Clustering and enhanced classification using a hybrid quantum autoencoder

Maiyuren Srikumar, Charles D. Hill, and Lloyd C.L. Hollenberg

1School of Physics, University of Melbourne, VIC, Parkville, 3010, Australia.
2School of Mathematics and Statistics, University of Melbourne, VIC, Parkville, 3010, Australia.

(Dated: January 13, 2022)

Quantum machine learning (QML) is a rapidly growing area of research at the intersection of classical machine learning and quantum information theory. One area of considerable interest is the use of QML to learn information contained within quantum states themselves. In this work, we propose a novel approach in which the extraction of information from quantum states is undertaken in a classical representational-space, obtained through the training of a hybrid quantum autoencoder (HQA). Hence, given a set of pure states, this variational QML algorithm learns to identify – and classically represent – their essential distinguishing characteristics, subsequently giving rise to a new paradigm for clustering and semi-supervised classification. The analysis and employment of the HQA model are presented in the context of amplitude encoded states – which in principle can be extended to arbitrary states for the analysis of structure in non-trivial quantum data sets.

I. INTRODUCTION

In recent years, the amalgamation of Quantum Mechanics and Machine Learning (ML) has instigated extensive research into the field of Quantum Machine Learning (QML) [1]. With fault-tolerant quantum computation far from realisable in the near future, one of the areas in which researchers are looking for quantum advantage are variational algorithms. These variational approaches have demonstrated robustness in the regime of noisy intermediate-scale quantum (NISQ) devices, thus being a contender to first demonstrate quantum advantage [2]. With applications in QML variational methods most commonly employ a parameterised quantum circuit (PQC) [3, 4], where parameters are classically optimised in a feedback loop routine between optimiser and PQC.

Generally, QML methods can be categorised into two distinct groups: (i) models that obtain advantage through the learning of classical data – once embedded into a quantum system – or (ii) the learning of purely quantum data sets. This paper focuses on the latter task, and proposes an approach in which the learning of quantum states can be undertaken in a classical representational space. This allows for novel approaches to cluster and classify quantum states based on their classical representations. Such representations will be formed from the employment of – what we have termed – a hybrid quantum autoencoder (HQA), illustrated in Figure 1c.

Classically, an autoencoder is a specific artificial neural network (ANN) architecture that is trained to return its input as its output, whilst undergoing a crucial funnelling of its degrees of freedom [6] (shown in Figure 1a). This funnelling process generates compressed representations of data points that belong to a particular group of data.

The autoencoder is composed of two maps: an encoder, e, and decoder, h, that are both approximated using ANNs (a discussion on ANNs relevant to this work is presented in Appendix A). The encoder receives data \( \tilde{x} \) and outputs a lower dimensional latent vector, \( \tilde{z} \), such that, \( \tilde{z} = e(\tilde{x}) \). The decoder performs the inverse, \( \tilde{x}' = h(\tilde{z}) \). Subsequently, the autoencoder (both h and e) is trained to approximate \( \tilde{x} \approx h(e(\tilde{x})) \) for any \( \tilde{x} \) in the data set.

FIG. 1. (a) An autoencoder, where black dots represent neurons, and lines represent their weighted inputs/outputs (Appendix A). (b) Structure of a quantum autoencoder as initially proposed in [5]. \( \mathcal{E} \) and \( \mathcal{D} = \mathcal{E}^\dagger \) are unitary operators. Here the latent vector is an inaccessible compressed quantum state. (c) Structure of the hybrid quantum autoencoder (HQA). The encoder is composed of a quantum neural network (QNN) that is measured to return a classical latent vector, \( \xi \). The decoder employs an AN and a QNN to return quantum state. The output of all autoencoders, (a), (b) & (c) are trained to match any input from a particular data set.

* srikumarm@unimelb.edu.au
† cdhill@unimelb.edu.au
‡ lloydch@unimelb.edu.au
This would be trivial for if \( \vec{x} \) and \( \vec{z} \) had equal dimensionality; however, in the case where \( \dim(\vec{x}) > \dim(\vec{z}) \), the autoencoder is forced to encode the most important aspects of the input, \( \vec{x} \), into the latent space. The latent vector, \( \vec{z} \in \mathcal{Z} \), is in essence a representation of \( \vec{x} \in \mathcal{X} \) in the lower dimensional representational space of \( \mathcal{Z} \).

The main advantage of an autoencoder is that it is able to learn complex compression strategies through an unsupervised learning process. Such a process requires a human to have minimal prior information regarding the data set. Hence, it is often used in the context of denoising and compressing data that lack obvious methods of dimensionality reduction. Quantum autoencoders (Figure 1b), discussed further in Section (II B), are direct analogues and hence provide non-trivial compression maps for quantum states to a subset of its Hilbert space.

In essence, both ML and QML algorithms exploit the tendency that data aggregates in a low dimensional sub-manifold over the vast space of possible data points. We describe the data as lying on a sub-manifold to emphasise the fact that infinitesimal tangential translations, result in remaining on the sub-manifold. It should be noted that, although in mathematics a manifold has a more formal definition, in ML it is used to describe a set of points that can be well approximated by considering only a small number of degrees of freedom, embedded in a higher-dimensional space [6]. This manifold hypothesis is essential so that data points have a neighbourhood of highly similar examples that can be accessed by applying small transformations to traverse the manifold. Hence, the main objective of a quantum autoencoder is to learn the sub-manifold that describes a particular set of quantum states. The HQA in particular, represents the sub-manifold in an accessible classical vector space.

Quantum states can be represented as positive semi-definite operators on a complex Hilbert space, \( \mathbb{C}^{2^n} \), known as density matrices. Theoretically, one can imagine putting the elements of the density matrix through a classical ML algorithm to find similarities between quantum states or even classify states. However, the information stored in quantum states is notoriously inaccessible without exponential resources to characterise each (through quantum state tomography [7, 8]). The HQA, in Figure 1b, gets around this by using an encoder that is trained to output classical information about important aspects of a quantum state. This quantum to classical transformation is extreme in its dimensionality reduction, as for a pure input state we have \( \mathbb{C}^{2^n} \rightarrow \mathbb{R}^v \) where \( v \sim O(n) \) is the dimension of the classical space by measuring the \( n \) number of qubits. Hence one can see that if the state is able to be reconstructed from the classical space, then the classical space can only describe a relatively small set of quantum states. Nonetheless, the classical real space represents a manifold in \( \mathbb{C}^{2^n} \) that can be learned with the construction of the HQA. It will be seen that it is this perspective that distinguishes the HQA from the QAEs explored in literature thus far.

This paper is structured such that we first provide background into quantum neural networks and quantum autoencoders in Section (II); before then constructing the HQA in Section (III). This is then followed by applications of clustering and classification in Section (IV), including results from numerical simulations.

## II. BACKGROUND

### A. Quantum Neural Networks

Quantum neural networks (QNNs) are the extension of ANNs to QML. The precise form of the QNN, however, is quite non-trivial as it would need to take advantage of unique quantum mechanical properties while also retaining the non-linear functional features of classical ANNs [9]. Hence, there are various proposals for QNN designs that claim similar non-linear dissipative dynamics of ANNs, but are yet to present clear quantum advantage [10–13]. In this paper, the implementation of the HQA will use the simplest design of a QNN: a PQC coupled with a specified observable. The choice of QNN, however, is arbitrary to the overall approach of the HQA (as shown in Figure 1c). Hence a fair comparison of QNN complexity and expressibility – for the construction of the HQA specifically – is left for future work.

PQCs form the basis of hybrid quantum-classical algorithms that optimise a quantum circuit with respect to a problem-dependent cost function [14]. The optimisation is performed classically to determine a better estimate of parameters which define a variational circuit. The optimisation in this work is performed using the parameter shift rule [15], elaborated in Appendix B.

Since any quantum circuit can be defined as a gate sequence \( U(\theta) \), the \( m \) parameters of the circuit are the set of \( \theta_i \) which parameterise the unitaries. We define the measurement of a PQC as a function \( f : \mathbb{R}^m \rightarrow \mathbb{R} \), mapping the gate parameters to an expectation value,

\[
f_{\rho_0}(\theta) := \text{Tr} \left[ U(\theta) \rho_0 U(\theta)^\dagger B \right] := \text{Tr} [ \rho(\theta) \hat{B} ],
\]

where \( \hat{B} \) is a predetermined observable (most commonly a Pauli-\( Z \)) and \( \rho_0 \) is an initial arbitrary state of the circuit. For consistency, the state before measurement will have notation, \( \rho(\theta) = U(\theta) \rho_0 U(\theta)^\dagger \) as evident from the second line in equation (1). It should be noted that this functional form of the variational circuits hides the fact that on real devices, repeated measurements of the circuit are required to obtain this expectation value. As a result, there is a natural statistical uncertainty to the estimated \( f \), determined by the number of samples taken of the circuit. Caution is thus required when constructing algorithms that require arbitrarily large numerical precision of \( f \). These algorithms may have promising results in state-vector simulations, but become computationally infeasible to measure on real quantum devices, requiring an exponentially large number of samples.

The optimisation of PQCs remain an area of research as it is not clear that employing classical optimisers will
result in optimal solutions for quantum cost function landscapes. This has given rise to works suggesting optimisers that are aware of the underlying quantum structure of quantum states [16–18]. Furthermore, there exist limitations of barren plateaus in deep PQC-based algorithms, as initially realised in [19]. Here, the exponential suppression of gradient with increasing depth of circuit, has also been shown to be linked to the expressibility of PQCs [20]. In addition to barren plateaus, PQCs are seen to exhibit narrow gorges [21] which is the occurrence of the existence of cost landscape minima in narrow wells that get steeper with increasing depth. The effects are not only seen with an increase in qubits, but arise due to entanglement [22], certain cost functions [21, 23], and noise [24]. These phenomena clearly have implications on QAEs that employ PQCs, hence it is important that an implementation of the HQA is efficient in its circuit depth – i.e. $\mathcal{O}(n)$. This is largely dependent on the chosen quantum data set $\{\rho_a\}$, and the expressibility of the particular QNN component shown in Figure 1c.

B. Quantum autoencoders

Many recent works involving quantum autoencoders (QAEs) build on the structure first proposed by Romero et al. [5], using shallow PQCs to compress quantum states. In [25], QNNs are of the form in [26] to construct a QAE that successfully denoises Greenberger-Horne-Zeilinger states which are subject to spin-flip and random unitary noise errors. In [27], a QAE is constructed using approximate quantum adders that are obtained with classical genetic algorithms as opposed to more commonly used gradient methods for parameter optimisation.

All such applications work on the process of funnelling quantum states into a lower dimensional Hilbert space. This naturally returns compressed representations that disregard both stochastic noise fluctuations and irrelevant degrees of freedom. It is important to note that the set of states are assumed to have support on a subset of its Hilbert space, $\mathcal{S} \subset \mathcal{H}$. The existence of such support is not guaranteed, but is instead common in many sets of quantum states, due to symmetries inherent to physical processes. For example, in [5] a QAE is classically simulated to show the compression of ground states of the Hubbard model and molecular Hamiltonians.

QAEs have been experimentally realised for the compression of qutrits with photons in [28] and the compression of two-qubit states into two single-qubit states in [29]. Furthermore, experimental realisation of the QAE via quantum adders has been shown in [30]. These QAEs are promising, but are nonetheless distinct to the HQA proposed in this paper. Specifically, the HQA aims to generate a classical representation of the $\mathcal{S}$ sub-manifold for classical analysis. Hence, not only can the data be compressed, but the compressed representation is an accessible classical vector that can be analysed.

There are clear limitations on an autoencoder’s ability to compress data. On the compression rate of QAEs, there exist not only a fundamental limit due to the degrees of freedom in a dataset, but a quantum limitation related to the von Neumann entropy of the density operator representing the ensemble of training states [5]. In [31], it is further elaborated that the compressibility is related to the eigenvalues of the weighted ensemble density matrix. Crucially, this theoretical limit is intrinsic to all possible compression strategies and QAEs.

III. THE HYBRID QUANTUM AUTOENCODER

A. Design

This paper proposes a novel variation of the QAE that we have termed a Hybrid Quantum Autoencoder (HQA). The hybrid nature of this model arises from the incorporation of both ML, in the form of classical ANNs, as well as QML, through the use of PQC-based QNNs. Figure 2 illustrates the overall design of the model which is a combination of (i) an encoder that takes a quantum state from Hilbert space $\mathcal{H}_2^{\otimes n}$ to a subset of the real vector space $\mathcal{V}$ of dimension $v = \dim(\mathcal{V})$, and (ii) a decoder that performs the inverse of such an operation. In general, though quantum states are positive semidefinite (trace= 1) operators $\rho$ on $\mathcal{H}_2^{\otimes n}$, the HQA is equipped to identify only pure states. Hence, we associate the vector $|\psi_a\rangle$ to the pure state $\rho_a := |\psi_a\rangle \langle \psi_a|$. Mathematically, the encoder and decoder have the form of a map,

$$\mathcal{E} : \mathcal{H}_2^{\otimes n} \rightarrow \mathcal{V}, \quad \text{where} \quad \mathcal{E}(|\psi_{in}\rangle) = \xi,$$

$$\mathcal{D} : \mathcal{V} \rightarrow \mathcal{H}_2^{\otimes n}, \quad \text{where} \quad \mathcal{D}(\xi) = |\psi_{out}\rangle,$$

where $\mathcal{V} = [-1, 1]^v$ is termed the latent space and $\xi \in \mathcal{V}$ is referred to as the latent vector – analogous to the terminology used when dealing with classical autoencoders. This vector is in essence a classical representation of the quantum state – a perfect representation if $\mathcal{D} \circ \mathcal{E}(|\psi_{in}\rangle) = |\psi_{in}\rangle$ is achieved. This indicates that the information of state $|\psi_{in}\rangle$ was preserved in the latent vector to then be recreated without any loss of information. In information theory, this is lossless encoding of information into a compressed latent space.

Though the functional form of the encoder and decoder are defined, the models themselves have not been specified. As seen in Figure 2, the encoder $\mathcal{E}$ is a PQC parameterised by a vector, $\alpha$. The PQC receives some state $|\psi_{in}\rangle$, and applies unitary $U_1(\alpha)$ on the combined system of the input state with $(v - n)$ ancilla qubits. From this circuit, the $Z$ expectation value of every qubit is measured, to then form the latent vector $\xi$,

$$\mathcal{E}^{(i)}(|\psi_{in}\rangle) = \xi^{(i)} = \langle \tilde{\psi}_{in} | Z_i U_1(\alpha) | \tilde{\psi}_{in} \rangle,$$

where $|\tilde{\psi}_{in}\rangle = |\psi_{in}\rangle \otimes |0\rangle^{\otimes (v-n)}$, $Z_i$ is the Z-Pauli operator acting on the $i^{th}$ qubit and $\alpha$ parameterises the PQC.
that will be optimised. The set of $\xi = (\xi^{(1)}, ..., \xi^{(v)})$ forms the latent vector which is identified as the classical representation of an input quantum state. Following the encoder, the decoder is extremely similar to the one defined in equation (4). However, this time it is a mapping that returns a quantum state when given a latent vector, \[ \mathcal{D}_A(\xi) = \mathcal{U}_2(f(\xi)) \ket{0} \ket{\psi_0}, \] \[ (5) \]

where $f : \mathcal{V} \rightarrow \mathbb{R}^p$ is the functional from of the ANN that is parameterised by $\mathbf{w}_{ij}^{(1)}$ (discussed in Appendix A). Training changes only the weights of the ANN and not the parameters of the PQC. The PQC parameters are designed to be the output of the ANN, $\ket{\psi_0}$ in Eq. (5) is an appropriate $n$-qubit ansatz for the type states involved. In this paper, we simply take $\ket{\psi_0} = \ket{0}^\otimes n$, which requires no additional operations for ansatz preparation.

It is important to note, both encoder and decoder have hyper-parameters (such as the number of parameters that define the PQCs, the number of neurons and depth of the ANN, etc.) for which one must optimise. From here on we will assume that these hyper-parameters are accounted, realising that there is possible future work in rigorously addressing the exact optimisation for this HQA model.

Now that we have defined the encoder and decoder, the HQA model $\mathfrak{A}$, is the combination defined as, \[ \mathfrak{A} : \mathcal{H}^\otimes n \rightarrow \mathcal{H}^\otimes n, \quad \mathfrak{A}(\ket{\psi_{in}}) = \mathcal{D} \circ \mathcal{E}(\ket{\psi_{in}}), \] \[ (6) \]

where we will refer to the output of the HQA as $\mathfrak{A}(\ket{\psi_{in}}) = \ket{\psi_{out}}$. Though the HQA looks as though it is a single run through both the encoder and decoder, there is an implicit sub-routine for the encoder where the PQC must be sampled multiple times to obtain $\xi$.

Now that the components of the HQA have been pieced together, the model is trained to copy the input such that $\ket{\psi_{out}} \approx \ket{\psi_{in}}$. To do this we find a measure that can identify the distance between quantum states which will be the foundation of the HQA cost function. There are many possible ways in which to construct a sensible loss function, the one which we will be considering is one minus the fidelity $\mathcal{F}$ between the model output and the expected training output. Hence for a chosen training data set of $K$ quantum states, $\{\ket{\psi^{(i)}_i}\}_{i=1}^K$, and the fidelity defined as $\mathcal{F}(\ket{\phi}, \ket{\psi}) = \langle \psi | \phi \rangle \langle \phi | \psi \rangle$, we define a loss, \[ \mathcal{L}(\{\ket{\psi^{(i)}_i}\}_{i=1}^K) = 1 - \mathcal{F}(\{\mathfrak{A}(\ket{\psi^{(i)}_i})\}_{i=1}^K), \] \[ (7) \]

where \[ \mathcal{F}(\{\ket{\phi_i}, \ket{\psi_i}\}_{i=1}^K) = \frac{1}{K} \sum_{i=1}^K \mathcal{F}(\{\ket{\phi_i}, \ket{\psi_i}\}_{i=1}^K), \] \[ (8) \]
is the average fidelity across all the training instances. In practice, the learner does not calculate the loss over all training instances per iteration, but rather a small batch or even over a single instance. This is because, (i) it is computationally expensive having a loss function that sums over all training instances, and (ii) doing so may result in over-fitting to the training data.

The fidelity between two states is maximum when input states are identical, and minimum when they are orthogonal. Hence, the aim is to maximise the fidelity to achieve $\ket{\psi_{in}} \approx \ket{\psi_{out}}$, and thereby minimise the loss, which lies in the range $[0, 1]$. This is a natural choice for a loss function, as the fidelity is a common distance measure between quantum states. The fidelity has also successfully been used for the construction of denoising quantum autoencoders [25] and is hence a great starting point for the construction of the HQA loss function. Selecting a method to measure the fidelity now becomes a hyper-parameter of the model – for which this paper will use the swap test (discussed in Appendix D).

Using the swap test, we can make an estimate of the training complexity. The sampling complexity per iteration (derivation in Appendix E) is given by, \[ \mathcal{O}\left(\frac{1 + P_\mathfrak{E}}{\varepsilon_\xi} + \frac{1 + P_D}{\varepsilon_{\text{fid}}}\right), \] \[ (9) \]

where $P_\mathfrak{E} = \dim(\mathfrak{A})$ is the number of parameters in the encoder and $P_D = \dim(\theta)$ in the decoder. $\varepsilon_\xi = \Delta \xi_i$ is
the uncertainty in each component of the latent vector $\hat{\xi}$, and $\epsilon_{\text{fid}}$ is the uncertainty in the fidelity measurement. The required $\epsilon_L$ and $\epsilon_{\text{fid}}$ is quite non-trivial. This non-triviality will become evident when dealing with the application in Section IV, where the HQA is fundamentally unable to learn some states due to their stochasticity.

### B. Order in latent space

Now that the HQA has been constructed, one can observe the powerful nature of representing states in a classical latent space. Training the HQA gives rise to order in latent space that is created purely through matching the input quantum state to the output. In other words, even though we are not supplying the HQA with information about the states trained directly, the model is able to learn these differences and form patterns in latent space. It is this order that we can exploit to apply ML learning techniques to cluster and classify states in Section IV.

The HQA is trained for a training set of quantum states $\{|\psi_i^{(\text{in})}\rangle\}_{i=1}^{N}$ that have underlying symmetry. In this paper, $|\psi_i^{(\text{in})}\rangle$ will correspond to a set of distinct amplitude encoded Gaussian states that can be easily analysed. A Gaussian distribution is defined as,

$$
N(x; \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{x - \mu}{\sigma} \right)^2},
$$

where $\mu$ and $\sigma$ are the mean and standard deviation respectively. Quantising this function for $N$ equally spaced values in the range $\mu \in [-\frac{N}{2}, \frac{N}{2}]$, $\sigma \in [0, \frac{N}{3}]$, we have $|d_i(\hat{\mu}, \hat{\sigma})\rangle = N(i - \left\lfloor \frac{N}{2} \right\rfloor; \hat{\mu}, \hat{\sigma})$ where $i \in \{0, 1, ..., N - 1\}$ and $\lfloor a \rfloor$ is the ceiling function that returns the smallest integer above or equal to $a$. Now that we have discrete distribution, $d_i(\hat{\mu}, \hat{\sigma})$, it can be encoded into the $N = 2^n$ amplitudes of a $n$-qubit quantum state,

$$
|\psi_N(\hat{\mu}, \hat{\sigma})\rangle = \frac{1}{\sqrt{C}} \sum_{i=0}^{N-1} d_i(\hat{\mu}, \hat{\sigma}) |i\rangle,
$$

where $C = \sum_{i=0}^{N-1} |d_i(\hat{\mu}, \hat{\sigma})|^2$ is the normalisation constant. The ability for such encoded states to be variationally encoded has been shown in [32].

With this set of distinct quantum states, the HQA is employed to generate a classical latent space that represents the subset in which these states lie. Automatic differentiation is performed through this supervised model to update the PQC parameters in the encoder and the weights of the ANN that determine the decoder. The PQC architecture of $U_\alpha(\alpha)$ and $U_\theta(\theta)$ are shown in Appendix C. We take $\text{dim}(\alpha) = 4v$ and $\text{dim}(\theta) = 4n$, and a feedforward ANN with one hidden layer of size $2v$. The training was performed in batches of 2 states using the Adam ($\gamma = 0.1$) optimiser [33], for a fixed number of epochs – which is the number of iterations each training sample is used for optimisation.

In Figure 3a, we see that training convergence is robust to latent space dimension, with similar loss evolution for all $v$. However, in Figure 3b, larger $v$ is seen to decrease the loss in testing. As expected we see greater expressibility of the quantum state with a larger latent space.

Once trained, to analyse this model we can apply only the trained encoder, $E$, to an input quantum state and observe its location in latent space – this latent vector is what we refer to as the classical representation of the input quantum state. An illustration of this process is shown in Figure 4. The latent vector can then be obtained for all the amplitude encoded Gaussian states used for training, and then plotted. Figure 5 shows such a plot for $n = 5$ and a latent size, $v = 12$ where first two components (or parameters) of the vector are illustrated. This figure shows that Gaussian encoded states are distributed with a pattern distinguishing their mean and standard deviations. In this specific example, we see that an outward radial movement in the presented latent space corresponds to decreasing the standard deviation; and a positive polar rotation corresponds to increasing the mean. Such elegance is evident from patterns in the original set of quantum states; the two degrees of freedom: $\mu$ and $\sigma$. However, these patterns have the ability...
FIG. 5. Figures show a HQA trained with Gaussian distributions over 5 qubits and a latent space dimension, \( v = 12 \). Noting that the HQA is not explicitly supplied with information about the states, the model can be seen to decode states into latent space, such that patterns are formed based on aspects of the trained distributions. Here we observe patterns formed in latent space distinguishing the mean and standard deviation of the Gaussian amplitude encoded states.

FIG. 6. Illustration of the Gaussian distributions displayed with 2 parameters in Figure 5, extended to a 3rd dimension. Its projection onto Parameter 1 & 2 space is shown in grey. Note: only some of the states with chosen means are plotted.

to be extremely non-trivial to visualise and this can be seen when we extend the results of Figure 5 to a 3rd dimension as shown in Figure 6. Hence this non-triviality suggests the use of ML to learn patterns in latent space.

It is not necessarily true that patterns will be visible when plotting the first two parameters of latent space. A plot of, say, the 9th and 10th latent parameters shows no patterns at all, as points seem to all congregate on a line or point. This indicates that these latent parameters are not being used to distinguish the quantum states, suggesting that a possible dimensionality reduction is possible for the latent space. This is where one can use principal component analysis (PCA) [6] that will both, allow for a clearer understanding of how the latent space is used, and also transform the space so that the most principal components can be plotted.

Now that we have identified a method of systematically analysing latent space, we can ask the question of whether this pattern would still occur if one simply trained on Gaussians with different mean values. Or similarly, if the model was built on training on Gaussians with only varying standard deviations. The results of doing so are shown in Figure 7 where the two most principal components from PCA dimensionality reduction are plotted. The formation of patterns in latent space can only be observed when training has seen the variations in state. For example, in Figure 7 where only \( \mu \) was varied and \( \sigma \) was kept constant, there is order formed distinguishing \( \mu \) but not \( \sigma \). The opposite occurs where the training is switched. This suggests the HQA naturally attempts to allocate areas of similar quantum states without the need for additional supervision. In the context of applications, this is extremely useful as one can exploit the latent space location of a particular unknown state in reference to other known states – where similarity was before not necessarily obvious. In general, this means

FIG. 7. (a) The distribution of Gaussian amplitude encoded states in latent space when the HQA is trained with varying mean, but fixed standard deviation of \( \sigma = 3 \). (b) Illustrates the reverse, with training over a range of standard deviations but with a fixed mean, \( \mu = 0 \). Both figures employ a HQA with a latent size of 12 over a 5 qubit amplitude encoded state. The two parameters shown are in the direction of the two most principle components which accounts for 55% of the variation in (a) and 71% of the variation in (b). This variation refers to the spread of data in the plotted components. One should also note, since the latent space has been transformed to the basis of the 2 principle components, the parameters of the latent space are not necessarily between \(-1\) and \(1\).
FIG. 8. Two HQA models are constructed with 7 qubit input states and a latent size of $v = 12$. Figure (a) shows the actual location of states in latent space when the HQA model is trained with only smooth states of skewed Gaussians. While (b) shows a similar result, however, in this case where the HQA is trained on both classes. Both (a) and (b) include plots for the principal and minor components of the latent vector. In total, 1600 states are plotted, split evenly between the classes.

that we can infer information about states by applying ML algorithms to the states in latent space, as will be explored in Section IV.

IV. APPLICATION

It is evident that clear patterns emerge in latent space from training Gaussian distributions. An extension to merely observing these patterns is obtaining information about quantum states through their latent representation. This includes both, understanding what it means for states to be located near each other in latent space, and also seeing if one can infer information about states using ML on their latent space representation.

In order to test these methods, we construct a toy problem involving two classes of states. We define amplitude encoded skewed Gaussian distributions of the form,

$$d_i = N_i \cdot v_i \cdot \eta,$$

where $d_i$ is the amplitude of the $i$th orthogonal state, $N_i$ is a Gaussian distribution, $v_i = \max(0, a \cdot i + b)$ is a linear function, and we have the class label definitions,

$$\eta = \begin{cases} 1, & \text{class = "smooth"} \\ \eta, & \text{class = "non-smooth"} \end{cases}$$

where $\eta \in [0, 1]$ is a uniform stochastic term that fluctuates as a distribution is called for training or testing.

To illustrate the power of the HQA, we introduce the artificial objective of clustering states by either smooth or non-smooth, simply from applying classical ML techniques to their latent space representations. However, before one can obtain these representations, one needs to train the HQA, raising the question of selecting training instances. In general, there is no clear answer to how one should proportion the training instances between the two classes. However, it was seen in Section III b that order was formed when the HQA had seen different distributions, without which the latent representations appear to not separate deferring distributions. Hence we look at two HQAs: (i) one that is trained with only smooth states, and (ii) a HQA trained with both classes in equal proportion. An important point to know here is that the HQA will fundamentally not be able to reproduce the states with the applied stochastic term as the set of such distributions is far too large. Nevertheless, it will be shown – through both clustering and classification – that the HQA does not need to recreate states perfectly to be useful in the context of ML in latent space.

Post the HQA training, it is possible to analyse the latent representations of the quantum states that are obtained by applying $C$ on a sample of the states (the results of which are shown in Figure 8). Using both classes of distributions for training, constructs a HQA (Figure 8b) that is still able to allocate regions of varying mean in its principal components. However, the HQA trained on only smooth states lacks this order (Figure 8). Interest-
FIG. 9. These figures show the prediction results of clustering on the latent space representations of quantum states. The classical clustering was done using Gaussian Mixture Modelling, post the training of the HQA. The clustering fits were taken using: (i) all components and, (ii) the last 4 minor components (least principal components). The plot shows four possible clustering outcomes by also distinguishing the HQA model trained with only smooth states in (a), and the HQA trained with both classes in (b). Each of the four outcomes, have plots showing the predictions in both the principal and minor components. Note: the class labels of smooth and non-smooth are added later, as clustering merely predicted two groups of blue and orange.

ingly, minor components of the vector clearly distinguish the two classes of states, regardless of the its training.

Having constructed classes of states, in this section we will look at (i) clustering states based on their latent representations, and (ii) providing an enhancement on classifying quantum states with semi-supervised learning.

A. Clustering

In classical ML the most common algorithm for clustering is kmeans [34] (also referred to Lloyd’s algorithm) which attempts to find clusters in a data set by observing some classical distance measure (an introduction to kmeans is presented in Appendix F). The quantum equivalent was first proposed in [35], in the form of a so-called quantum kmeans. Here they are able to produce a quantum state corresponding to the $k$-clusters with complexity that grows linearly with the number of qubits, $n$. However, obtaining a classical description becomes exponential as the quantum states need to be measured using quantum state tomography (QST) techniques, such as compressed sensing [7, 8], which requires $O(kN^2 \log N)$ where $N = 2^n$. In addition, the authors exploit the fact that kmeans can be expressed as a quadratic programming problem which can be solved using a quantum adiabatic algorithm. In [36], an approach is presented with efficient quantum methods of calculating the Euclidean distance. Alternatively, the quantum approximate optimisation (QAOA) algorithm [37] is used in [38], for clustering by association to the maximum cut problem.

Many of the methods in literature suffer from requiring QST techniques to classically obtain clusters, which is not required when using the HQA. We define a method of clustering states using classical representations generated by the HQA, after which classical clustering algorithms are used on a space that is exponentially smaller.

For the classical clustering of latent space, it is required that a more sophisticated method than kmeans is used. There are many advanced clustering methods that allow non-linear clustering [39], but in general such clustering is not natural. In this work we will use Gaussian Mixture Modelling which is an extension of the kmeans clustering algorithm that has the flexibility to change the importance of certain parameters of the vector $\{40\} – \text{kmeans simply uses a Euclidean measure of distance.}$

The predicted labels from clustering latent space are shown in Figure 9. The results show remarkable agreement with true labels when considering the minor components of the latent vector: 85.1% accuracy for the HQA trained with only smooth states and 84.4% accuracy for the HQA trained with both classes. On the other hand, clustering based on the principal components are seen to amount to guessing the class of state. The reason
for this is evident when looking at the principal components. Fitting using all components is seen to identify clusters relating to the mean of the distributions (grouping negative and positive means), which is also a valid clustering process. Importantly this is only seen for the HQA trained with both classes. Such a deficiency is attributed, not to a limitation of the algorithm, but rather to the non-uniqueness of the task’s solution.

At this point, one should retrace the steps of this clustering method in the context of an application. It is conceivable that a quantum experiment is conducted that produces quantum states about which the user has no information. This stream of states could hence be used for the training of the HQA. Importantly, however, one should note that a single sample of a quantum state is not sufficient. There is both, the sub-routine for the encoder, as well as the fidelity computation, that requires multiple copies of the state being produced from the experiment. Once a level of convergence has been reached, a clustering algorithm – such as kmeans or Gaussian Mixture Modelling – could be used on the latent representations of these states to identify possible groups. Finally, one can learn to identify – possibly highly non-trivial – distinctions between quantum states. This was shown in the distinction between smooth and non-smooth states, however more work is required to extend such a method to further applications.

### B. Semi-supervised classification

In a similar process to using classical clustering methods on latent space, we now use classical supervised learning models to classify quantum states. Specifically two ML algorithms are used: Support Vector Machine (SVM) [41] and Logistic Regression (LR) [42]. These are both supervised learning algorithms that are successfully used for classification. The former attempts to obtain a separating hyper-plane splitting the smooth and non-smooth classes, while the latter is a form of binary regression. It is enough for the reader to understand that these are supervised learning algorithms that are fundamentally linear, but where the SVM can be extended to non-linear decision boundaries with the use of, what is called, a kernel. The decision boundaries of these models are shown in Figure 10.

Splitting the 1600 classical data points (seen in Figure 8) in a 3 : 10 testing/training ratio, the SVM and LR models are trained in two ways: (i) on all components of the latent vectors, and (ii) on just the principal components. The accuracy on the test data for all these variations (including kernel) is shown in Table I.

In general, both SVMs with non-linear kernels are seen to have a higher classification accuracy than LR. This makes sense, as the distribution of states on latent space was seen to be highly non-linear. More importantly however, from the performance of the SVM, the interesting result is that the HQA trained on only the smooth distributions, is comparable to the HQA trained on both when considering all components of the vector. At the same time, the HQA trained on both classes performs better when considering ML on only the principal components. To understand this behaviour, we realise that the HQA model attempts to separate states in only a few principal components. This means that when both classes were used for training the HQA, this separation was identifiable by the algorithm.

Considering all components, the polynomial kernel SVM with a HQA trained on only smooth distributions has a 0.95 accuracy rate. This occurs as a result of the non-smooth distributions – that were not seen by the HQA – being stored off the learned manifold and into the distinct unutilised regions of the latent space. It was hence easy for both ML algorithms to distinguish between the classes, with even a linear decision boundary from LR achieving an accuracy of 0.87.

These are significant results to keep in mind, however, not necessarily the most natural use of the HQA. If such a classification of quantum states was the main objective given a set of labelled states, one could instead – potentially – train the encoder $\mathcal{E}$ rather than an entire HQA. However, the power of the HQA comes down to its ability to be trained unsupervised – i.e. without any labelling of the states that are being fed. For example, it is possible to train a HQA with the output states of some quantum experiment without knowing anything about the states themselves. Post-training one would only be required to label a few states and simply perform classical ML to obtain a working quantum state classifier. The reason that this works lies with the ability of the HQA to learn a manifold in which relevant states lie. Such a manifold is far smaller than the space of all states, hence requiring far fewer labelled instances to train. In ML literature, this process is known as semi-supervised learning, where only a portion of instances are labelled but where the unlabelled instances are also able to help with the overall classification process.
Table I. Results of the binary classification problem of distinguishing smooth and non-smooth states from their latent space representations. This is done for two HQA models, with \((n,v) = (7,12)\), that identify different latent vectors: a HQA trained on only smooth and another HQA trained on both. The accuracy is simply defined as the proportion of correct classifications on a separate group of test data (size=480). Importantly, both the SVM and the LR models are by no means optimised for the performance in this classification problem. Rather, these classical models are used as an illustration of the possibility of classifying states based on their latent space representation. For completeness, the hyper-parameters used for SVM are: \(C = 1.0\) and \(\gamma = 2\) (Linear), 20 (RBF). Finally, PCA reduction takes the 4 most principal components from the latent vector. The PCA reduced space accounts for 55\% of the variance for HQA trained on both classes and 48\% trained only smooth.

| ML Model           | Training type | Kernel | Latent space          | Accuracy |
|--------------------|---------------|--------|-----------------------|----------|
| Support Vector Machine | Only smooth  | Poly (degree=4) | All components  | 0.95    |
|                    |               | RBF    | All components        | 0.55    |
|                    | Both          | Poly (degree=4) | All components       | 0.92    |
|                    |               | RBF    | All components        | 0.65    |
| Logistic regression | Only smooth  | -      | All components        | 0.52    |
|                    |               |        | PCA reduction         | 0.58    |
|                    | Both          | -      | All components        | 0.73    |
|                    |               |        | PCA reduction         | 0.75    |

V. CONCLUSION

QML algorithms are yet to conclusively demonstrate advantage in the NISQ-era. There remain crucial problems that must be resolved for the application of these methods on real devices. One such problem is the emergence of barren plateaus and narrow gorges in gradient-based optimisation. This is common to all models that involve the optimisation of PQCs and is hence critical that it is solved for the specific HQA design implemented in this work. Yet another impediment is the difficulty of obtaining continuous values from qubit-based devices with measurements that are fundamentally digital. In this work, this is overcome with repeated measurement. However, it is possible to apply similar principles to Continuous-Variable (CV) Quantum devices \([43]\) for a more efficient measurement of continuous latent space vectors. This approach is left for future work, in align with the QML models proposed in \([44]\). Finally, the effect of noise on the HQA requires further research, with potential robustness seen from training stochastic states.

Due to these problems, it was important that the proposal of the HQA was made arbitrary in its specific implementation (as illustrated in Figure 1c). Therefore, the crucial aspect of this work is the proposed paradigm of learning quantum states through the application of ML techniques on their classical representations – representations that are generated through training a hybrid quantum autoencoder (HQA).

To demonstrate its successful application, the HQA was constructed using PQCs on a training set of Gaussian amplitude encoded states. Patterns – associated with the mean and standard deviation of the Gaussian encoded quantum state – were visually recognisable in their latent space representations. The emergence of order in latent space was exploited for the implementation of clustering and semi-supervised classification. In the context of (non-)smooth states (defined in Eq. (13)), we were able to achieve 84% accuracy for clustering and 93% for classification. Though the accuracy is highly problem dependent, the applied states under question had non-trivial distinctions and hence demonstrates the robustness of this clustering and classification approach.

Finally, it is assumed that an end-to-end application of the HQA will involve a set of training states obtained from supplemental quantum algorithms – such as the quantum variational eigensolver \([3]\). In \([5]\) the constructed quantum autoencoder is classically simulated to compress ground states of the hydrogen molecule with various \(r\). Furthermore, there have been attempts for entanglement classification \([45,46]\), a process which can potentially employ variational methods. Recently, QAEs were suggested for use in low-rank state fidelity estimation \([47]\), for which the structured classical latent space constructed by the HQA could be exploited. These examples motivate further uses of the HQA from the contrived application studied in this work. Nonetheless, the novel paradigm proposed in this paper, lays the framework for a unique approach to extracting information and obtaining underlying structure from sets of quantum states.

ACKNOWLEDGMENTS

The authors acknowledge the support provided by the University of Melbourne through the establishment of an IBM Network Q Hub at the University. CDH is supported by a research grant from the Laby Foundation.

The large number of simulations required for this work were made feasible through access to the University of Melbourne’s High Performance Computer, Spartan \([48]\). The HQA implementation was carried out employing the
QML framework provided by the PennyLane [49] library.

[1] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature 549, 195 (2017), 1611.09347.
[2] J. Preskill, Quantum Computing in the NISQ era and beyond, Quantum 2, 79 (2018), 1801.00862.
[3] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O’Brien, A variational eigenvalue solver on a photonic quantum processor, Nature Communications 5, 4213 (2014), 1304.3061.
[4] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, New Journal of Physics 18, 023023 (2016), 1509.04279.
[5] J. Romero, J. P. Olson, and A. Aspuru-Guzik, Quantum autoencoders for efficient compression of quantum data, Quantum Science and Technology 2, 045001 (2017), 1612.02806.
[6] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning (The MIT Press, 2016).
[7] D. Gross, Y.-K. Liu, S. T. Flammia, S. Becker, and J. Eisert, Quantum state tomography via compressed sensing, arXiv 105, 150401 (2009), 0909.3304.
[8] S. T. Flammia, D. Gross, Y.-K. Liu, and J. Eisert, Quantum tomography via compressed sensing: error bounds, sample complexity and efficient estimators, New Journal of Physics 14, 095022 (2012), 1205.2300.
[9] M. Schuld, I. Sinayskiy, and F. Petruccione, The quest for a Quantum Neural Network, Quantum Information Processing 13, 2567 (2014), 1408.7005.
[10] K. H. Wan, O. Dahlsten, H. Kristjánsson, R. Gardner, and M. S. Kim, Quantum generalisation of feedforward neural networks, npj Quantum Information 3, 36 (2017), 1612.01045.
[11] P. Rebentrost, T. R. Bromley, C. Weedbrook, and S. Lloyd, Quantum Hopfield neural network, Physical Review A 98, 042308 (2018), 1710.03599.
[12] J. Zhao, Y.-H. Zhang, C.-P. Shao, Y.-C. Wu, G.-C. Guo, and G.-P. Guo, Building quantum neural networks based on a swap test, Physical Review A 100, 012334 (2019), 1904.12697.
[13] Y. Cao, G. G. Guerreschi, and A. Aspuru-Guzik, Quantum Neuron: an elementary building block for machine learning on quantum computers, arXiv (2017), 1711.11240.
[14] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn, A. Kandala, A. Mezzacapo, P. Müller, W. Riess, G. Salis, J. Smolin, I. Tavernelli, and K. Temme, Quantum optimization using variational algorithms on near-term quantum devices, Quantum Science and Technology 3, 035003 (2018), 1710.01022.
[15] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, Evaluating analytic gradients on quantum hardware, Physical Review A 99, 032331 (2019), 1811.11184.
[16] J. M. Kübler, A. Arrasmith, L. Cincio, and P. J. Coles, An Adaptive Optimizer for Measurement-Frugal Variational Algorithms, Quantum 4, 263 (2020), 1909.09083.
[17] R. Sweke, F. Wilde, J. J. Meyer, M. Schuld, P. K. Fährmann, B. Meynard-Piganeau, and J. Eisert, Stochastic gradient descent for hybrid quantum-classical optimization, Quantum 4, 314 (2020), 1910.01155.
[18] J. Stokes, J. Izaac, N. Killoran, and G. Carleo, Quantum Natural Gradient, Quantum 4, 269 (2020), 1909.02108.
[19] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Barren plateaus in quantum neural network training landscapes, Nature Communications 9, 4812 (2018), 1803.11173.
[20] Z. Holmes, K. Sharma, M. Cerezo, and P. J. Coles, Connecting ansatz expressibility to gradient magnitudes and barren plateaus, arXiv (2021), 2101.02138.
[21] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, Cost function dependent barren plateaus in shallow parametrized quantum circuits, Nature Communications 12, 1791 (2021), 2001.00550.
[22] C. O. Marrero, M. Kieferová, and N. Wiebe, Entanglement induced barren plateaus (2021), arXiv:2010.15968 [quant-ph].
[23] A. Arrasmith, Z. Holmes, M. Cerezo, and P. J. Coles, Equivalence of quantum barren plateaus to cost concentration and narrow gorges (2021), arXiv:2104.05868 [quant-ph].
[24] S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio, and P. J. Coles, Noise-induced barren plateaus in variational quantum algorithms (2021), arXiv:2007.14384 [quant-ph].
[25] D. Bondarenko and P. Feldmann, Quantum Autoencoders to Denoise Quantum Data, Physical Review Letters 124, 130502 (2020), 1910.09169.
[26] K. Beer, D. Bondarenko, T. Farrelly, T. J. Osborne, R. Salzmann, D. Scheiermann, and R. Wolf, Training deep quantum neural networks, Nature Communications 11, 808 (2020), 1902.10445.
[27] L. Lamata, U. Alvarez-Rodriguez, J. D. Martín-Guerrero, M. Sanz, and E. Solano, Quantum autoencoders via quantum adders with genetic algorithms, Quantum Science and Technology 4, 014007 (2018), 1709.07409.
[28] A. Pepper, N. Tischler, and G. J. Pryde, Experimental Realization of a Quantum Autoencoder: The Compression of Qutrits via Machine Learning, Physical Review Letters 122, 060501 (2019), 1810.01637.
[29] C.-J. Huang, H. Ma, Q. Yin, J.-F. Tang, D. Dong, C. Chen, G.-Y. Xiang, C.-F. Li, and G.-C. Guo, Realization of a quantum autoencoder for lossless compression of quantum data, Physical Review A 102, 032412 (2020), 1903.08699.
[30] Y. Ding, L. Lamata, M. Sanz, X. Chen, and E. Solano, Experimental Implementation of a Quantum Autoencoder via Quantum Adders, Advanced Quantum Technologies 2, 1800065 (2019), 1807.10643.
[31] H. Ma, C.-J. Huang, C. Chen, D. Dong, Y. Wang, R.-B. Wu, and G.-Y. Xiang, On compression rate of quantum autoencoders: Control design, numerical and experimental realization (2020), arXiv:2005.11149 [quant-ph].
[32] C. Zoufal, A. Lucchi, and S. Woerner, Quantum Gen-
ceptive Adversarial Networks for learning and loading random distributions, npj Quantum Information 5, 103 (2019), 1904.00043.

[33] D. Kingma and J. Ba, Adam: A Method for Stochastic Optimization, International Conference on Learning Representations (2014).

[34] S. Lloyd, Least squares quantization in PCM, IEEE Transactions on Information Theory 28, 129 (1982).

[35] S. Lloyd, M. Mohseni, and P. Rebentrost, Quantum algorithms for supervised and unsupervised machine learning, arXiv:1307.0411 (2013), arXiv:1307.0411 [quant-ph].

[36] N. Wiebe, A. Kapoor, and K. M. Svore, Quantum Algorithms for Nearest-Neighbor Methods for Supervised and Unsupervised Learning, Quantum Info. Comput. 15, 316–356 (2015).

[37] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm (2014), arXiv:1411.4028 [quant-ph].

[38] J. S. Otterbach, R. Manenti, N. Alidoust, A. Bestwick, M. Block, B. Bloom, S. Caldwell, N. Didier, E. S. Fried, S. Hong, P. Karalekas, C. B. Osborn, A. Papageorge, E. C. Peterson, G. Prawiroatmodjo, N. Rubin, C. A. Ryan, D. Scarabelli, M. Scheer, E. A. Sete, P. Sivarajah, R. S. Smith, A. Staley, N. Tezak, W. J. Zeng, A. Hudson, B. R. Johnson, M. Reagor, M. P. da Silva, and C. Rigetti, Unsupervised machine learning on a hybrid quantum computer (2017), arXiv:1712.05771 [quant-ph].

[39] S. Maekawa, K. Takeuch, and M. Onizuka, Non-linear attributed graph clustering by symmetric nmf with pu learning, arXiv:1810.00946 [cd.LG] (2018).

[40] S. Dasgupta, Learning mixtures of gaussians (IEEE Computer Society, USA, 1999) p. 634.

[41] C. Cortes and V. Vapnik, Support-vector networks, Machine Learning 20, 273 (1995).

[42] Model-building strategies and methods for logistic regression, in Applied Logistic Regression (John Wiley & Sons, Ltd, 2013) Chap. 4, pp. 89–151.

[43] S. Lloyd and S. L. Braunstein, Quantum computation and continuous variables, Phys. Rev. Lett. 82, 1784 (1999).

[44] N. Killoran, T. R. Bromley, J. M. Arrazola, M. Schuld, N. Quesada, and S. Lloyd, Continuous-variable quantum neural networks, Phys. Rev. Research 1, 033063 (2019).

[45] M. Sanz, I. L. Egusquiza, R. D. Candia, H. Saberi, L. Lamata, and E. Solano, Entanglement classification with matrix product states, Scientific Reports 6, 30188 (2016), 1504.07524.

[46] C. Harney, S. Pirandola, A. Ferraro, and M. Paternostro, Entanglement classification via neural network quantum auto-encoder (2017).

[47] V. Bergholm, J. Izac, M. Schuld, C. Gogolin, M. S. Alam, S. Ahmed, J. M. Arrazola, C. Blank, A. Delgado, S. Jahangiri, K. McKiernan, J. J. Meyer, Z. Niu, A. Száva, and N. Killoran, PennyLane: Automatic differentiation of hybrid quantum-classical computations (2020), arXiv:1811.04968 [quant-ph].

[48] F. Rosenblatt, The perceptron, a perceiving and recognizing automaton, Cornell Aeronautical Laboratory report Report 85-460-1 (1957).

[49] P. G. Breen, C. N. Foley, T. Boekholt, and S. P. Zwart, Newton vs the machine: solving the chaotic three-body problem using deep neural networks, arXiv 10.1093/mnras/staa713 (2019), 1910.07291.

[50] T. Chen, H. Chen, and R.-w. Liu, Approximation capability in C(R) by multilayer feedforward networks and related problems, IEEE Transactions on Neural Networks 6, 25 (1995).

[51] J. J. Meyer, J. Borregaard, and J. Eisert, A variational toolbox for quantum multi-parameter estimation, arXiv:2006.06303 (2020), arXiv:arXiv:2006.06303v1.

[52] B. Meade, L. Lafayette, G. Sauter, and D. Tosello, Spar-tan hpc-cloud hybrid: Delivering performance and flexibility (2017).

[53] V. Bergholm, J. Izac, M. Schuld, C. Gogolin, M. S. Alam, S. Ahmed, J. M. Arrazola, C. Blank, A. Delgado, S. Jahangiri, K. McKiernan, J. J. Meyer, Z. Niu, A. Száva, and N. Killoran, PennyLane: Automatic differentiation of hybrid quantum-classical computations (2020), arXiv:1811.04968 [quant-ph].

[54] L. Banchi and G. E. Crooks, Measuring Analytic Gradients of General Quantum Evolution with the Stochastic Parameter Shift Rule, arXiv 10.22331/q-2021-01-25-386 (2020), 2005.10299.

Appendix A: Artificial Neural Network

The idea of Artificial Neural Networks (ANNs) was first proposed in 1957 in an attempt to mimic the way in which the human brain processes visual data [50]. Since then, the use of ANNs has been ubiquitous in many fields ranging from, image classification to numeric calculations in computationally expensive regions of phase-space [51]. It is important to note that ANNs are also regularly used in the context of unsupervised algorithms and in all cases, play the ubiquitous role of a function approximator.

Though ANNs come in various architectures, their fundamental unit is the neuron that accept inputs $\vec{x}$ and outputs a scalar known as the activation, $a$, of that neuron. It has the mathematical form,

$$a = \sigma \left( \sum_i w_i x_i + b \right) = \sigma (\vec{w} \cdot \vec{x} + b), \quad (A1)$$

where $w_i$ is the weighting for input $x_i$ and $b$ is a bias term. These parameters will be tuned in the learning stage of the model such that the outputs correspond to the labels of the inputs. The function $\sigma$ is known as the activation function which is a non-linear mapping $\sigma : \mathbb{R} \rightarrow [0,1]$, that outputs the activation of a neuron. In practice common activation functions include sigmoid, hyperbolic tan, rectified linear unit (ReLU) functions [6].

A network of these neurons form an Artificial Neural Network, with the output neurons commonly giving a probability distribution over the possible classifications. There are many different types of architectures for the way in which the neurons can be connected, but a common example is the feed-forward ANN, shown in Figure 11. The layered fashion of the feed-forward ANN means that the whole network has a relatively simple mathematical form,

$$f(\vec{x}) = \sigma_1 \circ g_1 \circ \sigma_{L-1} \circ g_{L-1} \cdots \sigma_1 \circ g_1(\vec{x}), \quad (A2)$$

where we have, $g_i(\vec{x}) = \vec{W}_i \cdot \vec{x} + \vec{b}_i$ is the sum of previous layer neurons appropriately weighted by matrix $\vec{W}_i \in \mathbb{R}^{l_i \times l_{i-1}}$. These weights are, most commonly, learned through gradient optimisation.
The real utility of ANNs is in their ability to learn and approximate any continuous function with sufficient data. The Universal Approximation Theorem states that a feed-forward neural network with a single hidden layer is able to approximate any continuous function on $\mathbb{R}^N$ [52]. It is important to note that non-linear activation functions are required – without which we have a complex linear model. This is evident from equation (2) where it can be that linear $\sigma_i$ implies a linear $f$ as function compositions preserve linearity. The non-trivial aspect is that there is no restriction on the non-linearity of the activation function and how this relates to the number of neurons required in the hidden layer. Due to this universality of function approximation, ANNs do not have the same hyper-parameter tuning problem as other ML models. Hyper-parameters refer to parameters of the model that must be identified by the user – such as the number of neurons in the case of an ANN. The hyper-parameter that an ANN does not require is a choice of the type of decision boundary, which in other models is crucial. At the same time a significant problem with ANNs its tendency to approximate too closely the training instances rather than generalising upon these data points. This is known in ML literature as over-fitting and there are regularisation techniques to mitigate this problem.

Appendix B: The parameter shift rule

In this paper the computation of quantum gradients of continuous parameters are computed using parameter-shift differentiation proposed in [15]. It is shown that the analytical gradient of a variational circuit, $f$, defined in equation (1), can be found when it is composed of a Hermitian operator $G$ with strictly two eigenvalues $\pm r$. It is shown that $\partial_\mu f$ can be estimated using two additional evaluations of the quantum device by placing either gates $G(\pm \frac{\mu}{r})$ in the original circuit next to the gate that we are differentiating. Since for unitarily generated one-parameter gates we have $G(a)G(b) = G(a + b)$, we simply have a shift of the parameters by $s = \frac{\mu}{r}$ to find the gradient,

$$\partial_\mu f = r[f(\mu + s) - f(\mu - s)]. \quad (B1)$$

This is aptly named the parameter shift rule. For generator $G$ with more than two eigenvalues this strategy fails; however one is able to use an ancilla qubit and perform a decomposition of the derivative of the gate to obtain a gradient as elaborated in [15]. As an important additional benefit, the parameter-shift rule has been shown to hold on noisy quantum devices [53]. Furthermore, recent works have attempted to generalise this method of obtaining an analytic gradient for more intricate unitaries using stochastic techniques [54].

Having acquired the derivative of parameters in VQCs, one is now able to explore its use in QML – especially hybrid models involving both classical and quantum processing units. The ability to compute gradients of VQCs means that it is possible to attach a VQC and ANN as components of a larger ML algorithm, over which the back-propagation algorithm will still be applicable (shown in Figure 12). To implement these models, an open-source Python3 software framework for hybrid quantum-classical optimisation and ML, termed PennyLane [49], is used. The library interfaces with popular machine learning libraries such as Tensorflow, PyTorch, autograd while also providing APIs for the access of publicly available quantum devices such as those by Rigetti and IBM – alleviating some of the tedious programming.

Appendix C: PQC architecture

The decomposition unitary $U(\theta)$ need not be understood in terms of the commonly used single and two qubit circuit gates. Rather, they can be generalised to any tunable parameter in a particular quantum device and optimised accordingly. Though this is possible for specific devices, this paper will address PQCs from a general perspective, such that $U(\theta)$ is composed of only single and two qubits gates, with parameters $\theta$ defining the rotation on single qubit gates. Though $U(\theta)$ can, in theory, be set as a random set of U3 gates and CNOT gates, it is generally desired to have a pattern of gates so that one can compare and generalise PQCs made from certain patterns. These patterns will be referred to as PQC architectures, separate from device architectures that describes the physical qubit layout of a device. In this work, we employ an alternating structure of rotation layers and entangling layers, as shown in Figure 13.

Appendix D: Swap test

To measure the fidelity between two coherent quantum states, one can introduce an ancilla qubit and use the swap test. To find the overlap between an output state, $|\psi\rangle = \sum_i \psi_i |i\rangle = \sum_i |\psi_i\rangle$, and a reference state, $|\phi\rangle = 

\[
\begin{array}{cccc}
\text{Input neurons} & z_1 & w_1 & \cdots \\
\text{Hidden layers(} \times \text{N}\text{)} & \sigma(z_1w_1 + z_2w_2 + b) & \vdots & \vdots \\
\text{...} & \vdots & \vdots & \vdots \\
\text{Output neurons} & ... & \vdots & \vdots \\
\end{array}
\]

FIG. 11. The structure for a feed-forward ANN.
FIG. 12. Diagram of an example hybrid QML algorithm, with the combination of an ANN and a PQC. For simplicity, both models only have a single parameter that is trained. For input, $x$, and label, $y$, the diagram highlights the derivatives required for the update of each parameter using the back-propagation algorithm.

FIG. 13. The architecture is defined by alternating rotation and entangling layers – with $q$ number of such repetitions. Here, $R_Y := \exp(-i Y \theta)$, where $Y$ is the Pauli-Y operator. With only nearest neighbour couplings, such a structure is also known as a hardware efficient ansatz.

FIG. 14. This is a circuit diagram depicting the swap test algorithm that is used to compute the overlap, $|\langle \phi | \psi \rangle|^2$ through repeated measurements of the ancilla qubit.

$$\sum_j \phi_j |j\rangle = \sum_j |\tilde{\phi}_j\rangle,$$

one repeatedly measures the ancilla qubit from the circuit shown in Figure 14. This works since the state before measurement is,

$$\frac{1}{2} \sum_{i,j} \left\{ |0\rangle \left( |\tilde{\psi}_i\rangle |\tilde{\phi}_j\rangle + |\tilde{\phi}_j\rangle |\tilde{\psi}_i\rangle \right) + |1\rangle \left( |\tilde{\psi}_i\rangle |\tilde{\phi}_j\rangle - |\tilde{\phi}_j\rangle |\tilde{\psi}_i\rangle \right) \right\}. \quad (D1)$$

Now measuring the ancilla qubit in the Z-basis, the probability of measuring eigenvalue, $z = \pm 1$ is,

$$\Pr(z = \pm 1) = \frac{1}{4} \sum_{ijkl} \left( \langle \tilde{\psi}_k | \tilde{\psi}_i \rangle \langle \tilde{\phi}_l | \tilde{\phi}_j \rangle \pm \langle \tilde{\phi}_l | \tilde{\psi}_i \rangle \langle \tilde{\psi}_k | \tilde{\phi}_j \rangle \right), \quad (D2)$$

where we have used $\langle \tilde{\psi}_i | \tilde{\psi}_j \rangle = |\psi_i|^2 \delta_{ij}$ due to orthogonality and the normalisation condition, $\sum_i |\psi_i|^2 = 1$. Similarly for the $|\tilde{\phi}_j\rangle$ states. Therefore by measuring and recording the ancilla qubit enough times, we can work out the overlap between the two states in terms of the probability $\Pr(z = \pm 1)$,

$$\mathcal{F}(|\phi\rangle, |\psi\rangle) = \left| \langle \phi | \psi \rangle \right|^2 = \mp \left[ 1 - 2\Pr(z = \pm 1) \right]. \quad (D3)$$

Under certain conditions, such a measure is equivalent to the mean-squared error (MSE) between amplitude encodings, though in general it is a far stronger statement. The MSE Loss, $L_{\text{MSE}}$, can in this context be written as,

$$L_{\text{MSE}} = \sum_i (|\psi_i| - |\phi_i|)^2 \quad \text{Equivalent to minimizing} \quad -\sum_i |\psi_i||\phi_i|, \quad (D4)$$

where again we have made a simplification using the fact that $\psi_i$ and $\phi_i$ are normalised. This means that minimising the fidelity implies the minimisation of $L_{\text{MSE}}$ – with the assumption that the amplitudes of the two states are non-negative real values. This can be shown by using the triangle inequality on equation (D3) to show that $|\langle \phi | \psi \rangle|^2 \leq \sum_i (\psi_i \phi_i)^2$ revealing that the maximisation of the fidelity implies the minimisation of $L_{\text{MSE}}$ in equation (D4). However, the fidelity statement is stronger due to the inequality. It should, however, be noted that the swap test requires large numbers of SWAP gates, and hence further research is required to enhance this fidelity measurement.

**Appendix E: Training complexity**

Given the particular design of the HQA (proposed in Section III A), one can obtain a complexity for the training process. Each evaluation of the HQA would require
\( M_\varepsilon \) samples to determine the latent vector and \( M_{\text{swap}} \) samples to obtain the fidelity. We can relate the error in fidelity, \( \varepsilon_{\text{fid}} \), to \( M_{\text{swap}} \) such that we have \( M_{\text{swap}} = 1/\varepsilon_{\text{fid}}^2 \). This relation relies on the fact that the swap test measurement is a Bernoulli trial. Similarly, we relate the uncertainty in the latent vector components to the number of encoder samples: \( M_\xi = 1/\varepsilon_{\xi}^2 \), where \( \xi \) is the uncertainty in each component of the latent vector.

A single evaluation of the HQA requires \( M_\varepsilon + M_{\text{swap}} \) samples. However, training requires the calculation of gradients through the parameter shift rule (Section B). Specifically, one must obtain the gradients with respect to all parameters in the model: \( \{ \frac{dL}{d\alpha_i} \}, \{ \frac{dL}{dw_{ij}} \} \) and \( \{ \frac{dL}{d\theta_i} \} \), where \( \alpha_i \) are the \( P_E \) encoder parameters, \( w_{ij} \) are the ANN parameters (weights) and \( \theta_i \) are the \( P_D \) decoder parameters. The gradient evaluations are of the form,

\[
\frac{dL}{d\theta_i} = \frac{dL}{d\mathcal{F}} \frac{d\mathcal{F}}{d\theta_i} \tag{E1}
\]

\[
\frac{dL}{dw_{jk}} = \sum_i \frac{dL}{d\theta_i} \frac{df_i}{dw_{jk}} \tag{E2}
\]

\[
\frac{dL}{d\alpha_l} = \sum_i \frac{dL}{d\theta_i} \frac{d\mathcal{E}}{d\alpha_i} \tag{E3}
\]

where the output of the encoder, \( \mathcal{E} \), becomes the input to the ANN, \( f \), and the output of the ANN becomes \( \theta_i \). Note, only \( \alpha_i \) and \( w_{ij} \) are optimised, but we also require the gradient with respect to \( \theta_i \).

Sampling a quantum circuit is required to obtain the gradients \( \frac{dL}{d\alpha_i} \) and \( \frac{dL}{dw_{ij}} \) for all \( P_D \) parameters and \( \frac{dL}{d\mathcal{E}} \) must be calculated for all \( P_E \) parameters. The number of circuit samples required for \( \frac{dL}{d\alpha_i} \) and \( \frac{dL}{d\mathcal{E}} \), is proportional to \( M_\varepsilon \) and \( M_{\text{swap}} \), respectively. In addition, an evaluation of the HQA is required for the calculation of the loss for that particular iteration. This therefore requires an additional \( M_\varepsilon + M_{\text{swap}} \) samples. Putting all this together we have sampling complexity per iteration of,

\[
\mathcal{O}\left((1 + P_\varepsilon)M_\varepsilon + (1 + P_D)M_{\text{swap}}\right) = \mathcal{O}\left(1 + P_\varepsilon \frac{1}{\varepsilon_{\xi}^2} + 1 + P_D \frac{1}{\varepsilon_{\text{fid}}^2}\right) \tag{E4}
\]

**Appendix F: Kmeans clustering**

Clustering is an unsupervised ML algorithm that attempts to group a set of data points into distinct clusters. The most commonly used clustering algorithm is the kmeans algorithm (or Lloyd’s algorithm) [34] – used mainly due to its elegance and simplicity.

Kmeans is an iterative algorithm that partitions a data set into \( k \) groups. The algorithm works as follows:

1. Select \( k \) points randomly from the set to act as seed clusters, \( \{ \bar{\mu}_i \}_{i=1,...,k} \).
2. Assign each data point in the set, \( \{ \bar{x}_j \} \), to the closest cluster based on some distance measure \( D(\bar{x}_j, \bar{\mu}_i) \).
3. Average the group of points associated with each cluster, to form \( k \) new clusters.
4. Go back to 2, or stop when no reassignments of data points from their cluster groups are made.

Note that the distance measure, \( D \), is arbitrary but usually chosen to be the Euclidean distance. The average of the \( m^{\text{th}} \) cluster group, with \( M \) associated data points \( \{ \bar{x}_i^{(m)} \}_{i=1,...,M} \), is taken as \( \frac{1}{M} \sum_{i=1}^{M} \bar{x}_i^{(m)} \).