Quantum Autoencoders

Ruhi Yusuf

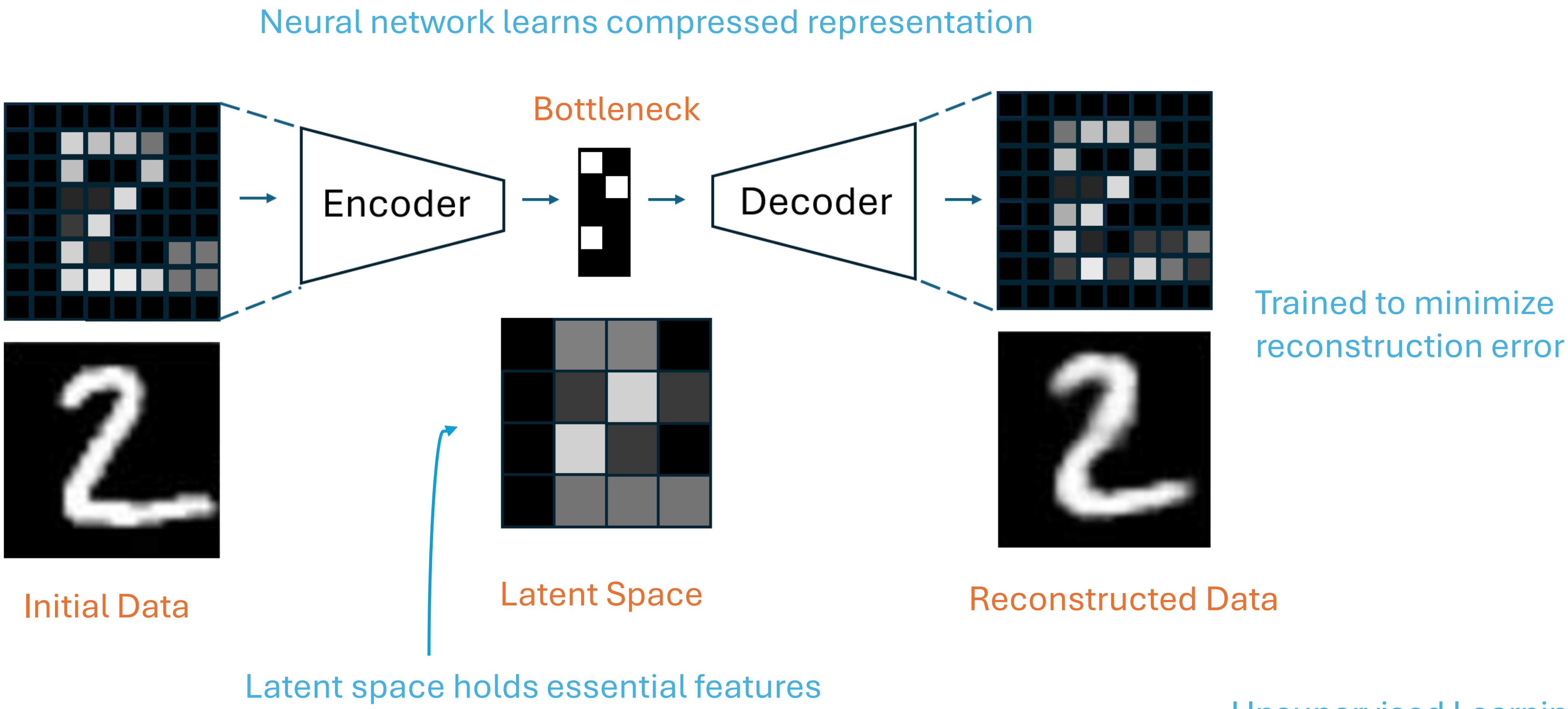
ISIT JUNE 22-27 ANN ARBOR MICHIGAN

Hackathon
where quantum computing meets information theory

Outline

- Classical Decoders
- Why Quantum Autoencoders?
- Converting Classical Data into Quantum States
- What is QAE?
- Variational Circuits
- Implementation and Circuit Architecture
- Building an example
- Q&A

Classical Encoding



Unsupervised Learning

Why use Quantum Autoencoders on Classical Data?

Classical Input
(e.g. image)

↓

Classical-to-Quantum Encoding
(e.g. angle or amplitude encoding)

↓

Quantum Autoencoder (QAE)
- Entanglement-aware
- Parametrized circuits

↓

Compressed Quantum State
(Latent Qubits)

↓ Reconstructed Output (Classical)

Why Quantum?

✓Models richer feature correlations

✓Enables hybrid classical-quantum ML

✓Smaller latent space in qubits vs. neurons

✓Leverages entanglement + interference

✓Built for future quantum pipelines

Converting Classical Data into Quantum States

- Basis Encoding
  - Encode each *n*-bit feature into *n* qubits
    - $x = (x_{n-1}, \cdots, x_1, x_0) \rightarrow |x\rangle = |x_{n-1} \cdots x_1 x_0\rangle$
- Amplitude Encoding
  - Encode into quantum state amplitudes
    - $x = \begin{pmatrix} x_0 \\ \vdots \\ x_{n-1} \end{pmatrix} \rightarrow |x\rangle = \sum_{j=0}^{n-1} x_j |j\rangle$
- Amplitude Encoding
  - Encode values into qubit rotation angles
    - $|x\rangle = \bigotimes_j \cos(x_i)|0\rangle + \sin(x_i)|1\rangle$
- Arbitrary Encoding (Feature Map)
  - Encode N features on N rotation gates in constant-depth circuit with n qubits
    - $x = \begin{pmatrix} x_0 \\ \vdots \\ x_{n-1} \end{pmatrix} \rightarrow |\psi_x\rangle = U_{\Phi(x)}|0\rangle$

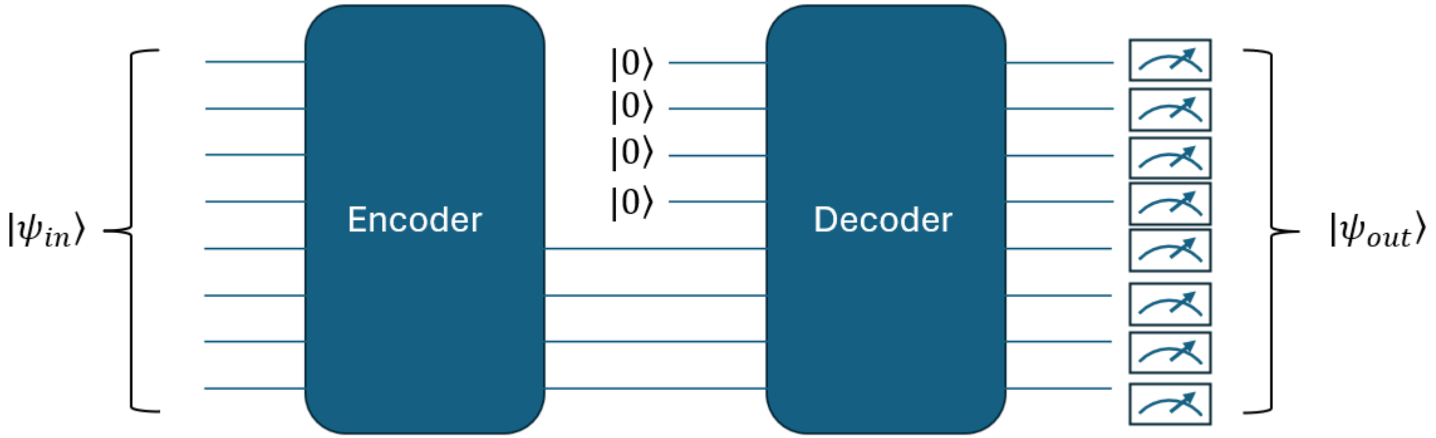
ADD REFERENCE

What is a Quantum Autoencoder (QAE)?

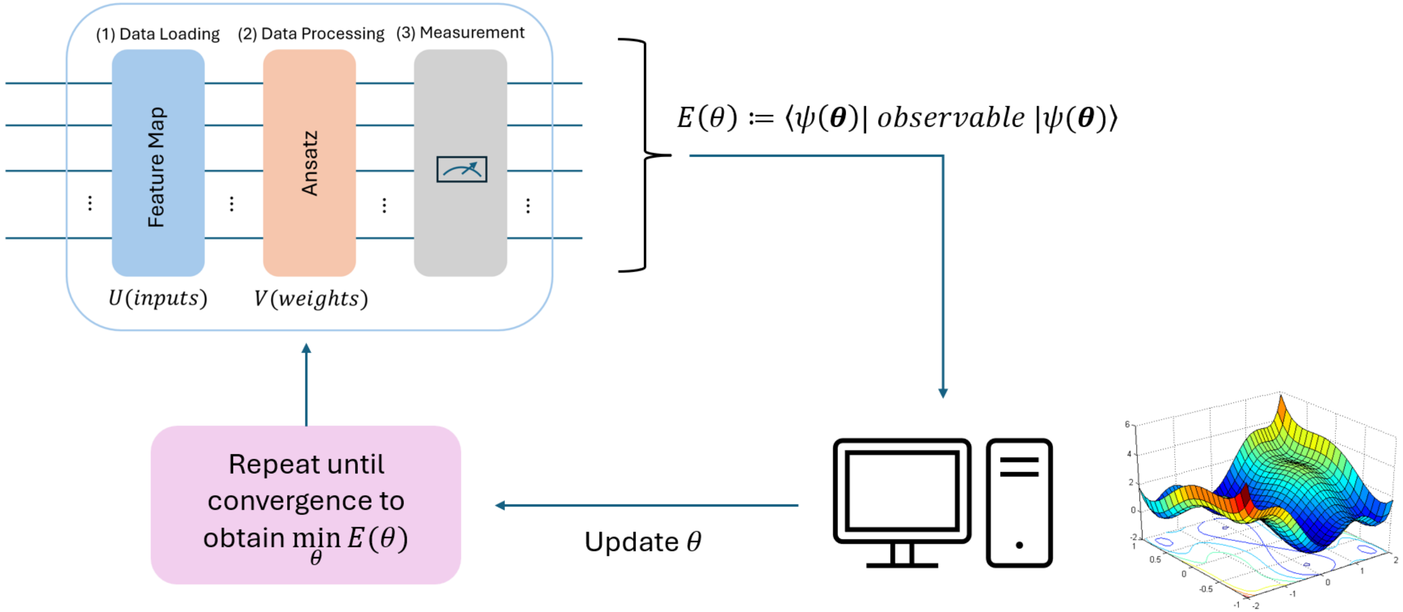
A Quantum Autoencoder is a variational quantum circuit that learns to compress quantum states by reducing the number of qubits needed to represent them, while preserving essential information.

Core Architecture:

- Input: quantum state $|\psi\rangle$
- Encoder: compresses input into fewer latent qubits
- Trash: remaining qubits ideally reset to $|0\rangle$ and discarded
- Decoder: reconstructs original state from latent qubits
- Training objective: maximize fidelity between input and reconstructed state



Variational Circuits

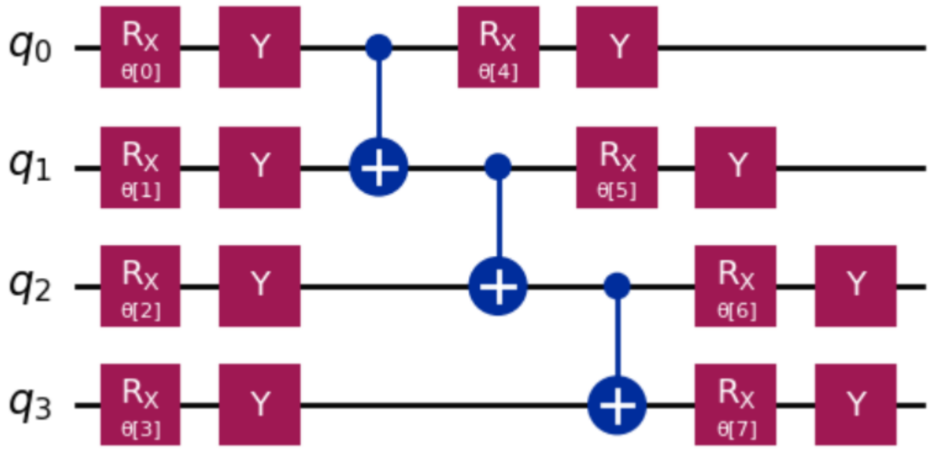


- Parameterized quantum circuit (Ansatz)
- Objective: Minimize or maximize this expectation
- Optimized via classical loop (gradient-based or gradient-free)

e.g. Variational Quantum Eigensolver (VQE), Quadratic Unconstrained Binary Optimization (QUBO)

Variational Circuits

- Example of variational circuits trained in classification

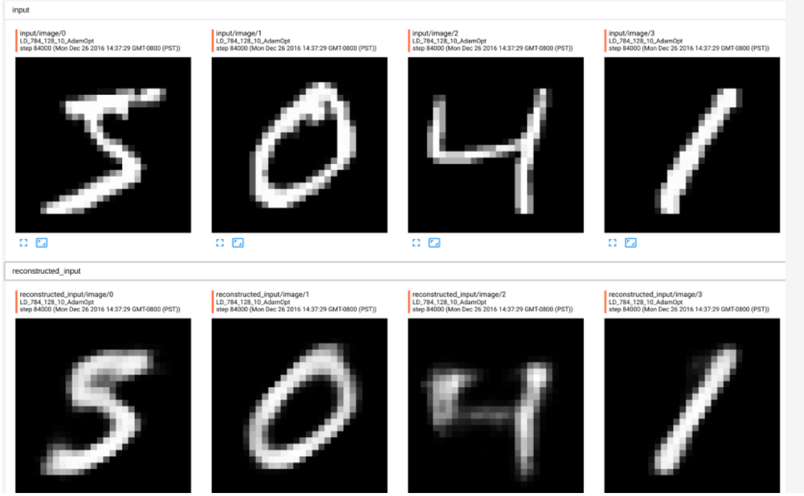


Fixed vs. Variational Quantum Circuits

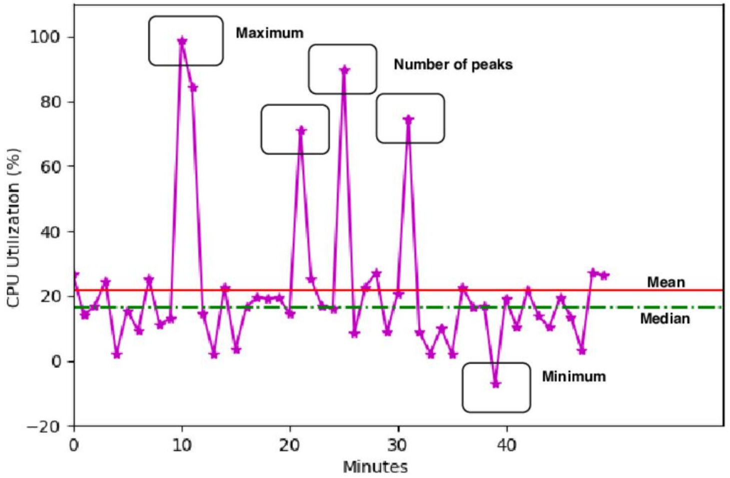
Feature	Fixed Quantum Algorithm (e.g., Grover)	Variational Quantum Circuits (e.g., QAE)
Circuit Design	Hardcoded unitary blocks	Learnable ansatz (e.g., RealAmplitudes)
Optimization	No training; exact logic	Trained via classical feedback loop
Adaptability	Rigid to input noise and imperfections	Learns patterns from data
Use Case	One-shot query/search	Compression, classification, denoising
NISQ Compatibility	Poor (deep + fragile)	Better (shallow, tunable)

Why use a QAE?

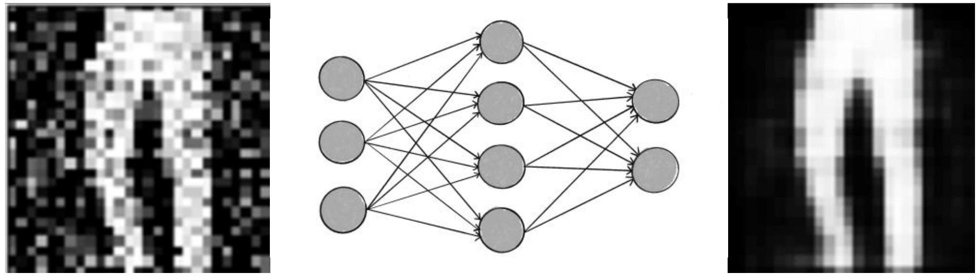
Efficient use of qubits (compression)



Feature Extraction

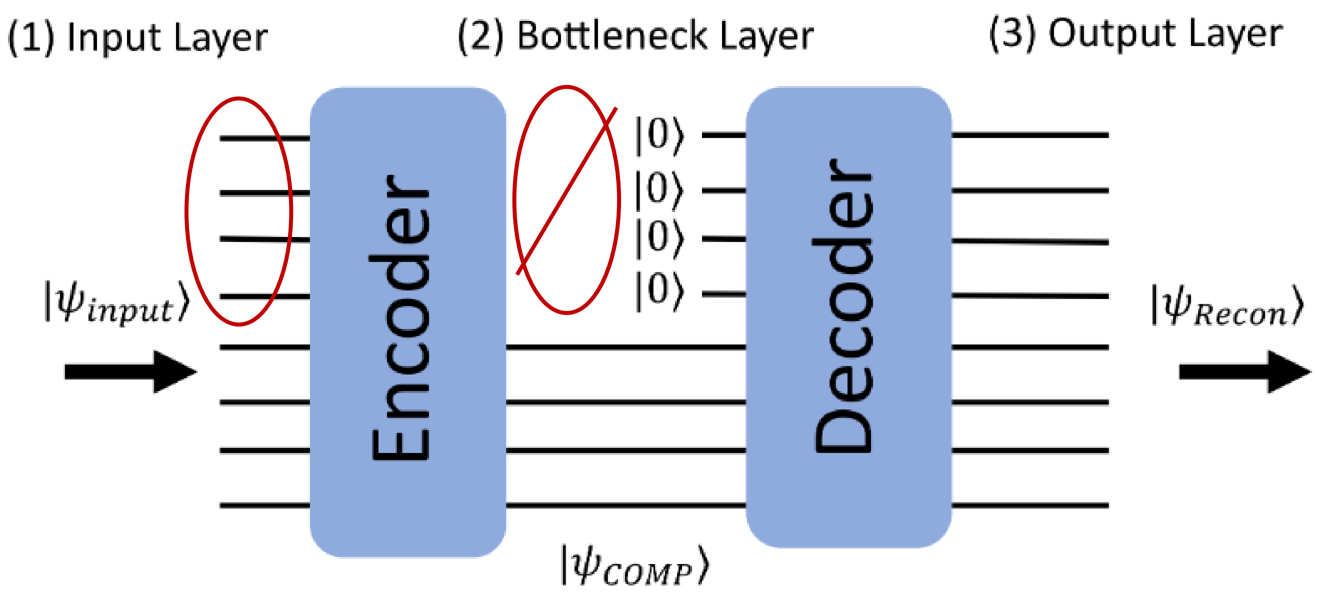


Efficient use of qubits (denoising)



Typical QAE Design

- QAEs compress quantum input into fewer qubits (latent space)
- Trash space is reset or discarded
- Decoder reconstructs full input from latent space
- Trained to minimize difference between input and reconstructed output



QAE Circuit Architecture

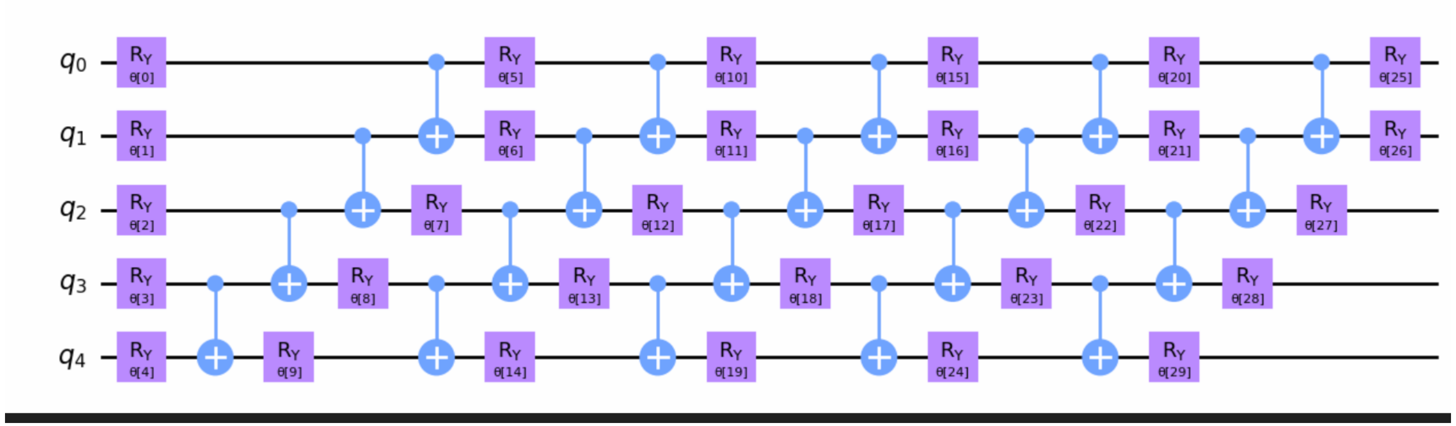
Structure:

- Both encoder and decoder use **parameterized variational circuits**
- Ansatz: **RealAmplitudes** with entangling layers (CNOTs)
- Same architecture for both encoder and decoder (symmetry)

Key Features:

- Works with angle encoding of input state
- Number of qubits = latent + trash
- Parameters θ are trained to maximize output fidelity

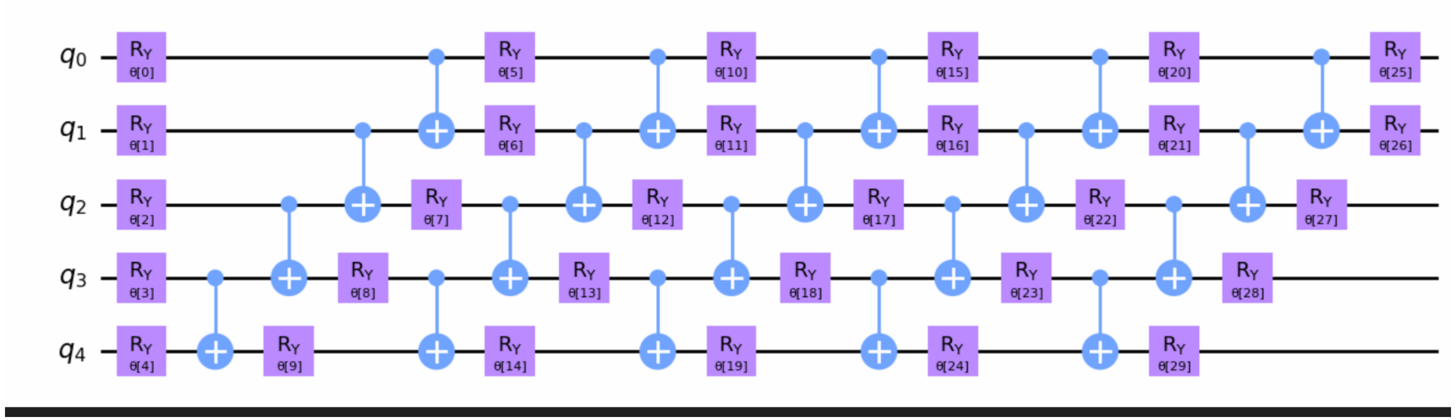
QAE Circuit Architecture



Structure:

- **Variational ansatz** built with parameterized R_Y gates and entangling CNOT layers
- Used for both encoder and decoder circuits (shared structure = symmetric QAE)
- Trains parameters θ to compress input TFIM states
- Entanglement structure enables modeling of quantum correlations

QAE Circuit Architecture



Key Features:

- Based on Qiskit's Real Amplitude template
- Number of qubits = latent + trash
- Trained to **maximize fidelity** between input and reconstructed state

Loss Function – Fidelity as a Measure of Compression Quality

Objective:

Train the QAE to **maximize fidelity** between the input state $|\psi_{input}\rangle$ and the reconstructed output $|\psi_{out}\rangle$

Loss Function:

$$\mathcal{L} = 1 - \left|\langle\psi_{input}|\psi_{out}\rangle\right|^2$$

Why fidelity?

- Quantum states can't be directly compared component-wise
- Fidelity gives a single number from 0 to 1
- High fidelity = accurate reconstruction = good compression

The SWAP Test: Comparing Quantum States

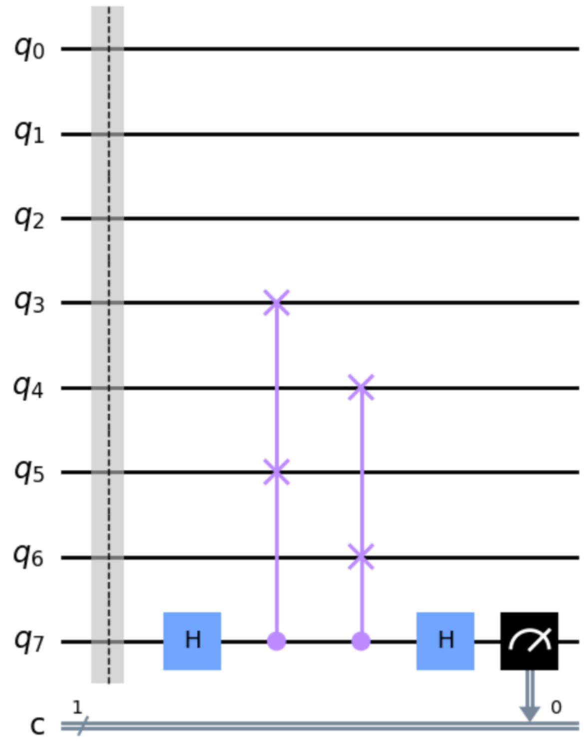
Run the swap test circuit M times
Count how often the ancilla qubit is measured in state $|1\rangle$, denoted L

Then compute:

$$S = 1 - \frac{2L}{M}$$

Where:

- S approximates the **fidelity** between two states
- M: total number of circuit runs
- L: number of runs where ancilla $=|1\rangle$



How the QAE Learns to Compress: Entropy Flow

Key Idea:

- During training, the QAE learns to **localize information** into latent qubits
- The **trash qubits** are pushed into a fixed, low-entropy state (ideally)
- This compression aligns with a **reduction in entropy flow** through the trash subsystem

What we observe:

- Entanglement and correlations migrate toward the **latent space**
- The trash space becomes disentangled, indicating successful compression

Wrap Up

- Quantum Autoencoders can compress structured quantum states
- Variational circuits (e.g. RealAmplitudes) are expressive enough for compression
- Swap test provides an efficient fidelity-based loss for training
- QAE achieved competitive reconstruction performance with fewer parameters than classical DL

Demo Time!

Questions?