PAPER • OPEN ACCESS

Information loss and run time from practical application of quantum data compression

To cite this article: Saahil Patel *et al* 2023 *Phys. Scr.* **98** 045111

Physica Scripta

PAPER

Information loss and run time from practical application of quantum data compression

Saahil Patel[1], **Benjamin Collis**[1,2], **William Duong**[3], **Daniel Koch**[1,2], **Massimiliano Cutugno**[1], **Laura Wessing**[1] **and Paul Alsing**[1]

[1] Air Force Research Laboratory, Rome, NY, United States of America
[2] Griffiss Institute, Rome, NY, United States of America
[3] Rochester Institute of Technology, Rochester, NY, United States of America

E-mail: saahil.patel@us.af.mil

OPEN ACCESS

RECEIVED
20 September 2022

REVISED
22 February 2023

ACCEPTED FOR PUBLICATION
15 March 2023

PUBLISHED
23 March 2023

Keywords: quantum, quantum computing, quantum algorithms, quantum information, experimental

Abstract

We examine information loss, resource costs, and run time from practical application of quantum data compression. Compressing quantum data to fewer qubits enables efficient use of resources, as well as applications for quantum communication and denoising. In this context, we provide a description of the quantum and classical components of the hybrid quantum autoencoder algorithm, implemented using IBMs Qiskit language. Utilizing our own data sets, we encode bitmap images as quantum superposition states, which correspond to linearly independent vectors with density matrices of discrete values. We successfully compress this data with near-lossless compression using simulation, and then run our algorithm on an IBMQ quantum chip. We describe conditions and run times for training and compressing our data on quantum devices, and relate trainability to specific characteristics and performance metrics of our parametric quantum circuits.

1. Introduction

In the Information Age [1], data compression has become a powerful tool for dealing with datasets large in number, volume, and variability. Reducing the dimensionality of large datasets allows for efficient use of resources, as well as extraction of useful features. Classical autoencoders are a type of artificial neural network used extensively for feature extraction [2, 3], denoising [4, 5], and data compression or dimensionality reduction [6–8].

Classical autoencoders, as well as variational autoencoders [9], can be utilized to help mitigate some of the limitations we find in quantum computing [10]. With small numbers of noisy qubits in current quantum chips, implementation of quantum algorithms can benefit greatly from efficient use of qubits for data encoding and computation. Quantum data compression has been studied extensively as a means to efficiently use qubits [11], denoising quantum states[12], as well as image-related autoencoders for classification and anomaly detection [13–15]. We can also utilize quantum data compression in the field of quantum communication. Compressed quantum data can be sent between two parties on a public channel, with a private key used to compress or decompress the data, similar to sending classical compressed data over a network [16].

The theoretical limit to losslessly compressing quantum information down to a latent space requires the number of maximum linearly independent vectors from the input state to not exceed the size of the latent space [17]. In practice however, even if we meet this criteria, we find that the performance and efficiency of quantum data compression is limited by the choice of parametric circuit, the classical optimizer, and the gate depth of the circuit.

In this study, we investigate information loss in application of quantum data compression utilizing the quantum autoencoder as formulated in Romero *et al* (2017) [18]. We study the data compression performance of three parametric circuits on a quantum simulator, and compare the times required to train the network. We then look at a simpler quantum data compression scheme to implement on real qubits that can yield a

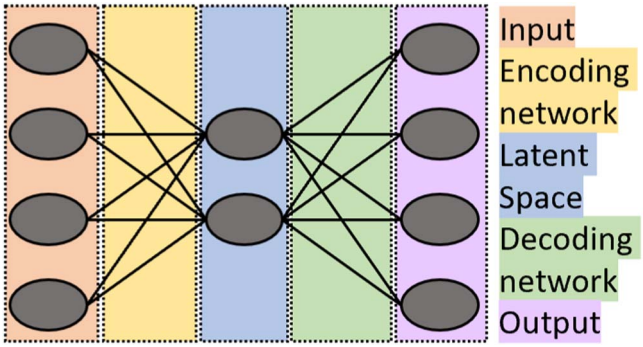

Figure 1. The classical autoencoder schematic broken into 5 major components; we start with 4 nodes, compress down to 2 nodes, and decompress back to 4 nodes.

reasonable training time, with results subject to noise from the quantum devices. Because of long training times necessary for quantum data compression, results from a real device face considerable information loss from noise when compared to simulations. Thus, reduction of gate depth can help mitigate expensive resource costs. We test the parameterized circuits on an IBMQ quantum device [19] in order to understand the running time of the data compression training network when compared to simulations.

In total, we study three parameterized circuits, and compare their performance against measures of expressibility and entangling capability as described in Sim *et al* (2019) [20]. Our simulations demonstrate near-lossless compression of quantum data using an image dataset, where each image is encoded as a unique superposition state, using a scheme similar to [21], described in section 3. Complex datasets incorporating color pixels have been encoded to qubits using flexible representation of quantum images (FRQI) [22–24]. In general, when using simulators, complex data sets can be encoded using translation transforms [22], or using enhanced data loading, which encode feature vectors that characterize a model [13]. These data encoding procedures rely on the ideal conditions provided by simulators. In order to utilize NISQ-era quantum chips, we are constrained by limited connectivity, and a need for small gate-depths. Hence, our dataset was constructed to easily encode simple images using one to three applications of CNOT and CZ gates, described in section 3. Our results gauge the efficacy of modern quantum hardware in compressing quantum data, and aim to improve our understanding of quantum autoencoders on NISQ devices.

The layout of this paper is as follows: section 2 covers the data compression method, which includes the general Quantum Autoencoder framework, the parameterized quantum circuits, and the optimizer we use for this study. Section 3 covers our dataset and data encoding for the simulation and quantum chip. Section 4 covers results from all our simulations and experiments, and discusses the results. In section 5 we summarize the paper and provide future prospects for this research.

2. Method

2.1. Classical autoencoder

An autoencoder is an artificial neural network that is used to learn the representation (or features) of an unlabeled dataset. The encoder improves and validates the representation by regenerating the input. It is a dimensionality reduction algorithm that takes in data and compresses it to a latent space. We can think of the latent space as containing the 'essence' of the input data - a compressed representation where similar data points are closer together in space. The decoding network then extracts features from the latent space and reconstructs the image, ideally without any noise or unwanted artifacts. A full autoencoder network consists of an encoding neural network, a latent space, and a decoding network, as shown in figure 1.

2.2. Quantum autoencoder

The quantum autoencoder (QAE) is a hybrid algorithm, where the encoding and decoding neural networks of a classical autoencoder are replaced by quantum subroutines that use quantum data, utilizing parameterized unitary gates and entanglement. The circuit's aim is to compress information from an input state $|\varphi\rangle$ to a mixed compressed state ρ_B, labeled in figure 2(A) as B. With optimal parameters $\vec{\theta}$, we can get lossless compression, where all the information from $|\varphi\rangle$ is compressed down into state ρ_B. The remaining 'trash' state is an empty state ($|0\rangle^{\otimes n}$), shown as A in figure 2(A). To do this, the overlap between the trash state and a reference $|0\rangle^{\otimes n}$ state is computed, shown as $|\psi\rangle$ in figure 2(A). Extracting the trash state A requires taking the partial trace on the

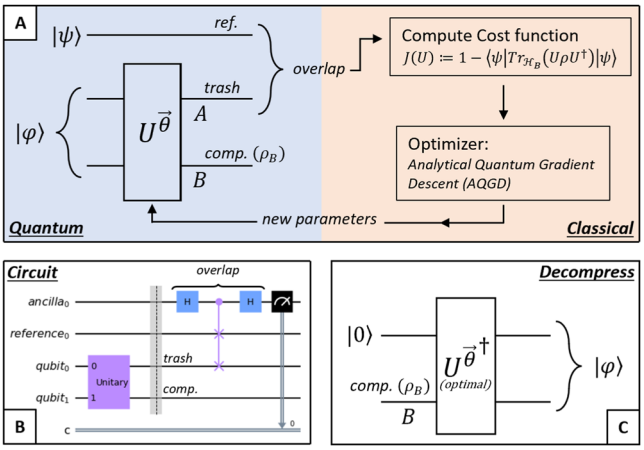

Figure 2. (A): the QAE schematic with the quantum component on the left containing the input state, PQC, and the overlap. The classical component is on the right containing the computation of the cost function and the optimizer. (B): The Qiskit circuit for the quantum subroutine, showing the Swap Test to compute the overlap. (C): Decompressing the compressed quantum state by bringing in a new qubit, and using optimal parameters for lossless decompression.

mixed state $U\rho U^\dagger$ with respect to B, where U is our parameterized circuit, and $\rho = |\varphi\rangle\langle\varphi|$. The overlap, $\langle\psi|Tr_B(U\rho U^\dagger)|\psi\rangle$, is a measure of fidelity between $|\psi\rangle$ and A. The maximum of this fidelity measure is 1, which ensures that the trash state is empty, and the compression is lossless. So the task of the classical optimizer is to minimize the cost function $J(U)$, defined as

$$J(U) = 1 - \langle\psi|Tr_B(U\rho U^\dagger)|\psi\rangle. \tag{1}$$

Within the quantum circuit, the overlap between the reference ($|\psi\rangle$) and trash (A) states is computed using the Swap Test [25], as shown in figure 2(B). The Swap Test is a quantum operation that measures the difference between two quantum states. From figure 2(B), we measure out the ancilla qubit, which computes the overlap between the two states, reference and trash, using a control-swap gate. If the two states are orthogonal, then the probability of the ancilla qubit measured as $|0\rangle$ is 1/2 ($J(U) = 1/2$), while if the two states are the same, the probability of measuring the ancilla as $|0\rangle$ is 1 ($J(U) = 0$) [26].

Once the training circuit minimizes the cost function, we get the final set of parameters $\vec{\theta}$ that compresses the dataset as efficiently as the parameterized quantum circuit (PQC) allows. The performance of the PQC is measured by two circuit descriptors, entangling capability and expressibility [20], discussed in the next section. The success of the algorithm is gauged by using new qubits to decompress the latent space back to the original state. Since we are gauging the efficiency of the compression, the decompression circuit is not a training network, unlike a classical decoding network. In our case, we simply use the optimal set of parameters to implement $U^\dagger$ on the qubits, as shown in figure 2(C), thereby avoiding a second training network. The optimal parameters take in new qubits and the compressed state ρ_B, and decompresses the data back to the original input state $|\varphi\rangle$. For this study, we measure the fidelity between the original input and the final decompressed output to understand information loss and compression efficiency.

2.3. Parameterized quantum circuit

In order to efficiently compress our images into fewer qubits, we quantitatively describe certain properties of the PQC such as performance and resource costs. Using these properties aids us in constructing our circuit with optimal gate sets, which in turn minimize the cost function of the training circuit while limiting computational complexity. First, we look at two circuit descriptors, expressibility and entangling capability, as defined in Sim *et al* (2019) [20]. The authors show expressibility and entangling capability measures for 19 PQCs in their paper [20], one of which we use for our simulations (figure 3(A) Circuit 2).

After encoding the data as quantum states (a step we will cover in the following section), we apply the parameterized circuit that best ensures that the data encoding preserves all features of the original data. Thus, an ideal PQC must have the ability to generate states that are well representative of the full Hilbert space. For a single qubit, this means that the PQC must have the ability to explore the full Bloch Sphere. The quantity that measures this property is called the expressibility ϵ of a circuit [20]. Expressibility is measured by generating a sampled distribution of state fidelities of the parameterized circuit, and a sampled distribution using Haar random states (uniform distribution of random states) [27]. We then measure the Kullback-Leibler (KL) divergence [28] between the two distributions, given by equation (2). The KL divergence, or relative entropy, measures how two probability distributions, P and Q, differ from one another, given by equation (3). A divergence measure of

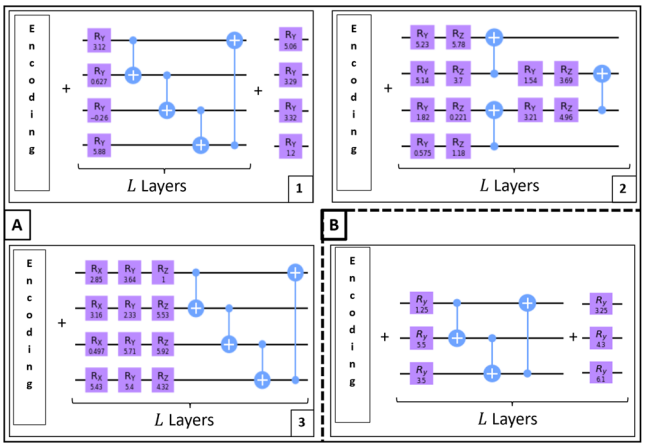

Figure 3. (A): the 3 Parameterized Quantum Circuits (PQCs) we use for our simulated data compression schemes. The circuits are named Circuit 1, Circuit 2 (from [20]), and Circuit 3, indicated by the label on the bottom right of each circuit. (B): PQC for 3 qubits based on Circuit 1, used for implementing 3 to 2 qubit data compression on an IBMQ device.

$D_{KL} = 0$ implies the two distributions are the same. Hence, the smaller the KL divergence, or ϵ, the more expressible the parameterized circuit.

In equation (2), $P_{PQC}(F; \theta)$ is the probability distribution from fidelities resulting from sampling states from the parameterized circuit; F is the fidelity, calculated using sampled parameters $\vec{\theta}$. $P_{Haar}(F)$ is the probability distribution from fidelities resulting from sampling states from Haar random states.

$$\epsilon = D_{KL}(P_{PQC}(F;\ \theta)||P_{Haar}(F)). \tag{2}$$

$$D_{KL}(P||Q) = \sum_{x} P(x)\ log_2 \frac{P(x)}{Q(x)}. \tag{3}$$

For the entangling capability measurement, the Meyer-Wallach (MW) measure [29] is used, a single scalar measurement for pure-state entanglement [30]. The entanglement of each individual qubit is measured with the remaining qubits in the system. One limitation with this version of entanglement measure is that it cannot distinguish between entangled states that are fully inseparable, and entangled states that can be separated into subsystems. This is in contrast to bipartite entanglement measures such as concurrence [30]. However, in this study, we are measuring the global entanglement of the PQCs, and the ease of computation and scalability [20] of the MW measure makes it fitting for our experiments. The following derivation of entangling capability relies on the MW measure. In the Brennen form [31], this distance measure is written as

$$Q(|\psi\rangle) = \frac{1}{n}\sum_{k=1}^{n} 2(1\ -\ Tr[\rho_k^2]), \tag{4}$$

where $Tr[\rho_k^2]$ traces out all but the one-qubit reduced density matrix of the kth qubit. For set S of sampled circuit parameters θ_i, the entangling capability E [20], derived from the MW measure, is then given by:

$$E = \frac{1}{|S|}\sum_{\theta_i \in S} Q(|\psi_{\theta_i}\rangle). \tag{5}$$

For this study, we examine three different PQCs to implement into the quantum subroutine. Note that in principle it is possible to build an arbitrarily complex PQC capable of achieving lossless or near-lossless compression. However, complex PQCs with large gate depths are detrimental to quantum data compression in the Noisy Intermediate Scale Quantum (NISQ) era [32]. Hence, we aim to build PQCs with certain characteristics in mind that make quantum data compression more achievable.

In principle, one can generalize a relationship of PQC performance to a metric like expressibility in the context of device architecture and trainability. In this case, we can perhaps conjecture in advance whether a PQC is trainable on specific hardware, and how that scales with problem size. Studies on quantum neural network training landscapes show how randomized PQCs can be detrimental to trainability as number of qubits increases[33]. Generalizing these relationships can be difficult with experimental realization due to hardware and data specific challenges, and is outside the scope of our study. Using architecture-agnostic simulators, we statistically estimate measures of expressibility and entangling capability similar to the more general studies of PQCs by [20]. Our study numerically verifies the relation between expressibility to the performance of PQCs in

Table 1. Circuit costs.

Circuit	Number of parameters	Number of two qubit gates	Circuit depth
1	$n(L+1)$	nL	$(n+1)$ $L+1$
2	$4(n-1)L$	$(n-1)L$	$6L$
3	$3nL$	nL	$(n+3)L$

Table 2. PQC descriptors.

Circuit	Expressibility (D_{KL})	Entangling Capability
1 ($n=4, L=3$)	0.130	0.800
2 ($n=4, L=3$)	0.008	0.743
3 ($n=4, L=3$)	0.005	0.826

the context of compressing quantum data. Our results empirically verify that in an ideal simulator (lossless all-to-all connectivity), the more expressible the circuit, the better the efficiency of quantum data compression.

Along with expressibility and entangling capability, we describe the performance of our circuits by examining the resource costs for each run. Figure 3(A) shows the three PQCs used for simulations in this study, each composed of 4 qubits; initial parameters are chosen randomly from full range of theta values. Table 1 shows the resource costs per circuit for the general case of n qubits and L layers for each of the three circuits. Table 2 shows the expressibility and entangling capability of each PQC studied in this paper with layers $L=3$. Intuitively, increasing the number of layers L of a PQC should improve the performance of the compression, consequently also requiring more computational time while training the circuit. However, for expressibility and entangling capability, the circuits can reach a saturation point, whereby increasing the number of layers of the PQC does not improve the metrics or the PQC's performance. In this study, we look at PQCs constructed using the metrics mentioned above, as well as ease of implementation using IBM's Qiskit programming language. The differences between the PQCs is further discussed in the results section of the paper. Two of our PQCs are more resource intensive and better suited for simulations (PQCs 2 and 3 in figure 3(A)), and one PQC that can be applicable on NISQ devices for small input sizes (PQC 1 in figure 3(A)). figure 3(B) shows this implementation for 3 qubits, run on an IBMQ device [19].

2.4. Classical optimizer

It is important to understand how a classical optimizer works in order to compute the run time of our algorithm. We calculate the number of times our circuit runs per epoch by examining the Analytical Quantum Gradient Descent (AQGD), an IBM Qiskit component that performs gradient descent optimization [34–36]. One epoch involves running every image in the training dataset once through the compression circuit. After an image is run, the optimizer takes the output observable, $\langle \hat{B} \rangle$, and separately conducts two more runs of the circuit for each parameter in order to calculate the gradient. We employ the parameter shift rule, where for each parameter θ_j, the circuit is run once with a positive shift $\theta_j + \Delta\theta_j$, giving the output $\langle \hat{B} \rangle_j^+$, and another run with a negative shift $\theta_j - \Delta\theta_j$, giving the output $\langle \hat{B} \rangle_j^-$; $\Delta\theta = \theta_j \cdot \pi/2$. A schematic of this procedure is shown in figure 1 of [35]. The gradient of the observable $\langle \hat{B} \rangle$ is then given by equation (6):

$$\frac{\partial \langle \hat{B} \rangle}{\partial \theta_j} = \frac{\langle \hat{B} \rangle_j^+ - \langle \hat{B} \rangle_j^-}{2}. \tag{6}$$

$$\langle \hat{B} \rangle' = \langle \hat{B} \rangle - \eta \cdot \nabla \langle \hat{B} \rangle. \tag{7}$$

Calculating equation (6) for all parameters gives us the total gradient $\nabla \langle \hat{B} \rangle$. We calculate the next step in the gradient descent with equation (7), where $\langle \hat{B} \rangle'$ is the new observable, $\langle \hat{B} \rangle$ is the current observable, and η is the learning rate. The learning rate is the coefficient of the gradient update, whereby increasing its value results in larger step sizes for the gradient optimization. Once the new observable $\langle \hat{B} \rangle'$ is found, gradient descent is used to find new adjusted parameters θ_j' used for the next iteration of training, i.e. $\theta_{t+1} - \theta_t \propto -\eta \nabla \langle \hat{B} \rangle'$ [37]. In this study, $\langle \hat{B} \rangle$ is the expectation value of the measured ancilla qubit (from the SWAP test), as shown in figure 2(B).

The total number of times a circuit is run per image involves 1 run of the original circuit, and 2 runs of the circuit per parameter to calculate $\partial \langle \hat{B} \rangle / \partial \theta_j$. This operation can be performed for many iterations of the optimizer, adjustable by an AQGD hyper-parameter. So for the full training dataset, the number of times we run

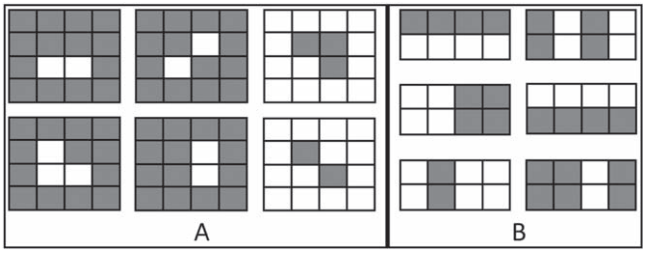

Figure 4. A (for simulations): 6 sample images from our 4 × 4 pixelated image dataset. The rows and columns are labeled by {00, 10, 01, 11}, and $|(c_1, c_2), (r_1, r_2)\rangle$ indicates the pixel label by the two column (row) bits c_1 c_2, (r_1 r_2). **B (for device)**: 6 sample images from the 2 × 4 bar-and-stripe pattern dataset. Here, $|(c_1, c_2), r_1\rangle$ indicates the pixel label by the two column bits c_1 c_2, and single row bit r_1.

the circuit per epoch is given by:

$$N_{\text{jobs}} = ((2 \cdot N_{\text{params}} + 1) \cdot N_{\text{images}}) \cdot N_{\text{iter}}, \tag{8}$$

where N_{iter} is the maximum number of iterations of the optimizer, N_{images} is the number of images in the training dataset that the training circuit goes through in one epoch, and N_{params} is the total number of parameters in the PQC. Here, N_{iter} is equivalent to number of shots (circuit runs + measurement) the optimizer takes to calculate the average cost function in equation (1). For all simulations in this paper, the maximum number of iterations, N_{iter}, was set to 10.

3. Data encoding

3.1. Data for simulations

In order to train the network to efficiently compress quantum data, we use a data set consisting of black and white pixelated images following specific rules. For the simulations, we use 4 × 4 pixel images where all squares touching the border are either black or white, creating a 'frame' around the 4 central squares, thereby functioning as a single pixel. The number of arrangements in the full dataset is then dependent on the central 4 squares and the frame, yielding a total 2^5 variations. Figure 4(A) shows 6 examples out of 32 total images. Using 4 × 4 images (16 pixels) allows us to represent each image as an equal superposition state of 4 qubits, giving us 16 total states for the 16 pixels. The phases on each state encode the different images by the following sign convention: phases on the superposition states go as $(-1)^n$, where $n = 0$ for a black pixel (positive phase), and $n = 1$ for a white pixel (negative phase). Phases on the pixels are assigned left to right, and top to bottom, following the convention in Tacchino *et al* (2019) [21]. This encoding of pixels creates a dataset of unique linearly independent vectors encoded using 4 qubits, which we then compress down to 3, 2 and 1 qubit in our simulations. The border rule on the dataset ensures a specific pattern for the images, and limits the dataset to size 2^5, instead of 2^{16}. The density matrices generated from the quantum states are discrete, and allow for efficient and lossless compression to a more continuous latent space. In figure 4(A), the top-right image would have the following associated superposition state: $-0.25|0000\rangle - 0.25|1000\rangle - 0.25|0100\rangle - 0.25|1100\rangle - 0.25|0010\rangle + 0.25|1010\rangle + 0.25|0110\rangle - 0.25|1110\rangle - 0.25|0001\rangle - 0.25|1001\rangle + 0.25|0101\rangle - 0.25|1101\rangle - 0.25|0011\rangle - 0.25|1011\rangle - 0.25|0111\rangle - 0.25|1111\rangle$.

Having established our dataset and our methodology for distinguishing different quantum states for each image, we now utilize IBM's Qiskit programming language to produce these states for the simulator. Qiskit's simulator includes a built-in function for complex amplitude initialization, called Initialize [36]. This Initialize function first takes the desired initialized quantum state to the zero state $|0\rangle^{\otimes n}$ in the computational basis. The gate sequence that accomplishes this is then implemented backwards after resetting the qubits, taking the $|0\rangle^{\otimes n}$ quantum state to the desired initialized state. Hence, Initialize is not a unitary gate, but a resource intensive Qiskit instruction that iteratively disentangles qubits from the register one by one. Since manually assigning control-not and control-phase gates to get the desired state for each 4-qubit image can become cumbersome, we opted for using the Initialize function for our 4-qubit simulations.

Encoding of complex image datasets has been studied extensively [22–24], as well as using QRAM as a future prospect for algorithm implementation (QRAM currently hasn't been physically realized) [38, 39]. Generally speaking, for any arbitrarily large or complicated dataset, exp(n) operations would be necessary to encode the information into n qubits. Our focus for this paper is studying the efficiency in training and compressing quantum data using not only Qiskit's simulator, but IBMQ's quantum chips as well; efficiency in data loading via simulation has been studied by [13]. Hence, for our experimental realization, we opted for a dataset that is easily

implementable on a quantum chip with encoding via just a few CNOT and CZ gates, as discussed in the next section.

With a total 32 images, we split our dataset with 14 images for training, and 18 images for testing. We augmented the training images by duplicating the dataset, shuffling the list, and picking a small batch of images at a time to train, thereby enabling the training network to train on each image multiple times through one epoch. In total, we trained the network using 42 images with batch size of 7. All simulations were run for 40 epochs with a fixed learning rate of 0.05.

3.2. Quantum device implementation

Due to limitations in quantum hardware (number of qubits and qubit connectivity), we created a smaller 2×4 pixel image dataset for compression when working with IBMQ. This involved encoding 8 pixels into a 3-qubit superposition state, using the same phase encoding scheme as done for the 4-qubit images. The dataset consists of 10 training images with bar and stripe patterns, where the black/white pixels form horizontal bars, or vertical stripes, but never both. Sample of 6 images are shown in figure 4(B). The top-right image would have the following associated superposition state: $0.35355|000\rangle - 0.35355|100\rangle + 0.35355|010\rangle - 0.35355|110\rangle + 0.35355|001\rangle - 0.35355|101\rangle + 0.35355|011\rangle - 0.35355|111\rangle$. Similar to the 4×4 image dataset, we augmented this dataset to 20 total training images. We used the PQC shown in figure 3(B), and compressed the 3-qubit superposition state down to 2 qubits. For comparison, this image dataset was compressed with a simulator as well, using the same optimizer and training network hyper-parameters as for the quantum device. For this study, we used the IBMQ 'Casablanca' chip, which is one of the IBM Quantum Falcon Processors [19].

Unlike ideal simulators, we are constrained by limited connectivity and noisy qubits on real devices. Hence, experimental realization for QAE meant that generalizing data encoding for arbitrary image data sets was outside the scope of this paper. Similar to[21], we opted to tailor a specific data set whose images could be individually encoded via one to three applications of CNOT and CPHASE gates. To maximize the efficiency of the training network, the data encoding initialization circuit (our combination of CNOT and CPHASE gates), and the parameterized quantum circuit, were transpiled first before training the network on the quantum chip using Qiskit's Transpiler[36].

4. Results

Results from our simulations and device experiments are split into two subsections. The first covers the performance and run time of the compression algorithm for the simulated runs. The second subsection goes over the performance and run times of the algorithm for the IBMQ experiment.

4.1. Simulations

4.1.1. Compression efficiency

Visualizing an image undergoing compression and decompression allows us to gauge the performance of our compression algorithm. The original input image and the latent space are visualized via density matrices to show the discrete or continuous nature of the Hilbert space. For the 3 PQCs outlined in figure 3(A), each one was trained with the Qiskit simulator using layer values of $L = 3, 5, 7$, along with three compression ratios (4 qubits compressed to 3, 2, 1 qubit(s), where the ratio is defined as $n_{\text{initial}}/n_{\text{compressed}}$), giving a total of 27 full training simulations. We cover all experimental results when discussing figure 7 later on. Figure 5 illustrates sample of our simulation results for a single test image, shown as density matrices. The leftmost column shows the example 4×4 image and the associated density matrix visualized as a 16×16 matrix ($\rho = \sum_i p_i |\psi\rangle\langle\psi|$). The density matrices take on values in the range $\{-0.0625, 0.0625\}$, with ρ_{initial} taking discrete values $\pm 1/16$. Similar to the original QAE paper [18], we only show the real component of the latent and decompressed density matrices in order to compare to the original, which has no complex components to begin with. For our purpose, this visualization should work with the complex components as well.

The top third section of figure 5 shows the simulation results for Circuit 1. The six cells display the latent space and the decompressed density matrix of the example image for two different compression ratios and PQCs at depth of 7 layers. The size of the latent space corresponds with the compression ratio; for the 4 to 3 qubit compression, the latent space is an 8×8 matrix. In general, the latent space is $2^m \times 2^m$, where m is the number of compressed qubits. The forms of the latent space differ for each PQC, regardless of achieving a high fidelity compression. In each case, we decompress back to a 16×16 density matrix, with the aim of recreating the original density matrix shown in the leftmost column. The latent space is continuous, while the 16×16 density matrices is discrete. The discrete nature in the initial density matrix is what allows for data compression down to a continuous mixed state, potentially without losing information. Theoretically, the data meets the criteria for

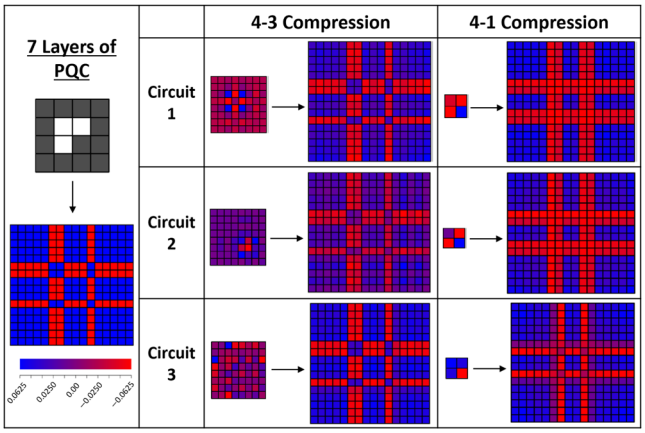

Figure 5. Simulations: density Matrices for single test image through different compression ratios and PQCs. We show a sample of six results from all our experiments, showing a visualization of the latent space, and the subsequent decompressed image via the real components of their density matrices.

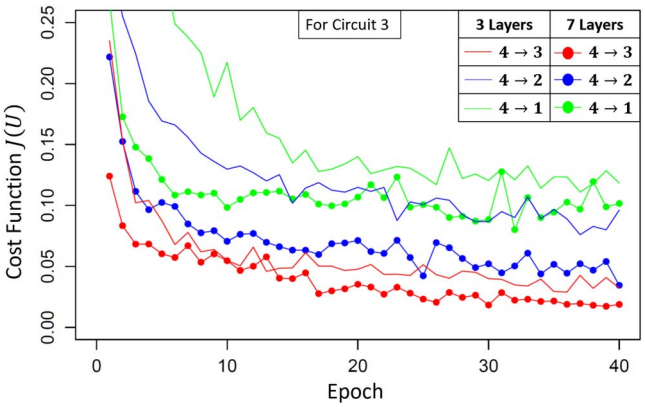

Figure 6. Simulations: loss curves from simulations for Circuit 3 with the three compression ratios, and layers $L = 3,7$. We minimize the cost function to approximately 0.02 for 4-to-3 qubit compression with 7 layers.

lossless compression even for the 1 qubit compression since we would have one linearly independent vector compressed down to a latent space size 2.

Hence theoretically, there should be no information loss across each compression ratio. In practice, figure 5 shows that the greater the compression ratio, the worse the compression algorithm performs We show later in figure 7 that conversely, increasing the number of layers of the PQC decreases loss of information, and improves compression efficiency. Figure 5 shows results from 7 layers of each PQC, where Circuit 3, the circuit with the best measurements for expressibility and entangling capability, performs best across different compression ratios for this image. Ultimately, for any image in the dataset, information loss is proportional to increase in compression, and inversely proportional to number of PQC layers.

This relationship can be seen more clearly in figure 6 for Circuit 3. The cost function relates to the amount of information lost from data compression via equation (1). Hence, minimizing the cost function is directly related to minimizing information loss, and true lossless compression would see the cost function go down to zero. Figure 6 shows loss curves from simulation of Circuit 3 for the full dataset. At the end of 40 epochs the smallest compression ratio (4 to 3 qubits), with 7 PQC layers, yields the least information loss, minimizing the cost function to 0.02.

While figure 5 visualizes density matrices for a single image, and provides an insight into how the compression algorithm performs across a sample of the 27 simulations, figure 7 gives further insight by quantitatively describing how well the PQCs performed on the full test dataset across all experiments. Figure 7 plots the fidelity between the original and decompressed images from the full test dataset. Here, we compare the direct results of compression efficiency from simulation per each test image. The box plots show the spread in fidelity measurements across the dataset, with the box containing 50% of the data, and the horizontal black bar as the median value. The highest fidelity measure across all simulations is 0.98, while the lowest is 0.65. For the

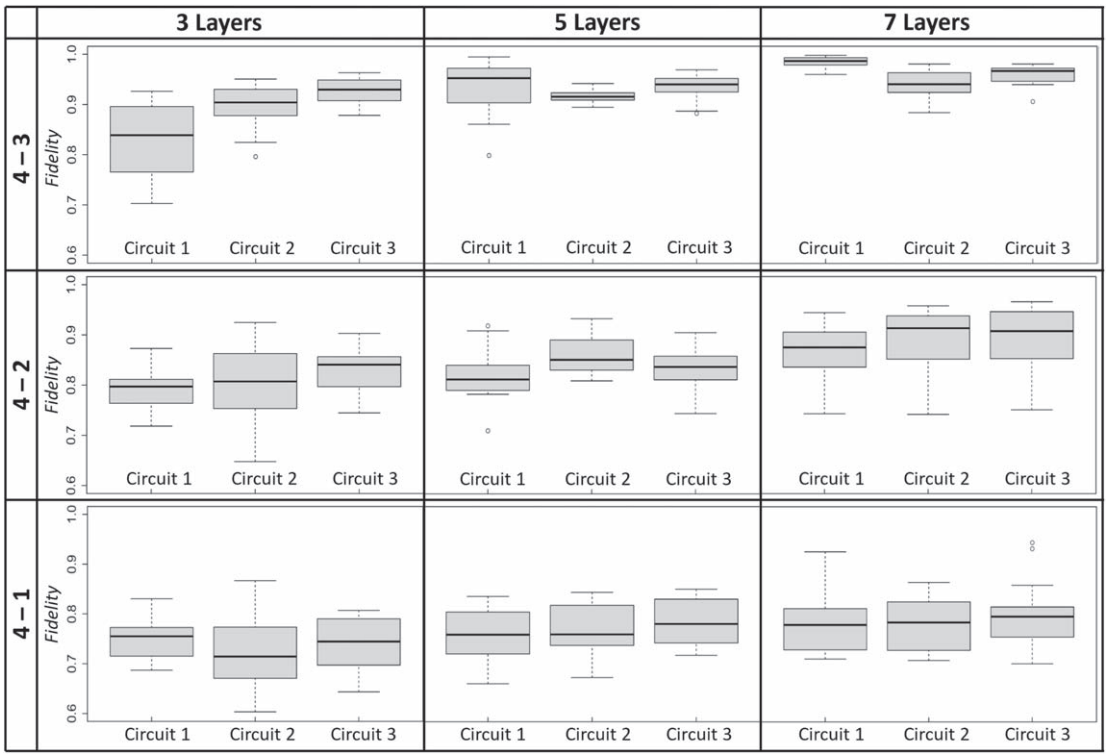

Figure 7. Simulations: fidelity measurements between the original and decompressed density matrices of the full test dataset for the 27 simulations. A score of 1.00 indicates that the original and decompressed image are identical. We show the spread in fidelity scores across the test dataset for each compression ratio, PQC, and number of layers. The circled dots in some of the box plots are marked as outliers (outside 1.5 times the interquartile range above the upper quartile and below the lower quartile).

original density matrix of an image ρ, and the decompressed density matrix σ, the fidelity is calculated using equation (9) [40]. The maximum fidelity of 1 means the final and initial image is identical, and the compression is lossless.

$$F(\rho, \sigma) \equiv Tr(\sqrt{\rho^{1/2}\sigma\rho^{1/2}}). \tag{9}$$

In figure 7, the general trend for PQCs with 3 layers (the leftmost column) shows Circuit 3 performing slightly better than the other two. Since the entangling capability and expressibility measures for Circuits 2 and 3 are more favorable compared to Circuit 1 (table 2), we see this reflected in the fidelity measures, where Circuit 2 and 3 generally outperform Circuit 1. This can clearly be seen in the simplest case of 4 to 3 qubit compression with 3 layers of PQC (top left of figure 7). This result agrees with Hubregtsen *et al* (2021), who found a strong correlation between classification accuracy of their circuit with expressibility of the circuit [41]. Even in the general case, Circuit 3 performs slightly better than the other two circuits given that its median fidelity measure is the highest or near highest in all cases. Additionally, aside from the simulation with the smallest compression ratio and the most number of PQC layers (top right of figure 7), Circuit 1 has the largest spread in fidelity measures. Increase in spread of fidelity measures generally corresponds with increase in compression ratios, but not with increase in number of layers, even if the median fidelity score improves with number of PQC layers.

While the increase in number of layers can improve the compression efficiency, each PQC behaves differently to this increase, as evident in figure 8. The overall compression efficiency of Circuit 1 is most susceptible to change with number of layers, shown as red lines in the plot. Increasing the number of layers for Circuit 2 and 3 (blue and green lines, respectively) does not significantly change the compression efficiency of the algorithm using the simulator on the full test data set, albeit increasing the running time.

4.1.2. Run time

One major focus of this study was to successfully compress quantum data near-losslessly, and quantify how changing the training network hyper-parameters affect information loss and compression efficiency. However, in practice, one must also take into account the running time of the algorithm in question when working with real data. In figure 9, for the 4 to 3 qubit compression, we compare simulation data for the total time taken to train the three PQCs across 40 epochs, with 10 iterations of the optimizer. Circuit 1 is the most practical of the three, taking the least time to train and minimize the cost function. It also has the least spread in training time

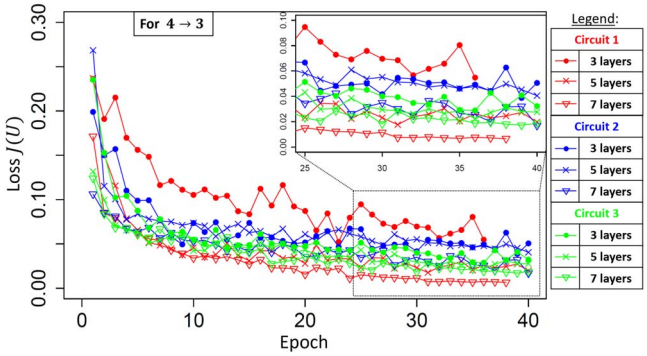

Figure 8. Simulations: loss curves from simulations for all three circuits for the 4-to-3 qubit compression, with layers $L = 3, 5, 7$. The spread in the loss curve is largest with Circuit 1 (red), where increasing the number of layers has the largest effect in minimizing the cost function.

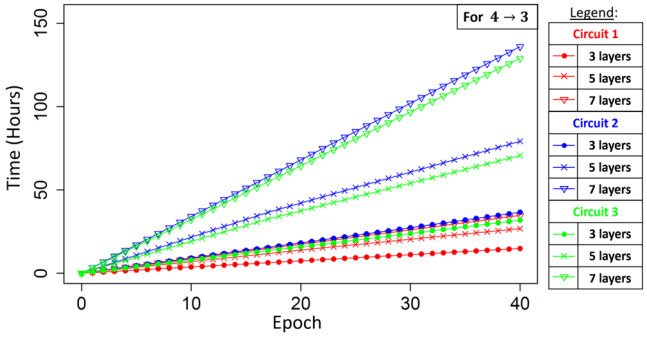

Figure 9. Simulations: total time (in hours) across 40 epochs for the 4-to-3 qubit compression with the 3 PQCs. Circuit 1 has the smallest spread. Circuit 3 with 3 layers is has a faster training time than Circuit 1 with 7 layers, while Circuit 2 is slowest overall.

Table 3. Simulations: avg. total time per epoch (Hours).

	Circuit 1	Circuit 2	Circuit 3
3 Layers	0.37	0.91 ± 0.01	0.79
5 Layers	0.67 ± 0.03	1.98 ± 0.15	1.76 ± 0.01
7 Layers	0.87 ± 0.01	3.40	3.22

when increasing layers. These characteristics are attributed to its shallow gate depth, which is also reflected in poorer entanglement and expressibility scores. Conversely, Circuit 2 is the slowest circuit across all layers, with the slowest iteration of Circuit 1 being faster than the fastest iteration of Circuit 2.

To study the applicability between the other two circuits, 1 and 3, we examine two competing aspects between the two PQCs: speed versus performance. The median fidelity measure for Circuit 3 in the the top left box (3 layers) of figure 7 is 0.92, while the median fidelity measure for Circuit 1 in the top right box (7 layers) is 0.98. That is approximately a 6% improvement in the median fidelity for Circuit 1. Consequently, the average time per epoch between these two circuits, shown in table 3, shows Circuit 3 (3 layers) is approximately 9% faster than Circuit 1 (7 layers). So for simulated quantum data compression, to minimize the run time while maximizing performance, Circuit 3 with fewer layers is more favorable. Generally, PQCs with better expressibility and entangling capability measures, like Circuit 3, tend to have higher gate depths, and are only suited to simulated data compression. Provided a 6% increase in information loss can be tolerated by a particular dataset or problem, simulating a more complex PQC with few layers becomes useful. Training networks run on quantum devices, however, requires both faster run times and shallower gate depths, with a preference for overall shallower gate depth over number of layers implemented. Hence, Circuit 1's design (shown in figure 3(B)) was chosen to run data compression on the IBMQ quantum device.

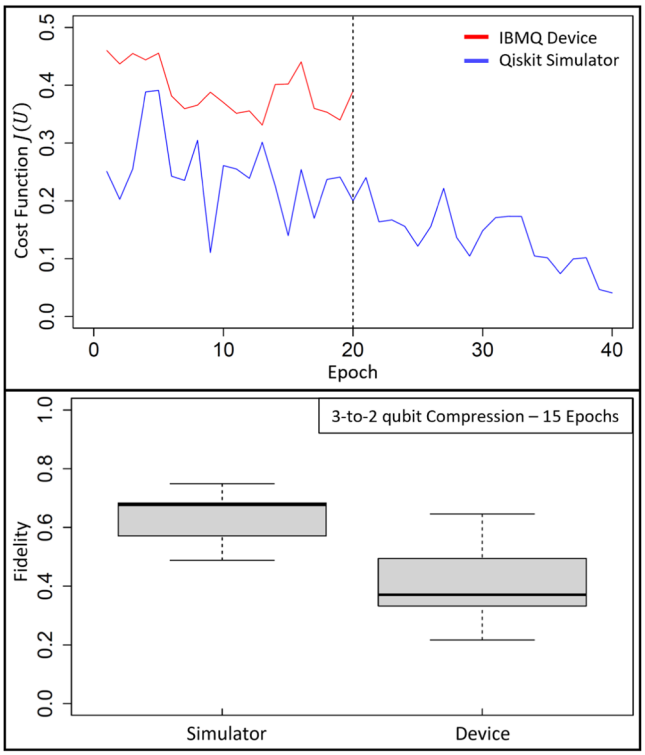

Figure 10. Top: loss curves from IBMQ Device and simulator for 3-to-2 qubit compression with 1 iteration of the optimizer, learning rate of 0.05, and 3 layers of PQC in figure 3(B). Bottom: fidelity measures from IBMQ Device and simulator of test dataset with parameters from cost function minima at the end of 20 epochs.

4.2. IBMQ device

4.2.1. Compression efficiency

Due to the longer training time on the quantum device - primarily due to Queue Time - the network was trained on the device for 20 epochs. Figure 10 (top) shows the loss curves for the device and simulator, with the vertical line at 20 epochs. We compare the two to show the difference in compression efficiency between the latest superconducting quantum hardware and a simulator. At the end of 20 epochs, the minima of the cost function for the device was over 3 times higher than the cost function for the simulator. The simulator trained the network out to 40 epochs, whereby the cost function was minimized to 0.04, indicating that the 3-to-2 qubit compression of this 2×4 dataset is indeed near-lossless with this particular set of hyper-parameters. Figure 10 (bottom) shows the fidelity measurements after training with the simulator and the device for 20 epochs. The median fidelity for the IBMQ device is 0.37, while the median fidelity for the simulator is 0.68, around 1.84 times the value.

4.2.2. Run time

For our experiment on IBMQ, we have to calculate the time cost in more detail. This requires a closer look at the number of jobs the training network sends to the device. In our case, three layers of the circuit from figure 3(B) means $N_{\text{params}} = 12$, and from the training dataset, $N_{\text{images}} = 20$. So number of jobs sent to device is given by equation (8) as:

$$N_j = ((2 \cdot 12 + 1) \cdot 20) \cdot N_{\text{iter}} = 500 \cdot N_{\text{iter}}. \tag{10}$$

So for the simplest PQC with 3 layers, as well as our simple image data set compressing 3 qubits down to 2, the number of jobs with just one iteration of the optimizer means 500 jobs were sent to the device per epoch. For 40 epochs, that is 20,000 total jobs sent to the device. Each job is one run of the entire circuit for a set amount of shots (set to a maximum of 8192 shots). The time per job for the training network run on the device, with one iteration of the optimizer, is broken up into several processes, shown in table 4. Omitting the queue time for the job sent to the device, since it can vary greatly depending on number of users on the device, the average time per job is then 22.18 seconds. For comparison, we ran this dataset through the compression algorithm using a simulator with 1 iteration of the optimizer. The simulator took an average of 7.46 seconds per job (one run of the circuit with 8192 shots). Therefore, the device takes roughly 3 times longer per job compared to the simulator for the same dataset, PQC, and hyper-parameters.

Table 4. IBMQ quantum chip: avg. job time (seconds).

Create time	Queue time	Run time	Total time
0.66	46.09	20.28	68.27

5. Summary and conclusion

In the Noisy Intermediate Scale Quantum (NISQ) era, qubits are expensive resources, and multi-qubit operations are fragile. By leveraging classical dimensional reduction techniques such as autoencoders, quantum data can be compressed to fewer qubits. Reducing the dimensionality of this problem can allow us to utilize fewer qubits, aiding in more efficient use of resources. However, quantum autoencoders face challenges in efficient data compression even if data sets match the theoretical limit for lossless compression [17]. Application of quantum data compression needs to address consequences from choice of data, parameterized quantum circuits, and optimizer. In this paper, we successfully demonstrate near-lossless quantum data compression using a simulator. We chose datasets with linearly independent vectors following specific patterns, visualized as black-and-white pixel images. Wavefunctions generated from encoding these images create density matrices with discrete values, allowing for efficient compression down to continuous latent spaces. Three PQCs were tested on the quantum autoencoder framework [18], using Qiskit's Analytical Quantum Gradient Descent optimizer. The PQCs were chosen for different gate depths, number of parameters, number of two-qubit gates, and their performance quantified using expressibility and entangling capability measures [20].

While in principle one can make a direct connection between real-device trainability and expressibility, thereby allowing for more general predictions in PQC performance[33], we found that hardware-specific challenges tailored the scope of our paper to studying trainable PQCs on IBMQ's quantum chips. For our experimental realization, we found that contrary to simulations, complex, more expressible circuits tended to do poorly due to larger gate depth. Hence on IBMQ chips, it became balancing act of designing a circuit that had high enough expressibility, but low gate depth. For our experimental studies, we don't discuss in detail predictability in compression efficiency as a function of expressibility, since gate depth of the PQCs ended up being far more important than any other metric. We therefore don't make a direct connection between real-device trainability and expressibility, but instead connect trainability to circuit depth, compression ratio, and choice of classical optimizer. Our results show that experimentally, these end up being more influential metrics for data compression for the specific architecture of IBMQ devices.

The summary of our compression results is as follows: our results show that while all three PQCs manage to efficiently compress the test data set with near-lossless compression using a simulator, their performance depended on the compression ratio, number of layers, and running time. For example, Circuit 3 with 3 layers had a 6% higher cost function minimum compared Circuit 1 with 7 layers, but was 9% faster in training time. Only if a 6% increase in information loss can be tolerated, does a more complex PQC with few layers becomes useful, since the speed-up in training can be beneficial. However, for a fixed number of layers, training on a quantum device finds a faster circuit more useful than an efficient one. So although circuits with higher expressibility and entangling capability, like Circuits 2 and 3, generally perform better with larger compression ratios, their complexity leads to noise washing out our results. For devices with limited qubits and connectivity, a simple circuit with smaller gate depth, like Circuit 1, is more NISQ-friendly.

The summary for run times is as follows: run times and training performance between an IBMQ quantum device and the simulator was done using the 3-qubit version of Circuit 1 for the PQC, shown in figure 3(B). Using a simulator with this PQC on our smaller data set, and one iteration of the optimizer, we compressed from 3 to 2 qubits near-losslessly by minimizing the cost function to 0.04, as seen in figure 10 (top). When run on an IBMQ device, the loss curve minima was a little over 3 times larger than the minima for the simulator after 20 epochs. Additionally, the device took three times longer per job compared to the simulator, when omitting queue time. Ideally, increase in total run time due to the job queue can be mitigated by reserving time on the device. We are still at a stage in the NISQ era where not only the noise and limited connectivity affect quantum algorithms, but processes within classical optimizers and data encoding as well.

In conclusion, efficient compression of quantum data on a quantum device will require careful construction of PQCs based on parameters examined in this paper, as well as choice of classical optimizer. In this paper, we examined the performance of our compression algorithm for different PQCs, and IBMQ's Analytical Quantum Gradient Descent optimizer.

Future work will involve integrating other gradient descent optimizers into Qiskit that promise a faster convergence, such as the Quantum Natural Gradient proposed by Stokes *et al* (2019) [37]. Upon compressing quantum data efficiently, future work will also involve decompressing latent states into any desired output state by implementing a parameterized circuit for decompression as well. If these compressed states have missing

information from the original input, any operation performed using the latent space would compound inaccuracies in the final result. It is essential that we understand how quantum autoencoders operate on NISQ devices, specifically in terms of run times, and compression efficiency, as shown by our experimental results. Hence, the goal of this paper to practically achieve lossless compression is important for future practical applications of quantum data compression.

Acknowledgments

The views expressed are those of the authors and do not reflect the official guidance or position of the United States Government, the Department of Defense or of the United States Air Force. The appearance of external hyperlinks does not constitute endorsement by the United States Department of Defense (DoD) of the linked websites, or the information, products, or services contained therein. The DoD does not exercise any editorial, security, or other control over the information you may find at these locations.

Data availability statement

The data cannot be made publicly available upon publication because no suitable repository exists for hosting data in this field of study. The data that support the findings of this study are available upon reasonable request from the authors.

Author declarations

The authors have no conflicts of interest to disclose.

ORCID iDs

Saahil Patel https://orcid.org/0000-0002-5646-3472
Daniel Koch https://orcid.org/0000-0002-7664-9194

References

[1] Castells M 1996 *The Rise of the Network Society* (Blackwell Publishers)
[2] Meng Q, Catchpoole D, Skillicorn D and Kennedy P J 2017 Relational autoencoder for feature extraction *2017 International Joint Conference on Neural Networks (IJCNN)* 364
[3] Vincent P, Larochelle H, Bengio Y and Manzagol P 2008 Extracting and composing robust features with denoising autoencoders *ICML 2008: Proceedings of the XXV International Conference on Machine Learning* 1096
[4] LeCun Y 1987 Modeles connexionistes de l'apprentissage *PhD thesis* Universite de Paris VI
[5] Vincent P, Larochelle H, Lajoie I, Benjio Y and Manzagol P 2010 Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion *J. Mach. Learn. Res.* **11** 3371
[6] Bourlard H and Kamp Y 1988 Auto-association by multilayer perceptrons and singular value decomposition *Biol. Cybern.* **59** 291
[7] Hinton G E and Zemel R S 1994 Autoencoders, minimum description length and helmholtz free energy *NIPS'93: Proceedings of the VI International Conference on Neural Information Processing Systems* 3
[8] Gallinari P, LeCun Y, Thiria S and Fogelman-Soulie F 1987 Distributed associative memories: a comparison *Proceedings of COGNITIVA 87 (Paris, La Villette)*
[9] Kingma D P and Welling M 2014 *Auto-Encoding Variational Bayes 2nd International Conference on Learning Representations, ICLR 2014, Conference Track Proceedings (Banff, AB, Canada, April 14-16, 2014)*
[10] Khoshaman A, Vinci W, Denis B, Andriyash E, Sadeghi H and Amin M H 2019 Quantum variational autoencoder *Quantum Sci. Technol.* **4** 014001
[11] Plesch M and Bužek V 2010 Efficient compression of quantum information *Phys. Rev.* A **81** 032317
[12] Bondarenko D and Feldmann P 2020 Quantum autoencoders to denoise quantum data *Phys. Rev. Lett.* **124** 130502
[13] Bravo-Prieto C 2021 Quantum autoencoders with enhanced data encoding *Machine Learning: Science and Technology* **2** 035028
[14] Sebastien P, Nairi U, Simone S, Mark H, Tommaso M and Peter M 2018 Image classification with quantum pre-training and auto-encoders *International Journal of Quantum Information* **16** 1840009
[15] Sakhnenko A, O'Meara C, Ghosh K J B, Mendl C B, Cortiana G and Bernabe-Moreno J 2022 Hybrid classical-quantum autoencoder for anomaly detection *Quantum Machine Intelligence* **4** 27
[16] Wilde M 2016 *From Classical to Quantum Shannon Theory* (Cambridge University Press)
[17] Huang C, Ma H, Yin Q, Tang J, Dong D, Chen C, Xiang G, Li C and Guo G 2020 Realization of a quantum autoencoder for lossless compression of quantum data *Phys. Rev.* A **102** 032412
[18] Romero J, Olsen J P and Aspuru-Guzik A 2017 Quantum autoencoders for efficient compression of quantum data *Quantum Sci. Technol.* **2** 045001
[19] IBMQ 2021 https://quantum-computing.ibm.com/
[20] Sim S, Johnson P D and Aspuru-Guzik A 2019 Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms *Adv. Quantum Technol.* **2** 1900070

[21] Tacchino F, Macchiavello C, Gerace D and Bajoni D 2019 An artificial neuron implemented on an actual quantum processor *npj Quantum Information* **5** 26
[22] Sanchez M A M, Sun G and Dong S 2019 Correlation property of multipartite quantum image *Int. J. Theor. Phys.* **58** 3773–96
[23] Yan P, Iliyasu A M and Venegas-Andraca S E 2015 A survey of quantum image representations *Quantum Inf. Process.* **15** 1
[24] Le P Q, Dong F and Hirota K 2011 A flexible representation of quantum images for polynomial preparation, image compression, and processing operations *Quantum Inf. Process.* **10** 63–84
[25] Buhrman H, Cleve R, Watrous J and Wolf R D 2001 Quantum fingerprinting *Phys. Rev. Lett.* **87** 167902
[26] Schuld M, Sinayskiy I and Petruccione F 2015 An introduction to quantum machine learning *Contemp. Phys.* **56** 172
[27] Zyczkowski K and Kus M 1994 Random unitary matrices *J. Phys. A: Math. Gen.* **27** 4235
[28] Kullback S and Leibler R A 1951 On information and sufficiency *Ann. Math. Stat.* **22** 79
[29] Meyer D A and Wallach N R 2002 Global entanglement in multiparticle systems *J. of Math. Phys.* **43** 4273
[30] Love P J, van den Brink A M, Smirnov A Y, Amin M H S, Grajcar M, Il'ichev E, Izmalkov A and Zagoskin A M 2007 A characterization of global entanglement *A.M.: Quantum. Inf. Process.* **6** 187
[31] Brennen G K 2003 An observable measure of entanglement for pure states of multi-qubit systems, Quantum *Inf. Comput.* **3** 619
[32] Preskill J 2018 Quantum computing in the nisq era and beyond *Quantum* **2** 79
[33] McClean J R, Boixo S, Smelyanskiy V N, Babbush R and Neven H 2018 Barren plateaus in quantum neural network training landscapes *Nat. Commun.* **9** 4812
[34] Mitarai K, Negoro M, Kitagawa M and Fujii K 2018 Quantum circuit learning *Phys. Rev.* A **98** 032309
[35] Schuld M, Bergholm V, Gogolin C, Izaac J and Killoran N 2019 Evaluating analytic gradients on quantum hardware *Phys. Rev.* A **99** 032331
[36] Qiskit (0.33.1) documentation, https://qiskit.org/documentation/ (2021)
[37] Stokes J, Izaac J, Killoran N and Carleo G 2020 Quantum natural gradient *Quantum* **3** 269
[38] Weigold M, Barzen J, Leymann F and Salm M 2021 Encoding patterns for quantum algorithms *IET Quantum Communication* **2** 141
[39] Giovannetti V, Lloyd S and Maccone L 2008 Encoding patterns for quantum algorithms *Phys. Rev. Lett.* **100** 160501
[40] Nielsen M A and Chuang I A 2011 *Quantum Computation and Quantum Information* X eda (Cambridge University Press) Chap. 9
[41] Hubregtsen T, Pichlmeier J, Stecher P and Bertels K 2021 Evaluation of parameterized quantum circuits: on the relation between classification accuracy, expressibility and entangling capability *Quantum Mach. Intell.* **3** 9