GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN IN PUBLICA COMMODA SEIT 1737

HPS

Seminar Report

Quantum Neural Networks: Libraries and Applications

An Example of Quantum Autoencoder

Zhuojing Huang

MatrNr: 12305011

Supervisor: Dr. Christian Boehme

Georg-August-Universität Göttingen

March 31, 2024

Abstract

The seminar report explores the field of Quantum Neural Networks (QNNs) with a specific focus on Quantum Autoencoders (QAEs). The introduction part of the report outlines the fundamental concepts of quantum computing, emphasizing the unique properties of qubits and quantum gates, essential for understanding QAEs' operation. In the following subsections of the introduction, basics of QAEs are also covered, as well as its advantages in the domain of High-Performance Computing (HPC). Apart from the bright side, the report also discusses existing challenges in realizing the full potential of QNNs. In the second section, through two detailed use cases-denoising Greenberger–Horne–Zeilinger states and real-life industrial applications-the report underscores QAEs' practical utility and efficiency in data compression and representation tasks. These examples highlight QAEs' ability to mitigate noise-induced errors in quantum systems and accurately compress complex datasets for classification tasks, demonstrating their potential to advance quantum machine learning research and real-world applications. In the next section, the report provides a simple tutorial for individuals interested in exploring QAEs using the Qiskit framework from IBM, showing the accessibility and resources available for newcomers to delve into quantum computing research. In the end, the report reemphasize the potential of the QNNs, especially QAEs, and provide outlook of the field.

Declaration on the use of ChatGPT and comparable tools in the context of examinations

In this work I have used ChatGPT or another AI as follows:

- ☐ Not at all
- ☐ During brainstorming
- ☐ When creating the outline
- ☑ To write individual passages, altogether to the extent of 10% of the entire text
- ☐ For the development of software source texts
- ☐ For optimizing or restructuring software source texts
- ☑ For proofreading or optimizing
- ☐ Further, namely: -

I hereby declare that I have stated all uses completely.
Missing or incorrect information will be considered as an attempt to cheat.

Contents

List of Tables

List of Figures

List of Abbreviations

GHZ Greenberger–Horne–Zeilinger

HPC High-Performance Computing

QAE Quantum Autoencoder

QNN Quantum Neural Network

1 Introduction

Driven by its remarkable potential, quantum computing is experiencing a growth in attention from both academia (Figure 1[ARO22]) and general public. The field is also rapidly advancing thanks to contributions from top enterprises, research institutions, startups, and well-resourced organizations worldwide [ARO22]. Among all the subfields within quantum computing, quantum machine learning stands out for its popularity, particularly fuelled by the upsurge in artificial intelligence in recent years. In this landscape, quantum autoencoder emerges as a particularly intriguing concept, as it offers a promising efficient approach to data representation and compression. Based on the above reasons, this seminar report provides an overview of quantum neural network, with a special focus on quantum autoencoder. Additionally, it highlights available resources to facilitate entry into the field.

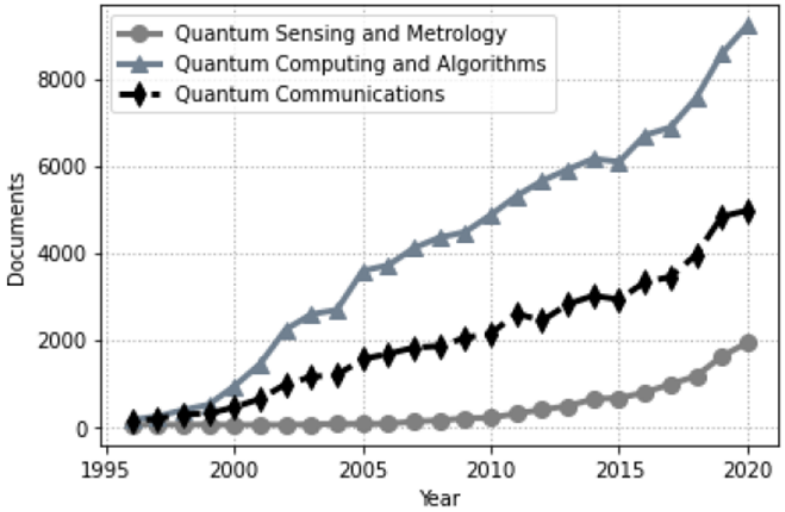

Figure 1: Publication growth in quantum-related fields

1.1 Basic Concepts of Quantum Computing

Before diving in quantum computing, it is important to know its foundation theory – quantum mechanics, as it is the framework for understanding quantum phenomena [Mar+18]. According to quantum mechanics, objects, prevalently limited to subatomic particles, have wave-like qualities [Mar+18]. It has however been forecasted that quantum mechanics will address computational challenges across various complex problems in domains like chemistry, physics, and mathematics [BFD19]. Quantum computing is therefore the result of the combination of quantum mechanics and information science.

Comparing to the classical systems, where binary bits are the cornerstone of computing, quantum computing works on the foundational unit of quantum bit, or **qubit**, which can be in one of two states and possibly a **superposition** of the two states, meaning a linear combination of 0 and 1 [ARO22]. Moreover, they can exhibit **entanglement**, where the values of specific qualities of one system are correlated with the values of the corresponding properties of the other system [Bia+17] —as famously described by Albert Einstein as "spooky action at a distance."

A qubit is most represented by a mathematical equation as follows:

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

Where α, β are complex numbers and $|0\rangle$, $|1\rangle$ are computational basis states that form an orthonormal basis in this vector space [ARO22]. A qubit can also be represented visually using a Bloch sphere (Figure 2 [ARO22]) which gives the angles and basis vectors for the $|\psi\rangle$ representation:

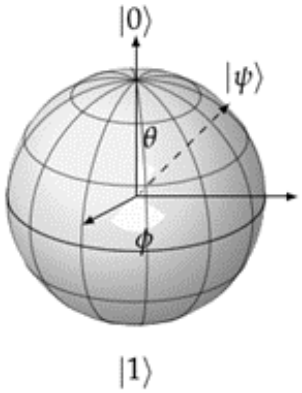

Figure 2: A Bloch sphere: it visualizes the qubit's geometric state

Another crucial aspect of quantum computing involves **quantum circuits** and **quantum gates**. A quantum circuit serves as a depiction of a sequential quantum operation. Logic qubits are conveyed along "wires" (illustrated by horizontal lines in Figure 3 [ARO22]), while quantum gates (depicted as blocks) manipulate the qubits within a standard quantum circuit, as shown in Figure 3 [ARO22].

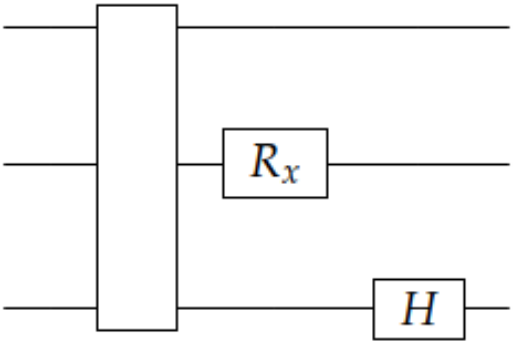

Figure 3: Quantum circuits and quantum gates

1.2 Current Situation of Quantum Computing

The drive to develop quantum computers capable of executing Shor's algorithm [Sho94] for large numbers has been a major force behind advancements in quantum computing. But generally, it's anticipated that quantum computers will also greatly benefit optimization problems, which are crucial in various fields such as defence, cryptography, financial trading, etc [Gam19]. Figure 4 [Ber04] shows the relative advantage of quantum computing when the task size exceeds a certain threshold.

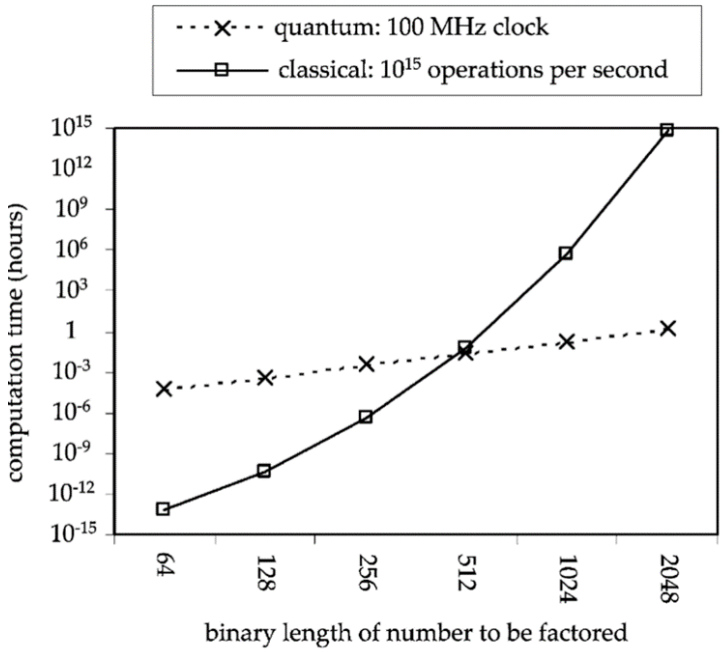

Figure 4: Computation times of classical and quantum computer

Figure 5 [Ezr23] shows how many qubits that the main companies in the field have at the time being.

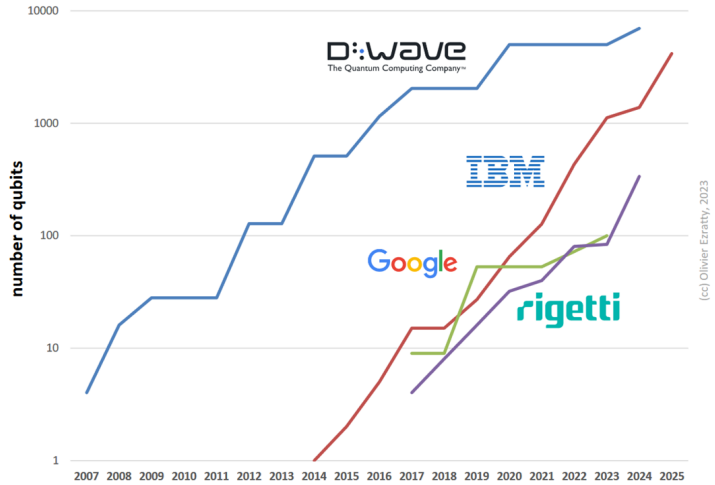

Figure 5: Number of qubits in major quantum companies

However, it's crucial to recognize that quantum computers are likely to provide significant speed-ups for specific types of problems only [Gam19]. And it's also essential to understand that quantum computing isn't about replacing classical computing but rather complementing it, addressing areas where classical methods may fall short [ARO22]. Despite the promising potential of quantum computing, several challenges must be addressed to realize its full capabilities. These challenges include error correction, decoherence, limited qubits, and qubit stability, which are inherent to quantum computing systems. Existing research has already tried to address some of the problems and there will be examples given later in the report.

1.3 Quantum Autoencoder and HPC

The pursuit of more efficient and powerful computational models has led to the emergence of Quantum Neural Networks (QNNs), a novel approach that combines the principles of quantum computing with neural network architectures [ARO22]. QNNs extend the capabilities of classical neural networks by leveraging quantum computing's inherent parallelism and entanglement. At its core, QNNs aim to process and manipulate quantum data for tasks such as classification, regression, and optimization. As QNNs cover a lot of different types of neural networks, this report is only going to focus on one of them, that is, Quantum Autoencoders (QAEs).

Building upon the foundations laid by classical autoencoders and QNNs, QAEs introduce unique challenges and opportunities in high-performance computing (HPC). Similar to classical autoencoders (Figure 6 [Bar23]), where compressed data from encoder aim to be efficiently and faithfully rebuilt to the original information through the use of decoder, the fundamental idea behind a QAE is to compress a quantum state onto a smaller number of qubits while preserving the original information content [Qis23]. This compression enables efficient representation and manipulation of quantum data.

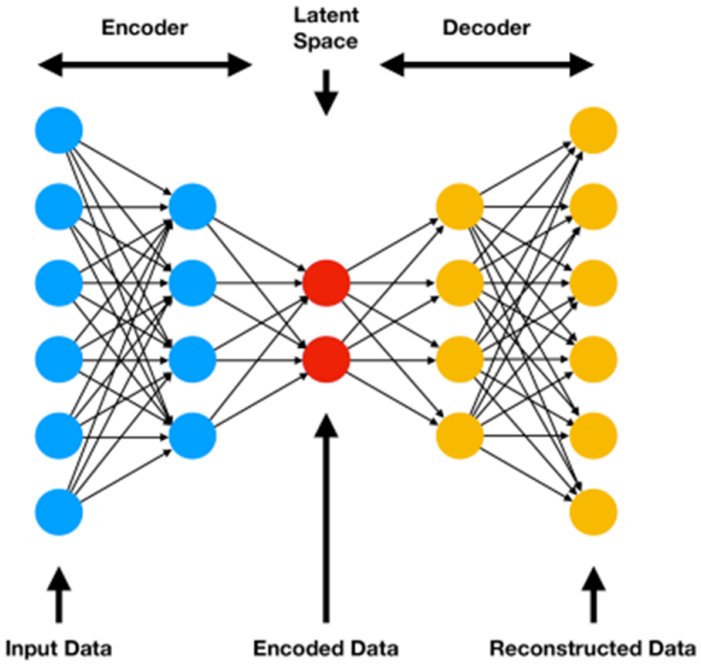

Figure 6: Architecture of a classical Autoencoder

However, the architecture of QAE (Figure 7 [Qis23]) differs significantly from classical autoencoders due to the constraints imposed by quantum computing: to be more specific, the unitary nature of quantum encoding make it impossible to eliminate or create new qubits during the process [Qis23].

The basic unit of a QAE usually composes of three parts, Input layer where input $|0\rangle$ contains n qubits, bottleneck layer to reduce the dimensionality to $n-k$ qubits, and output layer where k qubits (all in the state $|0\rangle$) plus the new qubits. Among all the n qubits, the k qubits that are set to $|0\rangle$ are referred to as the trash space and the rest as the latent space. Apart from the input $|\psi\rangle$, there are also other important subsystems: **reference space**, **auxiliary qubits** and a **classical register** (Figure 8 [Qis23]). In quantum computing, auxiliary qubits serve as crucial components that aid in specific quantum

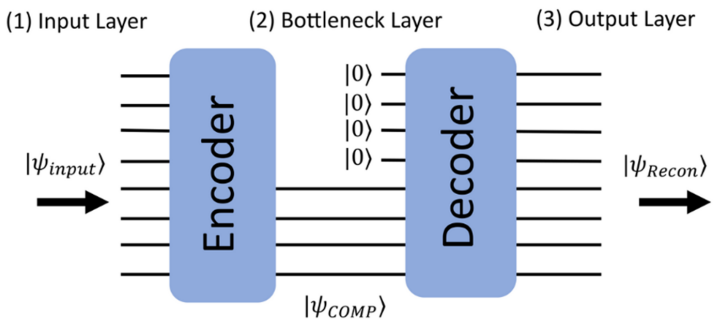

Figure 7: Architecture of a Quantum Autoencoder

operations by entangling with other qubits. They play a significant role in enhancing computational processes by facilitating entanglement and enabling various quantum operations. Reference states are utilized as benchmarks or standards for comparison within quantum algorithms, allowing for the measurement of deviations in the quantum system during computation. This comparison helps assess the progress and accuracy of computations. Meanwhile, classical registers are employed to store and process classical bits, providing a bridge between classical and quantum computing paradigms [Qis23]. In later sections of the report, there will be examples of how these subsystems work, which will make their functions easier to understand.

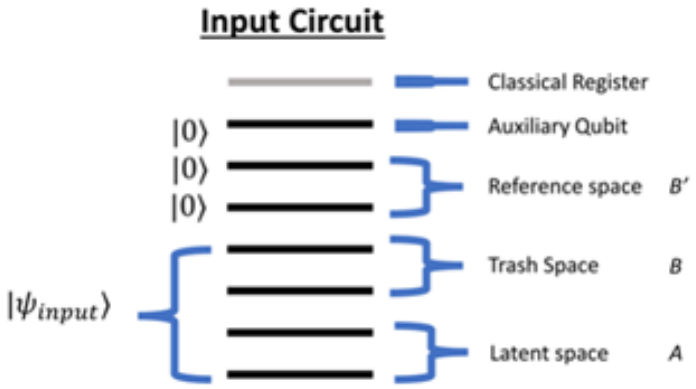

Figure 8: Quantum autoencoder input circuit

2 Use Cases of QAEs

Despite that QAEs are still at its early stage in practice, there are some attempts that reveal the potential of the field. In this section, I will briefly introduce two use cases: one is resolving the problem of noisy states of quantum data by using QAE, the other is a successful application of QAEs in industry.

2.1 Use Case 1: Denoising of Greenberger–Horne–Zeilinger States

In the pursuit of addressing the challenges posed by quantum noise in quantum systems, QAEs present a promising solution. One such use case involves the denoising of Greenberger–Horne–Zeilinger (GHZ) states subjected to random bit-flips and small unitary

noise [AHS20]. GHZ states, characterized by high levels of entanglement among at least three qubits, serve as valuable resources in quantum information processing [AHS20]. However, quantum noise, induced by random bit-flips and unitary noise, can degrade the fidelity of GHZ states, posing significant challenges in quantum error correction and quantum computation. The goal of utilizing QAE in this context is to reconstruct the original GHZ states from noisy versions, thereby mitigating the effects of quantum noise [AHS20].

This study by Achache et al. [AHS20] proposed a QAE architecture for such purpose. The architecture consists of a QNN with l layers, each denoted as $[m_1, \ldots, m_\ell]$. The quantum circuit comprises Q qubits, where $Q = 1 + m_1 + w$, with w being the width of the QNN, and employs techniques to reuse qubits from previous layers, optimizing resource utilization. The architecture of the QAE is shown in Figure 9 [AHS20]. Here, $w = 4$ and $Q = 8$.

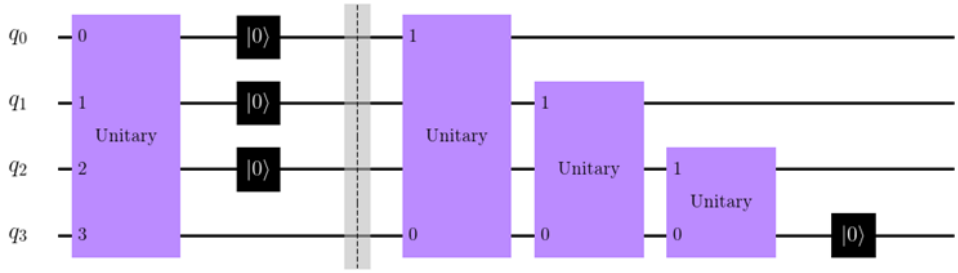

Figure 9: Circuit for the training of a [3,1,3] QAE

During training, the QAE operates in an unsupervised manner, learning from pairs of noisy and clean quantum states. The loss function is defined as the fidelity between the reconstructed state and the original state, quantifying the accuracy of denoising. Experimental results from the study demonstrate the effectiveness of QAEs in denoising GHZ states, achieving near-perfect reconstruction despite the presence of quantum noise (Figure 10 [AHS20]). These findings underscore the potential of QAE as valuable tools in mitigating noise-induced errors in quantum systems, thereby advancing the reliability and robustness of quantum computation and communication protocols.

2.2 Use case 2: QAE and classifier

The second use case is introduced by the paper *Quantum neural network autoencoder and classifier applied to an industrial case study*[Man+22]. Based on the title, the authors of the paper apply QAEs in real-life scenario, in particular using actual data from physical plants instead of pedagogical data from benchmark datasets, which indicates the potential of QNNs outside of academia.

The data of the paper is from a type of vessel that is designed to separate three output streams—water, oil and gas—from the input of crude oil (as indicted in Figure 11 [Man+22]). The separator is regulated with three controllers: a pressure controller for the gas stream, and controllers for the water and the oil stream. In an ideal machine learning scenario, as many variables as possible, including other upstream controllers, should be considered to reach a higher accuracy when predicting the behaviour of the separator—whether it is working normally or faultily. However, due to the limitation in available qubits and the complexity of the problem, the paper only considers four variables

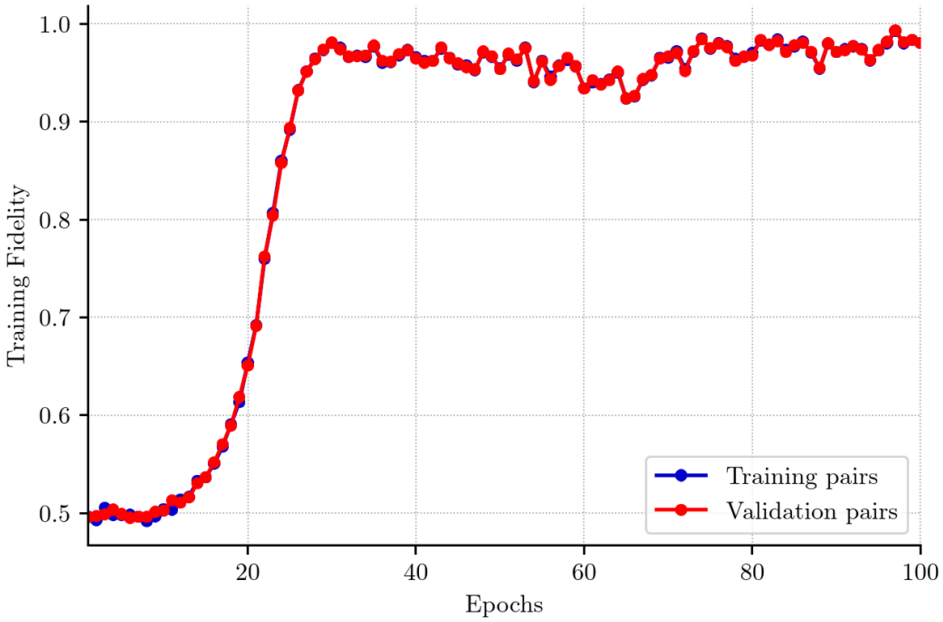

Figure 10: Training fidelity of the QAE

when training the model: the oil level, the oil output flow, the pressure and the opening of the oil output valve. Nevertheless, measurements coming from tens of sensors not not only include their instantaneous values, but also additional features such as moving averages, and minimal or maximal values trends, which leads a situation where too many input variables are available. Therefore, it is ineffective to directly feed all the available data into the neural network classifier. One strategy is to use a dimensionality reduction approach. Autoencoder has been improved to be an effective way for such purposes in classical neural networks.

Since the data form the industry is classical by default, the first step of the project is to encode such data on a quantum state to be later fed into a QAE. In this use case, the authors choose to use a phase encoding strategy, which provides an effective way to load classical data into a quantum state [Tac+19][Man+20]. According to this method, given a data sample $\mathbf{x} = (x_1, x_2, \ldots, x_N) \in \mathbb{R}^N$, this is encoded on the quantum state of $n = \log_2 N$ qubits as follows:

$$|\psi\rangle = \sum_{i=1}^{2^n} e^{ix_i}|i\rangle$$

The authors of the paper assessed the performance of their QAE by fully simulating the wavefunction, enabling us to gauge the average reconstruction error at approximately 5%, which is nearly identical to that of a classical autoencoder. This confirms the quantum autoencoder's ability to efficiently store a condensed version of the original dataset and recover it afterward. Moreover, they verified the accuracy of the quantum autoencoding process by assessing the quantum fidelity between the original and decoded quantum states, even in the presence of simulated stochastic measurement noise, finding them to be closely similar. After determining the optimal parameters for the quantum autoencoder during the training phase, the authors utilized the compressed quantum state as input for a quantum classifier to undertake a binary classification task. This algorithm achieved an accuracy exceeding 87%, comparable to the classical approach utilizing a neural network autoencoder followed by a nearest-neighbour classifier [Man+22]. This indicates once more the quantum algorithm's proficiency in accurately compressing pertinent data

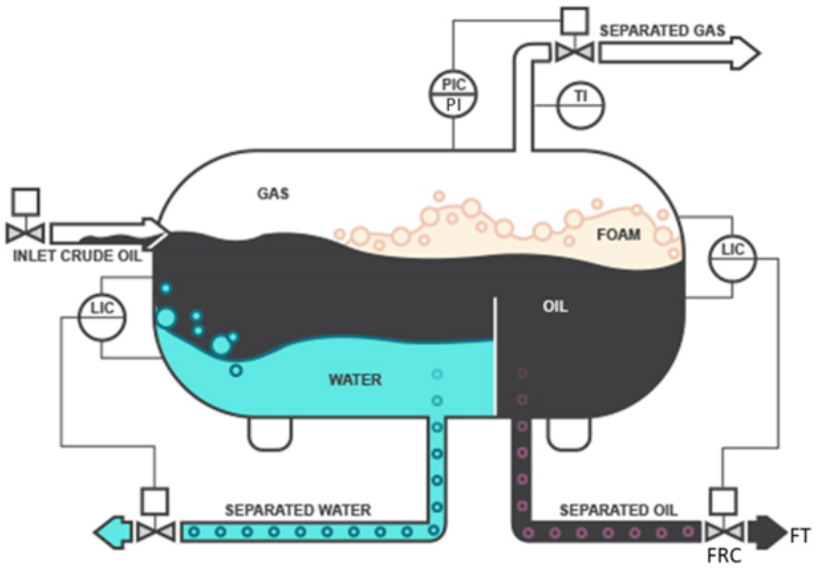

Figure 11: Industrial separator

information. Furthermore, the paper evaluated the performance of the entire quantum pipeline, comprising the quantum autoencoder and the classifier, on existing IBM quantum hardware, attaining a classification accuracy of 82%, only slightly lower than the ideal outcome [Man+22].

This paper marks a significant advancement in quantum machine learning, espicially QAEs, as it represents one of the first cases where both quantum computing software and hardware are directly utilized to examine genuine data sourced from industry.

2.3 Brief overview of Other Use Cases

Apart from the aforementioned use cases, the field of QAEs has also witnessed a growing popularity regarding scientific publication in recent years. In this section, I will give some more examples of recent research.

Being one of the first QAE paper, Romero et al. [ROA17] present an illustration of a straightforward circuit capable of being trained as a proficient autoencoder. Their model is employed within the domain of quantum simulation to condense ground states of the Hubbard model and molecular Hamiltonians. Based on former works, Srikumar et al.'s work [SHH21] introduces a unique approach using a hybrid quantum autoencoder to extract information from quantum states and represent it in a classical space. By training on pure states, this variational algorithm learns to identify key characteristics, enabling new methods for clustering and semi-supervised classification. The paper [SHH21] also demonstrates the hybrid quantum autoencoder's effectiveness with amplitude encoded states, with potential extensions to analyse complex quantum datasets. Another different type of QAE is quantum-enhanced variational autoencoder(QeVAE) [Rao+23]. The QeVAE combines quantum correlations to enhance fidelity without exponentially increasing parameters. The QeVAE paper [Rao+23] derives the QeVAE's output distributions and demonstrate its superiority over classical models across various quantum state classes, achieving over 2x fidelity improvement in some cases. Furthermore, the QeVAE model outperforms classical counterparts when implemented on the IBMq Manila quantum computer. This work [Rao+23] opens avenues for quantum generative learning algorithms and

understanding measurement distributions of high-dimensional quantum states. It is noteworthy that this overview of current works is far from complete. The purpose of this section is to serve as a window for the fast developing field of QAE.

3 Tutorial for Building QAEs

Although QNNs, or quantum computing in general, sounds obscure and difficult to be implemented for those who are not yet experts in the field, people can actually utilise various off-the-shelf frameworks in quantum computing. Figure 12 [ARO22] shows some of the open-source quantum computing tools and their corresponding languages. Among all these frameworks, Qiskit [Qis23], the quantum computing framework developed by IBM, stands out for its comprehensive documentation, tutorials, and examples, making it accessible for both beginners and experienced users. This report will only cover basic tutorials for building QAEs using Qiskit.

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

Figure 12: Off-the-shelf quantum libraries

Building a QAE using Qiskit involves several steps [Qis23]. In the following paragraphs I will offer a breakdown of the process:

1. Importing Libraries: Start by importing the required Qiskit libraries, including components for constructing quantum circuits, neural networks, and optimizers.

```
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
```

2. Defining the Quantum Circuit: Define the quantum circuit for the autoencoder, including the input, latent, trash, and reference spaces, along with auxiliary qubits. The reference state is not explicitly shown but is as the same number of our trash state. In this experiment, I chose to use 8 qubits to build my QAE by using this prebuilt ansatz function.

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

```
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
```

3. Calculate the loss: the SWAP test is a procedure commonly used to compare two states by applying CNOT gates to each qubit [Qis23]. By running the circuit M times, and applying the SWAP test, we then measure the auxiliary qubit. We use the number of states in the state 1 to compute:

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

where L is the count for the states in the 1 state. So, maximizing this function corresponds to the two states of which we are comparing being identical. We therefore aim to maximize this function, i.e. minimize $\frac{2}{M} \cdot L$. This value will therefore be our cost function. The SWAP test can be realised by the code listed below.

```
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
```

4. Application: in this step, I will show you how to apply QAEs to real-life scenarios. In total, I tried two tasks, which I showed during the presentation—1) Domain Wall; 2) Compressing and reconstructing MNIST images. In this report, I will only cover the second example. In the task of compressing and reconstructing MNIST, I randomly chose

two images from the MNIST dataset and cropped them into smaller size while keeping the important information of the image.

Since the total number of qubits is limited, I tried the hyperparameter of 6 input spaces and 2 trash spaces. Initially, the necessary quantum circuits are set up, including the feature map circuit (**fm**) responsible for encoding classical data into quantum states and the autoencoder circuit (**ae**) designed to compress and reconstruct the encoded data. These circuits are then combined within a larger QuantumCircuit (**qc**) that encompasses the entire computation, from input and latent qubits to trash qubits, along with an additional qubit for the output. The QNN is then is configured with the quantum circuit, input and weight parameters, an interpretation function, and the desired output shape (Figure 13).

Figure 13: QAE circuit for MNIST

Next, a cost function is defined to compute the probabilities of class 1 based on the output of the QNN, which is subsequently optimized using the COBYLA algorithm. Throughout the optimization process, the objective function values are tracked and plotted for visualization. After optimization, the code defines functions for generating datasets of images and preparing test quantum circuits for evaluating the trained model on new images. Finally, the performance of the model is assessed by sampling new images, computing their original and reconstructed state vectors, and visualizing the results . The reconstruct result can be seen in Figure 14. With limited qubits, the reconstructed images are not perfect yet going to the right direction. Therefore, with more quantum computing power, the task is expected to be solved without problem.

```
def cost_func_digits(params_values):
    probabilities = qnn.forward(train_images, params_values)
    cost = np.sum(probabilities[:, 1]) / train_images.shape[0]
return cost
opt = COBYLA(maxiter=150)
objective_func_vals = []
```

```
test_qc = QuantumCircuit(num_latent + num_trash)
test_qc = test_qc.compose(fm)
ansatz_qc = ansatz(num_latent + num_trash)
test_qc = test_qc.compose(ansatz_qc)
test_qc.barrier()
test_qc.reset(7)
test_qc.reset(6)
test_qc.barrier()
test_qc = test_qc.compose(ansatz_qc.inverse())
```

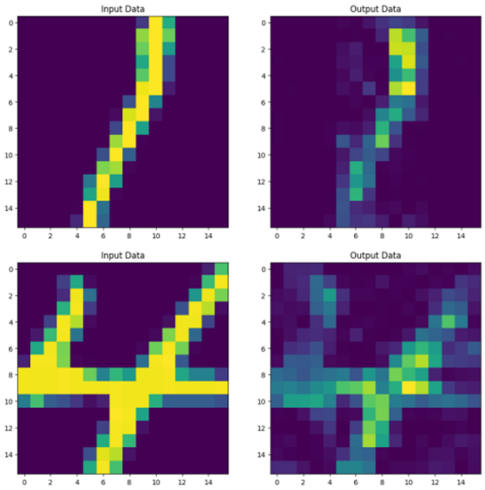

Figure 14: Reconstructed MNIST images

4 Conclusion

To wrap up, this seminar report discusses about the state-of-the-art in the field of QNNs with a specific lens on QAEs, as well as their advantages regarding HPC. Throughout the report, I have offered the foundational concepts of quantum computing, including qubits, superposition, entanglement, etc. Moreover, I also introduced existing research of applying QAEs in both academia and industry, such as denoising GHZ states and classifying data from industrial separator, illustrating how QAEs efficiently tackle data compression and representation tasks. These instances present the robustness of QAEs in handling noise-induced errors and compressing intricate datasets accurately.

While celebrating these advancements, we should not that there are still issues, e.g., the limited number of existing qubits, hindering the full potential of QNNs. It's essential to acknowledge the existing challenges for future innovative solutions and advancements.

Finally, for those eager to embark on their journey into QAEs or quantum computing in general, the report offers a brief tutorial utilizing the Qiskit framework from IBM, which shows the accessibility and wealth of resources available of the quantum community for researchers and enthusiasts.

References

[AHS20] Tomer Achache, Lior Horesh, and John Smolin. "Denoising quantum states with quantum autoencoders – theory and applications". In: (2020). arXiv: 2012.14714 [quant-ph]. URL: https://arxiv.org/abs/2012.14714.

[ARO22] O. Ayoade, 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.

[Bar23] N. Barla. *Representation Learning with Autoencoder*. 2023. URL: https://neptune.ai/blog/representation-learning-with-autoencoder (visited on 03/31/2024).

[Ber04] Karl Berggren. "Quantum Computing with Superconductors". In: *Proceedings of the IEEE* 92 (2004), pp. 1630–1638. DOI: 10.1109/JPROC.2004.833672.

[BFD19] J. Biamonte, M. Faccin, and M. De Domenico. "Complex networks from classical to quantum". In: *Commun. Phys.* 2 (2019), p. 53.

[Bia+17] J. Biamonte et al. "Quantum machine learning". In: *Nature* 549 (2017), pp. 195–202.

[Ezr23] Olivier Ezratty. *Is there a Moore's law for quantum computing?* 2023.

[Gam19] S. Gamble. *Quantum Computing: What It Is, Why We Want It, and How We're Trying to Get It*. National Academy of Engineering. 2019. URL: https://www.ncbi.nlm.nih.gov/books/NBK538701/.

[Man+20] S. Mangini et al. "Quantum computing model of an artificial neuron with continuously valued input data". In: *Machine Learning: Science and Technology* 1 (2020), p. 045008.

[Man+22] S. Mangini et al. "Quantum neural network autoencoder and classifier applied to an industrial case study". In: *Quantum Machine Intelligence* 4.2 (2022). DOI: 10.1007/s42484-022-00070-4.

[Mar+18] A. Marais et al. "The future of quantum biology". In: *J. R. Soc. Interface* 15 (2018), p. 20180640.

[Qis23] 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+23] A. Rao et al. *Learning hard distributions with quantum-enhanced Variational Autoencoders*. 2023. DOI: 10.48550/arxiv.2305.01592. arXiv: arXiv:2305.01592 [quant-ph].

[ROA17] J. Romero, J. P. Olson, and A. Aspuru-Guzik. "Quantum autoencoders for efficient compression of quantum data". In: *Quantum Science and Technology* 2.4 (2017), p. 045001. DOI: 10.1088/2058-9565/aa8072.

[SHH21] M. Srikumar, C. D. Hill, and L. C. L. Hollenberg. "Clustering and enhanced classification using a hybrid quantum autoencoder". In: *Quantum Science and Technology* 7.1 (2021), p. 015020. DOI: 10.1088/2058-9565/ac3c53.

[Sho94] P.W. Shor. "Algorithms for quantum computation: Discrete logarithms and factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 1994, pp. 124–134. ISBN: 0818665807. DOI: 10.1109/sfcs.1994.365700.

[Tac+19] F. Tacchino et al. "An artificial neuron implemented on an actual quantum processor". In: *npj Quantum Information* 5 (2019), p. 26.