GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN IN PUBLICA COMMODA SEIT 1737

HPS
https://hps.vi4io.org/

Zhuojing Huang

Quantum Neural Networks: Libraries and Applications

An example of quantum autoencoder

Table of contents

1 Introduction: Quantum Computing

2 Overview: Quantum Neural Network

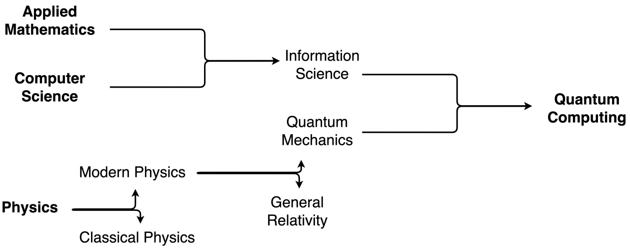
3 Use Case: Quantum Autoencoder

4 Getting into the Field

5 Main Takeaways

What is Quantum Computing?

- Multidisciplinary fields
  - Computer science
  - Physics
  - Mathematics



Source: Ayoade, Rivas, and Orduz, “Artificial Intelligence Computing at the Quantum Level”

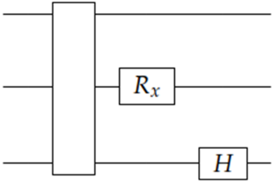
Essential Concepts in Quantum Computing

- Qubits and Supersupposition
  - Utilizes quantum bits, or **qubits**, instead of classical bits.
  - Superposition: qubits can exist in multiple states simultaneously.
- Entanglement
  - The state of one qubit can be dependent on another, even at a distance. "*Spooky action at a distance*" —- *Albert Einstein*
  - Enables faster information transfer and parallel processing.

Source: Ayoade, Rivas, and Orduz, “Artificial Intelligence Computing at the Quantum Level”

Essential Concepts in Quantum Computing

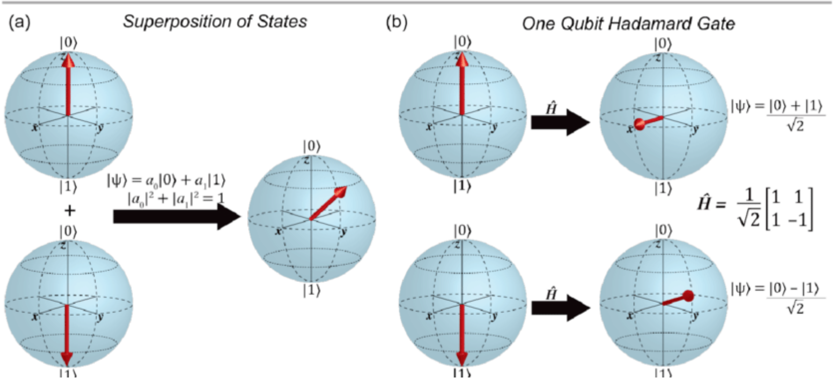
- **Quantum Gates**
  - Quantum gates act on the qubits in a quantum circuit



  - Exploit superposition to process multiple possibilities, allowing for complex operations
  - Create entanglement, where the state of two qubits become correlated

Source: Ayoade, Rivas, and Orduz, “Artificial Intelligence Computing at the Quantum Level”

Essential Concepts in Quantum Computing



Source: Moreno-Pineda et al., “Molecular Spin Qudits for Quantum Algorithms”

Essential Concepts in Quantum Computing

Source: https://arstechnica.com/science/2015/09/d-wave-unveils-new-quantum-computing-benchmark-and-its-fast/

:D

Essential Concepts in Quantum Computing

Source: https://arstechnica.com/science/2015/09/d-wave-unveils-new-quantum-computing-benchmark-and-its-fast/

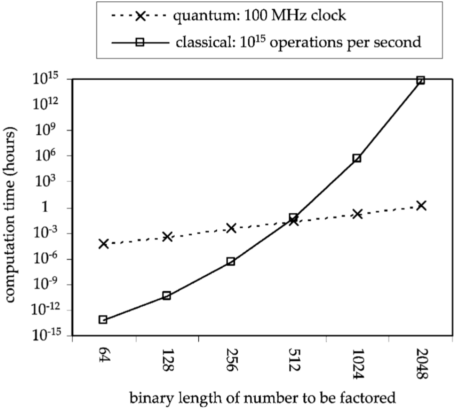
:D

Source: https://venturebeat.com/business/ibm-releases-qiskit-modules-that-use-quantum-computers-to-improve-machine-learning/

Why Do We Need Quantum Computers?

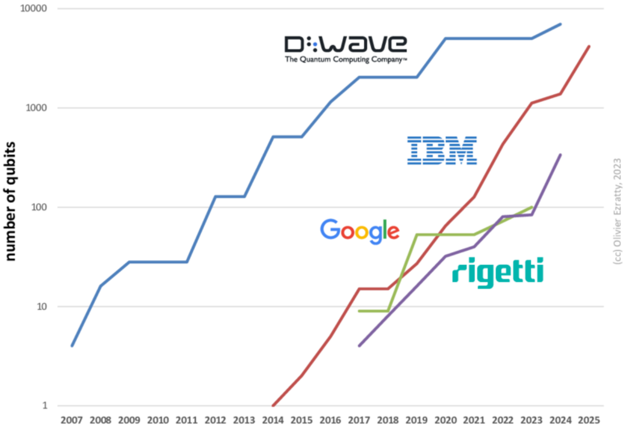
- Limitations of supercomputers
  - Reliant on 20th-century transistor technology
  - Slow or unable to solve some complex problems
- What quantum computers can do
  - Lead to more efficient computation for certain tasks with parallelism
  - Provide exponential speedup for certain computational tasks:
    - factoring large numbers
    - searching unsorted databases
  - Excel in solving complex optimization problems

Current stage of Quantum Computing



Source: Berggren, “Quantum Computing with Superconductors”

Current stage of Quantum Computing

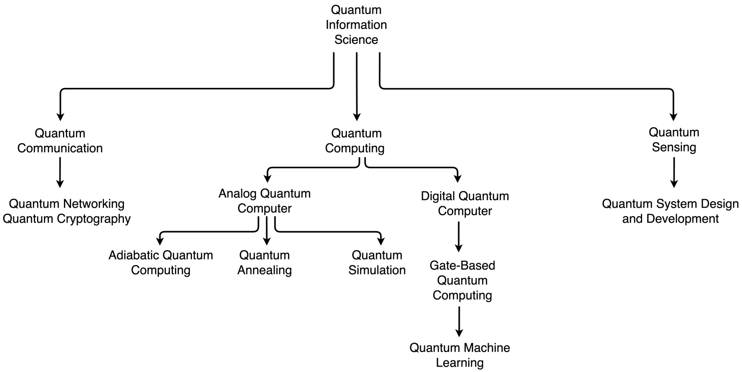


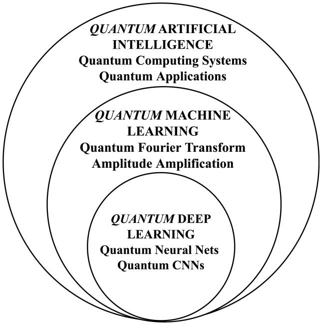
Source: Ezratty, “Is there a Moore’s law for quantum computing?”

What fields can Quantum Computing be useful?

- Optimization Problems
- Cryptography
- Quantum Simulation
- Machine Learning
- Molecular Modeling
- Integer Factorization
- ...

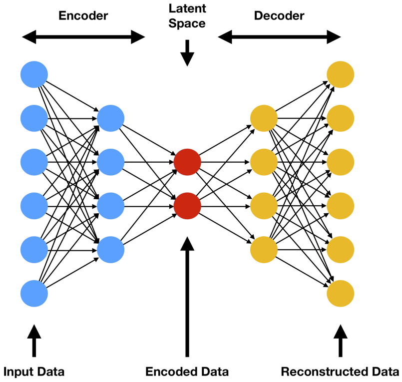
What is Quantum Neural Network?





Source: Ayoade, Rivas, and Orduz, “Artificial Intelligence Computing at the Quantum Level”

Recap on Classical Autoencoder: Basics

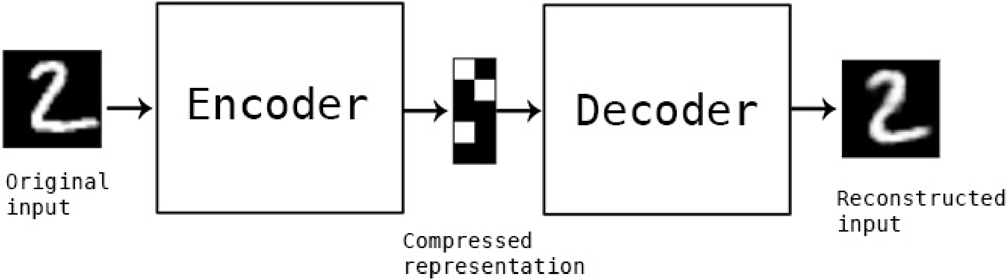


Source: Rao, “Learning hard distributions with quantum-enhanced Variational Autoencoders”

- Encoder and Decoder
  - Encoder maps input to a latent space
  - Decoder reconstructs data from the latent space
- Essential ideas
  - Unsupervised
  - Reconstruct original data from the compressed data
  - Minimize the difference between input and output

Recap on Classical Autoencoder: Applications

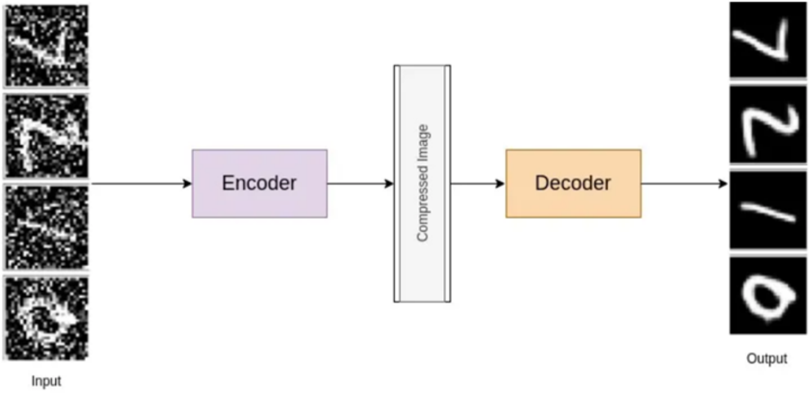
- Data Compression



Source: https://medium.com/edureka/autoencoders-tutorial-cfdcebdefe37

Recap on Classical Autoencoder: Applications

- Data Denoising



Source: https://medium.com/analytics-vidhya/reconstruct-corrupted-data-using-denoising-autoencoder-python-code-aeaff4b0958e

What is a Quantum Autoencoder (QAE)?

- Inspired by classical autoencoder

What is a Quantum Autoencoder (QAE)?

- Inspired by classical autoencoder
- Compress a quantum state onto a smaller amount of qubits, while retaining the initial information
- Can be used for the same purpose, i.e., dimentionality reduction

What is a Quantum Autoencoder (QAE)?

- Inspired by classical autoencoder
- Compress a quantum state onto a smaller amount of qubits, while retaining the initial information
- Can be used for the same purpose, i.e., dimentionality reduction
- Copy the structure of the classcial autoencoder?

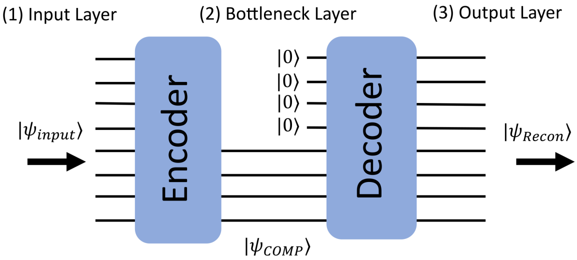
What is a Quantum Autoencoder (QAE)?

- Inspired by classical autoencoder
- Compress a quantum state onto a smaller amount of qubits, while retaining the initial information
- Can be used for the same purpose, i.e., dimentionality reduction
- Copy the structure of the classcial autoencoder?
  - No

What is a Quantum Autoencoder (QAE)?

- Inspired by classical autoencoder
- Compress a quantum state onto a smaller amount of qubits, while retaining the initial information
- Can be used for the same purpose, i.e., dimentionality reduction
- Copy the structure of the classcial autoencoder?
  - No
  - Quantum computations follow a unitary evolution
  - Not possible to eliminate or create new qubits during quantum computation

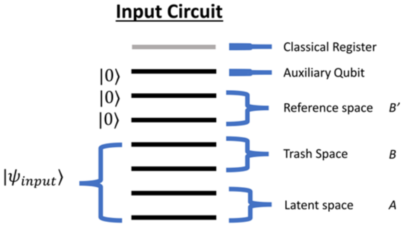
Basic Architecture of QAE



Source: QiskitCommunity, *Quantum Autoencoder Tutorial*

- Input layer: input $|\psi\rangle$ contrains n qubits
- Bottleneck layer: Reduce the dimensionality to $n - k$ qubits
- Output layer: k qubits (all in the state $|0\rangle$) plus the new qubits

Basic Architecture of QAE



Source: QiskitCommunity, *Quantum Autoencoder Tutorial*

- Auxiliary Qubit
  - Aid in certain quantum operations
  - Entangle quantum state with auxiliary qubits
- Reference State
  - A reference for comparison in quantum algorithms
  - Measure deviations in the quantum system during computation
- Classical Register
  - Store and process classical bits

Current Challenges of QAE

- Error Correction
- Decoherence
- Limited Qubits
- Qubit Stability

Use case: Denoising Quantum States using QAE

- Motivation: a possible solution to tackle the quantum noise problem
- Goal: reconstruct Greenberger–Horne–Zeilinger (GHZ) states subject to random bit-flips and small unitary noise.
- What are GHZ states?
  - A certain type of entangled quantum state that involves at least three qubits
  - A M-qubit GHZ state can be defined by

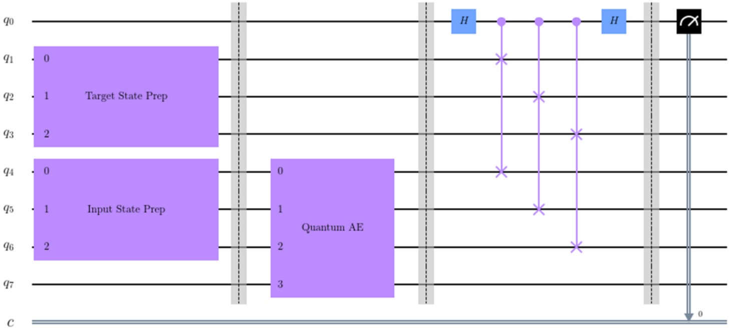
$$|\mathrm{GHZ}\rangle = \frac{|0\rangle^{\otimes M} + |1\rangle^{\otimes M}}{\sqrt{2}}$$

Source: Achache, Horesh, and Smolin, “Denoising quantum states with quantum autoencoders – theory and applications”

Use case: Denoising Quantum States using QAE

- Why do we do this?
  - GHZ states are highly entangled quantum states
  - Random bit-flips introduce errors and decoherence in quantum systems
  - Crucial for developing effective quantum error correction codes

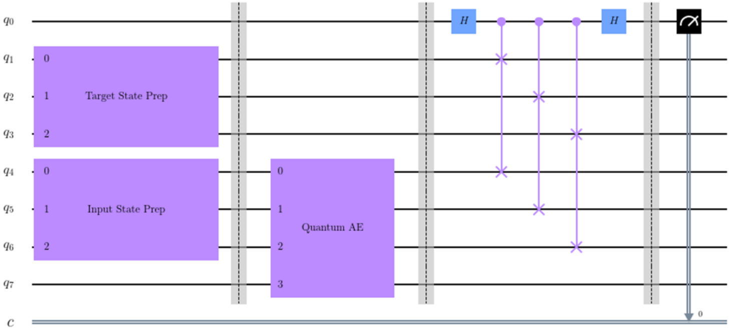
Use Case: Denoising Quantum States using QAE



- Architecture
  - A QNN with ℓ layers, denoted as $[m_1, \ldots, m_\ell]$
  - Each layer has size m_i, $1 \leq i \leq \ell$
  - The quantum circuit is made up of Q qubits, where $Q = 1 + m_1 + w$

Source: Achache, Horesh, and Smolin, "Denoising quantum states with quantum autoencoders – theory and applications"

Use case: Denoising Quantum States using QAE

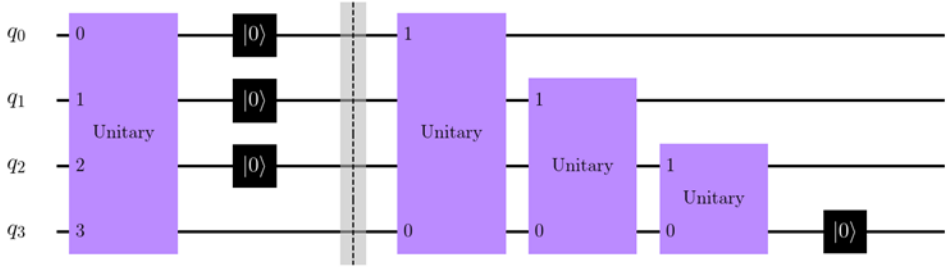


$$F(|\phi_i\rangle, \rho) = \langle\phi|\rho|\phi_i\rangle$$

$$p_0 = \frac{1}{2}(1 + F(|\phi_i\rangle, \rho))$$

Source: Achache, Horesh, and Smolin, "Denoising quantum states with quantum autoencoders – theory and applications"

Use case: Denoising Quantum States using QAE



- Reuse the qubits of the precedent layers by resetting them
- Only need the qubits representing two consecutive layers

Source: Achache, Horesh, and Smolin, “Denoising quantum states with quantum autoencoders – theory and applications”

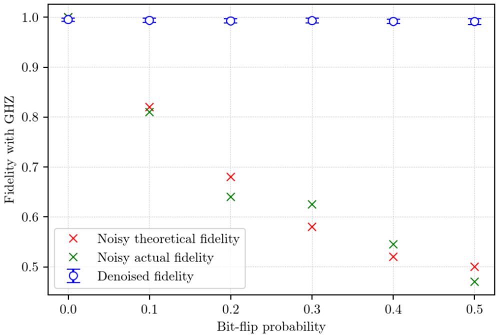
Use case: Denoising Quantum States using QAE

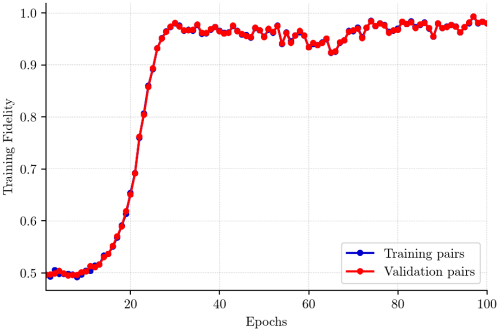
- Training
  - Unsupervised
  - Train on pairs (x, y); x and y are from the same noisy distribution
  - Loss function: $C(\kappa) = \frac{1}{N}\sum_{i=1}^{N} F(|\phi_i\rangle, \rho_i^{\kappa})$

Source: Achache, Horesh, and Smolin, "Denoising quantum states with quantum autoencoders – theory and applications"

Use Case: Denoising Quantum States using QAE

- Results
  - almost perfect





Source: Achache, Horesh, and Smolin, “Denoising quantum states with quantum autoencoders – theory and applications”

Frameworks support QAE out of the box

Tool	Language
Cirq	Python
Qiskit	Python
Dwave-system	Python
FermiLib	Python
Qbsolv	C
QGL.jl	Julia
Qiskit.js	JavaScript
Qrack	C++
Quirk	JavaScript
Strawberry Fields	Python

Source: Ayoade, Rivas, and Orduz, "Artificial Intelligence Computing at the Quantum Level"

Frameworks support QAE out of the box: Qiskit

- Open-source quantum computing framework developed by IBM
- Detailed documentation, tutorials, and examples are available
- Allows users to run quantum circuits on real quantum devices

Building QAE at Home

- Import necessary Qiskit libraries

Python

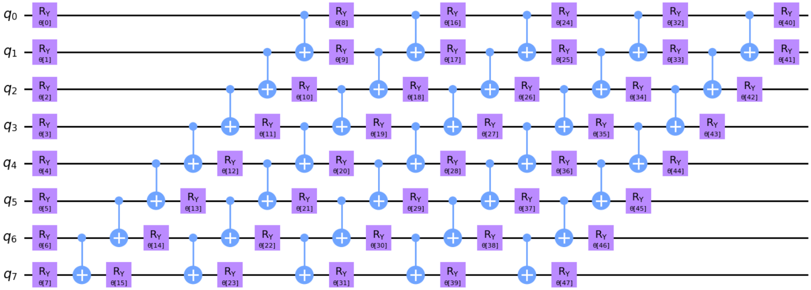
```python
from qiskit import ClassicalRegister, QuantumRegister
from qiskit import QuantumCircuit
from qiskit.circuit.library import RealAmplitudes
from qiskit.quantum_info import Statevector
from qiskit_algorithms.optimizers import COBYLA
from qiskit_algorithms.utils import algorithm_globals
from qiskit_machine_learning.circuit.library import RawFeatureVector
from qiskit_machine_learning.neural_networks import SamplerQNN
```

Source: QiskitCommunity, *Quantum Autoencoder Tutorial*

Building QAE at Home

Python

```python
num_qubits = 8
circ = ansatz(num_qubits)
circ.decompose().draw("mpl")
```



Building QAE at Home

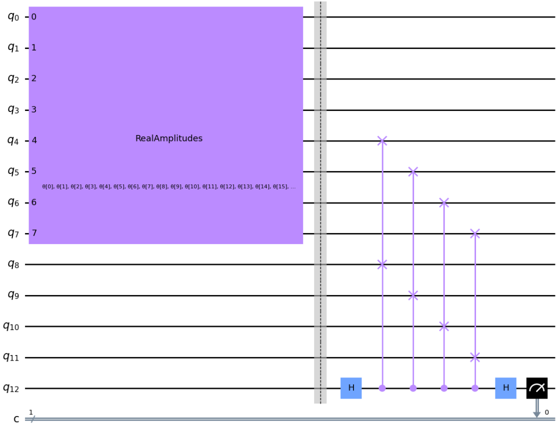
- Define the autoencoder circuit

Python

```python
def auto_encoder_circuit(num_latent, num_trash):
    qr = QuantumRegister(num_latent + 2 * num_trash + 1, "q")
    cr = ClassicalRegister(1, "c")
    circuit = QuantumCircuit(qr, cr)
    circuit.compose(ansatz(num_latent + num_trash), range(0, num_latent +
    ↪  num_trash), inplace=True)
    circuit.barrier()
    auxiliary_qubit = num_latent + 2 * num_trash
```

Source: QiskitCommunity, *Quantum Autoencoder Tutorial*

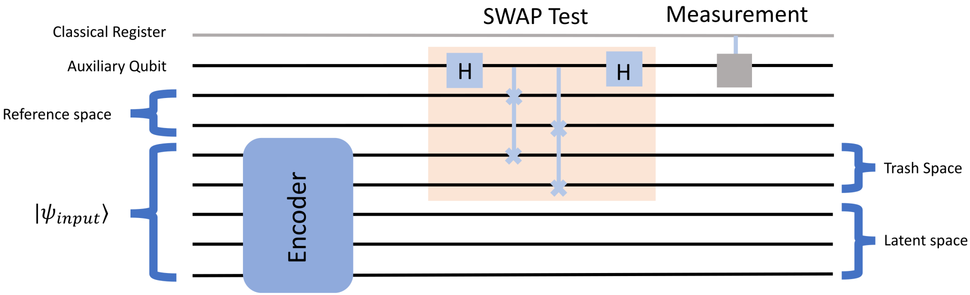
Building QAE at Home



- q_0 - q_3: Laten space
- q_4 - q_7: Trash space
- q_8 - $q_1$1: Reference space
- q_{12}: Auxiliary qubit
- c: Classical register

Source: QiskitCommunity, *Quantum Autoencoder Tutorial*

Building QAE at Home



Source: QiskitCommunity, *Quantum Autoencoder Tutorial*

Building QAE at home

- Maximizing this function corresponds to two states being identical

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

Python

```python
# swap test
circuit.h(auxiliary_qubit)
for i in range(num_trash):
    circuit.cswap(auxiliary_qubit, num_latent + i, num_latent +
    ↪  num_trash + i)
circuit.h(auxiliary_qubit)
circuit.measure(auxiliary_qubit, cr[0])
return circuit
```

Building QAE at Home: Domain Wall

- Apply the X gate (bit-flip)

Python

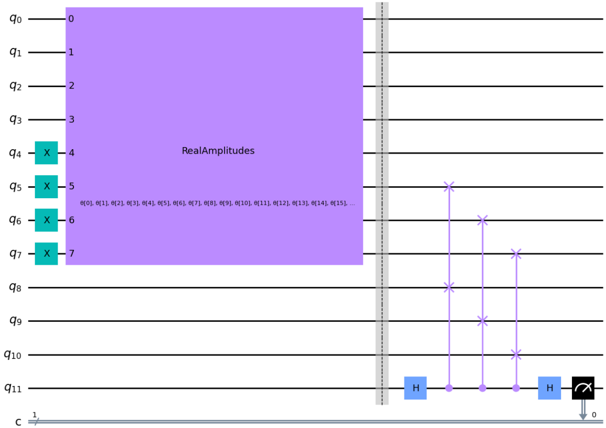
```python
def domain_wall(circuit, a, b):
    # Here we place the Domain Wall to qubits a - b in our circuit
    for i in np.arange(int(b / 2), int(b)):
        circuit.x(i)
    return circuit

ae = auto_encoder_circuit(num_latent, num_trash)
qc = QuantumCircuit(num_latent + 2 * num_trash + 1, 1)
qc = qc.compose(domain_wall_circuit, range(num_latent + num_trash))
qc = qc.compose(ae)
qc.draw("mpl")
```

Source: QiskitCommunity, *Quantum Autoencoder Tutorial*

Building QAE at Home: Domain Wall

- Initialize Qubits
  - Start with 8 qubits, initially set to the $|0\rangle$ state.
- Apply X-Gates
  - Apply an X-gate (bit-flip) operation on qubits 1 through 4.
  - This flips the states to $|1\rangle$, representing the "domain wall."

Building QAE at Home: Domain Wall



Source: QiskitCommunity, *Quantum Autoencoder Tutorial*

Building QAE at Home: Domain Wall

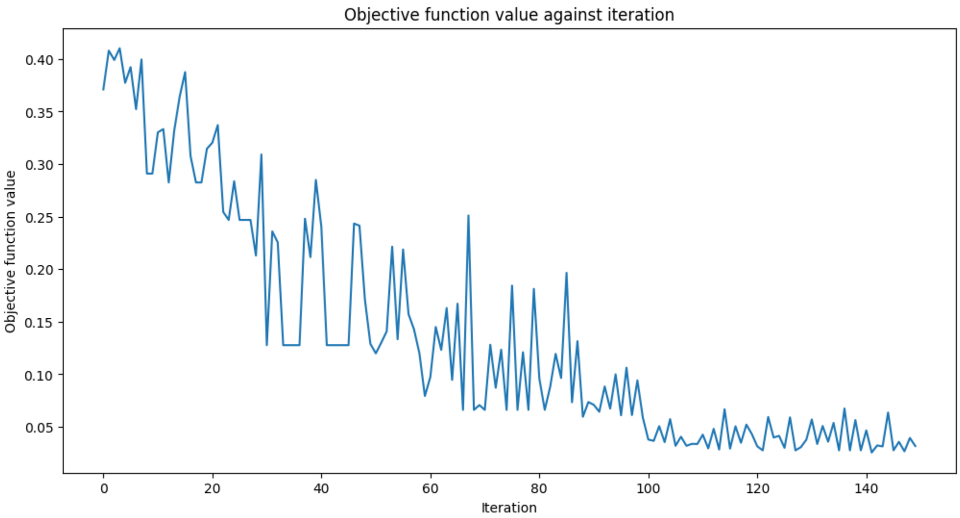
- Define loss function

Python

```
def cost_func_domain(params_values):
    probabilities = qnn.forward([], params_values)
    # we pick a probability of getting 1 as the output of the network
    cost = np.sum(probabilities[:, 1])
```

Source: QiskitCommunity, *Quantum Autoencoder Tutorial*

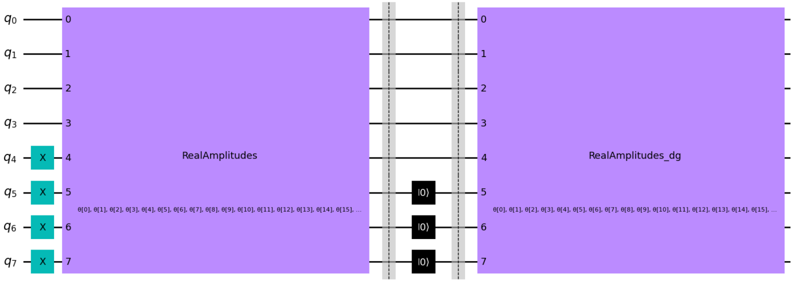
Building QAE at Home: Domain Wall



Building QAE at Home: Domain Wall

Python

```python
test_qc.reset(7)
test_qc.reset(6)
test_qc.reset(5)
```



Building QAE at Home: Domain Wall

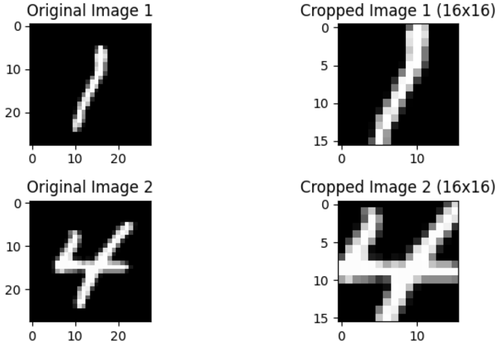
- Compare final fidelity between input and output

Python

```python
domain_wall_state = Statevector(domain_wall_circuit).data
output_state = Statevector(test_qc).data

fidelity = np.sqrt(np.dot(domain_wall_state.conj(), output_state) ** 2)
```

  - Fidelity of our Output State with our Input State: 0.9740570467513804

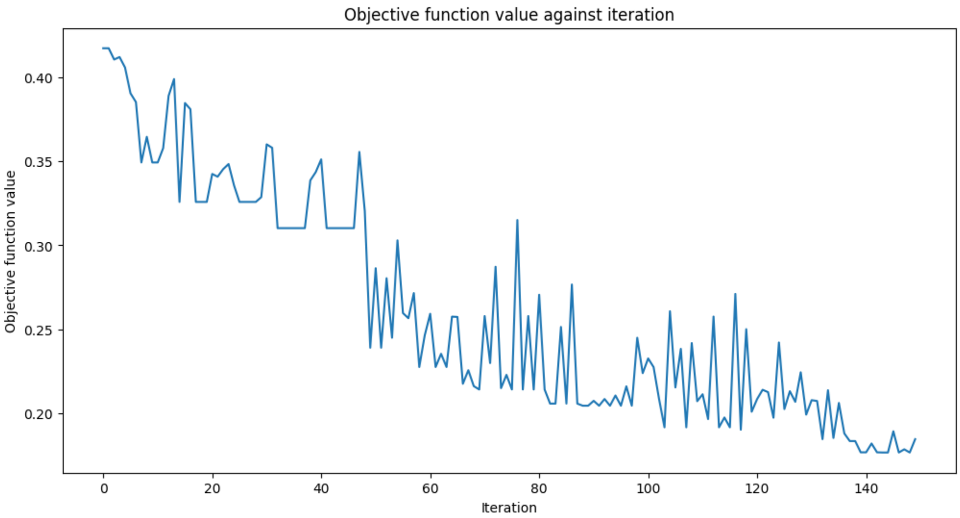
Building QAE at Home: Image Compression



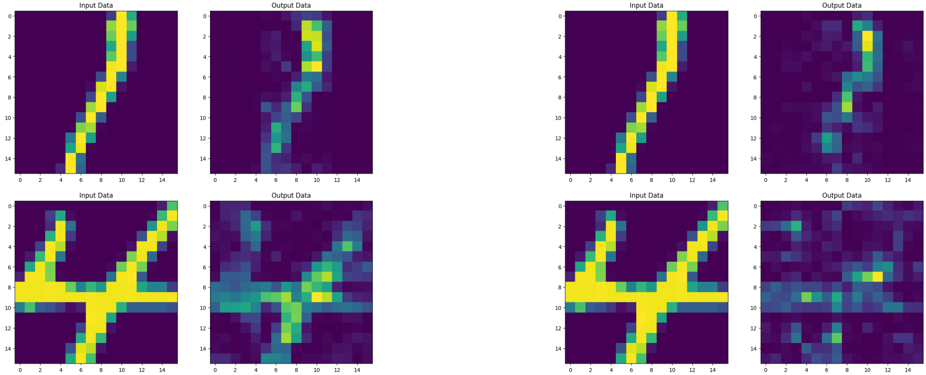
- Random image from MNIST
- Original size: 28×28
- Cropped size: 16×16

Building QAE at Home: Image Compression

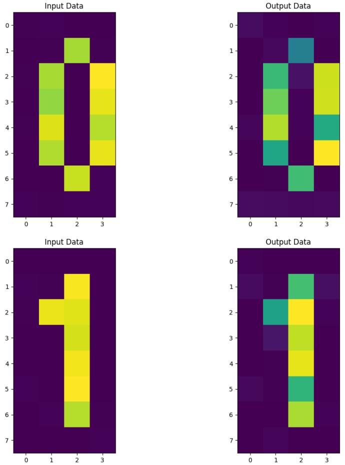
Building QAE at Home: Image Compression



Building QAE at Home: Image Compression



Building QAE at Home: Image Compression



- Results with smaller images
  - Image size: 4×8
  - Number of qubits: 5
  - Laten space: 3
  - Trash space: 2

Source: QiskitCommunity, *Quantum Autoencoder Tutorial*

Main Takeaways

- Push the boundaries of machine learning capabilities
- Applications: optimization, data compression, and data denoising
- Integrate into industry: the key to the full power of quantum computing
- Ongoing research promises enhanced efficiency and scalability
- Out-of-the-box frameworks like Qiskit facilitate experimentation

Conclusion

"*Nature isn't classical, and if you want to make a simulation of Nature, you'd better make it quantum mechanical*"

—- Richard Feynman

References

Achache, Tomer, Lior Horesh, and John Smolin. "Denoising quantum states with quantum autoencoders – theory and applications". In: (Dec. 2020). arXiv: 2012.14714 [quant-ph]. URL: https://arxiv.org/abs/2012.14714.

Ayoade, O., P. Rivas, and J. Orduz. "Artificial Intelligence Computing at the Quantum Level". In: *Data* 7 (2022), p. 28. URL: http://link.aip.org/link/?RSI/72/4477/1.

Berggren, Karl. "Quantum Computing with Superconductors". In: *Proceedings of the IEEE* 92 (2004), pp. 1630–1638. DOI: 10.1109/JPROC.2004.833672.

Ezratty, Olivier. "Is there a Moore's law for quantum computing?" In: (2023).

Moreno-Pineda, Eufemio et al. "Molecular Spin Qudits for Quantum Algorithms". In: *Chemical Society Reviews* 47 (2017). DOI: 10.1039/C5CS00933B.

QiskitCommunity. *Quantum Autoencoder Tutorial*. Qiskit Machine Learning Repository. 2023. URL: https://github.com/qiskit-community/qiskit-machine-learning/blob/stable/0.7/docs/tutorials/12_quantum_autoencoder.ipynb.

Rao, A. "Learning hard distributions with quantum-enhanced Variational Autoencoders". In: (May 2023). arXiv: 2305.01592 [quant-ph].