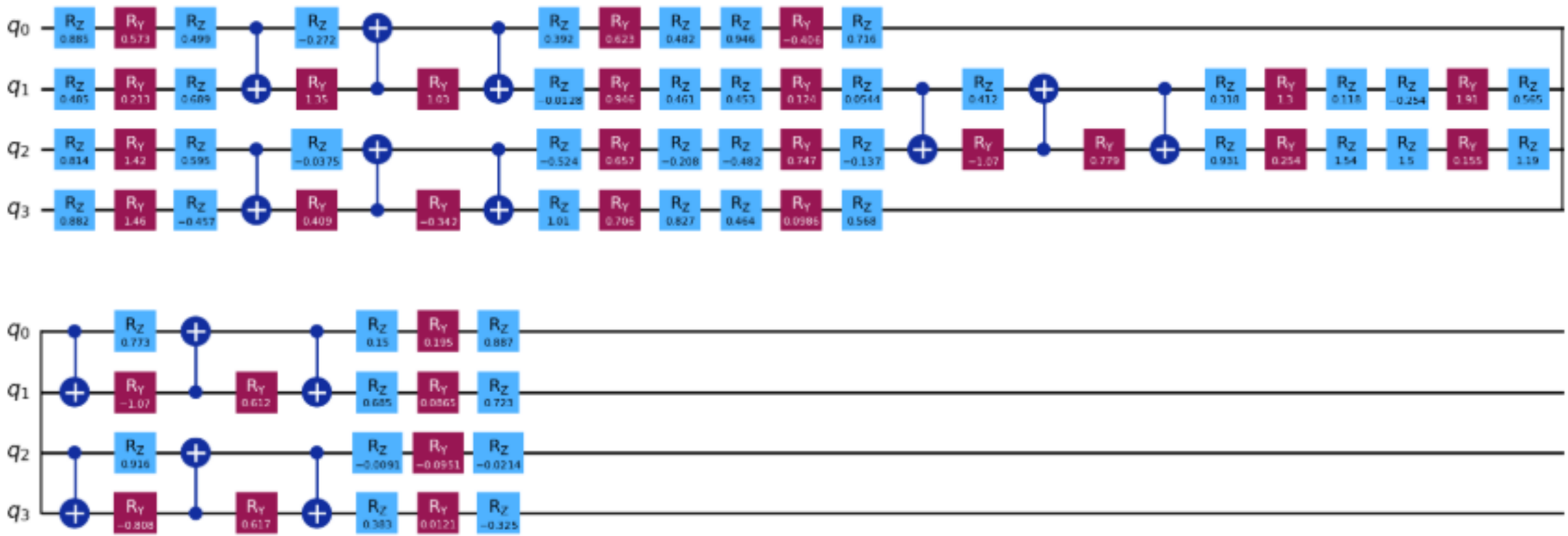
```python
theta_star = res.x
report(qc, params, theta_star, g)

# If you want the hardware-ready circuit with learned angles:
trained_qc = bind_params(qc, params, theta_star)
```

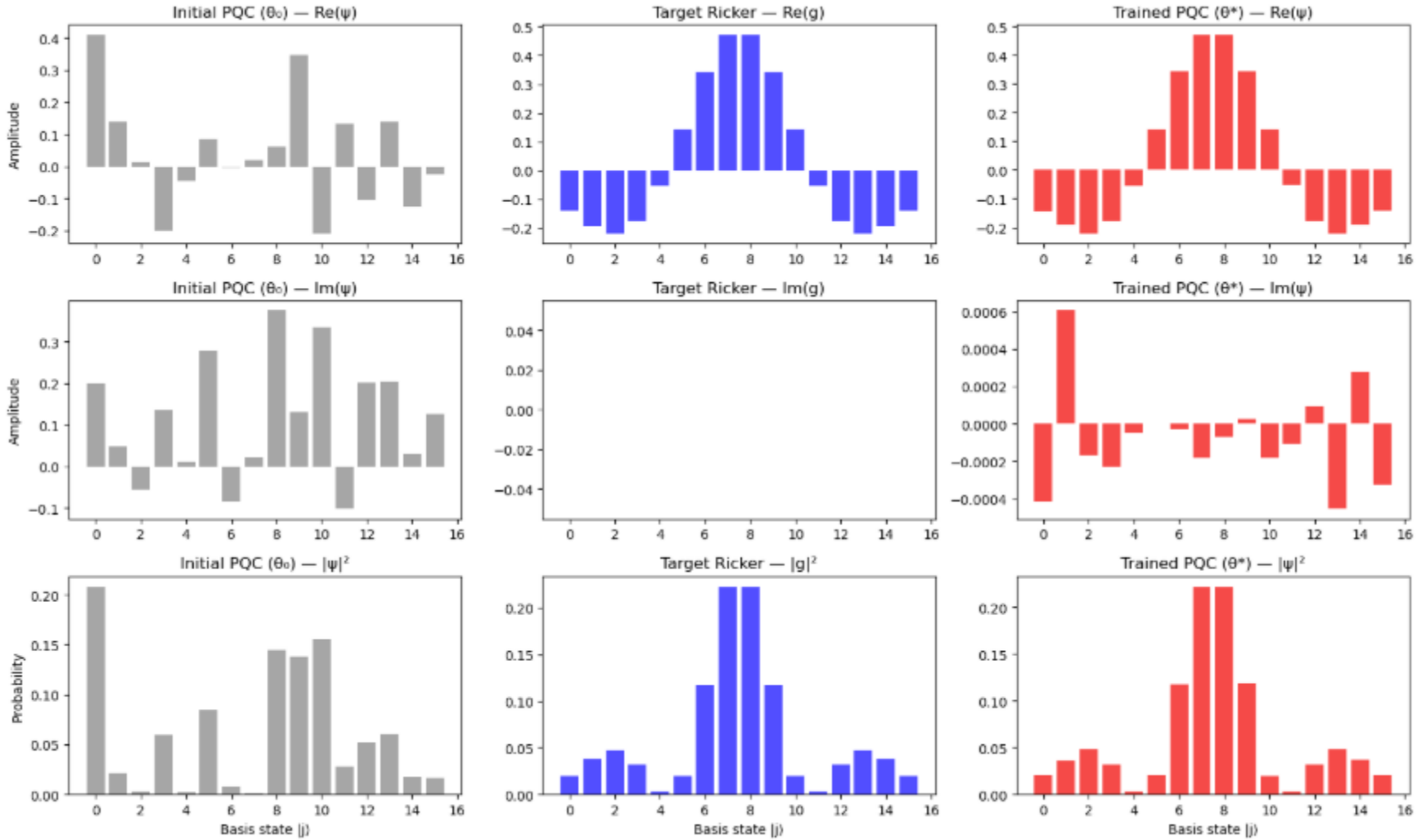
```
Final cost C(θ): 0.000024
Re(<g|ψ>): 0.999976,  Im(<g|ψ>): -0.000048
Fidelity |<g|ψ>|^2: 0.999952
```

In [67]:

```
trained_qc.draw('mpl')   # visualize, or submit to your backend of choice
```

Out[67]:




Initial PQC (θ₀) — Re(ψ)
Target Ricker — Re(g)
Trained PQC (θ*) — Re(ψ)
Initial PQC (θ₀) — Im(ψ)
Target Ricker — Im(g)
Trained PQC (θ*) — Im(ψ)
Initial PQC (θ₀) — |ψ|²
Target Ricker — |g|²
Trained PQC (θ*) — |ψ|²
Amplitude
Probability
Basis state |j⟩

Enough of geophysics, Let's move to why does quantum computing matter for AI ?

NOTE: The same quantum principles used to simulate physical systems can be applied to learning complex patterns in data.

Quantum application in AI

1. *BRIDGE: FROM CLASSICAL ML TO QUANTUM ML*
    1. Overview of Artificial Intelligence and Machine Learning
    2. Quantum Machine Learning
2. QUANTUM NEURAL NETWORKS
3. APPLICATIONS

1. Bridge: From Classical ML to Quantum ML

- Overview of Artificial Intelligence and Machine Learning
- Quantum Machine Learning

Machine Learning

Machine learning is a field of artificial intelligence where models learn patterns from data to make predictions or decisions.

Learning Model

$$y = f(x, \theta)$$

Key Components

- Data (x): input features
- Model f: mapping from input to output
- Parameters θ: learned during training

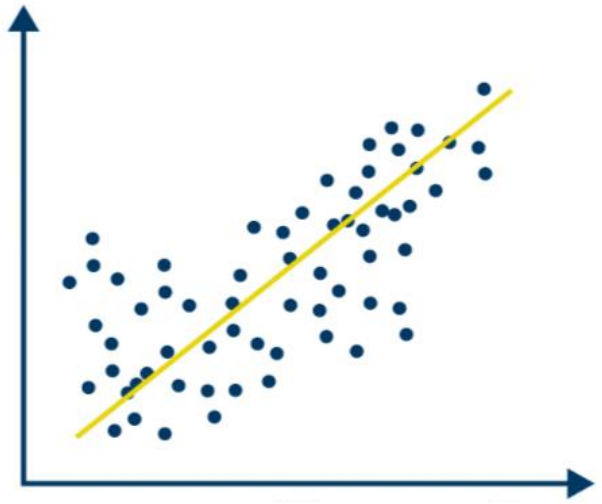
Learning Objective

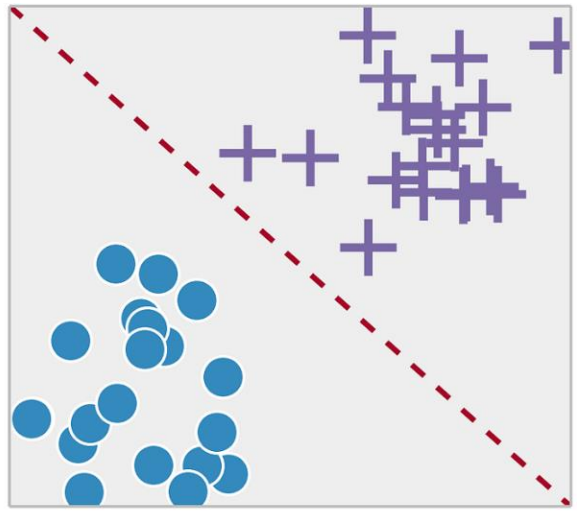
$$\theta^* = \arg\min_{\theta} L(\theta)$$

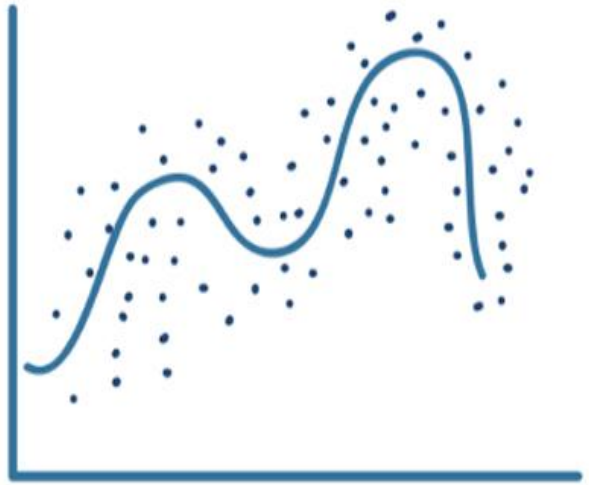
Loss function $L(\theta)$:measures prediction error

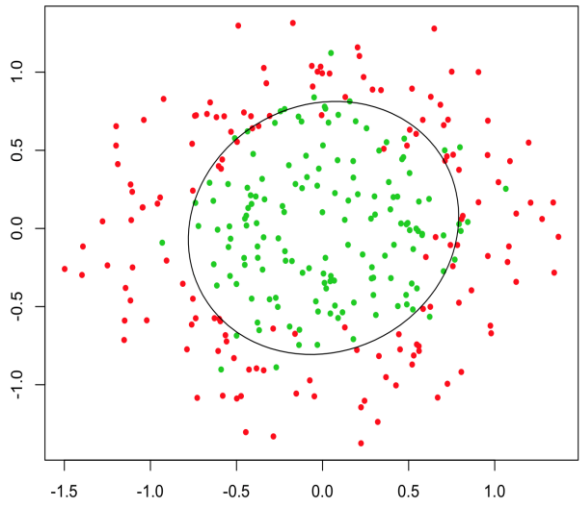
Common Types

- Supervised learning – classification, regression
- Unsupervised learning – clustering, dimensionality reduction
- Reinforcement Learning









Deep Learning Model

Deep learning is a machine learning approach that uses multi-layer neural networks to automatically learn complex patterns from data.

Learning Model

$$y = f(\mathrm{x}, \theta) = f_L(f_{L-1}(\ldots f_1(x)))$$

Key Components

- Input x: feature vector (data)
- Layers f_l :nonlinear transformations
- Weights $\theta = \{W_l, b_l\}$:trainable parameters

Example layer:

$$h_l = \sigma(W_l h_{l-1} + b_l)$$

Learning Objective

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L\,(f,(x_i, \theta) y_i)$$

Common Architectures

- Convolutional Neural Networks (CNN) – vision tasks
- Recurrent Neural Networks (RNN) – sequence data
- Transformers – large language models

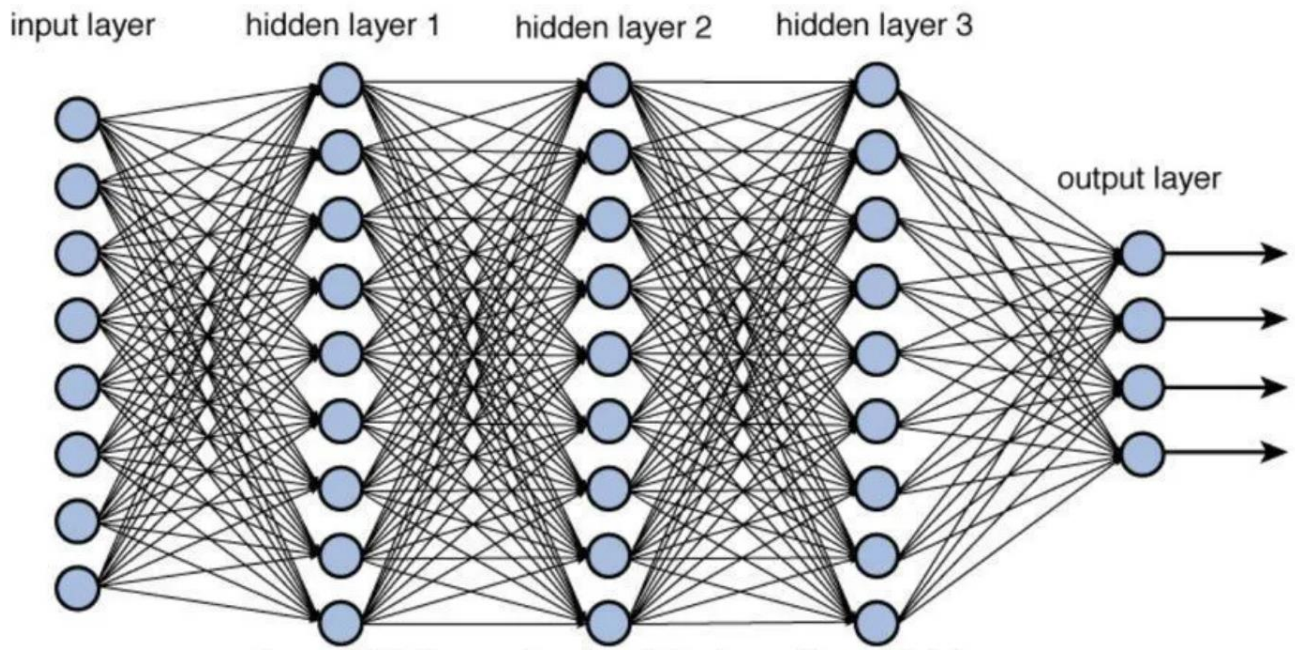
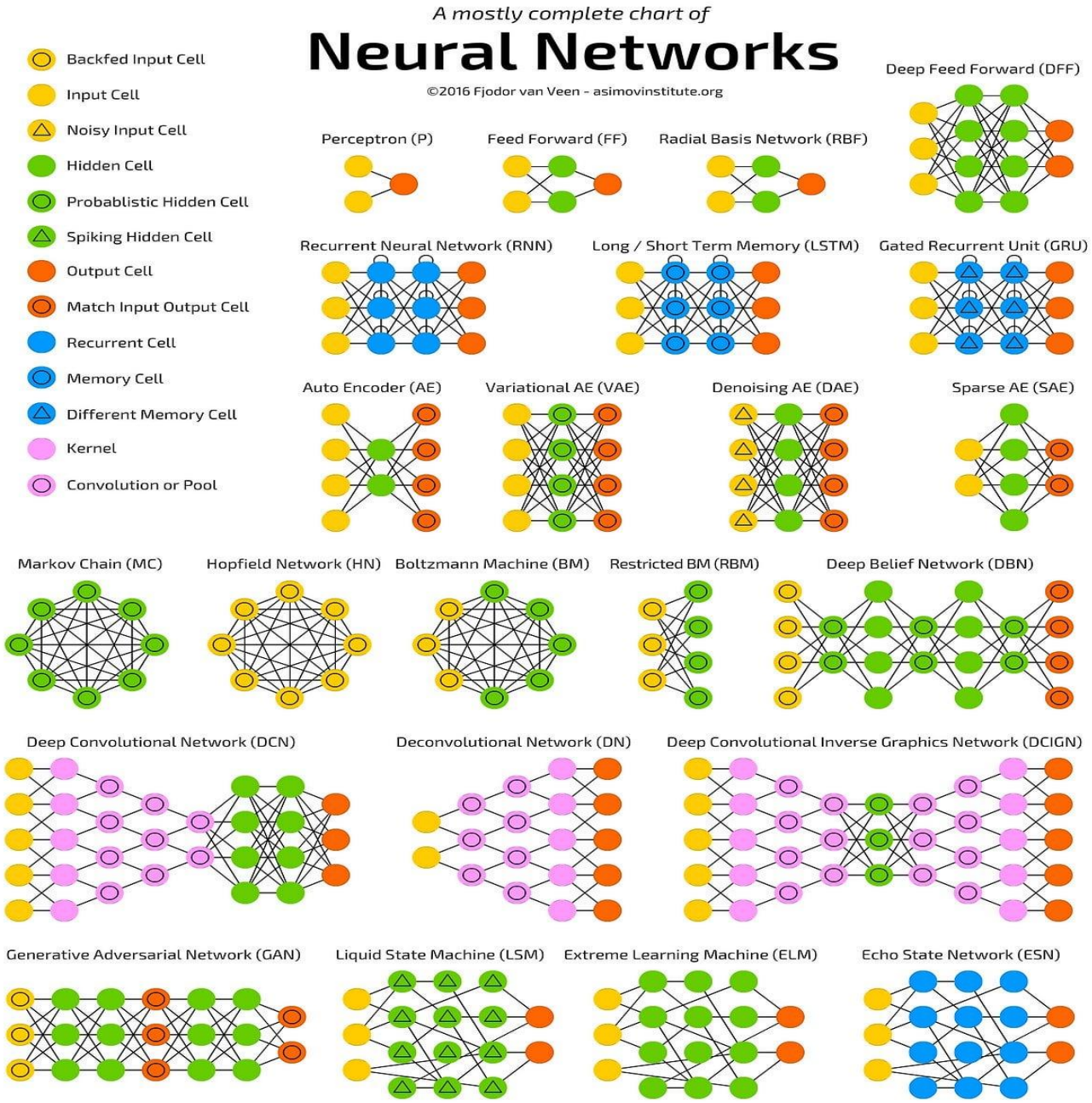


Figure 12.2 Deep network architecture with multiple layers.

Karataş, M. A., & Biberci, M. A. (2023). *Statistical analysis of WEDM machining parameters of Ti-6Al-4V alloy using Taguchi-based grey relational analysis and artificial neural network.* Experimental Techniques, 47, 851–870.

Deep Learning



Gadepally, V., Kepner, J., Milechin, L., & Samsi, S. (2020). *Artificial Intelligence and Machine Learning*. Massachusetts Institute of Technology

Why Quantum Computing in Artificial Intelligence?

Artificial intelligence systems face significant computational bottlenecks due to the challenges of high-dimensional data, large-scale optimization, and complex non-convex learning processes.

Curse of Dimensionality

- Volume grows exponentially with dimension
- Data becomes sparse ⇒ poor generalization

Optimization Complexity

- Training = minimize non-convex loss
- Many local minima, saddle points
- Gradient-based methods scale poorly with dimension

Computational Cost

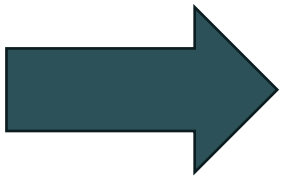
- Training cost scales or worse
- Large models ⇒ massive compute + memory requirements

Algebra Bottleneck

- Core ML operations:
- Matrix multiplication
- Eigenvalue problems

Classical Computing

Classical systems struggle with exponential scaling and large feature spaces



Quantum Computing

Represent 2^n-dimensional states efficiently giving potential speedups in linear algebra & sampling:

From Classical to Quantum Learning

From Classical ML

Classical ML learns a function:

$$y = f(x, \theta)$$

In ML, you minimize:

$$L(\theta) = \text{loss}(\text{prediction}, \text{target})$$

Training = optimize parameters, Goal:

$$\theta^* = \arg\min_{\theta} L(\theta)$$

to Quantum ML

we redefine: $E(\theta) \longrightarrow L(\theta)$

Instead of: "energy of a physical system" We now have: "error on data"

$$|\psi(x, \theta)\rangle = U(\theta)U_{\phi}(x)|0\rangle$$

$$L(\theta) = \sum_{i} \ell(y_i, \langle \psi(x_i, \theta) \mid M \mid \psi(x_i, \theta) \rangle)$$

The "cost" depends on: input data (x_i) and labels (y_i)

Quantum Comparison

In quantum In VQAs, you minimize:

$$E(\theta) = \langle \psi(\theta) \mid H \mid \psi(\theta) \rangle$$

$$\theta^* = \arg\min_{\theta} E(\theta)$$

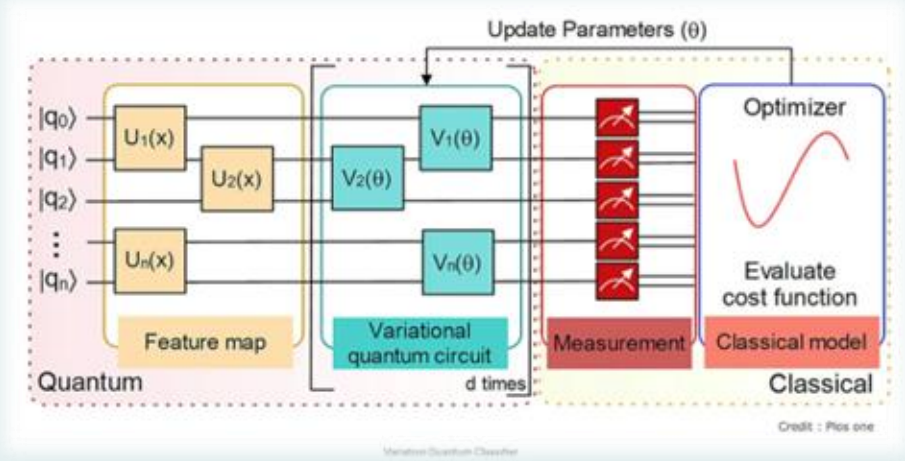
You are tuning the circuit so the quantum state has minimum energy

VQA in ML workflow

Variational Quantum Machine Learning uses parameterized quantum circuits (PQCs) trained with classical optimization to learn predictive models.

1. Quantum Feature Mapping: embed classical data into a high-dimensional Hilbert space.
   $$|\psi(x)\rangle = U_{\phi}(x)|0\rangle$$
2. Apply circuit $\rightarrow U_{\cdot}(\theta)$
3. Measure observable M
4. Update parameters θ

The Hamiltonian in VQAs is replaced by a data-dependent loss function in machine learning



Sen, P., Bhatia, A. S., Bhangu, K. S., & Elbeltagi, A. (2022). *Variational quantum classifiers through the lens of the Hessian*. PLOS ONE, 17(1), e0262346.

3. Quantum Neural Networks (QNNs)

- VQA
- QNNs
- QCNNs

VQCs → QNNs

Variational Quantum Circuits (VQCs) can build **Quantum Neural Networks (QNNs)** because they learn by **optimizing trainable parameters**, similar to classical neural networks.

Similarities to Classical Neural Networks

- Both models learn by adjusting parameters:

$$\theta^* = \arg\min_{\theta} L(\theta)$$

- Parameters are trained to minimize a **loss function**
- Both concpets perform **function approximation**

Key Differences

Classical Neural Networks	Quantum Neural Networks
Nonlinear activation functions	Entanglement operations
Irreversible operations	Reversible unitary operations
Classical feature space	Hilbert space representation
Classical weights	Parameterized quantum gates

Quantum Neural Networks (QNNs)

A **Quantum Neural Network (QNN)** is a machine learning model implemented using **parameterized quantum circuits** where quantum gates play the role of trainable weights.

QNN Architecture

Data Encoding Layer

Classical input data is encoded into a quantum state.

$$|\psi_0(x)\rangle = U_\phi(x)\,|0\rangle^{\otimes n}$$

Symbols

x— input feature vector

$U_\phi(x)$ —feature map circuit

n— number of qubits

$|0\rangle^{\otimes n}$ —initial quantum state

Objective

Embed classical data into a **high-dimensional Hilbert space**.

Quantum Hidden Layers

Parameterized quantum gates transform the quantum state.

$$|\psi_l\rangle = U_l(\theta_l)\,|\psi_{l-1}\rangle$$

Symbols

$U_l(\theta_l)$ —trainable unitary transformation

θ_l —circuit parameters

l— layer index

Typical Layer Structure

Rotation gates (trainable parameters)

$$R_y(\theta)$$

Entanglement gates

$$\text{CNOT}(i, j)$$

Objective

Learn complex feature transformations using **superposition and entanglement**.

Measurement Layer

Prediction is extracted through measurement.

$$y = \langle\psi_L\,|\,M\,|\,\psi_L\rangle$$

Symbols

M— measurement observable (Pauli operator)

L— final circuit layer

y— predicted output

Objective

Convert the quantum state into a **classical prediction**.

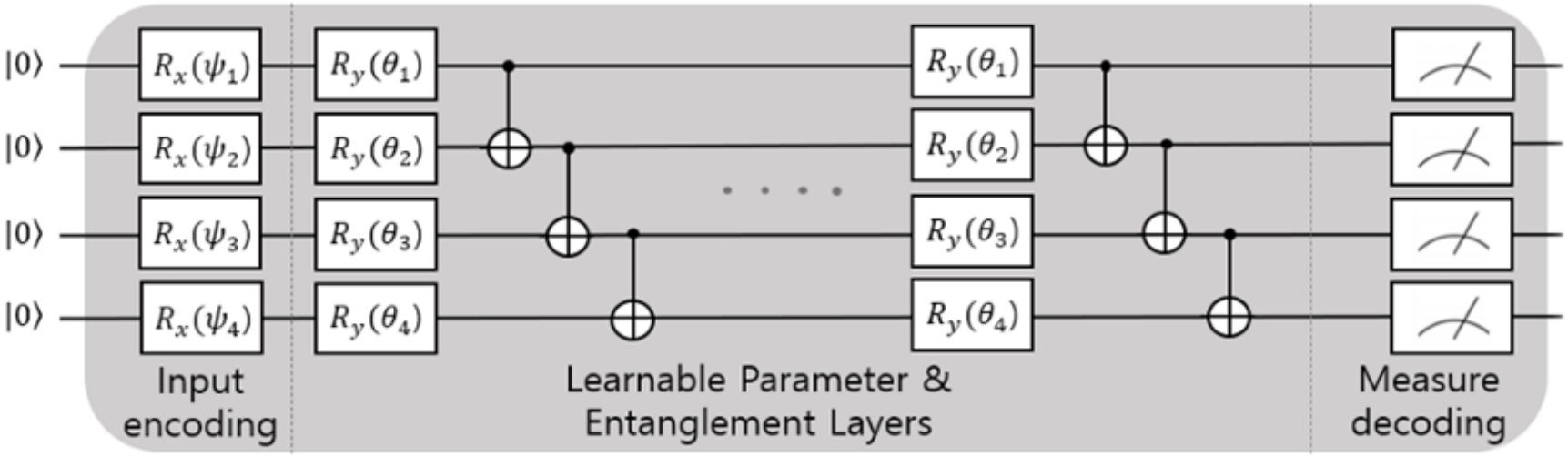
Learning Objective

$$\theta^* = \arg\min_\theta . L(f'(x'\theta)y)$$

Parameters are optimized using **classical optimizers**.

Potential **expressivity beyond classical models**

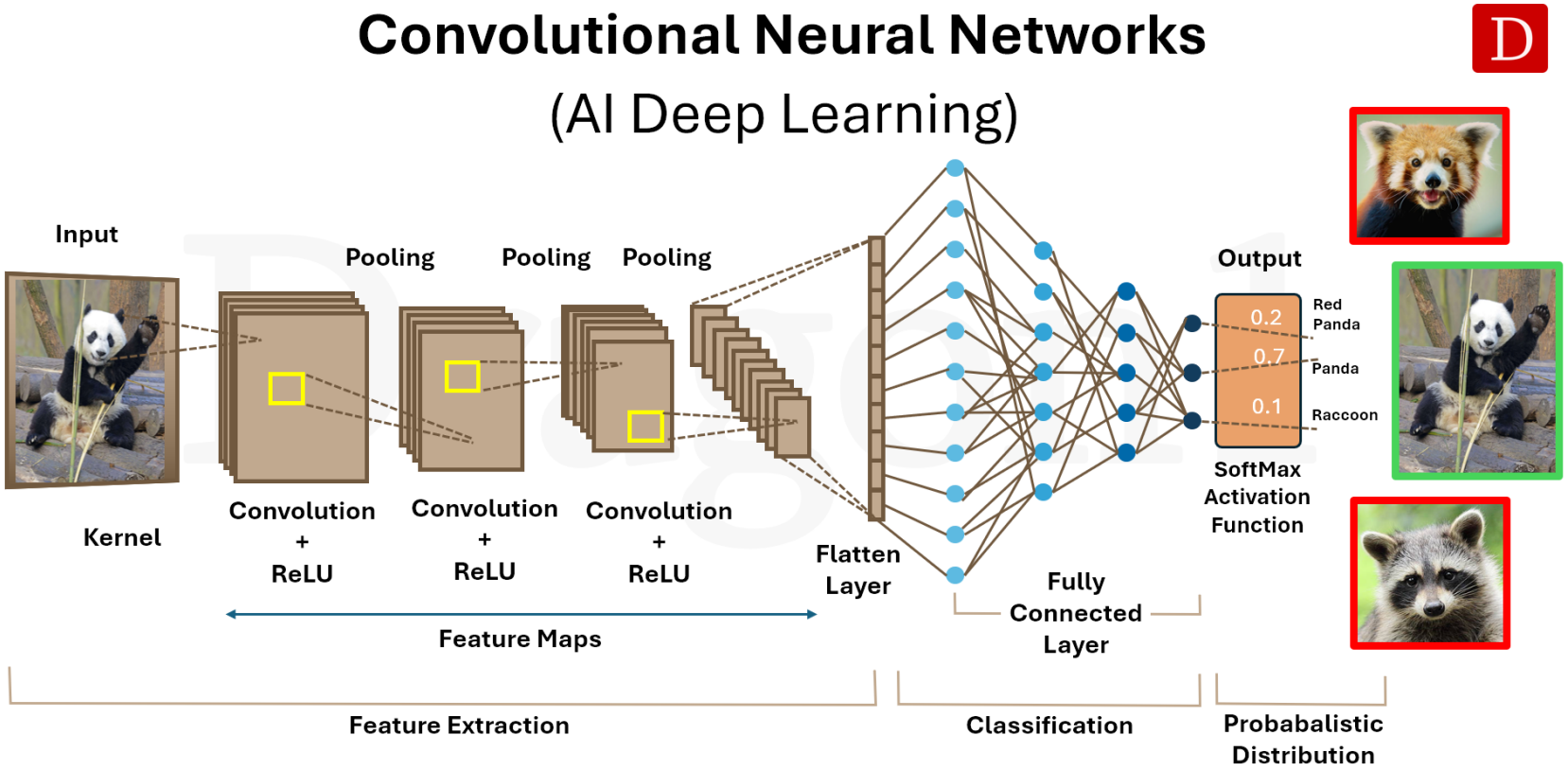
Quantum Neural Networks (QNNs)



Kwak, Y., Yun, W. J., Jung, S., & Kim, J. (2022). *Quantum neural networks: Concepts, applications, and challenges*.

Classical Neural Network	Quantum Neural Network
Input layer	Data encoding circuit
Hidden layers	Rotation + entanglement layers
Weights	Parameterized quantum gates
Activation functions	Entanglement & quantum interference
Output layer	Measurement

Convolutional Neural Networks



Dragon1. (2025). *Convolutional neural networks (CNN) architecture* Retrieved from https://www.dragon1.com

Quantum Convolutional Neural Networks (QCNNs)

A Quantum Convolutional Neural Network (QCNN) is a quantum machine learning architecture inspired by classical CNNs that uses parameterized quantum gates, convolution layers, and pooling operations to extract hierarchical features from quantum data.

QCNN Architecture

Input Encoding Layer

Classical input data is encoded into qubit states using rotation gates.

$$|\psi(x)\rangle = U_\phi(x)\,|0\rangle^{\otimes n}$$

Components

$R_x(\psi_i)$ —rotation gates encoding data

$|0\rangle$ —initial qubit states

x— classical input features

Objective

Embed classical data into a quantum feature space.

Quantum Convolution Layers

Parameterized unitary gates act locally on neighboring qubits.

$$|\psi_1\rangle = U(\theta)\,|\psi(x)\rangle$$

Components

$U_1(\theta_i)$ —trainable convolution gates

Local interactions between qubits

Objective

Extract local quantum features, similar to filters in classical CNNs.

Quantum Pooling Layer

Pooling is implemented using controlled quantum gates.

Operation

$$|\psi_2\rangle = V(\theta)\,|\psi_1\rangle$$

Components

Controlled rotations

Reduction of qubit number

Objective

Compress feature information and reduce circuit complexity.

Fully Connected Quantum Layer

Global unitary transformation combines extracted features.

$$|\psi_f\rangle = F\,|\psi_2\rangle$$

Objective

Integrate features for final prediction.

Measurement Layer

The final quantum state is measured to produce output.

$$y = \langle\psi_f\,|\,M\,|\,\psi_f\rangle$$

Components

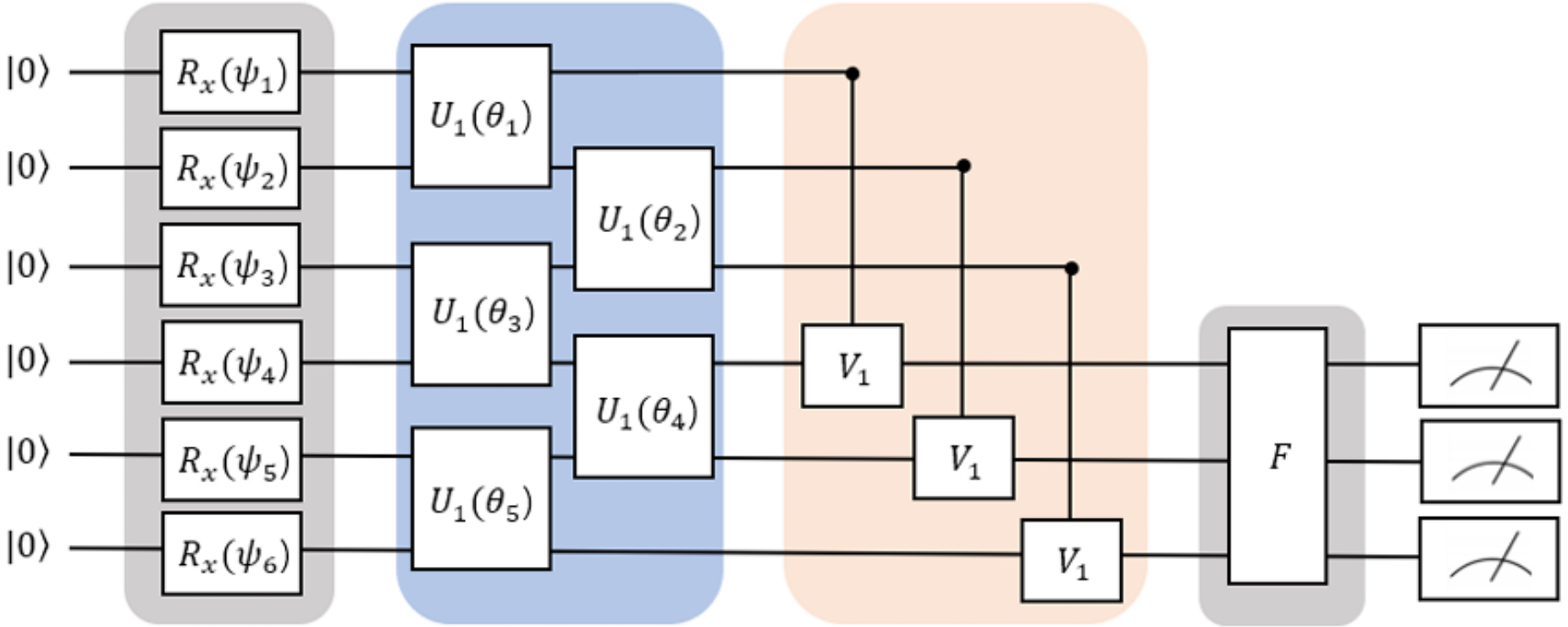
Measurement operators (Pauli gates)

Classical output values

Key Advantages

- Hierarchical feature extraction
- Efficient quantum data compression
- Potential exponential representation in Hilbert space

Quantum Convolutional Neural Networks (QCNNs)



Kwak, Y., Yun, W. J., Jung, S., & Kim, J. (2022). *Quantum neural networks: Concepts, applications, and challenges*.

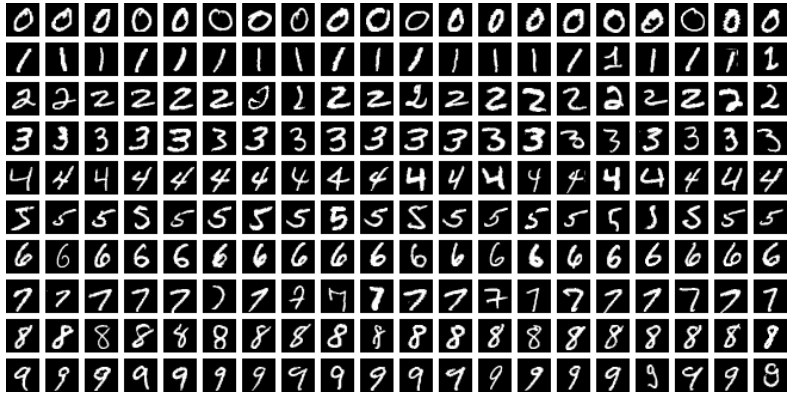
4. Applications and Limitations

Quantum Neural Networks: A Comparative Analysis and Noise Robustness Evaluation

Authors: Tasnim Ahmed, Muhammad Kashif, Alberto Marchisio, Muhammad Shafique
New York University Abu Dhabi (NYUAD), UAE

Briefly

Goal	Benchmark HQNN architectures and identify the most noise-robust one.
Models	QuanNN, QCNN, and QTL; each is a hybrid model built around a VQC.
Experimental setup	MNIST [0,1,2,3], 4 qubits, entanglement {basic, strong, weak}, depth 1–6.
Why it matters	Architecture choice depends on the device's dominant noise profile.



In this paper, we analyze three different HQNN algorithms to assess how the circuit architecture impacts performance in an ideal environment without noise. We then explore the effects of different device noise by evaluating the performance of the most effective models in noisy conditions. Our analysis is divided into two main sections: noise-free and noise robustness analysis. Figure 3 provides a detailed overview of our methodology.

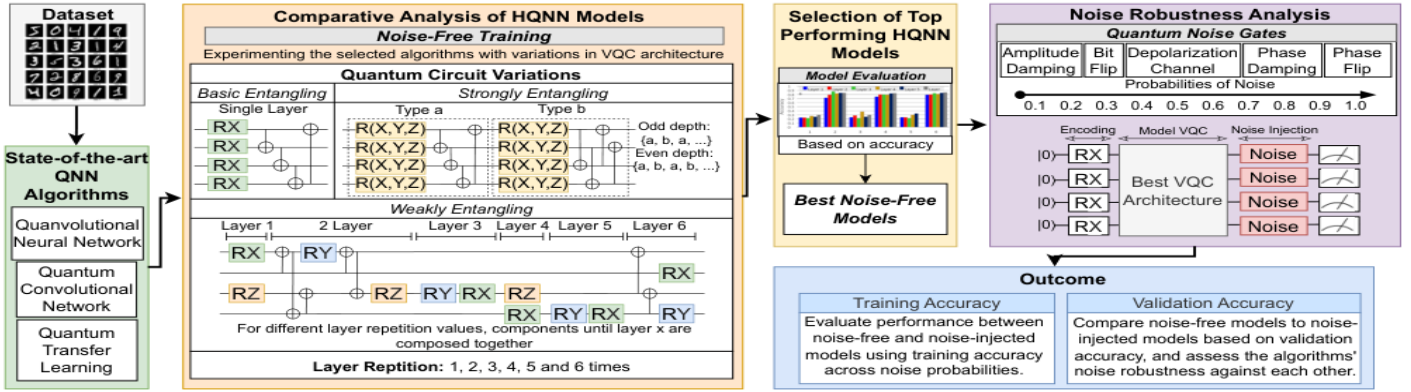
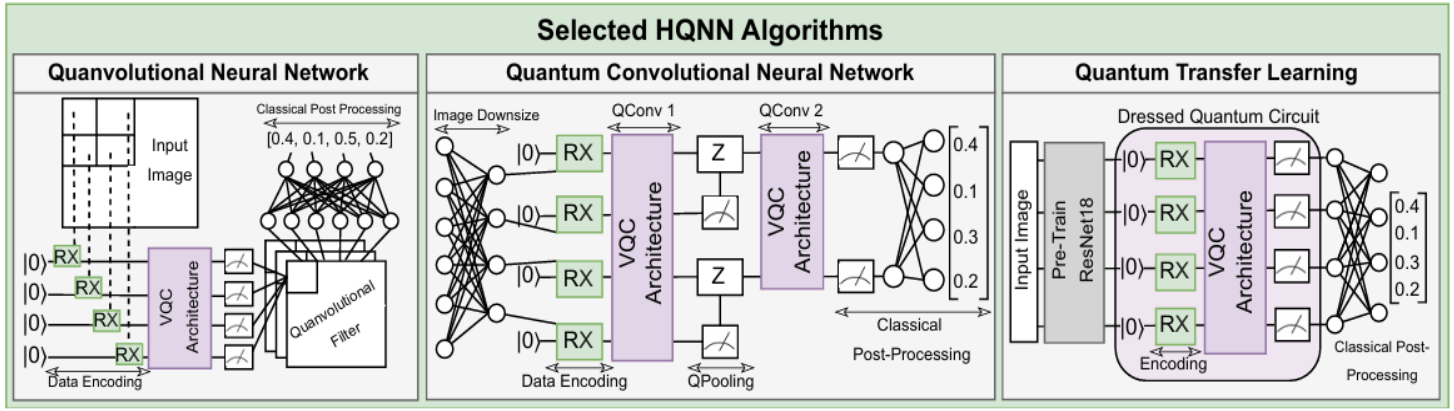
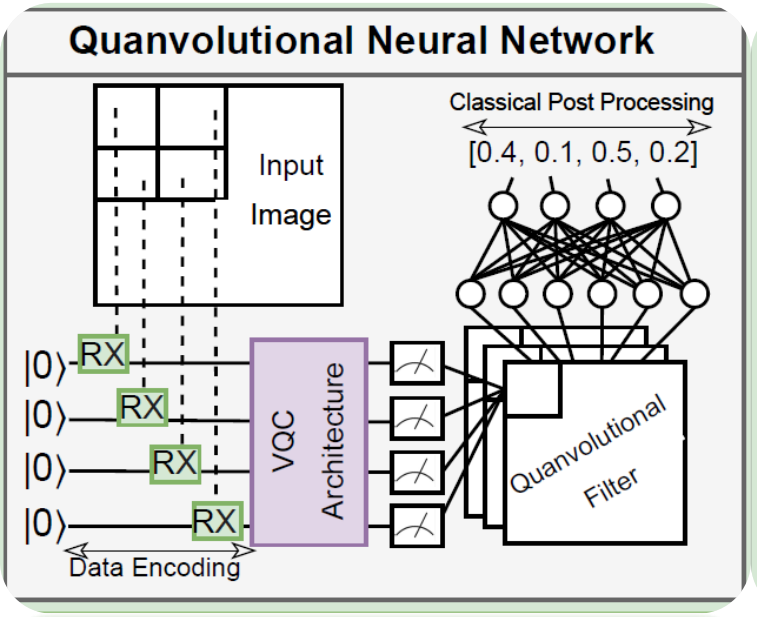
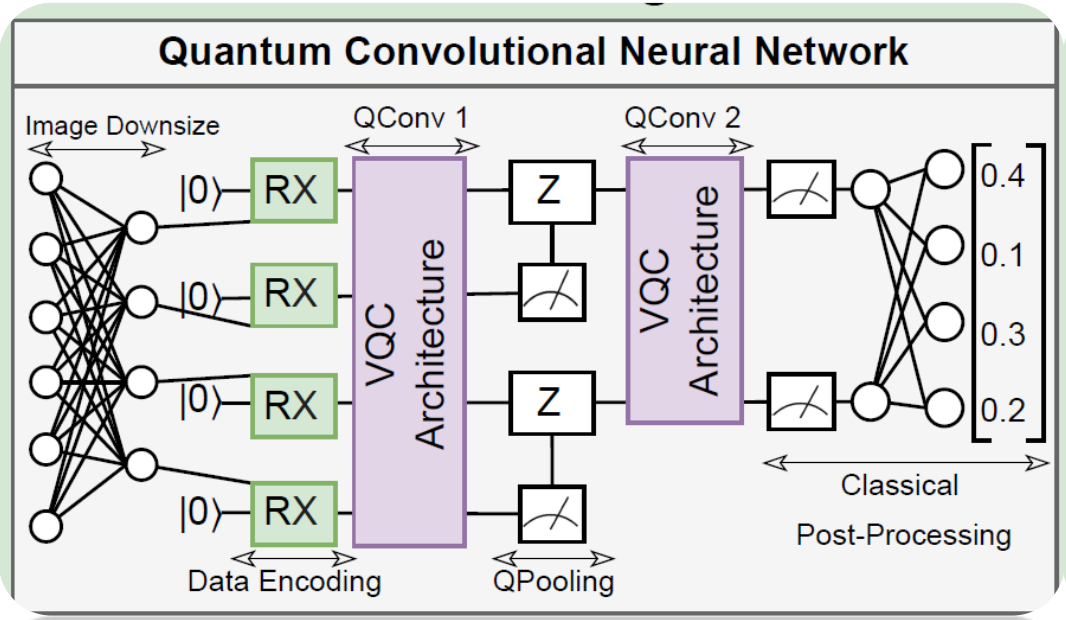


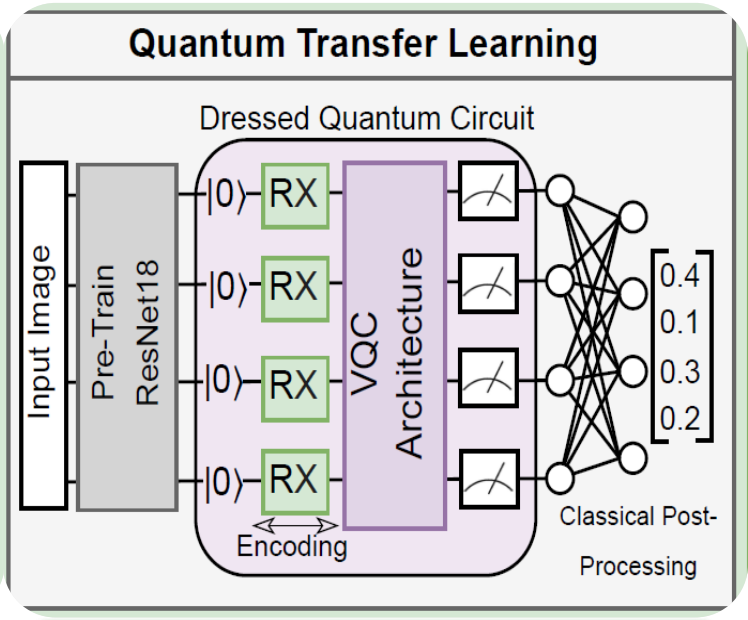
Figure 3. Our methodology. A comprehensive comparative analysis of three different variants of HQNNs is performed with different configurations of quantum layers mainly differing in degree of entanglement, rotation gates, and number of layers (depth of quantum layers). The odd and even depth in a strongly entangling configuration denotes how the layer is repeated



HQNN algorithms architectures.







Quantum Noise & Error Gates

Quantum noise refers to the unwanted random fluctuations that occur in quantum systems, due to factors such as environmental interference and the uncertainty principle.

Bit Flip

- A bit flip error in quantum computing is similar to a bit flip in classical computing
- A bit flip error causes a qubit initially in the state $|0\rangle$ to switch to $|1\rangle$, and vice versa

Phase Flip Noise

- Changes the phase of the qubit by flipping the sign of the $|1\rangle$ state while leaving $|0\rangle$ unchanged

Phase Damping

- Causes loss of quantum coherence without changing the actual state $|0\rangle$ or $|1\rangle$
- It reduces superposition by degrading phase relationships between states

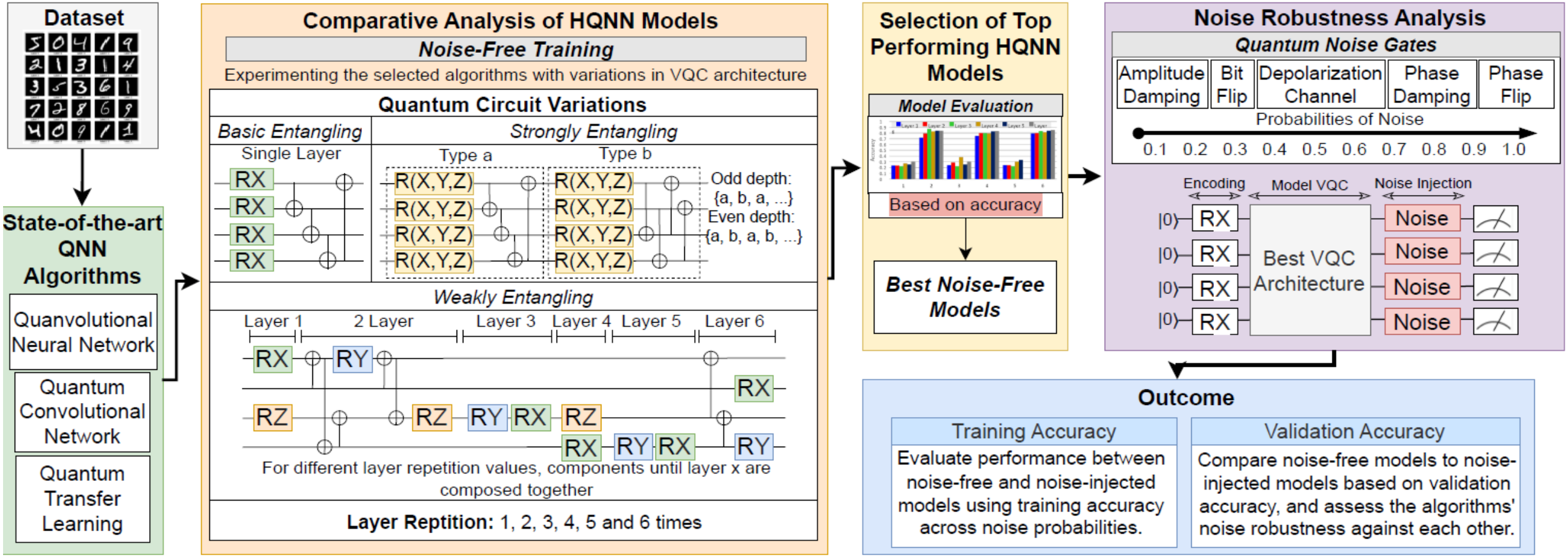
Amplitude Damping

- The amplitude damping represents a type of quantum noise that occurs when a qubit transitions from an excited state $|1\rangle$ to a ground state $|0\rangle$ due to the loss of energy.

Depolarizing Channel

- The depolarizing channel is a noise model in quantum computing that describes a process wherein qubits lose their quantum information to the environment,

Methodology



Hybrid Quantum Neural Network – Main Findings Noise Free

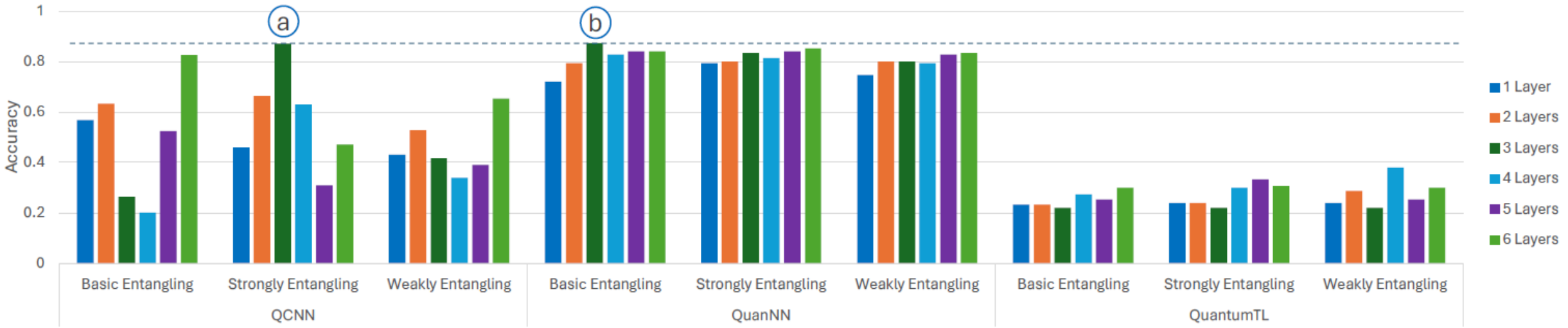
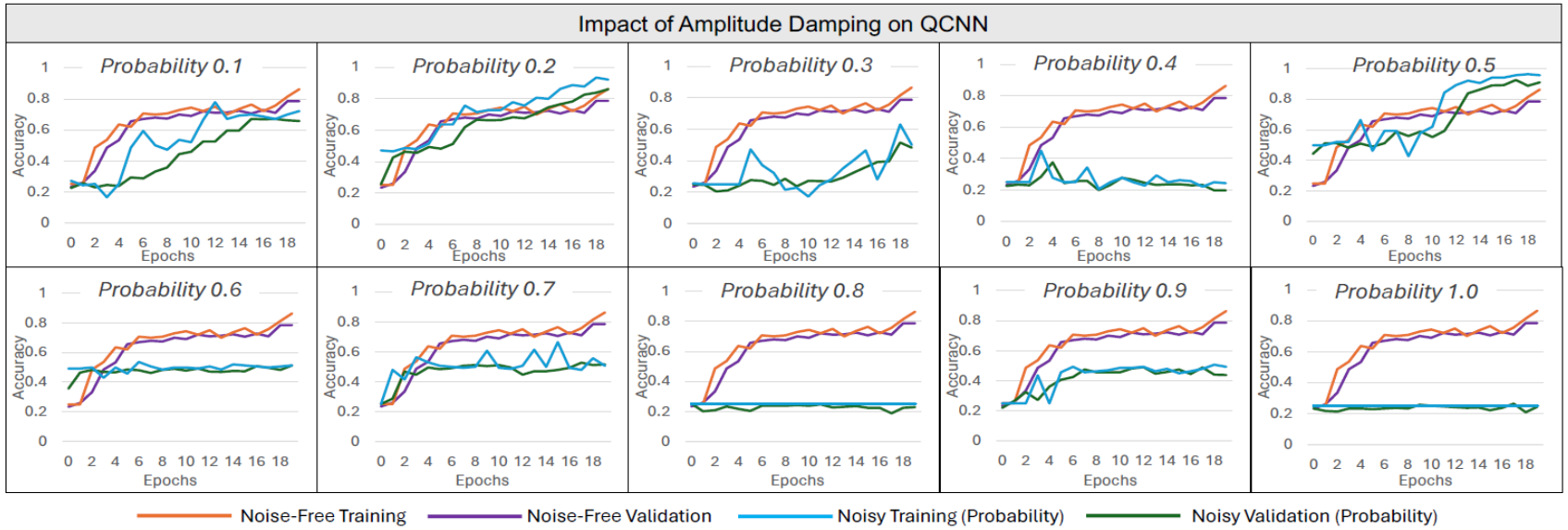


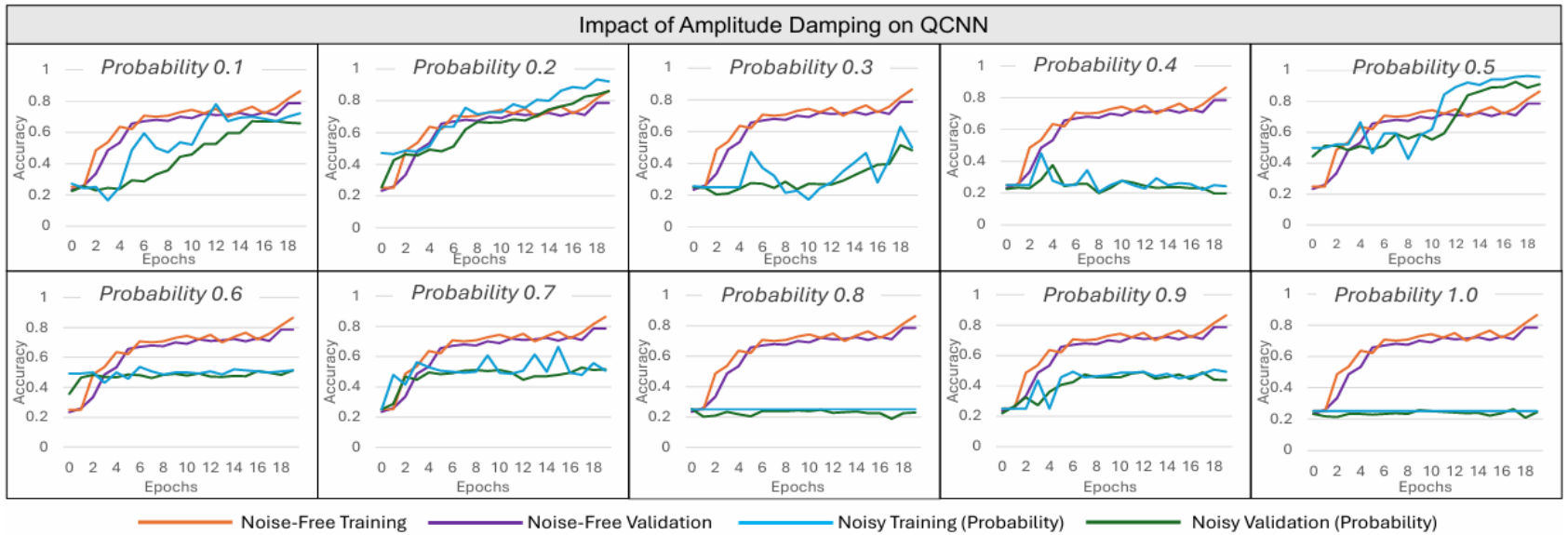
Table 1. Optimal Configurations for Best Performing Models

Model	VQC Design	# of Layers
QCNN	Strongly Entangling	3
QuanNN	Basic Entangling	3

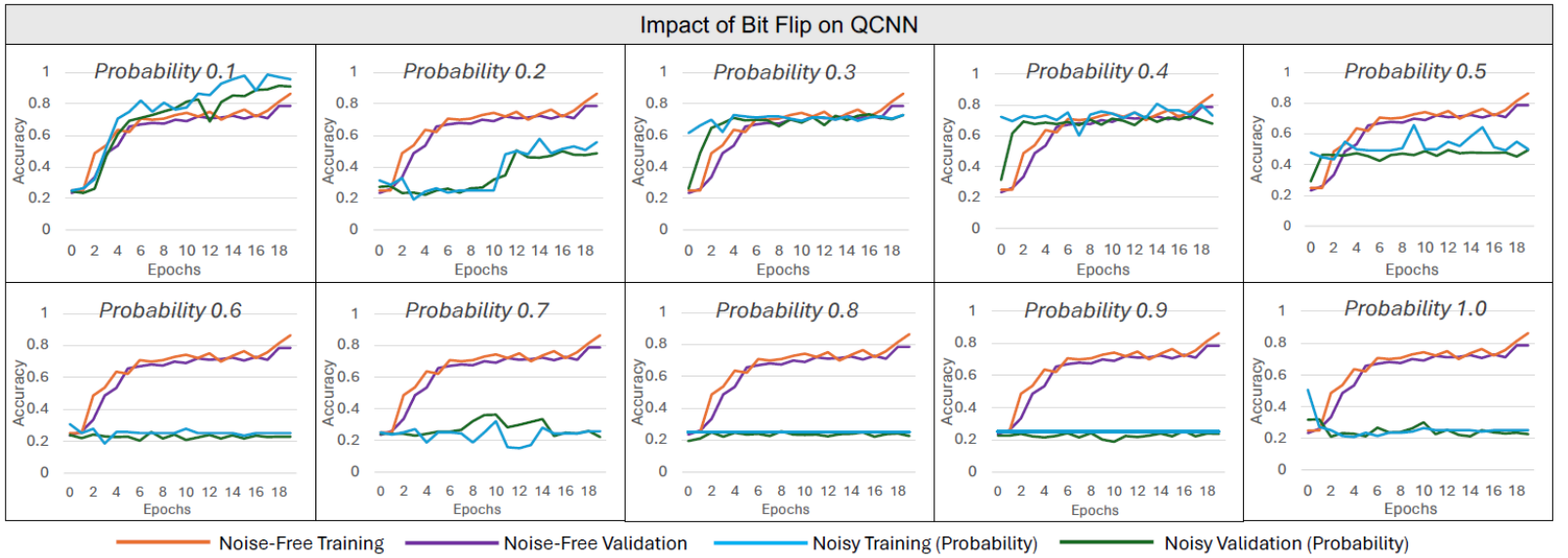
Paper references visible on slide: Figure 4; §4.1–4.2; Figure 12–14; Conclusion.

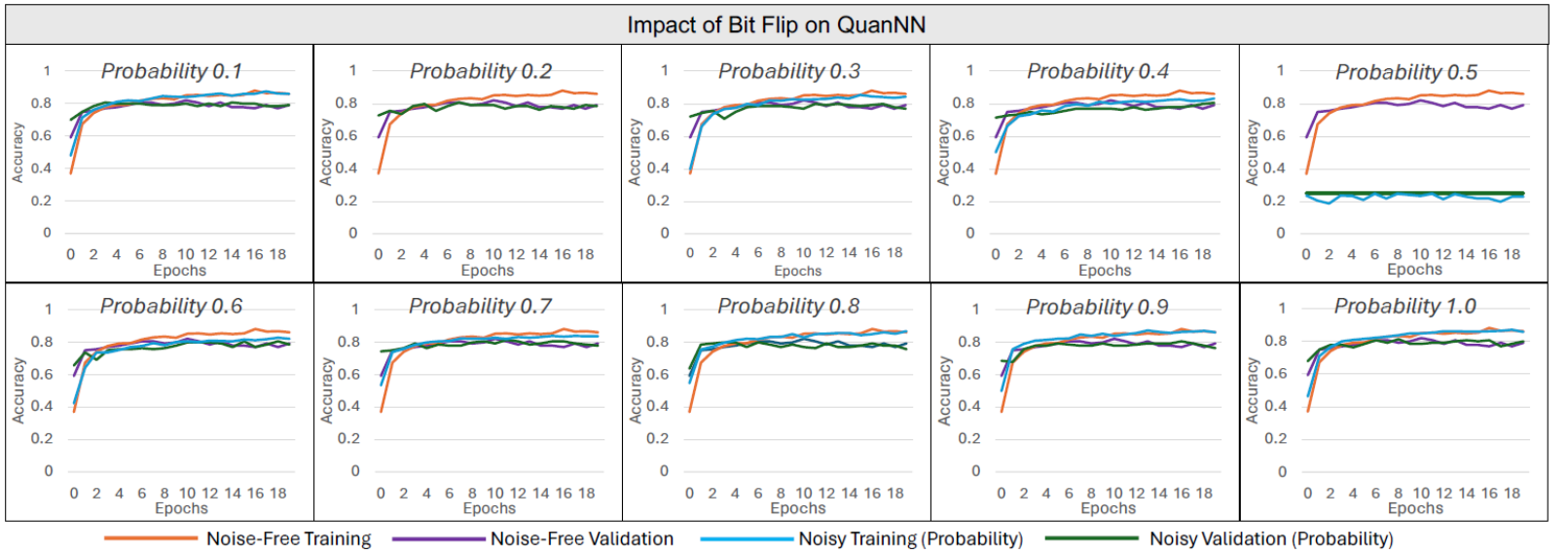
Hybrid Quantum Neural Network – Main Findings with Noise

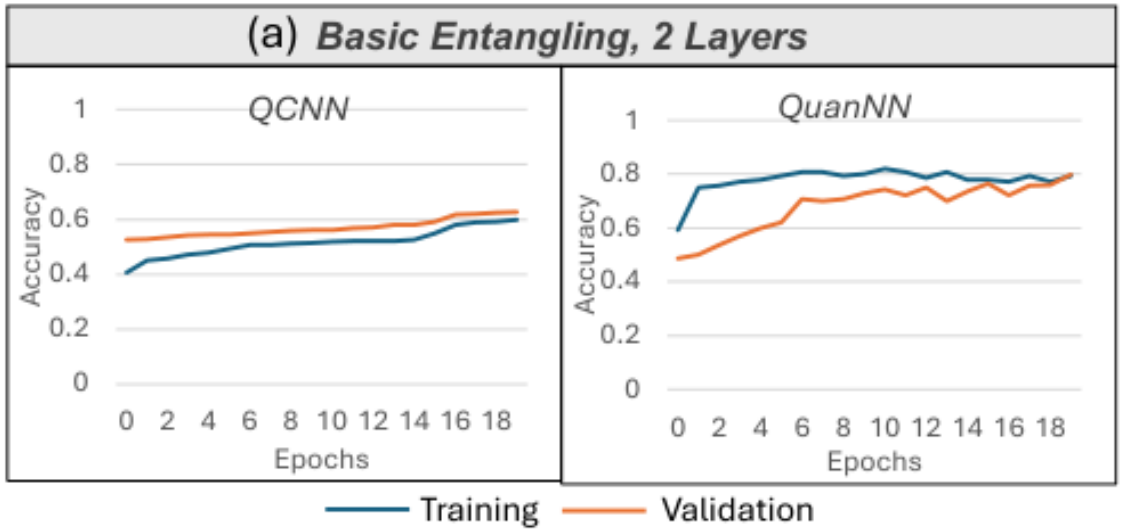


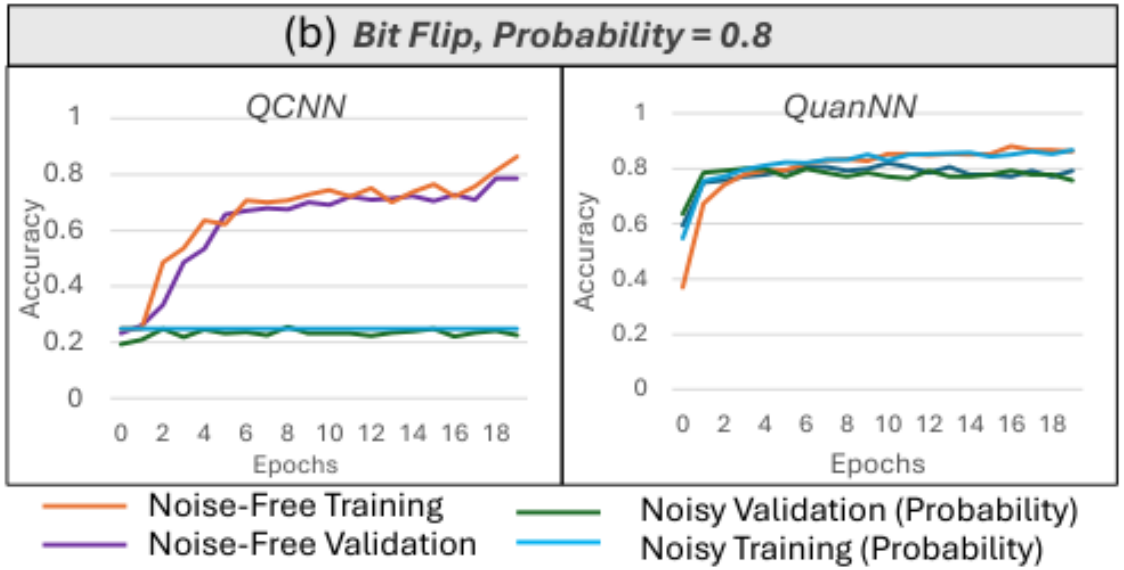


Hybrid Quantum Neural Network – Main Findings with Noise









	QCNN	QuanNN
Architecture	Quantum convolution + quantum pooling, with classical downsizing before the quantum layers.	Quanvolutional filters over local image patches; no pooling stage in the QCNN sense.
Best config	Strongly entangling, 3 layers.	Basic entangling, 3 layers.

Takeaway: for the specific task, dataset, and noise study in this paper, QuanNN is the safer architecture choice for NISQ deployment.

Discussion

- Quantum computing is not yet mature
- But it is already scientifically valuable
- Future impact lies at the intersection of:
  - Physics
  - Computation
  - Intelligence

Riding the Quantum Wave: From Earth To Intelligence

How quantum computing bridges physics-based modeling and AI?

Thank you

For your attendance and participation

SULAIMAN UREIGA

AI Eng & Researcher

AMNAH SAMARIN

HPC Developer & Computational Geophysicist