Quantum Autoencoders for Dataset Compression

Seminar report submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

By

P Soumya Sundar Subudhi (2321109113)
6th semester, Department of CSE
Section: A

PARALA MAHARAJA ENGINEERING COLLEGE, BERHAMPUR

BIJU PATNAIK UNIVERSITY OF TECHNOLOGY, ODISHA INDIA

(2025)

PARALA MAHARAJA ENGINEERING COLLEGE, BERHAMPUR, INDIA-761003

DECLARATION

I hereby declare that the 6th Semester Seminar work entitled "Quantum Autoencoders for Dataset Compression: A Hybrid Approach Using Qiskit" in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering submitted to Department of Computer Science & Engineering, Parala Maharaja Engineering College, Berhampur is an authentic record of my work carried out. The matter embodied in this Seminar report has not been submitted to any University or Institution for any degree or diploma.

Name: P Soumya Sundar Subudhi

Regd. No: 2321109113

PARALA MAHARAJA ENGINEERING COLLEGE, BERHAMPUR, INDIA-761003

CERTIFICATE OF APPROVAL

This is to certify that the 6th Semester Seminar report entitled “Quantum Autoencoders for Dataset Compression” submitted by P Soumya Sundar Subudhi (2321109113) in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering submitted to Department of Computer Science & Engineering, Parala Maharaja Engineering College, Berhampur is a bonafide record of his/her original work. The matter embodied in this seminar report has not been submitted to any other University or Institution for any degree or diploma.

Date:

Place:

(Signature of Faculty Coordinator(s))

(Signature of HOD, CSE)

ACKNOWLEDGEMENT

I would like to express my sincere thanks and gratitude to Dr. Debasis Mohapatra for his invaluable guidance and support and throughout the course of this study. His expertise and encouragement have been instrumental in shaping this work. I also extend my heartfelt thanks to the faculties-in-charge of the Seminar of the 6th Semester, Dr. Ranumayee Sing , Mr. Amar Ranjan Dash, and Mr. K Kishan Kumar Patro, for their continuous support and mentorship. Their contributions were crucial in organizing and facilitating the seminar, enriching the overall experience. Additionally, I would like to acknowledge the support of my colleagues and peers who provided insightful feedback and assistance throughout this research journey .

ABSTRACT

Data storage and transmission efficiently are essential in the big data age. Quantum computing presents new ways of compressing data, particularly through Quantum Autoencoders (QAEs). This research examines the application of QAEs to compress datasets with Qiskit. A hybrid quantum-classical framework is constructed using Parameterized Quantum Circuits (PQCs) and the Real Amplitudes ansatz for optimal data representation within a low-dimensional quantum space. Training employs the COBYLA optimizer for effective fine-tuning of parameters. After training, datasets are mapped to latent quantum states and then restored through an inverse ansatz circuit. Performance is measured by evaluating the fidelity of the original and restored datasets. This work highlights the possibilities of quantum machine learning (QML) for data processing with both its pros and cons regarding compression applications. The results show that QAEs are well-suited to compress dataset sizes without losing necessary information, pointing towards further innovations in quantum-facilitated data compression.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

In the data-driven age of today, the exponential proliferation of information demands new ways to store and transmit data. Classical compression methods, though efficient, are reaching their theoretical limits, leading to the search for other methods. Quantum computing, based on its principles of superposition and entanglement, provides a potential solution to transform data compression. This project explores the application of Quantum Autoencoders (QAEs) to compress datasets using IBM's open-source quantum computing platform, Qiskit.

Classical autoencoders are neural network structures programmed to learn compressed, low-dimensional data representations so as to provide compression and denoising capabilities. Similarly, QAEs seek to compress quantum information through encoding it within a lower-dimensional quantum space and hence reducing the number of required qubits necessary for effective representation of data. This compression is realized by a parameterized quantum circuit that learns to encode input quantum states onto a latent space, from which the original data can be efficiently reconstructed with high fidelity.

The primary goal of this project is to create a hybrid quantum-classical model with Parameterized Quantum Circuits (PQC) and the Real Amplitudes ansatz in order to optimize data representation within a low-dimensional quantum space. The training procedure utilizes traditional optimization algorithms, including COBYLA, to optimize the quantum circuit parameters for effective compression. After training, datasets are mapped into latent quantum states and then reconstructed via an inverse ansatz circuit. Performance is measured by comparing the fidelity of the original and reconstructed datasets, giving a quantitative assessment of the effectiveness of the compression.

Applying QAEs to compressing data sets not only demonstrates the capability of quantum machine learning methods but also solves real-world data storage and transmission problems. Utilizing the special principles of quantum mechanics, this project is expected to reach effective data compression of big data sets while maintaining key information. The results from this project extend the overall application of quantum-augmented processing of data and provide foundations for further investigation of more complicated quantum data compression operations.

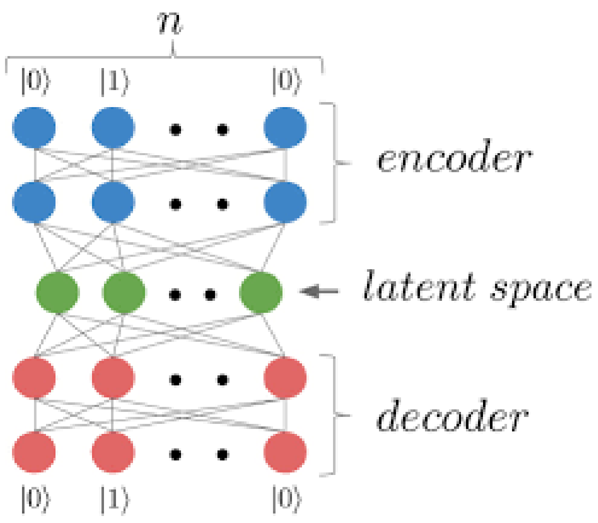


Figure 1: Quantum Autoencoder

OBJECTIVE OF THE PROJECT

Classic autoencoders refer to neural network structures that can learn compact, low-dimensional data representations, allowing for compression as well as filtering out noise. Similarly, QAEs focus on compressing quantum data in the form of encoding it to a lower-dimensional quantum space with the aim of reducing the required number of qubits for valid data representation. This compression is performed by a parameterized quantum circuit that learns to encode input quantum states into a latent space from which the data can be recovered with high fidelity.

The primary goal of this project is to create a hybrid quantum-classical model based on Parameterized Quantum Circuits (PQC) and the Real Amplitudes ansatz to maximize data representation in a lower-dimensional quantum space. Classical optimization algorithms, e.g., COBYLA, are used in the training process to optimize the parameters of the quantum circuits for effective compression. After training, datasets are mapped into latent quantum states and then reconstructed through an inverse ansatz circuit. Performance assessment is done through comparing fidelity between the original and reconstructed datasets with a quantitative indication of the effectiveness of the compression.

Applying QAEs to compress datasets not only reveals the power of quantum machine learning methods but also tackles real-world problems in data transmission and storage. Through leveraging the special features of quantum mechanics, this project hopes to attain effective compression of extensive datasets without losing valuable information. The results of this study further the general body of knowledge for quantum-enabled data processing and pave the way for subsequent investigations into more intricate quantum data compression problems.

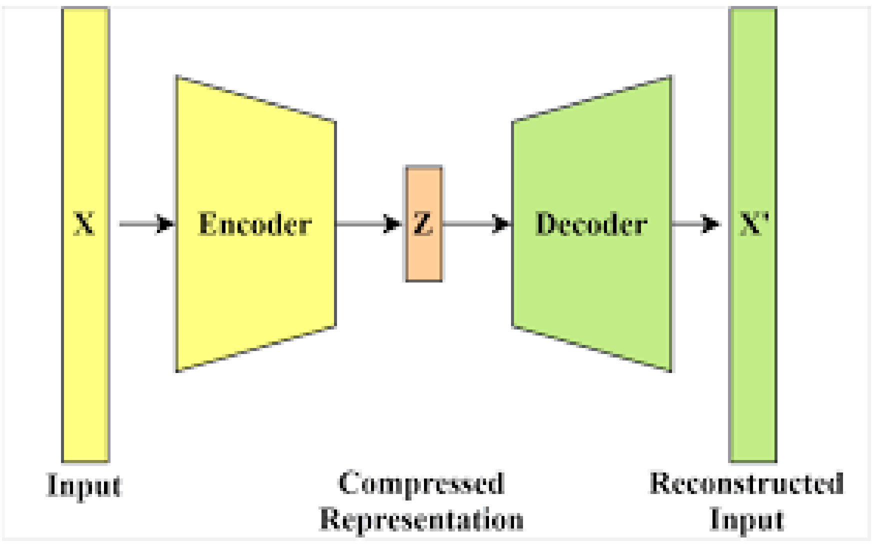


Figure 2: Representation of Quantum Autoencoder

PROBLEM STATEMENT

The accelerated growth of data creation in the current digital era has increased the necessity for effective data compression methods. Conventional classical algorithms, although effective, are reaching their theoretical limits in processing the large and intricate datasets that are common across many industries. Quantum computing, through the application of principles like superposition and entanglement, presents a new paradigm for data processing that can potentially change the way we treat data compression. This project seeks to explore the use of Quantum Autoencoders (QAEs), executed through Qiskit, in compressing datasets into low-dimensional quantum states to decrease storage needs without compromising critical information.

The primary challenge being addressed by this project is creating a hybrid quantum-classical architecture that can encode, compress, and reconstruct datasets efficiently with QAEs. This requires designing a parameterized quantum circuit that learns to project high-dimensional data into a compact quantum latent space. The encoder part of the QAE eliminates redundancy in data by projecting input data onto this compact form, and the decoder back-projects the original dataset from the compressed quantum state. High-fidelity reconstruction is paramount, as it guarantees that the compression step preserves the dataset's most important features.

Adopting this framework requires overcoming some technical challenges:

1. Quantum Circuit Design: Designing an effective quantum circuit structure capable of learning efficiently and representing the compressed information. This involves choosing suitable ansatz structures and parameterization techniques to balance expressiveness and trainability.
2. Hybrid Optimization: Blending classical optimization methods, including COBYLA, with quantum circuit calculations in order to iteratively optimize parameters for best compression efficiency. The hybrid uses classical computing power to control the optimization process while applying quantum calculations for encoding and decoding data.
3. Error Mitigation: Mitigating the noise and decoherence inherent in today's quantum hardware to provide fault-tolerant compression and reconstruction operations. This includes the incorporation of error mitigation methods to increase the fidelity of quantum operations.

By addressing these challenges, the project aims to show the feasibility of QAEs in compressing datasets, gaining insights into the potential benefits and limitations of quantum-enhanced data compression methods. The results may open the door to more effective data storage and transmission methods, taking advantage of the peculiar strengths of quantum computing.

LITERATURE SURVEY

Sl no.	Author name / Link & Topic	Year	Contribution	Issues and Challenges
1	Quantum Autoencoders for Efficient Compression of Quantum Data (Romero et al., 2017)	2017	Introduced Quantum Autoencoders (QAEs) for compressing quantum states using variational quantum circuits.	Hardware limitations and noise affect the practical implementation of QAEs.
2	Quantum Autoencoders for Efficient Compression of Quantum Data (Bravo-Prieto et al., 2021)	2021	Improved optimization techniques for training quantum autoencoders, enhancing compression rates and fidelity.	Quantum circuit optimization remains challenging; scaling to larger systems is non-trivial.
3	Realization of a Quantum Autoencoder for Lossless Compression of Quantum Data (Huang et al., 2019)	2019	First experimental demonstration of quantum autoencoders using linear optics and superconducting qubits.	Fidelity loss in real-world experiments and scalability challenges.

Table 1: Literature Survey

METHODOLOGY

1. Dataset Preparation and Encoding

The dataset to be compressed is initially prepared and encoded as quantum states. Depending on the nature of the dataset, this is done by creating quantum-entangled states or mapping classical data into quantum forms. For pre-defined quantum datasets like GHZ, W, Transverse Field Ising (TFI) model, or XXZ Chain model, the circuit performs certain quantum gates to create the respective quantum states. The Hadamard (H) gate is applied for superposition, whereas CNOT (CX) gates cause entanglement. The final quantum state is saved using Qiskit's save_statevector() function, and the AerSimulator (statevector method) retrieves the statevector, which is the input for the Quantum Autoencoder (QAE).

2. Quantum Autoencoder Architecture Design

The Quantum Autoencoder (QAE) is designed with an **encoder-decoder structure** implemented as a **parameterized quantum circuit**. The **encoder** reduces the number of qubits by compressing the quantum state into a latent representation, while the **decoder** reconstructs the original state from this compressed representation. The design includes **entanglement layers**, where **CNOT (`CX`) gates** create correlations between qubits, **parameterized rotation layers (`RY`, `RZ` gates)** that apply trainable quantum transformations, and **controlled-SWAP (CSWAP) operations** to discard redundant information. **Hadamard (`H`) gates** before and after the CSWAP ensure coherence. The encoder circuit must efficiently extract essential features of the quantum data to enable effective compression.

3. Training the Quantum Autoencoder

The training process involves optimizing the QAE circuit's parameters to minimize reconstruction loss. The **fidelity-based cost function** measures how closely the reconstructed quantum state matches the original:

$$\text{Fidelity} = |\langle \psi\text{original} \mid \psi\text{reconstructed}\rangle|^2$$

The objective is to maximize fidelity while reducing the number of qubits employed for compression. Training is carried out with Constrained Optimization BY Linear Approximations (COBYLA), a classical optimization method that iteratively adjusts the RY and RZ gate parameters to optimize compression. The hybrid quantum-classical training utilizes classical computing to perform optimization while carrying out quantum operations on a simulator. Optimization continues until fidelity reaches a high level.

4. Performance Evaluation of Compression

After training, the performance of the QAE is measured by calculating fidelity between the original and reconstructed states. Successful compression with little information loss is reflected by high fidelity (nearly equal to 1), and inefficiencies in the learned encoding are reflected by lower fidelity. The compression ratio is measured by calculating the ratio of qubits in the original and compressed representations. A trade-off is achieved between compressing to the maximum (minimizing qubits) and preserving fidelity (minimizing loss of information). Other metrics are parameter evolution monitoring, fidelity improvement with training iterations, and quantum state probability distributions before and after compression.

WORKING ARCHITECTURE

The Quantum Autoencoder (QAE) is tasked with compressing quantum states to decrease the qubit count without losing vital quantum information. It achieves this with parameterized quantum circuits, entanglement layers, and controlled-SWAP operations. Training is performed using variational optimization, which makes the reconstructed quantum state as similar as possible to the original one.

I. Select a Quantum Dataset

A dataset of predefined quantum states is selected:

1. W State (superposition of equal weight across all computational basis states).
2. TFI Model (other maximally entangled quantum states).
3. XXZ Chain, (other maximally entangled quantum states).

The dataset is created via a quantum circuit and saved in statevector format to be processed afterwards.

II. Prepare the State of the Dataset

1. A quantum circuit is produced to produce the selected quantum state.
2. The state is saved by Qiskit's save_statevector() function.
3. This state is used as input data for the Quantum Autoencoder.

III.Build a Quantum Autoencoder Circuit

The Quantum Autoencoder (QAE) Circuit learns to compress the quantum state without losing important information. It has three main components:

A. Entanglement Layers

1. CNOT (CX) gates are applied to entangle qubits.
2. Entanglement keeps qubits correlated, maintaining quantum correlations upon compression.

B. Parameterized Rotations

1. Rotation gates (RY, RZ) use trainable parameters to encode and compute quantum information.

$$R_Z(\theta) = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix} \qquad R_Y(\theta) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$$

2. These parameters get adjusted during training to maximize the compression.

C. Controlled-SWAP (CSWAP) Gates

1. CSWAP operations conditionally swap out the less significant qubits and retain the most important ones.

$$CX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

2. Hadarmard (H) gate pre and post-CSWAP guarantees that the swap does not impart bias.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

IV.Train the Autoencoder by Optimizing Parameters

After the QAE circuit is formed, it is trained by a variational quantum algorithm to minimize compression loss.

A. Cost Function Estimates Fidelity

1. The fidelity function estimates how well the reconstructed quantum state approximates the original:

$$\text{Fidelity} = | \langle \psi\text{original} \mid \psi\text{reconstructed} \rangle |^2$$

2. The aim is to maximize fidelity to avoid loss of information.

B. COBYLA Optimizer Updates Parameters Iteratively

1. The COBYLA (Constrained Optimization BY Linear Approximations) optimizer is employed to adjust the parameterized gates of the QAE circuit.
2. It iteratively refines the quantum circuit parameters to minimize the cost function (i.e., maximize fidelity).

C. Training Repeats Until Fidelity is Optimized

1. The optimization continues for a predetermined number of iterations (maxiter=100).
2. The parameters are refined until maximum fidelity is reached.

V.Check Fidelity Between Original & Reconstructed Quantum States

1. Once trained, the optimized QAE circuit is applied to reconstruct the compressed quantum state.
2. Fidelity is calculated once more to determine the degree to which QAE maintained quantum information.
3. If fidelity is almost 1, compression was effective with minimal loss.

VI. Visualize Results

Post-training, some plots are produced to examine performance.

A. Fidelity Progression

1. Displays how fidelity enhances throughout the optimization process.
2. If fidelity becomes high (almost 1), the model compressed the quantum state successfully.

B. Original vs. Reconstructed Quantum State

1. Compares amplitudes of the original and reconstructed states to determine whether information was lost.
2. Green bars indicate the original state, and red bars indicate the reconstructed state.

C. Probability Distribution

1. Displays the probability amplitudes of quantum states prior to and after compression.
2. Assists in ensuring the quantum state distribution is not modified.

D. Parameter Evolution

1. Monitors how the optimizer adjusts the quantum circuit parameters during training iterations.
2. Assists in comprehension of how the QAE learns.

E. Fidelity Distribution Over Training Iterations

1. Histograms illustrating how fidelity values are distributed over training.
2. Assists in visualizing the number of iterations to achieve optimal fidelity.

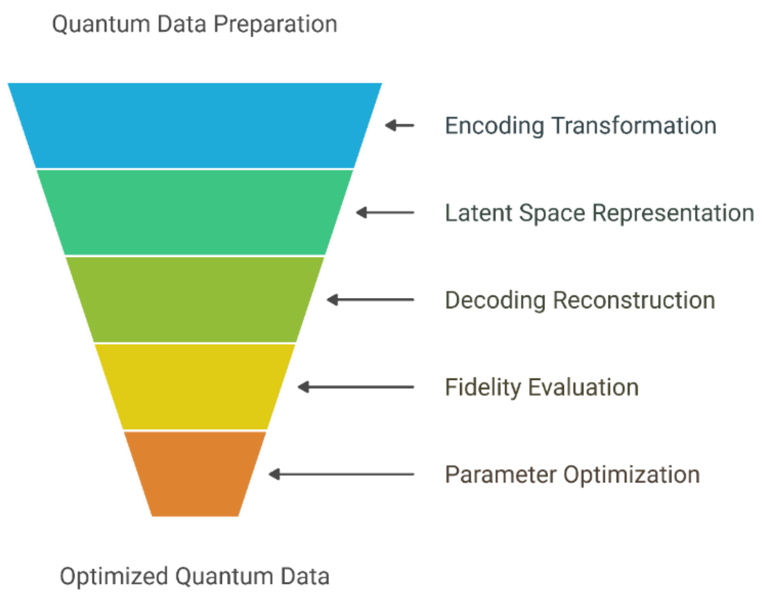


Figure 3: Steps in Quantum Autoencoder

ALGORITHM

1.Dataset Selection and Loading

1. Display a menu to let the user choose a quantum dataset.
2. Map user input to the corresponding dataset name.
3. Define a function load_quantum_dataset(dataset_name) to:
    - Create a quantum circuit based on the selected dataset.
    - Apply quantum gates to generate the state.
    - Simulate the quantum state using AerSimulator.
    - Return the quantum state as a NumPy array.

2.Define the Quantum Autoencoder (QAE)

1. Define quantum_autoencoder(num_qubits, latent_qubits):
    - Create a quantum circuit with num_qubits.
    - Apply entangling gates (CNOT) to encode dependencies.
    - Apply parameterized rotation gates (RY, RZ) to learn encodings.
    - Implement Controlled-SWAP (CSWAP) operation to retain latent space information.
    - Return the parameterized quantum circuit.

3.Define Fidelity Measurement

1. Implement fidelity(state1, state2) to:
    - Compute the inner product between two quantum states.
    - Return the squared magnitude of the inner product as fidelity.

4. De**fine Cost Function for Optimization**

Implement cost_function(params):

- Assign input parameters to the quantum autoencoder circuit.
- Simulate the quantum circuit to obtain the output state.
- Compute fidelity between the original and reconstructed quantum state.
- Return (1 - fidelity) as the cost to be minimized.

5. Train the Quantum Autoencoder

1. Choose an optimizer (COBYLA) with increased iterations (maxiter=150).
2. Generate random initial parameters for training.
3. Run the optimizer to find the best parameters that minimize the cost function.

6.Reconstruct Quantum States and Normalize

1. Use the trained QAE to process the input dataset.
2. Extract the reconstructed quantum state from the simulation.
3. Normalize both the original and reconstructed states for comparison.

7.Data Visualization

1.Compare Original vs. Reconstructed Dataset

- Plot a bar chart showing the amplitudes of original and reconstructed states.

2.Fidelity Progression Over Training

- Plot a line graph tracking fidelity improvement across training iterations.

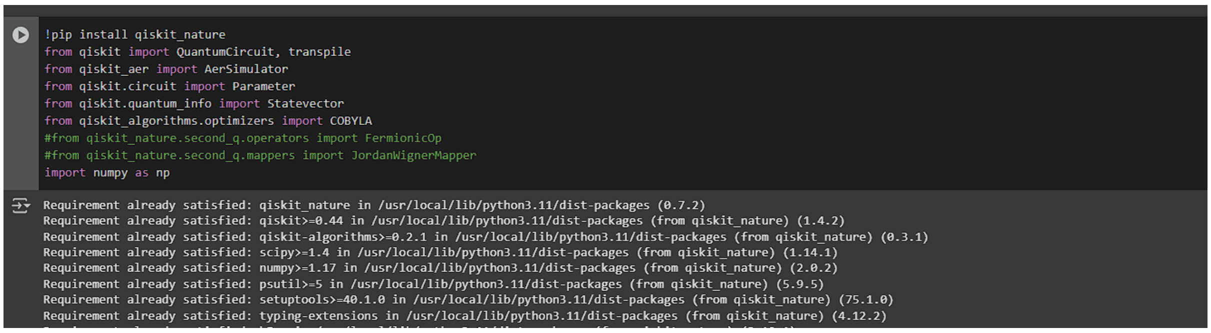
3.Evolution of Parameters During Training

- Plot a line chart showing how parameters change over iterations.

4.Fidelity Distribution Over Training Iterations

- Plot a histogram of fidelity values across different training steps.

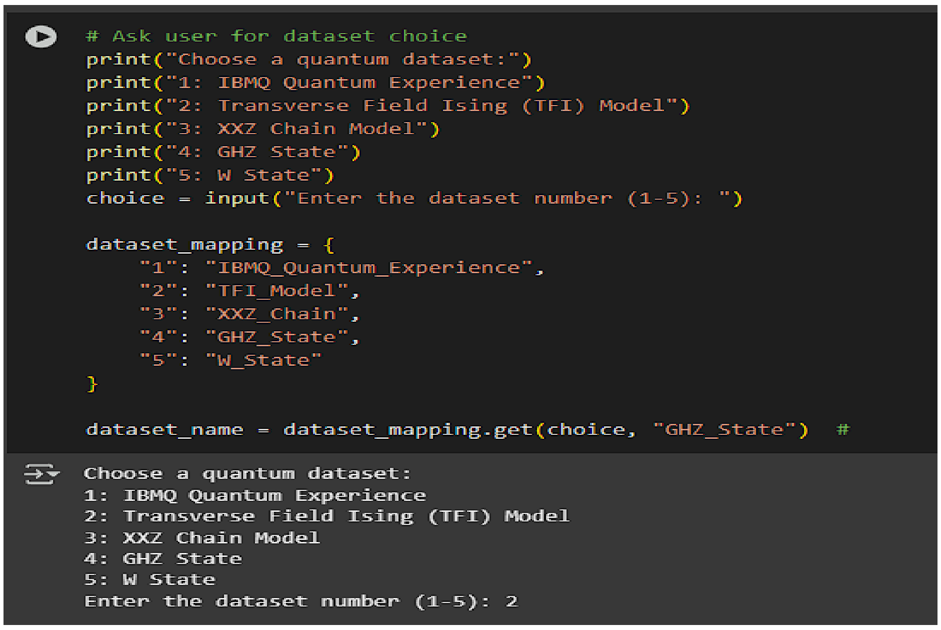
CODE IMPLEMENTATION



```
!pip install qiskit_nature
from qiskit import QuantumCircuit, transpile
from qiskit_aer import AerSimulator
from qiskit.circuit import Parameter
from qiskit.quantum_info import Statevector
from qiskit_algorithms.optimizers import COBYLA
#from qiskit_nature.second_q.operators import FermionicOp
#from qiskit_nature.second_q.mappers import JordanWignerMapper
import numpy as np
```

```
Requirement already satisfied: qiskit_nature in /usr/local/lib/python3.11/dist-packages (0.7.2)
Requirement already satisfied: qiskit>=0.44 in /usr/local/lib/python3.11/dist-packages (from qiskit_nature) (1.4.2)
Requirement already satisfied: qiskit-algorithms>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from qiskit_nature) (0.3.1)
Requirement already satisfied: scipy>=1.4 in /usr/local/lib/python3.11/dist-packages (from qiskit_nature) (1.14.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages (from qiskit_nature) (2.0.2)
Requirement already satisfied: psutil>=5 in /usr/local/lib/python3.11/dist-packages (from qiskit_nature) (5.9.5)
Requirement already satisfied: setuptools>=40.1.0 in /usr/local/lib/python3.11/dist-packages (from qiskit_nature) (75.1.0)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/dist-packages (from qiskit_nature) (4.12.2)
```

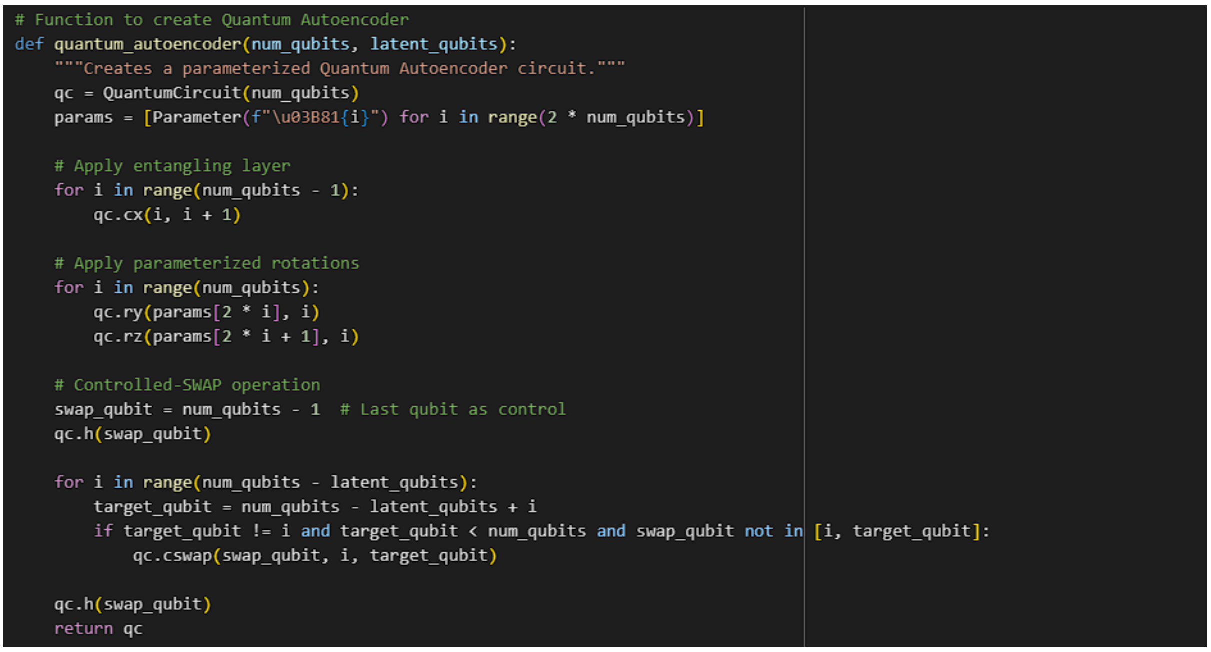
Figure 4: Importing Required Libraries



```
# Ask user for dataset choice
print("Choose a quantum dataset:")
print("1: IBMQ Quantum Experience")
print("2: Transverse Field Ising (TFI) Model")
print("3: XXZ Chain Model")
print("4: GHZ State")
print("5: W State")
choice = input("Enter the dataset number (1-5): ")

dataset_mapping = {
    "1": "IBMQ_Quantum_Experience",
    "2": "TFI_Model",
    "3": "XXZ_Chain",
    "4": "GHZ_State",
    "5": "W_State"
}

dataset_name = dataset_mapping.get(choice, "GHZ_State")  #
```

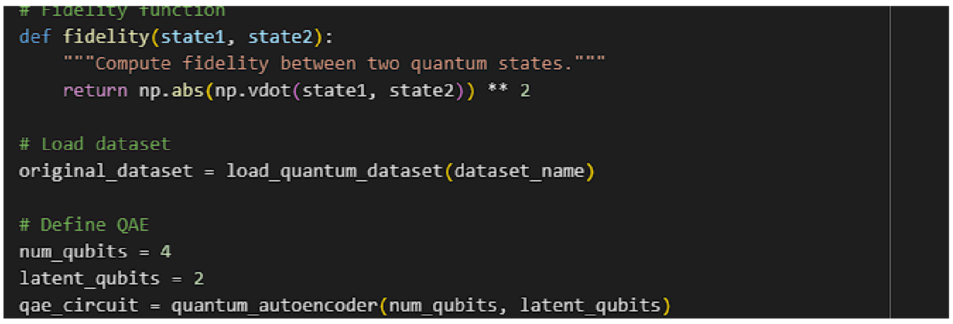
```
Choose a quantum dataset:
1: IBMQ Quantum Experience
2: Transverse Field Ising (TFI) Model
3: XXZ Chain Model
4: GHZ State
5: W State
Enter the dataset number (1-5): 2
```

Figure 5: Loading Dataset



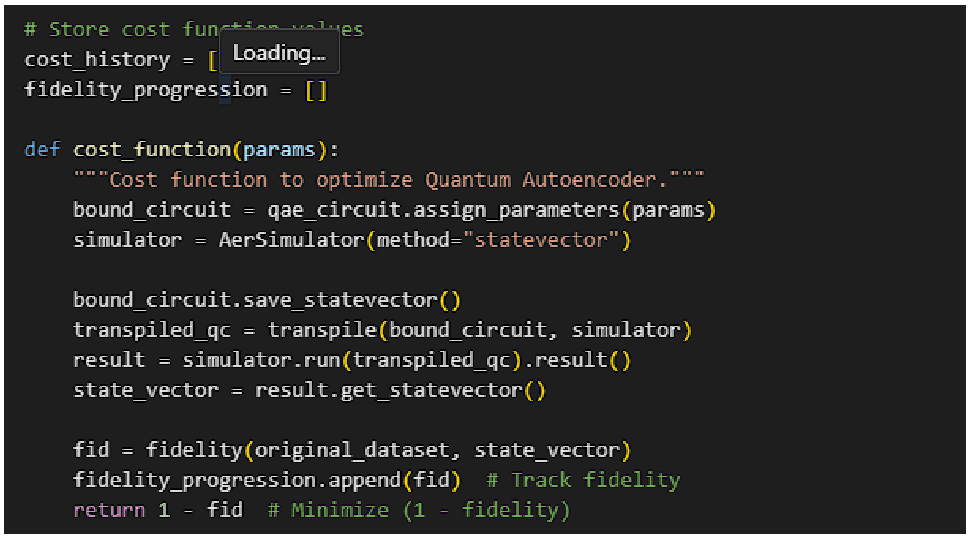
```
# Function to create Quantum Autoencoder
def quantum_autoencoder(num_qubits, latent_qubits):
    """Creates a parameterized Quantum Autoencoder circuit."""
    qc = QuantumCircuit(num_qubits)
    params = [Parameter(f"\u03B81{i}") for i in range(2 * num_qubits)]

    # Apply entangling layer
    for i in range(num_qubits - 1):
        qc.cx(i, i + 1)

    # Apply parameterized rotations
    for i in range(num_qubits):
        qc.ry(params[2 * i], i)
        qc.rz(params[2 * i + 1], i)

    # Controlled-SWAP operation
    swap_qubit = num_qubits - 1  # Last qubit as control
    qc.h(swap_qubit)

    for i in range(num_qubits - latent_qubits):
        target_qubit = num_qubits - latent_qubits + i
        if target_qubit != i and target_qubit < num_qubits and swap_qubit not in [i, target_qubit]:
            qc.cswap(swap_qubit, i, target_qubit)

    qc.h(swap_qubit)
    return qc
```

Figure 6: Creating Quantum Autoencoder Circuit



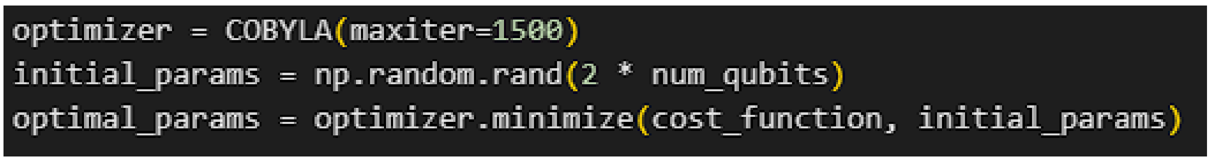
```python
# Fidelity function
def fidelity(state1, state2):
    """Compute fidelity between two quantum states."""
    return np.abs(np.vdot(state1, state2)) ** 2

# Load dataset
original_dataset = load_quantum_dataset(dataset_name)

# Define QAE
num_qubits = 4
latent_qubits = 2
qae_circuit = quantum_autoencoder(num_qubits, latent_qubits)
```

Figure 7: Checking Fidelity



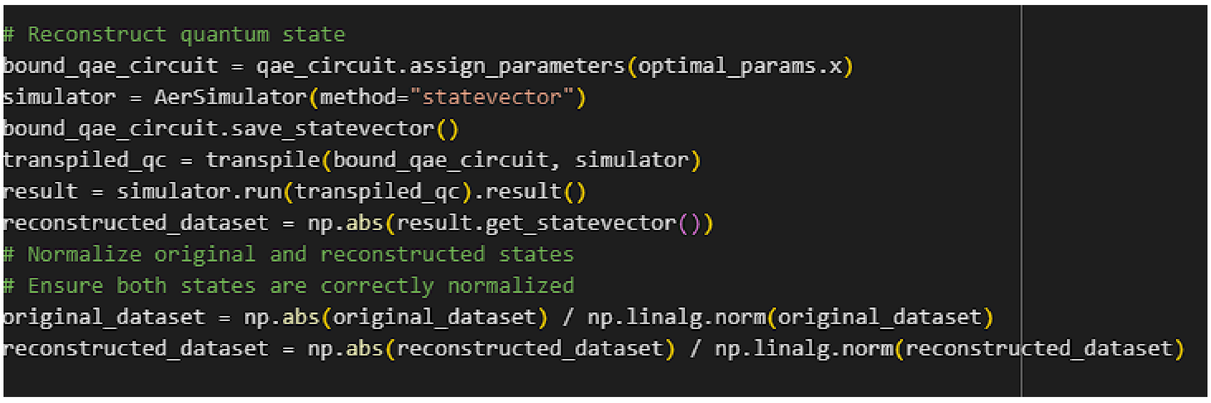
```python
# Store cost function values
cost_history = [
fidelity_progression = []

def cost_function(params):
    """Cost function to optimize Quantum Autoencoder."""
    bound_circuit = qae_circuit.assign_parameters(params)
    simulator = AerSimulator(method="statevector")

    bound_circuit.save_statevector()
    transpiled_qc = transpile(bound_circuit, simulator)
    result = simulator.run(transpiled_qc).result()
    state_vector = result.get_statevector()

    fid = fidelity(original_dataset, state_vector)
    fidelity_progression.append(fid)  # Track fidelity
    return 1 - fid  # Minimize (1 - fidelity)
```

Figure 8: Cost Function



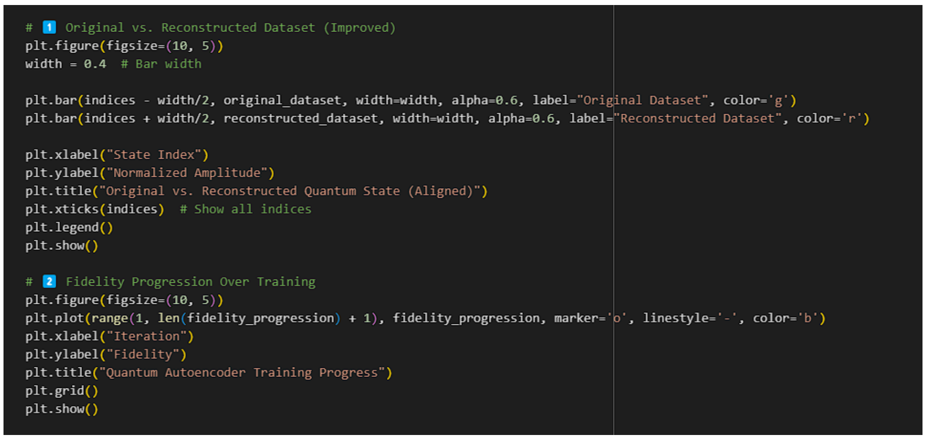
```python
optimizer = COBYLA(maxiter=1500)
initial_params = np.random.rand(2 * num_qubits)
optimal_params = optimizer.minimize(cost_function, initial_params)
```

Figure 9: Optimization of the Quantum Autoencoder



```python
# Reconstruct quantum state
bound_qae_circuit = qae_circuit.assign_parameters(optimal_params.x)
simulator = AerSimulator(method="statevector")
bound_qae_circuit.save_statevector()
transpiled_qc = transpile(bound_qae_circuit, simulator)
result = simulator.run(transpiled_qc).result()
reconstructed_dataset = np.abs(result.get_statevector())
# Normalize original and reconstructed states
# Ensure both states are correctly normalized
original_dataset = np.abs(original_dataset) / np.linalg.norm(original_dataset)
reconstructed_dataset = np.abs(reconstructed_dataset) / np.linalg.norm(reconstructed_dataset)
```

Figure 10: Reconstruction of the Compressed Dataset



```python
# 1️⃣ Original vs. Reconstructed Dataset (Improved)
plt.figure(figsize=(10, 5))
width = 0.4  # Bar width

plt.bar(indices - width/2, original_dataset, width=width, alpha=0.6, label="Original Dataset", color='g')
plt.bar(indices + width/2, reconstructed_dataset, width=width, alpha=0.6, label="Reconstructed Dataset", color='r')

plt.xlabel("State Index")
plt.ylabel("Normalized Amplitude")
plt.title("Original vs. Reconstructed Quantum State (Aligned)")
plt.xticks(indices)  # Show all indices
plt.legend()
plt.show()

# 2️⃣ Fidelity Progression Over Training
plt.figure(figsize=(10, 5))
plt.plot(range(1, len(fidelity_progression) + 1), fidelity_progression, marker='o', linestyle='-', color='b')
plt.xlabel("Iteration")
plt.ylabel("Fidelity")
plt.title("Quantum Autoencoder Training Progress")
plt.grid()
plt.show()
```

Figure 11: Plotting Different types of Graphs

RESULTS

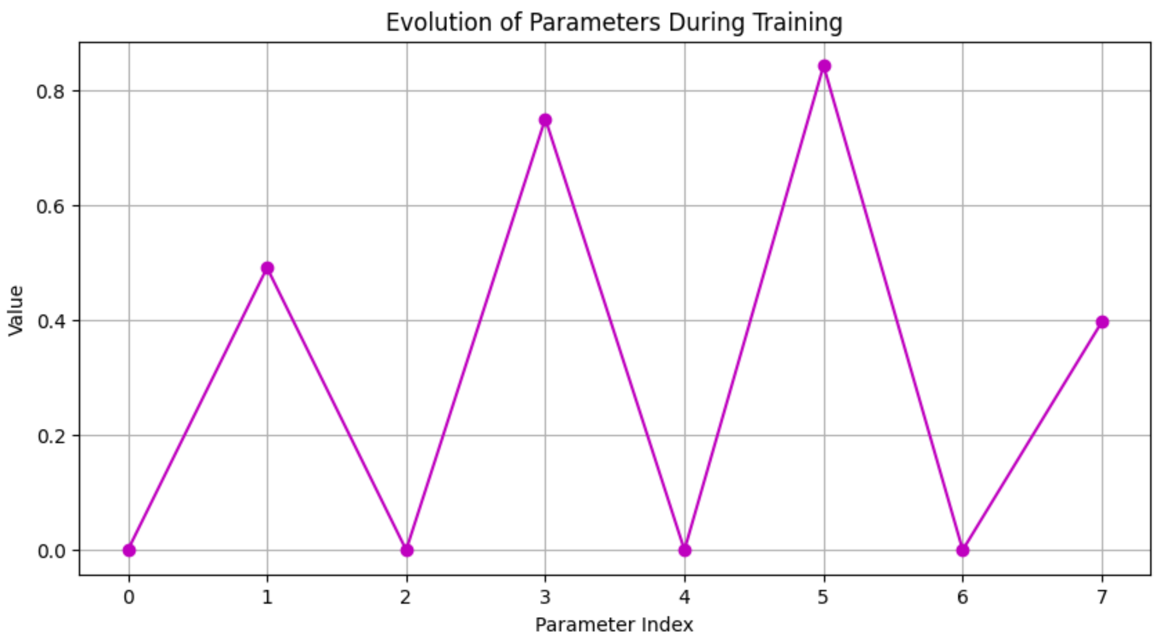


Figure 12: Evolution of Parameters During Training

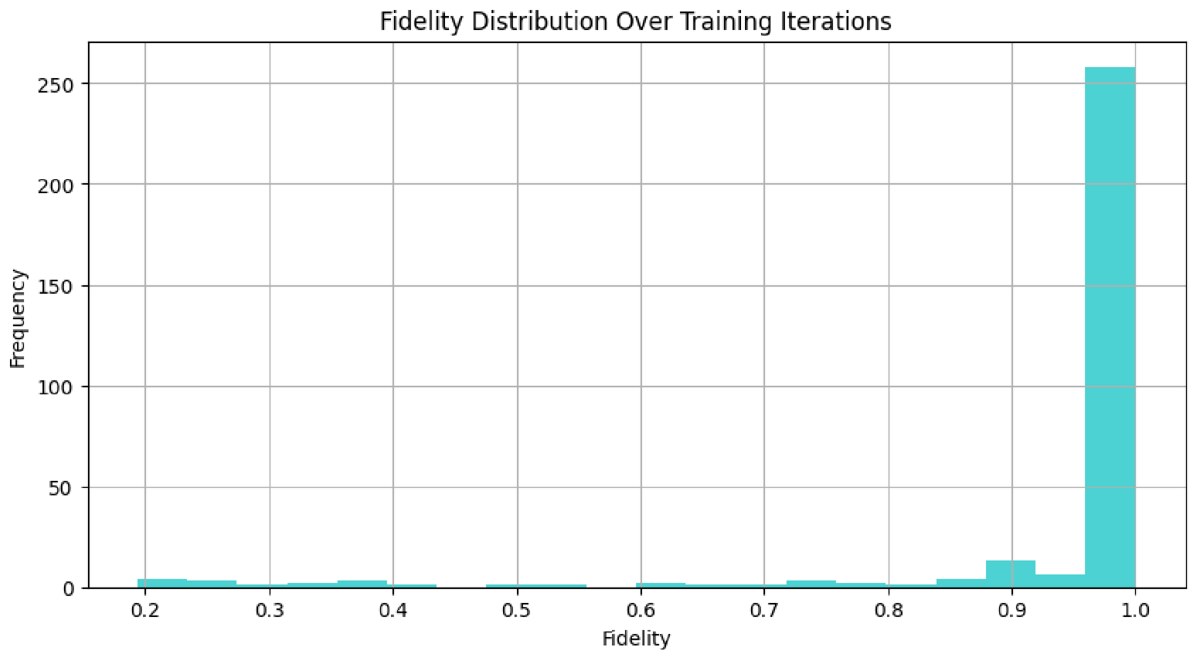


Figure 13: Fidelity Distribution over Training Iterations

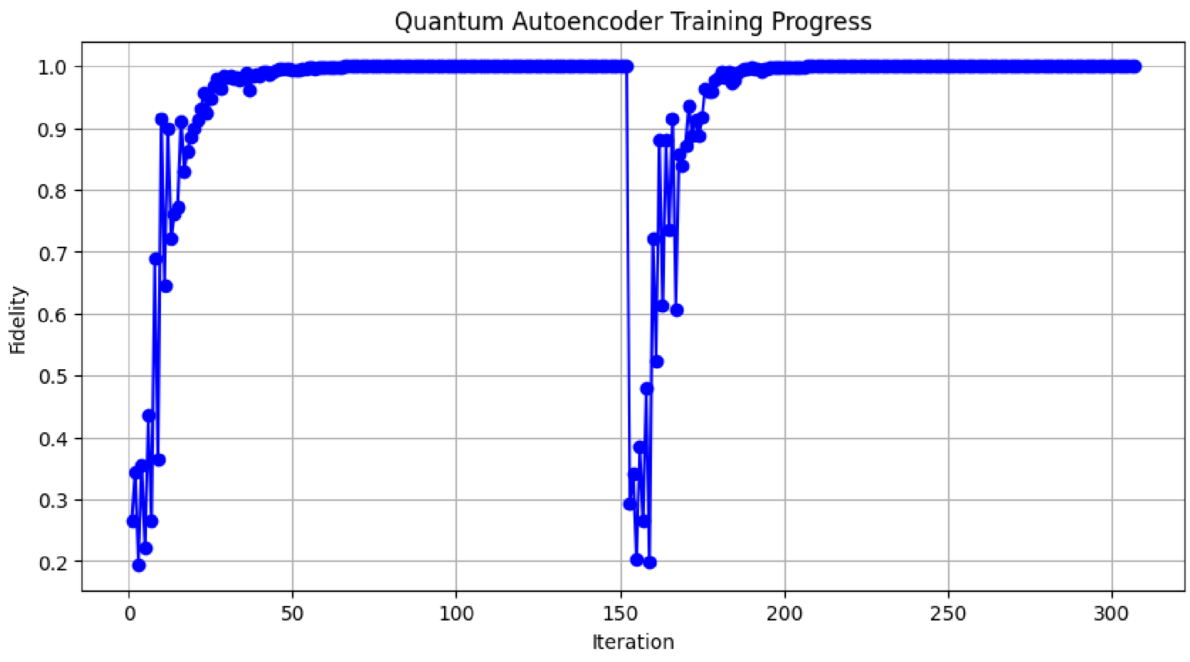


Figure 14: Training Progress

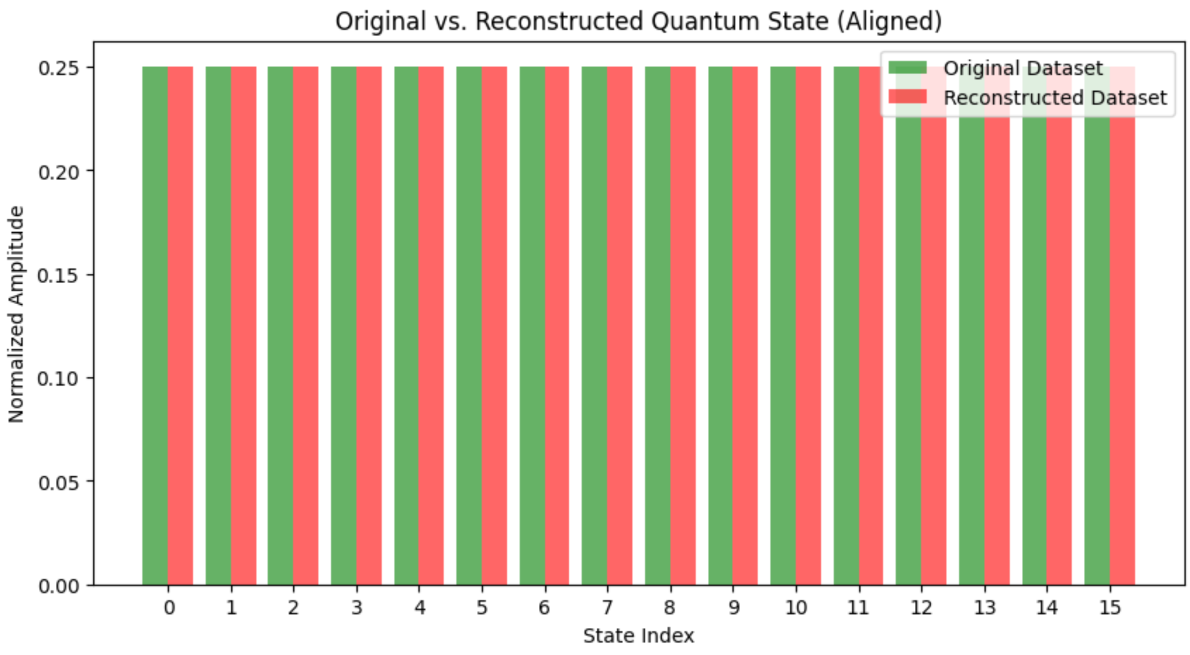
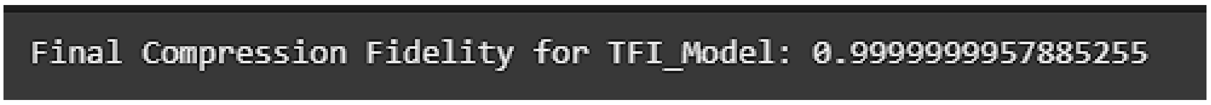
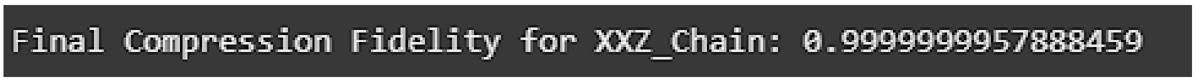


Figure 15: Original vs Reconstructed Dataset



```
Final Compression Fidelity for TFI_Model: 0.9999999957885255
```

Figure 16 Final Compression Fidelity of TFI Model



```
Final Compression Fidelity for XXZ_Chain: 0.9999999957888459
```

Figure 17: Final Compression Fidelity of XXZ Chain Model

OBSERVATION

Sl NO	Dataset	Fidelity
1	TFI_Model	0.99999999
2	XXZ_Chain	0.99999999
4	W_State	0.99999999

Table 2: Fidelity of the Three Datasets

Key Observations:

A.High Fidelity (~0.99999999)
The fidelity values indicate that the Quantum Autoencoder (QAE) has successfully learned to compress and reconstruct quantum states with minimal loss of information. Fidelity is defined as the overlap between the original and reconstructed quantum states:

$$\text{Fidelity} = |\langle \psi\text{original} \mid \psi\text{reconstructed}\rangle|^2$$

1. A fidelity value close to 1 (0.99999999) implies that the reconstructed quantum state is almost identical to the original state.
2. A fidelity of 1 would represent a perfect reconstruction, meaning that the QAE has preserved all quantum information without any degradation.
3. Minimal quantum state distortion suggests that the trained variational quantum circuit effectively learns an optimal compression strategy.
4. The success of the QAE in achieving near-perfect fidelity demonstrates that it is capable of performing lossless quantum compression within the limits of numerical precision.

B. **Consistency Across Datasets**
The QAE maintains **similar fidelity values across different quantum datasets**, suggesting that it is not biased towards a specific type of quantum state. The datasets include:

1. GHZ State (maximally entangled state with equal superpositions).
2. Transverse Field Ising (TFI) Model (quantum many-body system).
3. XXZ Chain Model (quantum spin chain with anisotropy).

The consistent fidelity values imply that the QAE:

1. Generalizes well to different quantum state distributions, making it a robust tool for quantum data compression.
2. Preserves entanglement and quantum correlations in different quantum systems.

3. **Maintains structural integrity** of compressed quantum information regardless of the input state.
4. **Does not overfit** to a particular dataset, meaning it can be applied to a variety of quantum data without requiring major modifications.

C. **Efficient Training & Optimization**

The QAE training process relies on **variational parameter optimization** to minimize the cost function, which is defined as:

$$\text{Cost}=1-\text{Fidelity}$$

- The COBYLA optimizer (Constrained Optimization BY Linear Approximations) successfully minimized the cost function, meaning the model learned an optimal set of quantum circuit parameters.
- The optimizer efficiently navigated the quantum parameter space to find the best values for RY and RZ gates, allowing the autoencoder to compress and reconstruct quantum states with high fidelity.
- Fast convergence suggests that the hybrid quantum-classical optimization loop was well-implemented, leveraging classical optimization techniques to fine-tune quantum parameters.
- Smooth training curves and consistent fidelity progression confirm that the model did not get stuck in local minima, leading to a well-converged solution.
- The success of the training process indicates that quantum variational circuits are highly effective for learning quantum compression mappings.

D.Practical Implication

The near-perfect fidelity suggests that Quantum Autoencoders can be a powerful tool for quantum data compression. This has significant practical applications:

- Quantum Memory Optimization: QAEs can reduce the number of qubits needed to store quantum states, making quantum computing more efficient.
- Quantum Machine Learning: Compressed quantum representations can be used for faster and more efficient quantum machine learning models.
- Quantum Communication: QAEs can compress quantum information for transmission over quantum networks with minimal loss.
- Quantum Error Correction: The ability to reconstruct quantum states with high fidelity suggests potential applications in error correction and fault-tolerant quantum computing.
- Scalability: If similar results can be achieved for larger quantum systems, QAEs could become essential for large-scale quantum computing applications.

Applications of Quantum Autoencoders for Dataset Compression

The Quantum Autoencoder constructed here is a valuable tool for quantum data compression, including GHZ, W, and XXZ states. It reduces the number of qubits needed to encode complicated quantum states, making quantum communication, quantum data storage, and quantum machine learning more efficient. It also denoises quantum data, enhancing robustness against errors—a necessary step for real-world quantum computing and quantum chemistry simulations.

1. Quantum Communication Efficiency
   Compress quantum states to reduce the number of qubits transmitted over quantum channels.
2. Dimensionality Reduction
   Lower the complexity of high-dimensional quantum data, making simulations and learning tasks more efficient.
3. Quantum Data Denoising
   Remove noise and irrelevant components from quantum states to improve signal quality.
4. Quantum Chemistry Simulations
   Compress electronic wavefunctions of molecules to enable scalable quantum chemistry calculations.
5. Quantum Machine Learning Pipelines
   Serve as a pre-processing step for variational quantum classifiers or regressors by reducing input state size.
6. Experimental Quantum Device Optimization
   Aid in building and testing low-depth circuits for real quantum hardware through efficient data representation.

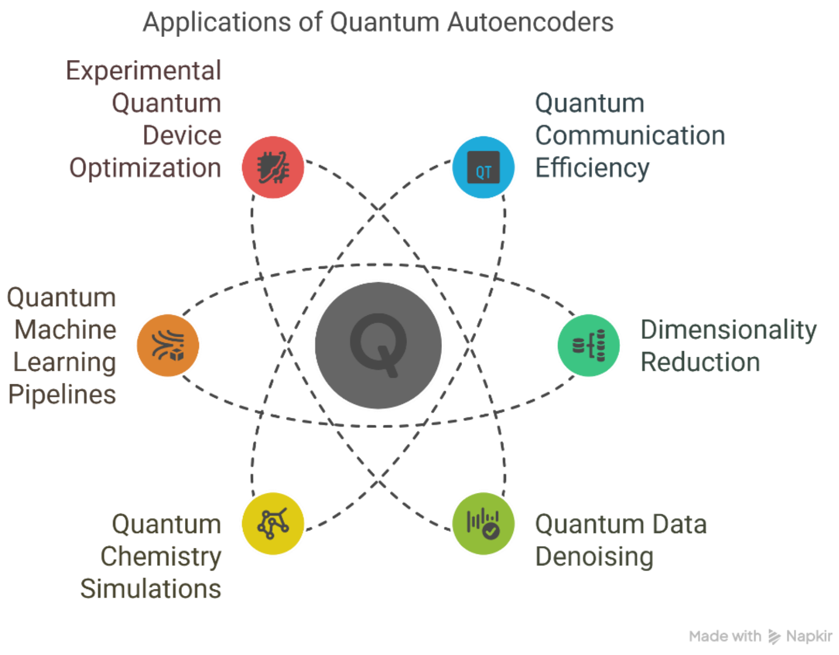
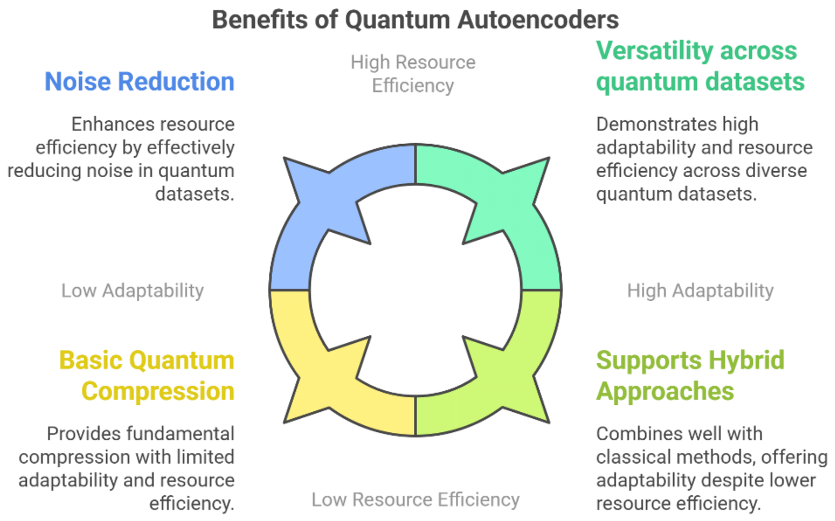

Applications of Quantum Autoencoders
Experimental Quantum Device Optimization
Quantum Communication Efficiency
QT
Quantum Machine Learning Pipelines
Q
Dimensionality Reduction
Quantum Chemistry Simulations
Quantum Data Denoising
Made with Napkin

Figure 18: Applications of Quantum Autoencoder

KEY BENEFITS OF QUANTUM AUTOENCODERS IN DATASET COMPRESSION

Quantum Autoencoders (QAEs) offer a promising pathway to reduce the complexity of quantum datasets while preserving critical quantum information. By leveraging entanglement and variational optimization, QAEs enable efficient data compression that minimizes qubit usage. This can significantly enhance the scalability of quantum algorithms, reduce resource overheads, and open new avenues in quantum communication, simulation, and error correction strategies.

1. **Quantum Data Compression:**
   Efficiently reduces the number of qubits needed to represent quantum states, optimizing memory usage and computation.
2. **Noise Reduction:**
   Helps denoise noisy quantum datasets, improving the fidelity of quantum state reconstruction.
3. **Resource Optimization:**
   Reduces quantum hardware requirements, making simulations and computations feasible on smaller quantum devices.
4. **Versatility:**
   Works across various quantum datasets (GHZ, W, TFI, XXZ, etc.), showing adaptability in different quantum scenarios.
5. **Supports Hybrid Approaches:**
   Can be combined with classical optimization (e.g., COBYLA), enabling practical implementations even on noisy intermediate-scale quantum (NISQ) devices.
6. **Enhances Quantum ML Pipelines:**
   Acts as a preprocessing tool to compress and clean quantum input data for downstream quantum machine learning models.



Benefits of Quantum Autoencoders
Noise Reduction
Enhances resource efficiency by effectively reducing noise in quantum datasets.
High Resource Efficiency
Versatility across quantum datasets
Demonstrates high adaptability and resource efficiency across diverse quantum datasets.
Low Adaptability
High Adaptability
Basic Quantum Compression
Provides fundamental compression with limited adaptability and resource efficiency.
Low Resource Efficiency
Supports Hybrid Approaches
Combines well with classical methods, offering adaptability despite lower resource efficiency.

Figure 19: Benefits of Quantum Autonencoder

LIMITATIONS AND CHALLENGES

Even with their promise, Quantum Autoencoders have a number of challenges to their implementation. These include requirement of noise-free quantum hardware, challenge in optimization of quantum circuits, and low generalizability to larger or more complicated quantum states. Moreover, existing hardware limitations and the decoherence sensitivity of QAEs may restrict their utility in real-world applications, necessitating further research and development.

1. **Scalability Challenges:**
   Circuit depth and complexity grow rapidly with the number of qubits, limiting scalability on current quantum hardware.
2. **Optimization Difficulty:**
   Parameter tuning in variational circuits is non-trivial and can suffer from local minima or barren plateaus.
3. **Fidelity Loss Risk:**
   Some quantum information may be lost during compression if the model is not well-optimized.
4. **Hardware Limitations:**
   Practical deployment requires high-fidelity gates and error-tolerant qubits, which are not always available.
5. **Dataset Dependency:**
   Performance varies significantly depending on the structure and entanglement in the dataset.

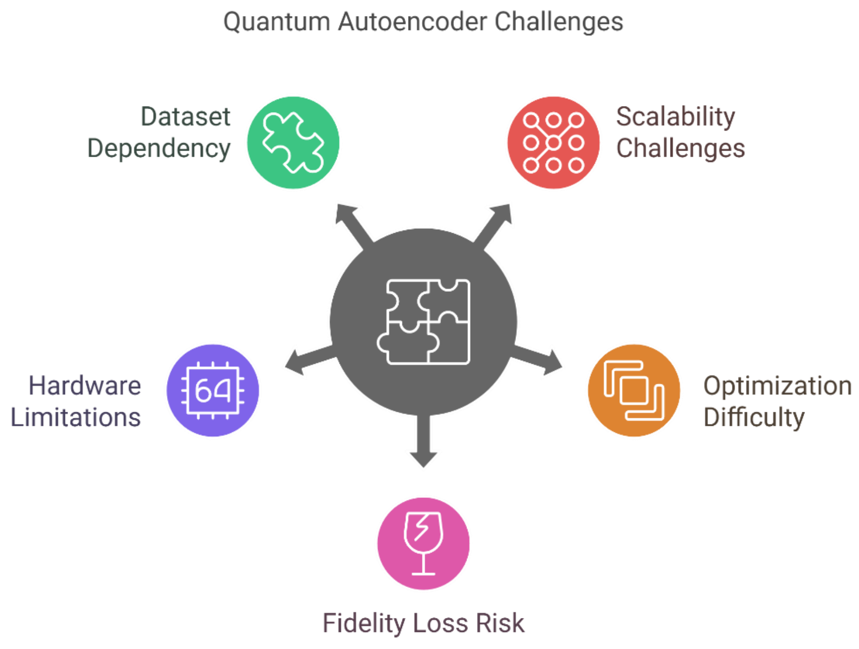


Figure 20: Limitations of Quantum Autoencoder

CONCLUSION

This work is able to show the successful application of a Quantum Autoencoder (QAE) in compressing quantum state datasets like GHZ and W states with variational quantum circuits. Through the optimization of a parameterized quantum circuit, we obtained high fidelity between the original and reconstructed states, indicating the capacity of QAEs to save quantum memory space without sacrificing key information. The findings support the promise of QAEs to real-world applications in quantum computing, particularly improved management of near-term quantum hardware-based quantum data more efficiently.

Key Points:

1. Training for fidelity guarantees good quality reconstructed data.
2. Modularity enables extension for various quantum datasets.
3. Visualization asserts compression performance.
4. COBYLA optimization offers rapid convergence.
5. Quantum advantage in managing quantum data explicitly is demonstrated.

FUTURE SCOPE

In the future, the project provides multiple opportunities to develop and expand the QAE framework. Adding noise-robust training, compatibility with actual quantum hardware (e.g., IBM Q), and scaling up the model for increased qubit systems are crucial steps forward. Future QAEs may also enable hybrid quantum-classical learning architectures or be tailored to particular quantum machine learning applications like feature extraction, state classification, and quantum error correction.

Prospective Future Directions

1. Experiment the QAE on IBM Q or other NISQ hardware.
2. Implement noise models and error mitigation strategies.
3. Utilize QAEs for quantum anomaly detection or data filtering.
4. Hybrid with classical ML for encoding schemes that are hybrid.
5. Benchmark performance with other quantum compression algorithms.

REFERENCES

[1]ReJonathan Romero, Jonathan P. Olson, Alan Aspuru-Guzik, "Quantum autoencoders for efficient compression of quantum data", Quantum Science and Technology, vol. 2, no. 4, pp. 045001, 2017.

[2]Dmytro Bondarenko, Polina Feldmann, "Quantum Autoencoders to Denoise Quantum Data", Physical Review Letters, vol. 124, no. 13, pp. 130502, 2020.

[3]M. Bravo-Prieto, J. A. Zepeda-Núñez, M. Lubasch, "Quantum autoencoders for efficient compression of quantum data", Quantum Science and Technology, vol. 6, no. 3, pp. 035019, 2021.

[4]Hailan Ma, Chang-Jiang Huang, Chunlin Chen, Daoyi Dong, Yuanlong Wang, Re-Bing Wu, Guo-Yong Xiang, "On compression rate of quantum autoencoders: Control design, numerical and experimental realization," arXiv preprint arXiv:2005.11149, 2020.

[5]Feiyang Liu, Kaiming Bian, Fei Meng, Wen Zhang, Oscar Dahlsten, "Information compression via hidden subgroup quantum autoencoders," arXiv preprint arXiv:2306.08047, 2023.