International Journal of Engineering Applied Sciences and Technology, 2021
Vol. 6, Issue 6, ISSN No. 2455-2143, Pages 311-319
Published Online October 2021 in IJEAST (http://www.ijeast.com)

A METHODOLOGY FOR ENCODER USING DATA COMPRESSION AND OBFUSCATION TECHNIQUES IN QUANTUM COMPUTING

Lokesh B S,[1] Hariharan,[2] Chaitanya,[2] Karthik U G,[2]Rakshith P[2,]
[1]Assistant Professor, Dept. of ECE, MIT Mysore, Karnataka, India
[2]Student, Dept. of ECE, MIT Mysore, Karnataka, India

Abstract - Classical data are often compressed by using simple procedure which allows a string of data to take up less space during a computer's memory. But, In Quantum Computing Data Compression is different because Quantum data are different and it's impossible to decide the frequencies of 1's and 0's in quantum information. The structure of the underlying autoencoder network is often chosen to represent the information on a smaller dimension, effectively compressing the input. Inspired by this concept, we introduce the model of a quantum autoencoder to perform similar tasks on quantum data. The quantum autoencoder is trained to compress a specific dataset of quantum states, where a classical compression algorithm cannot be employed. Using classical optimization algorithms, the parameters of the quantum autoencoder are trained. We show an example of an easy programmable circuit which will be trained as an efficient autoencoder with and without using dummy swap gates are implemented in IBMQ.

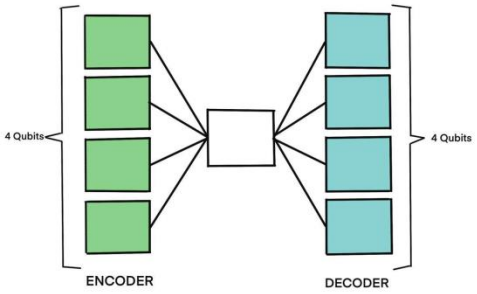
Keywords: Quantum computing, Autoencoder, IBMQ, Qubit.

I. INTRODUCTION

Quantum technologies, ranging from quantum computing to quantum cryptography, have been found to have powerful applications for a modern society. Quantum computing is the utilization of collective properties of quantum states, such as superposition and entanglement, to perform computation. Superposition is the ability of a quantum system to be in multiple states at the same time until it is measured and Quantum entanglement is a physical phenomenon that occurs when a swarm of particles are generated, interact, or share spatial proximity in a way such that the quantum state of every particle of the group can't be described independently of the state of the others, including when the particles are separated by an outsized distance. The topic of quantum entanglement is at the heart of the inequality between classical and quantum physics entanglement is a primary feature of quantum mechanics lacking in classical mechanics. The devices that perform quantum computations are referred to as quantum computers. There are a number of types of quantum computers which includes the quantum circuit model, quantum Turing machine, adiabatic quantum computer, one-way quantum computer, and various quantum cellular automata. The most widely used model is that the quantum circuit, supported the quantum bit, or qubit is somewhat analogous to the bit in classical computation.

For classical data processing, machine learning via an au- to encoder is one such tool for dimensional reduction [1-3].as well as having application in generative data models [4].A like the idea of classical auto encoders, a quantum autoencoder is a function whose parameters are improved across a training data such that given an (n+k)-qubit input x, the autoencoder attempts to reproduce x. Part of the process involves expressing the input data set using a fewer number of qubits (using n qubits out of n+k).This means that if the QAE is successfully trained, the circuit represents a compressed encoding of the input x, which may be useful to applications such as dimension reduction of quantum data.

In this paper, we introduce the concept of a quantum autoencoder which is caused by this design for an input of n+k qubits. Because quantum physics is in a position to get patterns with properties (e.g., superposition and entanglement) that's beyond classical physics, a quantum computer should even be ready to recognize patterns that are beyond the potential of classical machine learning. Thus, the motivation for a quantum autoencoder is simple, It allows us to perform corresponding machine learning tasks for quantum systems.



International Journal of Engineering Applied Sciences and Technology, 2021
Vol. 6, Issue 6, ISSN No. 2455-2143, Pages 296-303
Published Online October 2021 in IJEAST (http://www.ijeast.com)

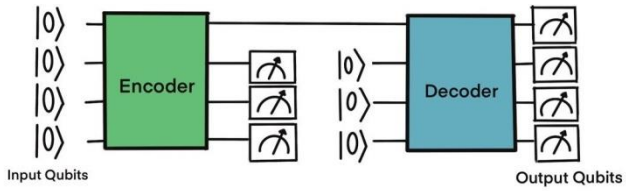


Figure 1. a) A graphical representation of a 4-bit autoencoder with a 1-bit latent space. The map encodes a 4-bit input (Green) into a 1-bit intermediate state (White dots), after which the decoder attempts to reconstruct the input bits at the output(Blude dots). b) Circuit implementation of a 4-1-4 quantum autoencoder.

without exponentially costly classical memory, for instance, in dimension reduction of quantum data.In this work, we provide a simpler model which we believe more easily captures the essence of an autoencoder.

A. *QUANTUM COMPUTING PRELIMINARIES*

Qubits: Qubit is similar to classical bits 0 and 1 with a fundamental difference being that a qubit can be in a superposition state i.e.,a union of 0 and 1 at the same time. Qubit state is expressed with a key Notation**.**

Quantum gates: Quantum gates are the operations that regulate the state of qubits and thus, perform computations. Quantum gates are represented by 2(pow(n)) *2(pow(n)) unitary matrices (n= number of qubits) and can work on a single qubit (e.g., X (NOT) gate) or on multiple qubits (e.g., 2-qubit CNOT gates).

Basis gates and coupling constraints: A practical quantum computer normally supports a limited number of single and multi-qubit gates referred to as basis gates or native gates of the hardware. For example, IBM quantum computers have the following basis gates: u1, u2, u3, id (single-qubit), and CNOT (two-qubit). However, the quantum circuit may contain high-level gates that are not native to the hardware e.g., the Toffoli gate is not native to the IBM quantum computers. Therefore, the gates in a quantum circuit are decomposed into the basis gates before execution. Besides, the two-qubit operation (CNOT) is only permitted between the connected qubits. These restrictions in two-qubit operations in any target hardware are also known as coupling constraints.

Compilation: Quantum circuit compilers e.g., Qiskit [5]. perform necessary modifications (e.g., insert SWAP gates) to the input circuits to satisfy coupling constraints of the hardware. Besides, compilers offer higher-level circuit optimization through single/multi-qubit gate cancellation, rotation merging, and gate-reordering [6]. Qiskit supports barrier between circuit partitions to limit these additional optimizations across circuit partitions [5].

B. *TOTAL VARIATION DISTANCE (TVD)*

TVD is a widely used metric to measure the difference between two quantum states [7-8],We use TVD as a measure of the distance between the desired output state of the original circuit and the corrupted output state of the obfuscated circuit. A higher TVD indicates functional corruption of the circuit. We use Total Variation Distance (TVD) as a measure of the difference between obfuscated output and original output. TVD is determined as $\sum i(|xorig,i - xobfus,i| / shots)$. Here, xi is the count of ith element of a distribution. One of the challenges with this technique is the lack of knowledge of the correct output state for a realistic maximization problem that is solved using the quantum circuit.

C. *RELATED WORK*

The works in [9-10]assume that corrival will insert trojans in the reversible circuit before fabrication and send it back to the design companies. However, this attack model is not applicable to quantum circuits since basis gates are realized using microwave or laser pulse. The quantum circuit is never physically fabricated in gate-based model of quantum computing even though a quantum circuit is reversible. Binary data and test pattern-based detection assumptions made in the work does not apply to quantum circuits.

Another work [11]assumes untrusted foundry that can locate ancillary and garbage lines in reversible circuit and can extract the circuit functionality. Dummy ancillary and garbage lines are added to the circuit which increases the ancillary and garbage lines post-synthesis. The attacker can identify only ancillary and garbage lines added post-synthesis, not the pre-synthesis. To bring down the overhead, reversible gates are added to the circuit judiciously to remove the “telltale” signs post-synthesis while keeping the logical functionality intact. The authors mentioned these approaches are only applicable for oracle-type or pure Boolean logic-based quantum circuits and not for quantum computing. This proposed approach is more generally applicable for any quantum circuits including circuits with and without ancillary and garbage lines whereas [11] crucially depends on the presence of the ancillary and garbage lines. Again, the work in [11] assumes the foundry and fabrication process of the reversible circuit cannot be trusted which is not directly applicable for gate-based quantum computing for reasons previously stated. In our work, we assume the compiler cannot be trusted.

Another work [12] assumes that malicious adversary in quantum cloud will report incorrect qubit quality to force erroneous computation. Our adversarial model considers the compiler to be untrusted whereas the model in [12] considers quantum cloud to be malicious. Therefore, we are addressing vulnerability in a different layer in the quantum computing

International Journal of Engineering Applied Sciences and Technology, 2021
Vol. 6, Issue 6, ISSN No. 2455-2143, Pages 296-303
Published Online October 2021 in IJEAST (http://www.ijeast.com)

stack.

Another work [13] assumes an autoencoder is a supervised learning algorithm that learns that output data reproduce input data [13]. It is composed of a three-layer neural network of an input layer, a hidden layer, and an output layer, and the data of the hidden layer is a data representing the features of the input data. The quantum autoencoder is not a neural network but is calculated using quantum adiabatic algorithm, which is one of optimization algorithms. First, in order to verify the performance of a normal quantum autoencoder, they will verify whether the same data as the original data can be output when the 9pixel black-and-white image is used as the original data and the noise mixed data obtained by adding noise to the original data is used as the input data.

Another work [14] assumes Classical autoencoders are neural networks that can learn efficient low dimensional representations of data in higher dimensional space. The duty of an autoencoder is, given an input x, is to map x to a lower dimensional point y such x can likely be recovered from y. The structure of the underlying autoencoder network is often chosen to represent the information on a smaller dimension, effectively compressing the input.

They show an illustration of a simple programmable circuit that can be trained as an efficient autoencoder. They apply the model within the context of quantum simulation to compress ground states of the Hubbard model. Inspired by this concept, we initiate the model of a quantum autoencoder to perform similar tasks on quantum data. The quantum autoencoder is trained to compress a specific dataset of quantum states, where in a classical compression algorithm can't be employed. The criterion of the quantum autoencoder are trained using classical optimization algorithms.

II. QUANTUM AUTOENCODER MODEL

In comparsion with the model of classical auto encoders, the quantum network features a graphical representation consisting of an interconnected group of nodes. In the graph of the quantum network, each node represents a qubit, with the primary layer of the network representing the input register and therefore the last layer representing the output register. In our representation, the sides connecting adjacent layers represent a unitary transformation from one layer to subsequent. Autoencoders, specially, shrink the space between the first and second layer, as depicted in Figure 1a.

For a quantum circuit to embody an autoencoder network, the data contained in a number of the input nodes must be discarded after the initial encoding E. We imagine this takes place by tracing over the qubits representing these nodes (in Figure 1b, this is represented by a measurement on those qubits). New qubits (initialized to some reference state) are then prepared and used to implement the final decoding evolution, which is then compared to the initial state [17].

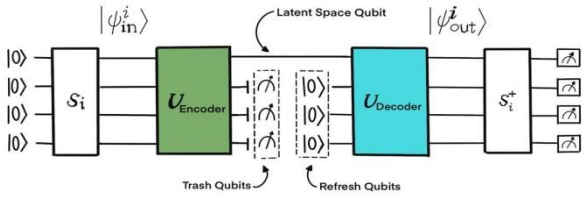


Figure 2. Quantum Autoencoder Model

We have an encoder circuit that encodes a particular state. The trick is that we measure some qubits that we call "Trash qubits", and we want the encoder circuit to produce those states that have zeroes on those trash qubits. This way if we apply a reversed autoencoder circuit, we will obtain the original quantum state at the output. To do this, we have Maximum sum of probabilities of states that have zeroes on the trash qubits.

The learning task for a quantum autoencoder is to find unitaries which preserve the quantum information of the input through the smaller intermediate latent space [15]. To this end, it is important to quantify the deviation from the initial input state, $|\psi_{i\ in}$, to the output, $\psi_{i\ out}$ .Here, we will use the expected for all the input states.

A formal description of a quantum autoencoder follows: Let, {ψi *in,* ψi *out*}be an ensemble of pure states on n + k qubits, where subsystems A and B are made of n and k qubits, respectively. Let {U E} be a family of unitary operators acting on n + k qubits, some set of parameters defining a unitary quantum circuit. Also be some fixed pure reference state of k qubits. Using classical learning methods, we wish to find the unitary U(decoder) which maximizes the average fidelity.

To implement the quantum autoencoder on a quantum computer we must define the form of the parametrized unitary, decomposing it into a quantum circuit suitable for optimization. It consists of layers composed of Controlled Z gates acting on alternating pairs of neighboring qubits which are preceded by R_y qubit rotations. After implementing, a final layer of R_y qubit gates is applied. Eventually, measurements on the desired discarded qubits have to be performed for the training.

International Journal of Engineering Applied Sciences and Technology, 2021
Vol. 6, Issue 6, ISSN No. 2455-2143, Pages 296-303
Published Online October 2021 in IJEAST (http://www.ijeast.com)

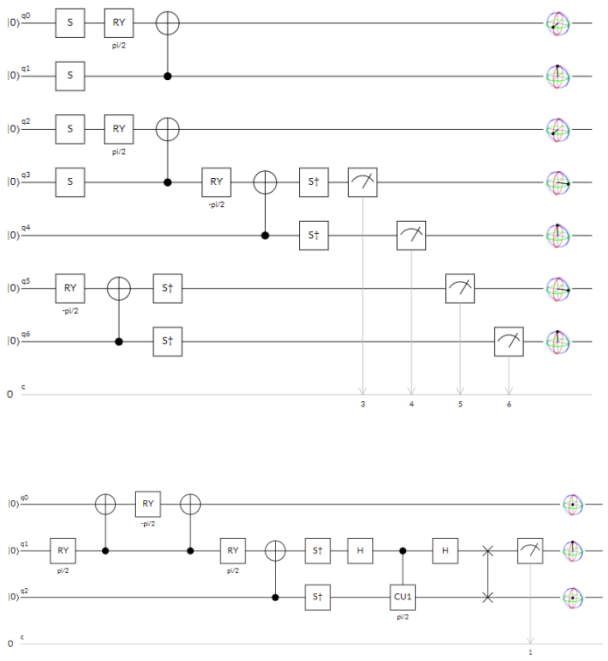


Figure 3. A network comprising all the possible controlled general single-qubit rotations (denoted by Ry) and a control qubit in a qubit set, plus single qubit rotations at the beginning of 2:1:2 and 4:1:4 Quantum Autoencoder.

It has been shown that, employing a circuit of exponential depth, one is usually able to perform a disentangling operation, but to perform this operation in constant or polynomial depth is tough , then classical heuristics are often used for finding quantum circuits that are as close to optimal as possible. Also, information as well as theoretic bounds are explored during this context before, both in the context of one-shot compression and one-shot decoupling. However, because the heuristics involved in choosing efficient-to-implement families of unitaries are largely adhoc, it is difficult to mention if these bounds are meaningful within the context of a quantum autoencoder.

The objective of a quantum autoencoder is to store the quantum information of the input through the lesser latent space. Therefore, it is important to quantify how well the information is preserved. Note that the motivation for a quantum autoencoder is to be able to recognize patterns beyond the capabilities of a classical autoencoder, given the different properties of quantum mechanics.

IMPLEMENTATION OF THE QUANTUM AUTOENCODER MODEL

To excute the quantum autoencoder model on a quantum computer we must define the form of the unitary, $U^E$and decompose it into a quantum circuit suitable foroptimization. For the execution to be efficient, the number of parameters and the number of gates in the circuit shouldscale polynomially with the number of input qubits. This requirement immediately eliminates the possibility of using a $(n + k)$ qubit general unitary as U^E due to the exponential scaling in the number of parameters needed to get them.

One alternative for the generation of U(encoder) is to employ a programmable quantum circuit [15-16]. This type of circuit construction consists of a hard and fast network of gates, where a polynomial number of parameters associated to the gates i.e. rotation angles, constitute theta. The pattern defining the network of gates is considered a unit-cell. This unit-cell can ideally be repeated to extend the pliability of the model. For the numerical evaluation presented in this work.

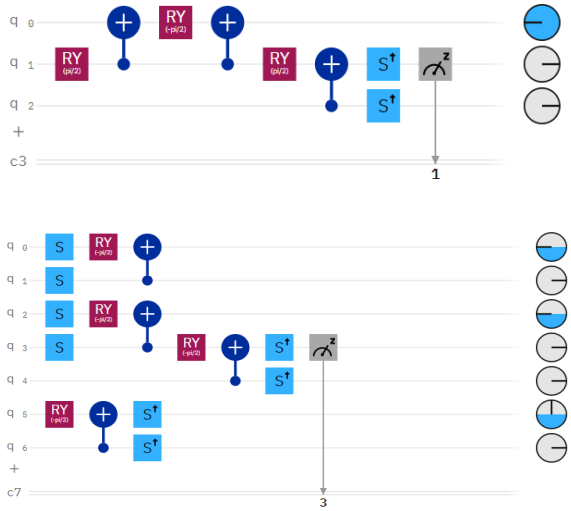


Figure 4. Implementation of Quantum Autoencoder Using IBMq

We begin with expressing each state using 4 qubits and try to compress the information to 1 qubit (implementing a 4-1-4 quantum autoencoder). we use 7 total qubits for the 4-1-4 autoencoder, using the last 3 qubits (qubits q6, q5 and q4)as refresh qubits. The unitary S represents the state preparation circuit, gates implemented to produce the input data set. There are 3 refresh qubits used to reconstruct the data from the smaller latent space. The unitary U represents the training circuit that will be responsible for representing the data set employing a fewer number of qubits, during this case employing a single qubit. The tilde symbol above the daggered operations indicates that the qubit indexing has been adjusted such that q0=q6, q1=q5, q2=q4 and q3=q3. So, qubit q3 is to be the latent space qubit, or qubit to hold the compressed information. Using the circuit structure above (applying S and U then effectively un-applying S and U) we train the autoencoder by propagating the QAE circuit with proposed parameters and counting the number of times the measurements of the refresh qubits and the latent space qubits (q3 to q6) are 0000 in Figure 4.

This circuit contains a unit-cell having all the possible controlled one-qubit rotations among a group of qubits, complemented with a group of single qubit rotations at the start

International Journal of Engineering Applied Sciences and Technology, 2021
Vol. 6, Issue 6, ISSN No. 2455-2143, Pages 296-303
Published Online October 2021 in IJEAST (http://www.ijeast.com)

and at the end of the unit-cell, as shown in for the four-qubit case. We get thinking about the rotations controlled by the primary qubit, followed by the rotations controlled by the second qubit and so on. Accordingly, our model comprises 2n(n 1)+4n training parameters per unit cell and can be implemented in state of the art quantum hardware using the standard decomposition of controlled unitaries into two CNOT gates and single-qubit rotations [17]. This model is also general and can be altered by adding constraints to the parameters. For instance, one could consider the initial and final layers of rotations to be all an equivalent.
In the current version of Quantum Compress, there are two main training schemes:

1. Halfway training - during this, we implement only the state preparation followed by the training circuit and count the probability of measure all 0's on the "trash" qubits (i.e. input qubits that aren't the latent space qubits).
2. Full training - In this scheme, we implement the entire circuit (state preparation, training, un-training, un-state preparation) and count the probability of measuring all 0's on the "output" qubits. There are two possible sub-strategies:

2a. Full training with reset: With the RESET feature in pyQuil, we reset the input qubits such that these qubits are the refresh qubits in the latter half of the QAE circuit. In total, this method requires n_{in} qubits.

2b. Full training without reset: Without the reset feature, we present new qubits for the refresh qubits. Therefore, this method requires 2 * n_{in} - n_{latent} qubits.

III. IMPLEMENTATION OF QUANTUM AUTOENCODER MODEL USING DUMMY SWAP GATES

In this section, we provide outline of the proposed procedure. First, we describe various features of the dummy SWAP locations using examples. Next, we explain various approaches to combine them to distill effective metrics.

A. *OVERVIEW*

Our main objective is to introduce dummy gates in a quantum circuit before it is compiled. This insertion should be done strategically to obtain a good TVD from the original circuit. The approach is to completely analyze all possible positions to insert dummy gates, the corresponding total variance distance values and identify their features. Once the features for a particular SWAP gates location and corresponding total variance distance is understood, a guided task can be developed to insert dummy gates for a fresh circuit to maximize corruption of the circuit. For the complete search based analysis, a quantum circuit is selected, and two qubit SWAP gates are inserted at locations while the circuit depth is unchanged.
To do this, first the circuit is divided into slices. Inside each slice, any two quantum gates are operated using different set of qubits which means the quantum gates can operate in parallel. Hence, assuming n free qubits in a slice, there are (n/2) possible ways the dummy 2-qubit SWAP gates are inserted into the slice if n >=2 (even on non-neighbouring qubits). To study suitable positions for the dummy gates, we insert one dummy SWAP gate at each of the possible positions of the standard circuits.

B. *PROPOSED WORK*

Step – 1: First step is to identify various features that can distinguish the dummy gate locations from each other in a given circuit. Some examples of these features include number of control gates on the qubits and depth of the dummy SWAP gate from output. These features are used individually or in combination to determine the best metric for guided selection of SWAP gate location for future unseen circuits. The features are explained below.

1) Depth of the dummy gate from primary outputs

We use this feature to calculate the number of slices present between the considered position and the output. It quantifies the influence of the SWAP gate on the output of the circuit. For example, the depths of SWAP positions 1, 2, and 4 are 7, and 6 respectively.

2) Measuring qubit

This feature checks if the two qubits associated with the dummy SWAP gate are being measured eventually or not. If both of these qubits are measured, then it is likely that the SWAP gate will impact the output significantly. The impact reduces if only one of the qubits is measured. For example, SWAP–6 involves 1 measured qubit (Q2; qubits in the figure are indexed from Q0 to Q1 from the top), SWAP–5 involves 2 measured qubits (Q0 and Q4), and SWAP–1 involves 0 measured qubits.

3) Number of times the qubit is used as control qubit

This feature counts the number of times the qubit involved in the dummy SWAP gate is used as control qubit in the circuit. This intuitively states that if this qubit is impacted, the other directly/indirectly controlled qubits will be affected as well.

4) Number of control qubits in the paths

This feature counts the number of control qubits that can be tracked in the paths from each qubit involved in the dummy gate (source) till one of the measured output qubits has been reached (destination). Multiple paths can be tracked from the two qubits in a dummy gate to one of the destinations measured qubit. The number of these control gates are added together for each dummy gate position. Considering SWAP–6 in Fig. 5. Two paths (one "top" and another "bottom") stemming from 2 qubits in the SWAP gate are shown. At the control qubit the path splits into two direction. One proceeds to the next time-step (if the next time-step has gates in it) and another takes a 90◦ bend towards the target qubit. For SWAP– 6, the top path splits into two. The split path moving to the next time-step has no gates in that time-step. Another

International Journal of Engineering Applied Sciences and Technology, 2021
Vol. 6, Issue 6, ISSN No. 2455-2143, Pages 296-303
Published Online October 2021 in IJEAST (http://www.ijeast.com)

split path takes the 90 bend and continues towards the target qubit, and in the process encounters another control qubit. Thus, the top path from SWAP–12 has 2 control qubits. Likewise, the bottom encounters 1 control qubit during traversal. However, that control qubit is already considered during the "top" path traversal, and hence, not counted again.

5) Constant qubit:

Some circuits have constant values for some qubits e.g., 1 or 0. These constants affect the output especially for some input combinations and when the dummy gate involves one constant qubit. They get swapped and as a result, affect the other gates, eventually corrupting the output. Hence whether the dummy gate has a constant qubit involved is used as a feature.

Step – 2: After collecting all the features described above, we focus on using them to develop an effective metric. To begin with, the dummy gate depth (or number of control qubit) feature is used one by one to select the dummy gate location. However, these basic metrics produce multiple positions with the same feature values. The TVD of these positions are averaged and compared with the average and the best TVD of each of the benchmark circuit the average across all benchmarks for 2 features namely, depth and number of control qubits. It can be noted that each of these features exhibit strong correlation towards output TVD. Further, some features might maximize the TVD more than others. Since these features produce multiple positions, we use their combination to refine a single best SWAP location for each circuit. If we consider a single feature, the difference from best TVD is relatively high.

Metric-1 and Metric-2: In the first pass, we remove SWAP positions with 0 depth and 0 measuring qubits involved. These cases are SWAP gates at primary output involving qubits that are not measured. The corresponding Total Variance Distances authenticate to the fact that such SWAP positions are in- effective for obfuscation and can be safely removed from consideration. After this pruning, we obtain a reduced set of SWAPs. We compute score for each position to select candidate SWAP location. Metric-1 picks the SWAP with highest score and Metric-2 picks the one with lowest score

Metric-3 and Metric-4: In the second pass (i.e., with after SWAP list reduction employed in Metric-1/2), we remove SWAP(s) not involving any constant qubits to further sanitize the SWAP list. Metrics-3 and 4 are based on the highest and lowest scores, respectively.

Metric-5 and Metric-6: Finally, we remove SWAP locations involving 0 control gates in the path to a measured qubit. The potential impact of the SWAP on the output is minimum without any control operations in the path. Thus, such SWAP locations are not ideal for obfuscation. Metrics-5 and 6 are based on the highest and lowest scores.

In denser circuits, higher depth means that the SWAP gate is inserted earlier in the circuit, and higher number of control qubits both add up to give highest score. Also note that, lowest score either indicates lower depth implying that the SWAP gate is located closer to the measured qubits and/or that the number of control qubits is less and can directly impact the output TVD significantly. We consider both highest and lowest scores in our metrics. However, we find that the highest score-based metric (Metric 5) performs better on a greater number of circuits. Therefore, this can be the metric of choice.

C. *CASE STUDY- 4:1:4 AUTOENCODER CIRCUIT*

The 4:1:4 Quantum autoencoder circuit (Fig. 5) is divided into 5 slices between barriers. For each experiment, one SWAP gate at a time is inserted in one of the 22 possible locations overlapping with the other gates in the slices. After compilation and simulation for individual SWAP gate positions, it has been categorised that the average TVD.

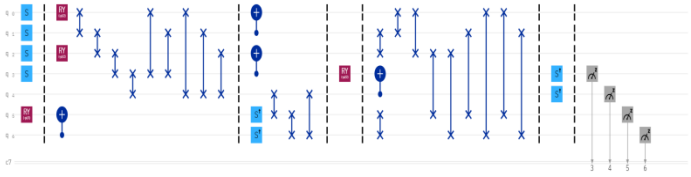


Figure 5: Circuit diagram of 4:1:4 Quantum autoencoder with annotated features implemented in IBMQ. For example, SWAP gate position 6 has depth = 4, number of control qubits = 7, measured or not = 1, constant qubits involved or not = 1, and number of control qubits in paths = 2, as shown. The path shown in blue, starts from each qubit of SWAP gate 6, and is continued in links by the control qubits, ending in target qubit that is measured. Doing this for both qubits in SWAP gate 6, we get 2 control qubits in the paths.

However, the final metric should be generic i.e., it should outperform the average TVD for most of the circuit and should be as close to best TVD as possible. This analysis shows that metric 3 meets this requirement even though it performs badly for this particular example circuit.

IV. DISCUSSION

We have introduced a general model for a quantum autoencoder a quantum circuit augmented with learning via classical optimization and have shown that it is capable of learning a single circuit which can ease compression of quantum data, particularly in the factors of quantum simulations. We imagine that the model can have other applications, such as compression protocols for quantum communication, error- correcting circuits, or perhaps to solve particular problems directly. An application for quantum autoencoders is state preparation. Once a quantum autoencoder has been trained to compress a specific set of states, the decompression unitary(U†) can be used to generate states similar to those originally used for training. This is achieved by preparing a state of the form ΨI a and evolving it under U†, where ΨI has the size of the latent space and a is the reference state used for training.

International Journal of Engineering Applied Sciences and Technology, 2021
Vol. 6, Issue 6, ISSN No. 2455-2143, Pages 296-303
Published Online October 2021 in IJEAST (http://www.ijeast.com)

In the specification of the autoencoder, we define the input states to be an group of pure states, and therefore the evolution of those states to be unitary [18-21]. The most generalized picture of the autoencoder, however, would leave inputs to be ensembles of mixed states and therefore the set to be a group of quantum channels. In the case of mixed state inputs, we remark that this formulation can in theory be captured by our model already.

It is questioned whether there are any obvious limitations to the quantum autoencoder circuit. One such consideration is that the von-Neumann entropy [22] of the density operator representing the ensemble ψi in, ψi out restricts the number of qubits to which it can be noiselessly compressed. However, finding the entropy of this density operator is not insignificant – in fact, given a circuit that constructs a density operator ρ, in general it is known that even estimating the entropy of ρ is QSZK complete [23]. This then opens the possibility for quantum autoencoders to efficiently give some approximate of the entropy of a density operator.

1) Consideration for multiple dummy gates

The addition of multiple dummy SWAP gates can be considered to obtain a higher Total Variance Distance and to increase the adversarial effort further. However, the overhead will also be higher. This feature can be investigated in future research.

2) Impact of dummy gates on coherence

SWAP gates are only inserted in the compilation phase. Before executing the circuit on the real hardware, the dummy gates will be removed. The device executable version of the circuit will not contain any additional gates. Therefore, the proposed approach does not affect coherence during execution but will secure the circuit during compilation on third-party compilers.

3) Consideration for non-arithmetic circuit

Non-arithmetic quantum circuits are often studied in future research to spot other metrics which will be used for obfuscation. These circuits will provide a far better understanding since measurement in X-basis and Hadamard basis don't seem to be commonly covered in arithmetic circuits.

4) Recognition and removal of added extra logic

After compilation, the designer needs to identify and remove the extra dummy gates to retrieve the original circuit functionality. This could pose a challenge since the dummy SWAP gate could be optimized with other gates in the circuit, removed during compilation if a reverse SWAP is required to meet the coupling limitation of the hardware and mixed up with other gates and could be difficult to identify due to change in circuit depth and other add SWAP gates. As an initial solution, we employed the barriers to enclose the dummy SWAP gate. This prevents the compiler from optimizing the SWAP gate with other gates. Although the added barrier could lead to slight degradation in optimization, it provides an easy mechanism to the designer to identify and remove the dummy SWAP gate post compilation. Note that the added barrier may provide clue to the adversary. One can obfuscate such clues by adding dummy barriers in the design.

Circuit	Original Circuit			SWAP gate position with best TVD			Overhead(%)		
	Compl. time (s)	Depth	Ops. count	Compl. time (s)	Depth	Ops. count	Compl. time	Depth	Ops. count
4:1:4 QAE	10.01	86	131	5.66	90	139	-30.36	4.40	3.47

TABLE 1: Design overhead in terms of circuit depth, number of basis operations, and compilation time for two benchmark circuits with the Qiskit compiler backend.

An additional topic of interest for any quantum computing model is the computational complication exhibited by the device. For our construction, it's clear that any complexity result would be dependent upon the family of one that is chosen for the learning to be optimized over. As the training itself is dependent on classical optimization algorithms this further obfuscates general statements regarding the complexity of the model.

V. EXPERIMENTAL RESULTS

For the training set, six states were selected. The circuit in Fig. 4 is run for each of the training states, calculated the probability of getting measurement outcome 0000, and averaged the probabilities over the training set (and negated) to determine the loss. We then used the parameters to get the optimal or near-optimal parameters for the training circuit. For the demo, we chose an easy training circuit containing a couple of single-qubit rotation gates (with two tunable parameters) alongside a few CNOT gates. Also, we implemented Quantum Autoencoders on IBMQ, We get a 100% probability on state vectors (2:1:2 QAE).

It is noted that one can perform comprehensive simulation in order to find the best positions to insert the dummy gates. However, simulation is prohibitively expensive for quantum circuits with large number of qubits. Moreover, the application loses any perceived quantum advantage if it can be classically simulated. One of the prime objectives of this article is to devise a scalable approach for quantum strength comparable to the exhaustive approach. In this Section, we show results on small quantum circuit benchmarks to demonstrate the benefit of the proposed work. we first

provide a brief description of our experimental/simulation framework, and benchmark.

Evaluation Framework and Benchmarks: We use the open-source quantum software development kit from IBM (Qiskit) [24]. For our simulations. The software runs locally on an Intel Core i5 CPU clocked at 2.40GHz machine. The default compiler backend in Qiskit is used f o r compilation. A Python-based wrapper is built around Qiskit to accommodate the proposed on the input circuits to the compiler backend.

In IBMQ or Quantum programming studio,By implementing 4:1:4 Quantum autonencoder without using dummy swap gates we get the more state vectors and less Probabilities but while implementing 4:1:4 Quantum autoencoder with using dummy swap gates we get the less state vectors and more Probabilities shown in fig 6.

Figure 6. State vectors and probabilities of 4:1:4 Quantum autoencoder with dummy swap gates.

ACKNOWLEDGEMENTS

We would like to show our gratitude to the Maharaja Institute of Technology Mysore and we thank teaching and non-teaching staff of department of electronics and communication engineering. Also, thanks to our parents and friends who all are directly or indirectly supported for this research.

VI. REFERENCES

[1] C.-Y. Liou, J.-C. Huang, and W.-C. Yang, Neurocomputing **71**,3150 (2008).

[2] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, and D.-R. Liou, Neuro- computing **139**, 84 (2014).

[3] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, arXiv preprint quant-ph/1611.09347 (2016).

[4] R. Gómez-Bombarelli, D. Duvenaud, J. M. Hernández-Lobato,
J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and
A. Aspuru-Guzik, arXiv preprint cs.LG/1610.02415 (2016).

[5] A. Cross, "The ibm q experience and qiskit open-source quantum computing software," in APS Meeting Abstracts, 2018.

[6] Y. Nam, N. J. Ross, Y. Su, A. M. Childs, and D. Maslov, "Automated optimization of large quantum circuits with continuous parameters," npj Quantum Information, vol. 4, no. 1, pp. 1–12, 2018.

[7] M. Sarovar, T. Proctor, K. Rudinger, K. Young, E. Nielsen, and
R. Blume-Kohout, "Detecting crosstalk errors in quantum information processors," Quantum, vol. 4, p. 321, Sep. 2020. [Online]. Available: https://doi.org/10.22331/q-2020-09-11-321.

[8] A. Ash-Saki, M. Alam, and S. Ghosh, "Experimental characterization,modeling, and analysis of crosstalk in a quantum computer," IEEE Transactions on Quantum Engineering, vol. 1, pp. 1–6, 2020.

[9] X. Cui, S. M. Saeed, A. Zulehner, R. Wille, K. Wu, R. Drechsler, and R. Karri., "On the difficulty of inserting trojans in reversible computing architectures," IEEE Transactions on Emerging Topics in Computing, 2018.

[10] Limaye, Nimisha, M. Yasin, , and O. Sinanoglu, "Revisiting logic locking for reversible computing," IEEE European Test Symposium (ETS), 2019.

[11] S. M. Saeed, A. Zulehner, R.Wille, R. Drechsler, and R. Karri, "Reversible circuits: IC/IP piracy attacks and countermeasures," IEEE Transactions on Very Large Scale Integration (VLSI), 2019.

[12] Acharya, Nikita, , and S. M. Saeed, "A lightweight approach to detect malicious/unexpected changes in the error rates of nisq computers," IEEE/ACM International Conference On Computer Aided Design (ICCAD), 2020.

[13] Nathan Hubens Deep learning and Data science in Autoencoder ,2018.

[14] Jonathan Romero, Jonathan P. Olson, and Alan Aspuru-Guzik- Quantum autoencoders for efficient compression of quantum data ,(Dated: February 14, 2017).

[15] P. B. Sousa and R. V. Ramos, arXiv preprint quant-ph/0602174 (2006).

[16] A. Daskin, A. Grama, G. Kollias, and S. Kais, J. Chem. Phys
137, 234112 (2012).

[17] M. A. Nielsen and I. L. Chuang, Quantum Computation and
Quantum Information (Cambridge University Press, 2010).

[18] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou,
P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, Nat. Commun.5, 4213 (2014).

[19] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-

Guzik,
New. J. Phys 18, 023023 (2016).

[20] D. Wecker, M. B. Hastings, and M. Troyer, Phys. Rev. A 92,
042303 (2015).

[21] Y. Li and S. C. Benjamin, arXiv preprint quant-ph/1611.09301
(2016).

[22] M. M. Wilde, Quantum Information Theory (Cambridge University Press, 2012).

[23] A.Ben-Aroya and A.Ta-Shma, arXiv preprint quantph/0702129 (2007).

[24] A. Cross, "The ibm q experience and qiskit open-source quantum computing software," in APS Meeting Abstracts, 2018.