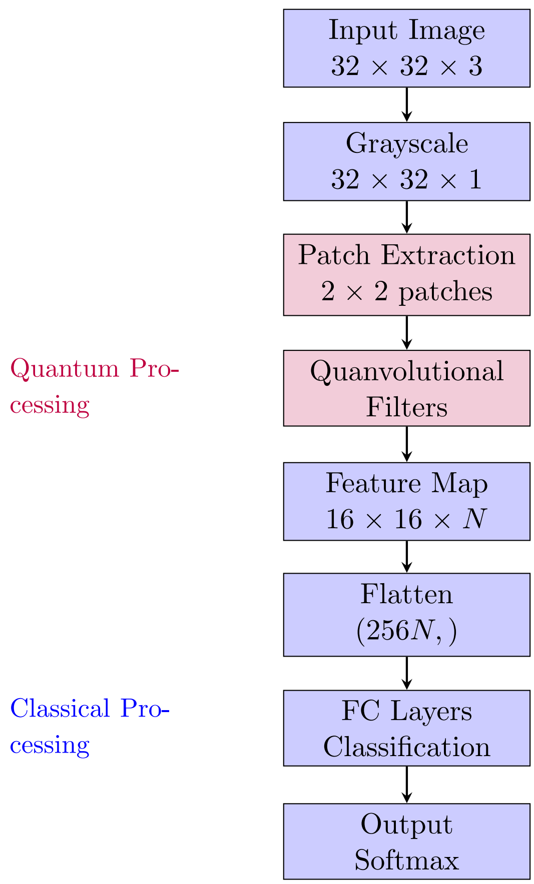
Quanvolutional Neural Network (QuanNN): Complete Architecture Guide

Hybrid Quantum-Classical Deep Learning

1 Architecture Overview



2 Step-by-Step Process

2.1 Step 1: Patch Extraction

Given input image $\mathbf{I} \in \mathbb{R}^{H \times W \times C}$ where $H = W = 32$, $C = 3$:

$$\mathbf{I}_{gray} = 0.299R + 0.587G + 0.114B \to \mathbb{R}^{32 \times 32 \times 1} \tag{1}$$

Extract patches with size $p \times p = 2 \times 2$ and stride $s = 2$:

$$N_{patches} = \left(\frac{H - p}{s} + 1\right) \times \left(\frac{W - p}{s} + 1\right) = 16 \times 16 = 256 \tag{2}$$

Each patch: $\mathbf{u}_x = [u_1, u_2, u_3, u_4]$ where $u_i \in [0, 255]$

2.2 Step 2: Quantum Encoding

For each patch, normalize pixel values:

$$\theta_i = \frac{u_i}{255} \times 2\pi, \quad i \in \{1, 2, 3, 4\} \tag{3}$$

Initialize quantum state and apply encoding:

$$|\psi_{enc}\rangle = \bigotimes_{i=1}^{4} RX(\theta_i)|0\rangle = RX(\theta_1) \otimes RX(\theta_2) \otimes RX(\theta_3) \otimes RX(\theta_4)|0000\rangle \tag{4}$$

where $RX(\theta) = e^{-i\theta X/2} = \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$

2.3 Step 3: Variational Quantum Circuit (VQC)

Apply parametric quantum circuit with L layers:

$$|\psi_{out}\rangle = U_L(\boldsymbol{\phi}_L) \cdots U_2(\boldsymbol{\phi}_2) U_1(\boldsymbol{\phi}_1)|\psi_{enc}\rangle \tag{5}$$

Each layer U_ℓ consists of:

- **Entanglement:** $CNOT_{i,i+1}$ for $i \in \{0, 1, 2\}$
- **Rotation:** $RY(\phi_i^\ell)$ and $RZ(\psi_i^\ell)$ on each qubit

2.4 Step 4: Measurement and Aggregation

Measure each qubit in Pauli-Z basis:

$$m_i = \langle\psi_{out}|Z_i|\psi_{out}\rangle, \quad i \in \{0, 1, 2, 3\} \tag{6}$$

Aggregate measurements to single feature value:

$$f_x = \mathcal{A}([m_0, m_1, m_2, m_3]) = \frac{1}{4}\sum_{i=0}^{3} m_i \tag{7}$$

Complete Quanvolutional Filter:

$$\boxed{f_x = \mathcal{Q}(\mathbf{u}_x, e, q, d) \in \mathbb{R}} \tag{8}$$

2.5 Step 5: Feature Map Construction

Apply N different quanvolutional filters (each with different parameters):

$$\mathcal{F} = \begin{bmatrix} f_1^{(1)} & f_2^{(1)} & \cdots & f_{16}^{(1)} \\ f_{17}^{(1)} & f_{18}^{(1)} & \cdots & f_{32}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ f_{241}^{(1)} & f_{242}^{(1)} & \cdots & f_{256}^{(1)} \end{bmatrix}_{16\times16} \quad \text{for filter 1} \tag{9}$$

Stack all filters: $\mathcal{F}_{total} \in \mathbb{R}^{16\times16\times N}$

2.6 Step 6: Classical Post-Processing

Flatten feature map:

$$\mathbf{v} = \text{Flatten}(\mathcal{F}_{total}) \in \mathbb{R}^{256N} \tag{10}$$

Fully connected layers:

$$\mathbf{h}_1 = \text{ReLU}(\mathbf{W}_1\mathbf{v} + \mathbf{b}_1) \in \mathbb{R}^{n_1} \tag{11}$$

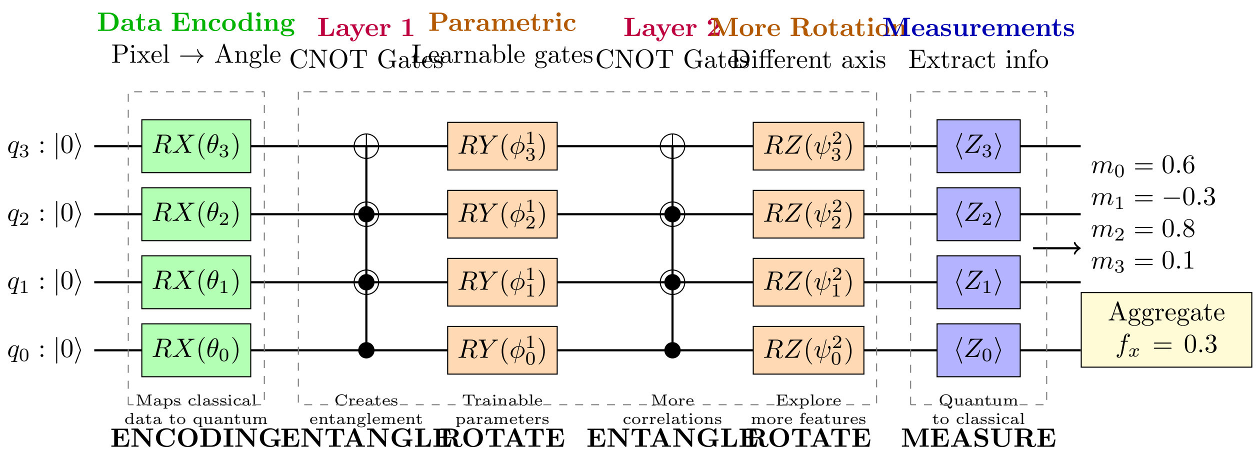
$$\mathbf{h}_2 = \text{ReLU}(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2) \in \mathbb{R}^{n_2} \tag{12}$$

$$\mathbf{z} = \mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3 \in \mathbb{R}^{n_{classes}} \tag{13}$$

Output probabilities:

$$P(y = c|\mathbf{I}) = \text{Softmax}(\mathbf{z})_c = \frac{e^{z_c}}{\sum_{j=1}^{n_{classes}} e^{z_j}} \tag{14}$$

3 Complete Quantum Circuit Diagram with Explanations



3.1 Why These Specific Components?

3.2 Why Layer 1, Layer 2, ...? (Circuit Depth)

Trade-off between expressiveness and noise:

- **More layers (deeper circuit):**
  - + More expressive (can learn complex functions)
  - + More trainable parameters
  - – More quantum gates → more noise accumulation (NISQ problem!)
  - – Longer circuit execution time
- **Fewer layers (shallow circuit):**
  - + Less affected by quantum noise
  - + Faster execution
  - – Limited expressiveness
  - – May not capture complex patterns

Typical choice for NISQ: $L = 2$ to 6 layers (balance between power and practicality)

Component	Purpose	Why This Choice?
RX Encoding	Map classical pixel values to quantum states	RX creates superposition. Angle θ encodes pixel intensity. Alternative: RY, RZ, or amplitude encoding
CNOT Gates	Create entanglement between qubits	Captures spatial correlations between adjacent pixels. Without this, qubits work independently (no quantum advantage!)
RY Rotations	Learnable transformations	Parameters ϕ are trained via optimization. RY chosen for expressibility. Could also use RX or RZ
RZ Rotations	Additional learnable features	Different rotation axis provides more expressiveness. Allows circuit to learn complex patterns
Pauli-Z Measure	Extract expectation values	$\langle Z\rangle \in [-1, +1]$ provides continuous output. Alternative: measure probabilities $\|0\rangle$ vs $\|1\rangle$
Aggregation	Reduce 4 values to 1	Mimics CNN filter: many inputs $\rightarrow$ one feature. Uses mean, sum, or first value

3.3 Alternative Circuit Designs Not Shown

1. **Why not use Hadamard gates?**
   - H gates create uniform superposition $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
   - Not suitable for data encoding (doesn't embed pixel values)
   - Could be used for feature mixing, but RX/RY/RZ are more flexible
2. **Why not amplitude encoding?**
   - Amplitude encoding: $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} x_i |i\rangle$
   - Would encode all 4 pixels in amplitudes
   - Problem: Requires normalization and complex state preparation
   - Angle encoding (RX) is simpler and more robust
3. **Why not use more complex entangling patterns?**
   - Could use: all-to-all CNOT, CZ gates, or custom entanglers
   - Trade-off: More entanglement vs. more gate errors
   - Linear chain (shown) is most hardware-friendly
4. **Why measure in Z-basis only?**
   - Could measure in X-basis: $\langle X\rangle$ or Y-basis: $\langle Y\rangle$

- Z-basis is hardware-native (fastest, most accurate)
- Some designs measure multiple bases for richer features

3.4 Mathematical Detail: What Each Gate Does

$$RX(\theta) = e^{-i\theta X/2} = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \quad \text{(Rotation around X-axis)}$$

$$RY(\phi) = e^{-i\phi Y/2} = \begin{pmatrix} \cos\frac{\phi}{2} & -\sin\frac{\phi}{2} \\ \sin\frac{\phi}{2} & \cos\frac{\phi}{2} \end{pmatrix} \quad \text{(Rotation around Y-axis)}$$

$$RZ(\psi) = e^{-i\psi Z/2} = \begin{pmatrix} e^{-i\psi/2} & 0 \\ 0 & e^{i\psi/2} \end{pmatrix} \quad \text{(Phase rotation)}$$

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad \text{(Controlled-NOT: flips target if control is } |1\rangle)$$

3.5 Why This Exact Sequence?

Design Philosophy:

1. **Encode data** (RX) → Classical info enters quantum realm
2. **Create correlations** (CNOT) → Qubits "talk" to neighbors
3. **Transform** (RY) → Learn features via trainable parameters
4. **Repeat** (CNOT + RZ) → Deepen the representation
5. **Extract** (Measure) → Return to classical world

This mimics classical neural networks: *Input* → *Transform* → *Transform* → *Output*

4 Algorithm: QuanNN Forward Pass

Algorithm 1 QuanNN Forward Pass

Require: Image $\mathbf{I} \in \mathbb{R}^{32\times32\times3}$, N quantum filters, $n_{qubits} = 4$
Ensure: Prediction probabilities $\mathbf{P} \in \mathbb{R}^{n_{classes}}$

$\mathbf{I}_{gray} \leftarrow \text{RGB2Gray}(\mathbf{I})$ ▷ $32 \times 32 \times 1$
$\mathcal{P} \leftarrow \text{ExtractPatches}(\mathbf{I}_{gray}, size = 2, stride = 2)$ ▷ 256 patches
$\mathcal{F} \leftarrow \text{zeros}(16, 16, N)$ ▷ Initialize feature map
for $k = 1$ to N **do** ▷ For each filter
    for $i = 1$ to 256 **do** ▷ For each patch
        $\mathbf{u}_i \leftarrow \mathcal{P}[i]$ ▷ Get patch $[u_1, u_2, u_3, u_4]$
        $\boldsymbol{\theta} \leftarrow [\theta_1, \theta_2, \theta_3, \theta_4] = \frac{\mathbf{u}_i}{255} \times 2\pi$
        $|\psi\rangle \leftarrow |0000\rangle$
        for $j = 1$ to 4 **do** ▷ Encoding
            $|\psi\rangle \leftarrow RX_j(\theta_j)|\psi\rangle$
        end for
        for $\ell = 1$ to L **do** ▷ VQC layers
            $|\psi\rangle \leftarrow U_\ell(\boldsymbol{\phi}_\ell^{(k)})|\psi\rangle$ ▷ Parametric gates
        end for
        $\mathbf{m} \leftarrow [\langle Z_0\rangle, \langle Z_1\rangle, \langle Z_2\rangle, \langle Z_3\rangle]$
        $\mathcal{F}[i, k] \leftarrow \text{mean}(\mathbf{m})$ ▷ Aggregate
    end for
end for
$\mathbf{v} \leftarrow \text{Flatten}(\mathcal{F})$ ▷ $256N$ dimensions
$\mathbf{h}_1 \leftarrow \text{ReLU}(\mathbf{W}_1\mathbf{v} + \mathbf{b}_1)$
$\mathbf{h}_2 \leftarrow \text{ReLU}(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2)$
$\mathbf{z} \leftarrow \mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3$
$\mathbf{P} \leftarrow \text{Softmax}(\mathbf{z})$
return P

5 Key Specifications

Component	Dimension	Type
Input Image	$32 \times 32 \times 3$	Classical
Grayscale Image	$32 \times 32 \times 1$	Classical
Patch Size	2×2	-
Number of Patches	256	-
Qubits per Filter	4	Quantum
Quantum Filters	N (e.g., 8)	Quantum
Feature Map	$16 \times 16 \times N$	Classical
Flattened Vector	$256N$	Classical
FC Layer 1	$256N \to 128$	Classical
FC Layer 2	$128 \to 64$	Classical
Output Layer	$64 \to n_{classes}$	Classical

6 Computational Complexity

6.1 Quantum Part

- **Parameters per filter:** $O(n_{qubits} \times L)$ where L is VQC depth
- **Circuit executions:** $256 \times N$ (sequential on single QPU)
- **Total quantum parameters:** $\approx 50N$ to $200N$

6.2 Classical Part

- **FC parameters:** $(256N \times 128) + (128 \times 64) + (64 \times n_{classes})$
- For $N = 8$, $n_{classes} = 10$: $\approx 270,000$ parameters
- **Dominant cost:** Classical layers (most learnable parameters)

7 Training Process

Loss Function:

$$\mathcal{L} = -\frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} \sum_{c=1}^{n_{classes}} y_i^{(c)} \log(P_i^{(c)}) \tag{15}$$

Optimization: Hybrid quantum-classical optimization

- Quantum parameters $\boldsymbol{\phi}$: Updated via parameter-shift rule
- Classical parameters $\mathbf{W}, \mathbf{b}$: Updated via backpropagation

8 Quick Reference: FAQ

Q1: Sequential or parallel patch processing?
A: Sequential on single QPU (256 executions). Parallel only in simulation.

Q2: What is the quanvolutional filter output?
A: Single scalar $f_x \in \mathbb{R}$ per patch, aggregated from 4 qubit measurements.

Q3: Why use FC layers after quantum processing?
A: To aggregate local quantum features into global classification decision.

Q4: Can I use more qubits?
A: Yes! Larger patches (e.g., $4 \times 4 = 16$ qubits), but needs more quantum resources.

Q5: What makes it "hybrid"?
A: Quantum feature extraction + Classical aggregation and classification.