



# Permutation invariant encodings for quantum machine learning with point cloud data

Jamie Heredge<sup>1</sup> · Charles Hill<sup>1,2</sup> · Lloyd Hollenberg<sup>1</sup> · Martin Sevier<sup>1</sup>

Received: 22 April 2023 / Accepted: 12 March 2024  
© The Author(s) 2024

## Abstract

Quantum computing offers a potentially powerful new method for performing machine learning. However, several quantum machine learning techniques have been shown to exhibit poor generalisation as the number of qubits increases. We address this issue by demonstrating a permutation invariant quantum encoding method, which exhibits superior generalisation performance, and apply it to point cloud data (three-dimensional images composed of points). Point clouds naturally contain permutation symmetry with respect to the ordering of their points, making them a natural candidate for this technique. Our method captures this symmetry in a quantum encoding that contains an equal quantum superposition of all permutations and is therefore invariant under point order permutation. We test this encoding method in numerical simulations using a quantum support vector machine to classify point clouds drawn from either spherical or toroidal geometries. We show that a permutation invariant encoding improves in accuracy as the number of points contained in the point cloud increases, while non-invariant quantum encodings decrease in accuracy. This demonstrates that by implementing permutation invariance into the encoding, the model exhibits improved generalisation.

**Keywords** Quantum machine learning · Quantum computing · Geometric quantum machine learning · Quantum encoding methods · 3D computer vision · Point cloud data

## 1 Introduction

Quantum machine learning (QML) is a promising candidate for real-world applications of quantum technology (Biamonte et al. 2017). In recent years, a multitude of different techniques have been developed with the aim of using quantum computers to perform machine learning tasks (Zeguendry et al. 2023; Sajjan et al. 2022) and QML tech-

niques have been applied to a wide range of fields, including particle physics (Heredge et al. 2021; Tüysüz et al. 2021), medical data (Pregolato and Zizzi 2023; Azevedo et al. 2022), aerodynamics (Yuan et al. 2022) and natural language processing (Meichanetzidis et al. 2023). In many QML techniques, classical data is encoded into an exponentially larger quantum space where there is the possibility that the data may be separated more easily (Havlíček et al. 2019). This is a proposed source of quantum advantage over classical routines in the case that the quantum circuit performing the encoding cannot be efficiently simulated classically (Liu et al. 2021). When trying to find suitable QML techniques for real-world data, it is important to use an advantageous encoding method for that data. A current active area of research is the search for methods to encode or represent different types of data in quantum devices, for example, finding techniques to represent two-dimensional images (Lisnichenko and Protasov 2022; Anand et al. 2022; West et al. 2022).

Many QML techniques, such as the quantum support vector machine (QSVM) (Havlíček et al. 2019), are motivated by the idea that encoding classical data into a higher dimensional

✉ Jamie Heredge  
heredgej@student.unimelb.edu.au

Charles Hill  
cdhill@unimelb.edu.au

Lloyd Hollenberg  
lloydch@unimelb.edu.au

Martin Sevier  
martines@unimelb.edu.au

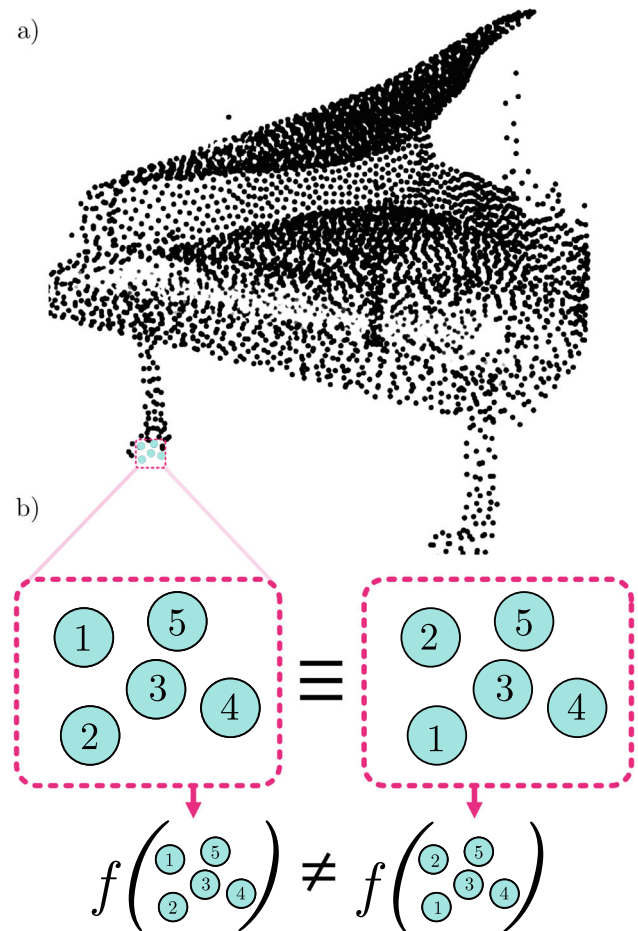
<sup>1</sup> School of Physics, University of Melbourne, Parkville, VIC 3010, Australia

<sup>2</sup> School of Mathematics and Statistic, University of Melbourne, Parkville, VIC 3010, Australia

quantum Hilbert space can simplify data classification. The circuit architecture used to encode classical data into a quantum state influences the class of functions that a QML algorithm can learn (Schuld et al. 2021). Therefore, it is critical to find an optimal quantum encoding for the data in any QML method. However, some QML techniques do not generalise well as the number of qubits, and hence, the dimensionality of the Hilbert space increases (Huang et al. 2021). To improve generalisation, attempts have been made to reduce the expressivity of QML methods by introducing some form of inductive bias into the method. These attempts include projected kernels, where only a limited number of qubits are measured, thus projecting to a lower-dimensional space (Kübler et al. 2021), introducing inductive bias through the tuning of quantum kernel hyperparameters (Shaydulin and Wild 2022), and variational state-based approaches that are capable of encoding inductive biases directly into quantum states, which have been shown to improve generalisation in the context of learning zero-sum games (Bowles et al. 2023). In this work, we present a method of introducing an inductive bias into a quantum encoding when the underlying data exhibits a permutation symmetry. By projecting our quantum-encoded state onto a symmetric subspace, this method exponentially reduces the encoding's dimensionality, leading to improved generalisation in our experiments.

In this study, we consider point cloud data types, which are three-dimensional images that consist of a set of three-dimensional points. The point cloud may represent various objects (e.g. piano, car, tree) that need to be classified. This could be in the context of identifying pedestrians in self-driving vehicles (Chen et al. 2021) or classifying different particle decay events in a high-energy physics experiment (Mikuni and Canelli 2021). Point clouds are a natural data type to study when investigating the effects of permutation invariant machine learning methods. While they may also occasionally exhibit internal data-specific symmetries such as rotation or translation symmetry, we focus here on their point order permutation symmetry when they are input into a model and present a method of encoding this symmetry into a quantum state. Permutation symmetry is a property point cloud data possesses that is not normally captured in a classical input vector. There is no inherent ordering to the points in a point cloud. Therefore, if an order is assigned to the points in a point cloud, then it should be invariant under any permutation of these point labels. Classical computers are generally forced to assign an order to the points  $\mathbf{p}_i$  when they are input, since the data must be stored in an array in memory that has an intrinsic ordering to the points. Consider the input array  $[\mathbf{p}_1, \mathbf{p}_2]$ ; exchanging two points in this array will produce a different array  $[\mathbf{p}_2, \mathbf{p}_1]$  which may give a different result when input into a given algorithm. In general, a machine learning model classifier model  $f([\mathbf{p}_1, \mathbf{p}_2])$ , with

out purposeful construction, will return a different answer if given a different permutation of the same points in the input,  $f([\mathbf{p}_1, \mathbf{p}_2]) \neq f([\mathbf{p}_2, \mathbf{p}_1])$ , while in reality the point cloud would be physically unchanged by this reordering. By creating an encoding that is invariant to the permutation of point ordering, we can exponentially reduce the effective dimensionality of the encoded states in a manner that respects an underlying symmetry of the data. An example point cloud is shown in Fig. 1 demonstrating the permutation invariance of points in the input.



**Fig. 1** (a) Example point cloud generated using the Point-E demo by OpenAI using the prompt “Grand Piano” (Nichol et al. 2022). Distinguishing between different objects could be a possible classification task that uses point cloud data. (b) Demonstration of point permutation symmetry in the input for a point cloud. Changing the order of points in a point cloud does not have an effect on the point cloud itself. However, when stored as a classical input array in computer memory, exchanging point order produces a different array. Unless it has been purposely constructed otherwise, as is the case for PointNet (Qi et al. 2016), a general machine learning classifier function, denoted by  $f$ , may produce a different classification output given a different order permutation of the point order in its input  $f(x_1, y_1, z_1, x_2, y_2, z_2) \neq f(x_2, y_2, z_2, x_1, y_1, z_1)$

There are other machine learning methods that aim to respect this permutation symmetry by building symmetric functions into the model, such as the max pool function in the classical PointNet (Qi et al. 2016) and the proposed quantum extension of PointNet (Shi et al. 2020). Similarly, techniques in geometric quantum machine learning have shown it is possible to construct a variational circuit that respects qubit permutation (and hence point permutation if one were to encode a single point per qubit) (Meyer et al. 2023; Nguyen et al. 2022; Schatzki et al. 2022; Kazi et al. 2023). The technique we discuss in this work differs in that we focus on implementing these symmetries into the encoding circuit, meaning that the classification part of the algorithm is free to take any form. Recent work suggests that permutation invariant operators, such as permutation equivariant variational circuits, may be classically tractable to simulate under certain conditions (Anschuetz et al. 2023). In our case, the classification part does not necessarily need to be permutation equivariant, as permutation invariance is captured in the encoding step. More broadly, the recent advancements in identifying the prerequisites for the emergence of barren plateaus (Fontana et al. 2023; Ragone et al. 2023) have prompted inquiries into the classical simulatability of variational quantum circuits devoid of such barren plateaus (Cerezo et al. 2023). In this work, we introduce techniques to include symmetry in the encoding step of the circuit, meaning the variational trainability/simulatability issue does not need to be considered here as we can avoid using a quantum variational classifier with this technique.

## 2 Permutation invariant encoding

In this section, we theoretically outline the structure and properties of a permutation invariant quantum state within the context of point cloud data. Point clouds were chosen as a natural use case of this encoding; however, this could be applicable to any data that exhibits permutation symmetry. We consider a point cloud data input, denoted as  $X$ , to be an array of values in the form  $X = [x_1, y_1, z_1, x_2, y_2, z_2, \dots, x_n, y_n, z_n]$ . Each point cloud therefore consists of  $n$  points, where a single point in the point cloud can be denoted as  $\mathbf{p}_i = [x_i, y_i, z_i]$ . Each point is first encoded into a quantum state  $|\mathbf{p}_i\rangle$  using a quantum circuit  $U$ , consisting of  $k$  qubits, that maps three-dimensional classical data  $\mathbf{p}_i$  to a  $2^k$  dimensional quantum state,  $U : \mathbf{R}^3 \rightarrow \mathbf{R}^{2^k}$ . We implement this gate on an initial  $|0\rangle^{\otimes k}$  state such that  $|\mathbf{p}_i\rangle = U(\mathbf{p}_i)|0\rangle^{\otimes k}$ . While in our experiments  $U$  was implemented using angle encoding, this general technique could be used for any encoding strategy. To enforce the point-exchange invariance, we construct a state that is in a symmetric superposition of all  $|\mathbf{p}_i\rangle$  permutation

states. For a point cloud  $X$  with only two points,  $n = 2$ , this can be represented as

$$|X_s\rangle = \mathcal{N}(|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle + |\mathbf{p}_2\rangle|\mathbf{p}_1\rangle), \tag{1}$$

which is invariant under permutation of the order of the two points. Regarding the normalisation constant  $\mathcal{N}$ , it should be noted that depending on the data and the encoding method  $U$ , that the initial states  $|\mathbf{p}_1\rangle$  and  $|\mathbf{p}_2\rangle$  may or may not be orthogonal. Hence, the normalisation constant  $\mathcal{N}$  for the 2 qubit case, in general, is

$$\mathcal{N} = \frac{1}{\sqrt{2(1+|\langle\mathbf{p}_1|\mathbf{p}_2\rangle|^2)}}. \tag{2}$$

The point order invariant encoded state  $|X_s\rangle$  can then be evaluated using techniques such as QSVM or passed to a variational method. As the input quantum state is now in a permutation invariant state, the quantum classification algorithm  $g(|X_s\rangle)$  is free to have any design and we are guaranteed to have permutation invariance under point order permutation as  $g(\mathcal{N}(|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle + |\mathbf{p}_2\rangle|\mathbf{p}_1\rangle)) = g(\mathcal{N}(|\mathbf{p}_2\rangle|\mathbf{p}_1\rangle + |\mathbf{p}_1\rangle|\mathbf{p}_2\rangle))$  regardless of the structure of the quantum classification function  $g$ . This contrasts to a general machine learning classifier  $f([\mathbf{p}_1, \mathbf{p}_2])$ , accepting its input as a classical array, that may give different results depending on the order of points in the input such that  $f([\mathbf{p}_1, \mathbf{p}_2]) \neq f([\mathbf{p}_2, \mathbf{p}_1])$ . In the case of  $n$  points, we wish to construct the symmetric state defined by

$$|X_s\rangle = \mathcal{N}_n \sum_{\sigma \in S_n} |\mathbf{p}_{\sigma_1}\rangle|\mathbf{p}_{\sigma_2}\rangle \dots |\mathbf{p}_{\sigma_n}\rangle, \tag{3}$$

where we sum over all permutations in the symmetric group  $S_n$ . This symmetric state will be identical under any permutation of points.

By combining the quantum states into a symmetric superposition state, we utilise the inherent quantum property of state superposition to implement an underlying symmetry of the data structure into the encoding. This symmetry exploitation allows for a reduction in the expressivity of the encoding, by exponentially reducing the effective dimensionality of the state. This can be demonstrated by considering a three-qubit state. In general, a three-qubit state is  $2^3 = 8$  dimensional and can be written as

$$\begin{aligned} |\psi\rangle = & \alpha_0|0\rangle|0\rangle|0\rangle \\ & + \alpha_1|1\rangle|0\rangle|0\rangle + \alpha_2|0\rangle|1\rangle|0\rangle + \alpha_3|0\rangle|0\rangle|1\rangle \\ & + \alpha_4|1\rangle|1\rangle|0\rangle + \alpha_5|1\rangle|0\rangle|1\rangle + \alpha_6|0\rangle|1\rangle|1\rangle \\ & + \alpha_7|1\rangle|1\rangle|1\rangle. \end{aligned} \tag{4}$$

If we now insist that the state  $|\psi\rangle$  is fully symmetric with respect to its qubits, then by exchanging qubits in Eq. 4 and

ensuring that the state remains unchanged under this action, it can be seen that  $\alpha_1 = \alpha_2 = \alpha_3$  and  $\alpha_4 = \alpha_5 = \alpha_6$  (Barenco et al. 1996). Hence, a three-qubit symmetric quantum state is effectively 4 dimensional with the following basis states

$$\begin{aligned}
 |\psi\rangle = & \beta_0|0\rangle|0\rangle|0\rangle \\
 & + \beta_1(|1\rangle|0\rangle|0\rangle + |0\rangle|1\rangle|0\rangle + |0\rangle|0\rangle|1\rangle) \\
 & + \beta_2(|1\rangle|1\rangle|0\rangle + |1\rangle|0\rangle|1\rangle + |0\rangle|1\rangle|1\rangle) \\
 & + \beta_3|1\rangle|1\rangle|1\rangle.
 \end{aligned} \tag{5}$$

It follows that an  $n$  qubit system that is permutation invariant with respect to its qubits has dimension  $n + 1$ , which is exponentially smaller than  $2^n$  (Barenco et al. 1996). In the general case, where each initial state  $\mathbf{p}_i$  has  $k$  qubits, it has been shown that the dimension of this symmetric state is

$$n+2^k-1 C_{2^k-1} = \frac{1}{(2^k-1)!} n^{2^k-1} + O(n^{2^k-2}), \tag{6}$$

which exhibits polynomial scaling in  $n$ , in contrast to the general case where the dimension is  $(2^k)^n$  and the dimension scales exponentially (Barenco et al. 1996). Note that if the data requires, such as in cases of underfitting the training data, we retain the ability to reintroduce some expressivity through breaking of the symmetry or by increasing the number of qubits  $k$  used per point.

For the theoretical results in this work, we used Qiskit *statevector\_simulator* (Abraham et al. 2019) along with an analytical symmetrisation process as described in Algorithm 1 that mathematically constructs the permutation invariant quantum states. A discussion around possible implementations of this procedure on a real quantum machine is the focus of Sect. 4.

### 3 Methodology

We compare the performance of various quantum and classical machine learning techniques when applied to the classification of two different point cloud data distributions: a sphere and a torus. Numerical results were found using Qiskit *statevector\_simulator* (Abraham et al. 2019) for the initial point encodings, followed by standard operations to construct the symmetrised states and kernel entry matrices, as will be further detailed in this section.

#### 3.1 Dataset specifications

To generate the data, we randomly sample  $n$  points from the surface of each shape to form a point cloud for each distribution. This sampling process is repeated until there are  $N$  point cloud samples in total, dividing the resulting data into training and testing sets with 80% and 20% of the data,

**Algorithm 1** Encode point cloud data with point permutation invariance using Qiskit *statevector\_simulator* and analytical symmetrisation. This algorithm demonstrates the mathematical structure of the encoding intuitively at the cost of being computationally inefficient by containing  $O(n!)$  classical computations.

---

```

Input: Array  $X$  of  $n$  points  $\mathbf{p}_i$  where  $0 < i \leq n$ 
1: for  $i$  in  $(0, n]$  do
2:    $U(\mathbf{p}_i)|0\rangle^{\otimes k} = |\mathbf{p}_i\rangle$  on separate registers
3:   Evaluate  $|\mathbf{p}_i\rangle$  using statevector_simulator
4: end for
5: Initialise empty symmetric statevector  $|X_s\rangle$ 
6: for all permutations  $\sigma$  in symmetric group  $S_n$  do
7:    $|X_\sigma\rangle \leftarrow |\mathbf{p}_{\sigma_1}\rangle$ 
8:    $j \leftarrow 2$ 
9:   while  $j \leq n$  do
10:     $|X_\sigma\rangle \leftarrow |X_\sigma\rangle \otimes |\mathbf{p}_{\sigma_j}\rangle$ 
11:     $j \leftarrow j + 1$ 
12:  end while
13:   $|X_s\rangle \leftarrow |X_s\rangle + |X_\sigma\rangle$ 
14: end for
15: Normalise  $|X_s\rangle \leftarrow \frac{|X_s\rangle}{\sqrt{\langle X_s | X_s \rangle}}$ 
16: Return  $|X_s\rangle$ 
Output: Permutation invariant statevector  $|X_s\rangle$ 

```

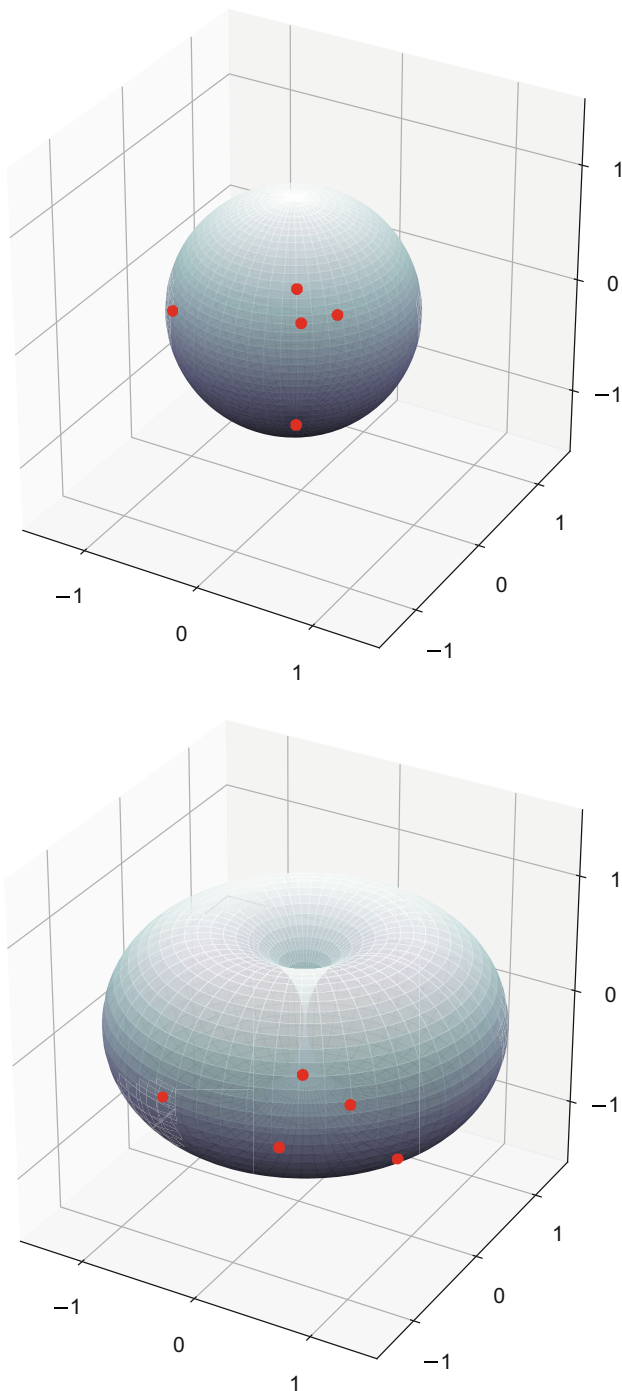
---

respectively. The performance of various models is then evaluated on the testing set. The entire process is then repeated with new randomly generated datasets and we record the average test accuracy for 10 repeated experiments.

The sphere and torus distributions were both centred at the origin. To ensure that the sphere and torus distributions are as similar as possible, the torus was scaled such that the average magnitude of the points that lie on the torus distribution surface matches the radius of the sphere. An illustration of the two distributions and an example point cloud sample is shown in Fig. 2. All data was normalised to be in the range  $\frac{\pi}{2}$  to  $-\frac{\pi}{2}$  to allow it to be encoded as rotation angles.

#### 3.2 Encoding process for individual points

We tested a variety of quantum and classical techniques, summarised in Table 1, where some of the algorithms contain permutation invariance and others do not. In order to focus entirely on the encoding method, without having to consider the structure of a variational component, we used a QSVM to classify the data with various choices of encoding circuits. For the order permutation invariant encoding, we tested several different point encoding circuits  $U$  for encoding the individual points, showing results for the best-performing circuit denoted  $U_\alpha$  alongside a more generic point encoding circuit that uses the instantaneous quantum polynomial (IQP) encoding (Havlíček et al. 2019), denoted by  $U_\beta$ . We provide a brief description of the different methods reported:



**Fig. 2** (Left) Sphere distribution and (right) torus distribution surfaces from which points are sampled to form point clouds. The red points in each figure represent an example point cloud with  $n = 5$  points drawn from the corresponding distribution. Each dataset is generated by randomly sampling these distributions to create a set of  $N$  different point clouds

- **Permutation Invariant QSVM (Best)** — a QSVM with the permutation invariant encoding method using the point encoding circuit  $U_\alpha$  that provided the highest accuracy, as shown in Fig. 3.

**Table 1** Summary of the various algorithms tested indicating whether they contain permutation invariance in their design and whether they are quantum or classical approaches

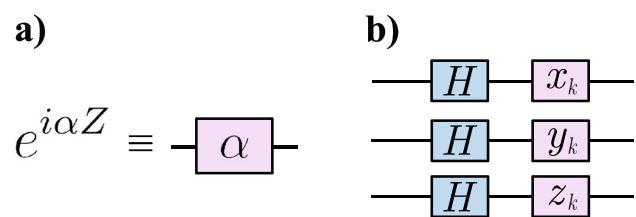
	Quantum	Classical
Permutation invariant	Permutation Invariant QSVM (Best)	PointNet
Non-invariant	Permutation Invariant IQP QSVM IQP Encoding QSVM	RBF Kernel SVM

- **Permutation Invariant IQP QSVM** — a QSVM with the permutation invariant encoding method where the point encoding circuit  $U_\beta$  is an IQP encoding, as shown in Fig. 4. This is to provide a more fair comparison between the regular IQP encoding and the invariant encoding.
- **IQP Encoding QSVM** — a QSVM using the IQP encoding applied to all variables in the input (without permutation invariance) as described by Havlíček et al. (2019).
- **PointNet** — classical point cloud classifier algorithm utilising neural networks with a symmetric max pool function to ensure point order permutation invariance (Qi et al. 2016). PointNet was run over 100 training epochs.
- **RBF Kernel SVM** — classical SVM using the radial basis function (RBF) kernel. Hyperparameters were optimised using grid search over a cross-validation set.

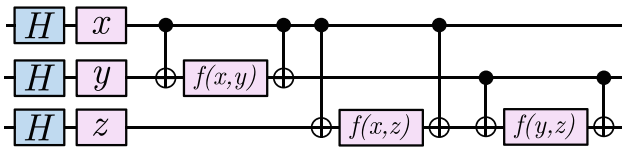
Our results were obtained from noiseless simulations using Qiskit *statevector\_simulator* (Abraham et al. 2019). This allows us to obtain a quantum state  $|p_i\rangle$  for each point in the point cloud. Notably as we only simulate a single point at a time, and each point uses  $k$  qubits, the scaling of this simulation step is at worst  $\mathcal{O}(n2^k)$  still maintaining a linear scaling in the number of points  $n$ . A fully quantum implementation would scale  $\mathcal{O}(nk)$  in the number of qubits used.

### 3.3 Symmetric state preparation

While a quantum circuit capable of probabilistically preparing symmetric superposition states is shown in Sect. 4, for



**Fig. 3** **a** Pink boxes represent parameterised Z rotation gates. **b** Single layer of the best-performing point encoding circuit  $U_\alpha$  found for the sphere/torus dataset during our investigation, which is used for results titled Permutation Invariant QSVM (Best)



**Fig. 4** The point encoding circuit  $U_\beta$  that uses the IQP encoding (Havlíček et al. 2019), which corresponds to results titled Permutation Invariant IQP QSVM. The entanglement function was defined as  $f(x, y) = \frac{1}{\pi}(\pi - x)(\pi - y)$

relatively small-scale classical simulations, a brute force method can be used to calculate all the permutations of the quantum states  $|\mathbf{p}_i\rangle$ , sum them together, and then normalise the resulting state to find the symmetrised state

$$|X_s\rangle = \mathcal{N}_n \sum_{\sigma \in \mathcal{S}_n} |\mathbf{p}_{\sigma_1}\rangle |\mathbf{p}_{\sigma_2}\rangle \dots |\mathbf{p}_{\sigma_n}\rangle. \tag{7}$$

This brute force classical technique scales as  $\mathcal{O}(n!)$ , which would quickly become infeasible for large  $n$ . The fully quantum implementation discussed in Sect. 4 utilises  $kn + \frac{1}{2}kn(kn - 1)$  qubits and hence would exhibit scaling of  $\mathcal{O}(k^2n^2)$  which is more feasible for large  $n$  than classical alternatives. An extra consideration in a real quantum device is that the state is only prepared with a certain success probability; hence, the true scaling of the quantum model may be worse than  $\mathcal{O}(k^2n^2)$ . However, our empirical investigation in Appendix F suggests the additional scaling would not be worse than polynomial.

### 3.4 Quantum support vector machine integration

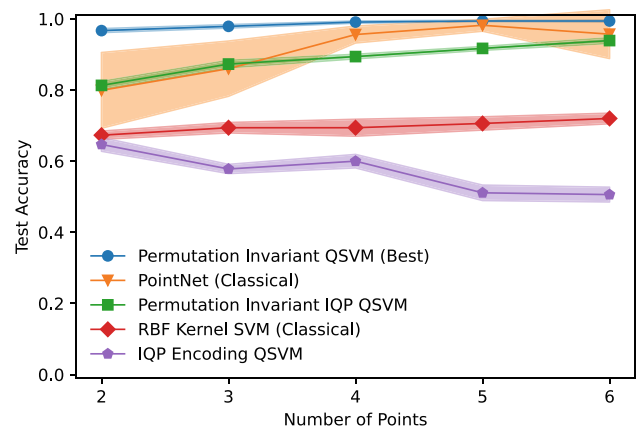
After preparing a symmetrised quantum state  $|X_s\rangle$ , the point cloud  $X$  has successfully been encoded in a manner which is invariant to permutations of the point orderings. This is a key novel proposal of this work, as it subsequently allows any classification algorithm that utilises this symmetric state as an input to remain permutation invariant. This is different to many other approaches in the literature that often rely on utilising a permutation equivariant encoding followed by a permutation equivariant variational ansatz to guarantee label invariance in the overall model (Meyer et al. 2023; West et al. 2024).

As we have implemented permutation invariance into the quantum encoding step itself, we are able to utilise any classification technique and retain permutation invariance. For this study, we chose to use a quantum support vector machine (QSVM) to perform the classification. A support vector machine (SVM) works by finding a hyperplane that maximally separates data  $x$  that has been encoded into some higher dimensional space as  $\phi(x)$ . A crucial feature of a SVM is that the explicit form of all  $\phi(x_i)$  need not be known, only the inner product between them for all data points

$K_{i,j} = \phi(x_i)^T \phi(x_j)$ , with the entries forming a matrix called the kernel matrix. It has been proposed that QSVMs can perform classification by using the overlap of quantum states as the kernel entries (Havlíček et al. 2019). In this case, the kernel entries are given by  $K_{i,j} = |\langle \psi(X_i) | \psi(X_j) \rangle|^2$ . Once we have prepared the symmetric quantum states  $|X_i\rangle$  for each data point  $X_i$ , then we can calculate the kernel entries by using a swap test, or otherwise, to calculate  $K_{i,j} = |\langle X_i | X_j \rangle|^2$  (Havlíček et al. 2019). In the case of classical simulations, we can directly calculate the inner product between the vector representation of the quantum states, which in general would scale as  $\mathcal{O}(2^n)$ .

### 3.5 Results

Figure 5 displays how the various techniques scale as the number of points in the point cloud increases. Although more points provide more information, the non-invariant IQP QSVM’s performance decreases as the number of points increases. This finding is consistent with previous research indicating that generic QSVM methods may struggle to generalise as the number of qubits increases (Huang et al. 2021). In contrast, the permutation invariant IQP encoding exhibits an improvement in accuracy as the number of points increases. This result indicates that the symmetrisation technique may help prevent poor scaling due to the reduced expressivity in the encoding. Additionally, our tests reveal that using the best encoding circuit  $U_\alpha$  produces a classifier that can outperform the classical PointNet algorithm for this dataset. This is further demonstrated in the results depicted in Table 2 which shows that for small datasets, the permutation invariant quantum encoding outperforms both



**Fig. 5** The average accuracy over 10 repeated experiments as the number of points in each point cloud increases. Shaded regions indicate the error bounds on the average accuracy. Each experiment consists of a random dataset sample of 500 point clouds. Each point cloud is generated by randomly sampling a number of points from either the torus or the sphere distribution. The training and testing data contained 80% and 20% of the total data respectively

**Table 2** The average accuracy over 10 repeated experiments

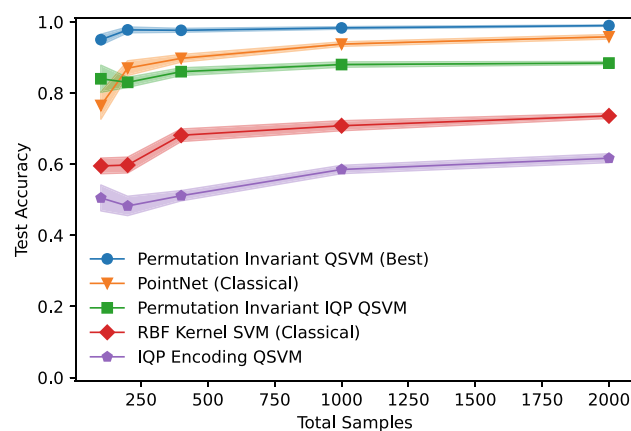
Algorithm	Accuracy
Permutation Invariant (Best)	$0.950 \pm 0.015$
PointNet (Classical)	$0.765 \pm 0.038$
RBF Kernel SVM (Classical)	$0.595 \pm 0.022$
Permutation Invariant IQP QSVM	$0.840 \pm 0.038$
IQP Encoding QSVM	$0.505 \pm 0.036$

Each experiment consists of a random dataset sample of 100 point clouds consisting of 3 points each run using Qiskit *statevector\_simulator*. The training and testing data contained 80% and 20% of the total data, respectively

non-invariant quantum/classical methods and the permutation invariant classical method PointNet.

The effect of increasing the size of the data sample is shown in Fig. 6. In this case, more data is available, but the number of points, and thus qubits is fixed. All algorithms tested generally improve with more data samples, as expected. Comparing again to Fig. 5 shows that the IQP encoding, while improving with more data samples generally, is specifically struggling when there is an increase in qubits. This problem is not apparent with the permutation invariant encodings.

For larger data samples, PointNet starts to approach the accuracy of the permutation invariant QSVM encoding. It is worth noting that PointNet uses deep neural networks consisting of a total of 3.5 million parameters that can be better utilised with a large amount of training data. There are also additional aspects to the PointNet algorithm that tackle



**Fig. 6** The average accuracy over 10 repeated experiments as the number of samples in the training and testing dataset increases. Shaded regions indicate the error bounds on the average accuracy. Each experiment consists of a random dataset sample of point clouds. Each point cloud is generated by randomly sampling 3 points from either the torus or the sphere distribution. The training and testing data contained 80% and 20% of the total data respectively

geometric symmetries, such as rotational invariance, in point cloud data that have not been implemented in this quantum approach. Implementing these geometric symmetries into the encoding could be a subject of further investigation.

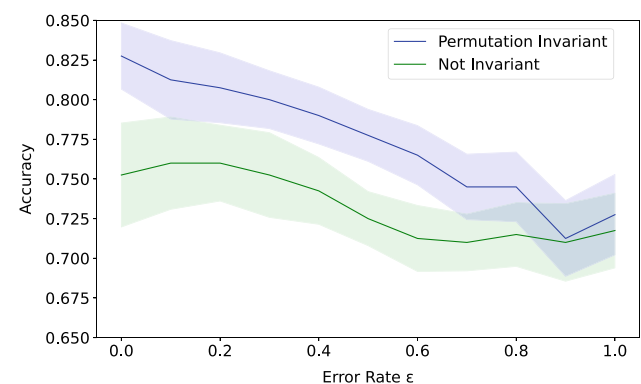
### 3.6 Quantum errors

If during the encoding step the points are subject to a source quantum noise, then it is expected that the effectiveness of the overall classification will decrease. In order to investigate this effect, we considered point clouds where the individual points had initially been encoded into a state  $|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle\dots|\mathbf{p}_n\rangle$ . We then introduced random errors to these initial point states by generating a random Hermitian matrix  $H_\gamma$  and subsequently applying the unitary matrix

$$U_\gamma = e^{i\epsilon H_\gamma}, \quad (8)$$

where  $\epsilon$  parameterises the magnitude of the error. We generate  $n$  different unitaries in this manner and one to each qubit in the state  $|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle\dots|\mathbf{p}_n\rangle$ .

Figure 7 shows how the QSVM classifier performs after errors have been applied to the initial states, either with or without the symmetrisation process applied to the state  $|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle\dots|\mathbf{p}_n\rangle$ . It can be seen that the advantage obtained by the symmetrisation procedure is diminished at a certain threshold of noise in the initial point states. This suggests a possible drawback of the symmetrisation method in this work is that it may not exhibit an advantage if the initial input states are accompanied by a certain amount of noise.



**Fig. 7** Mean test accuracy of the QSVM classifier as the noise applied to the initial point states is increased. Blue indicates that the permutation invariant symmetrisation suggested in this work was performed; green indicates no symmetrisation procedure was used (the encoding process simply consisted of the initial point states in one particular order). Mean test accuracy reported is the mean average of ten repeated experiments, with the shaded region indicating uncertainty of the mean. The points were encoded using the IQP encoding as described in Sect. 3. A total of 200 point cloud samples were used per experiment

### 4 Symmetric state projection circuit implementation

The results in Sect. 3 suggest that permutation invariant encodings could be useful for point cloud data. These results were however created using analytical simulations, which required  $\mathcal{O}(n!)$  classical operations. We now present a discussion on the practical aspects of implementing them the encoding directly onto a real quantum device.

#### 4.1 Symmetric projection for two points

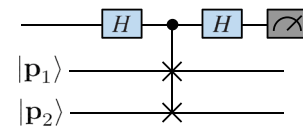
Initially let us consider a point cloud  $X$  consisting of only two points,  $\mathbf{p}_1$  and  $\mathbf{p}_2$ . As described previously, these points are separately encoded into quantum states represented by  $|\mathbf{p}_1\rangle$  and  $|\mathbf{p}_2\rangle$ . It has been shown that the symmetrisation process  $|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle \Rightarrow \mathcal{N}(|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle + |\mathbf{p}_2\rangle|\mathbf{p}_1\rangle)$  cannot be done perfectly by a unitary transformation (Buzek and Hillery 2000). However, it can be implemented in a probabilistic manner using controlled swap gates and ancilla qubits (Barenco et al. 1996). For a point cloud data sample that contains only two points, we start with each point having been encoded in a separate quantum state  $|\mathbf{p}_1\rangle = U(\mathbf{p}_1)|0\rangle^{\otimes k}$  and  $|\mathbf{p}_2\rangle = U(\mathbf{p}_2)|0\rangle^{\otimes k}$ , using an encoding circuit  $U$ . The symmetrisation procedure needs to produce the state

$$|X_s\rangle = \mathcal{N}(|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle + |\mathbf{p}_2\rangle|\mathbf{p}_1\rangle),$$

such that exchanging the ordering of the two points in the input will leave this new quantum state invariant. This can be achieved in the two-qubit case by preparing an ancilla qubit using a Hadamard gate in the state  $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$  and using it to apply a controlled swap gate to the two input states  $|\mathbf{p}_1\rangle$  and  $|\mathbf{p}_2\rangle$ , followed by another Hadamard gate applied to the ancilla qubit. This action leaves the system in the state

$$\frac{1}{2}|0\rangle(|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle + |\mathbf{p}_2\rangle|\mathbf{p}_1\rangle) + \frac{1}{2}|1\rangle(|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle - |\mathbf{p}_2\rangle|\mathbf{p}_1\rangle), \quad (9)$$

which contains both the permutation symmetrised and anti-symmetrised states. By measuring the ancilla qubit and discarding any result when the ancilla qubit is in the  $|1\rangle$  state (corresponding to the anti-symmetric state), we arrive at the permutation symmetrised state whenever the ancilla qubit is measured in the  $|0\rangle$  state. Inspecting Eq. 9, it can be seen that the probability of measuring the ancilla qubit in the desired  $|0\rangle$  state is  $\frac{1}{2}(1 + |\langle \mathbf{p}_1 | \mathbf{p}_2 \rangle|^2)$  (Buzek and Hillery 2000), which means that in the worst-case scenario, when the input states are orthogonal, the probability is  $\frac{1}{2}$ . This symmetrisation procedure for a two-qubit system is illustrated in Fig. 8.



**Fig. 8** Permutation symmetrisation circuit for two points. The circuit consists of a  $|\mathbf{p}_1\rangle$  state, a  $|\mathbf{p}_2\rangle$  state, and an ancilla qubit that performs a controlled swap operation. The final state of this circuit is given by  $|X\rangle = \frac{1}{2}|0\rangle(|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle + |\mathbf{p}_2\rangle|\mathbf{p}_1\rangle) + \frac{1}{2}|1\rangle(|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle - |\mathbf{p}_2\rangle|\mathbf{p}_1\rangle)$ . This state is symmetric when the ancilla qubit is measured in the state  $|0\rangle$  and anti-symmetric when it is measured in the state  $|1\rangle$ . By measuring the ancilla qubit and discarding any measurements in the state  $|1\rangle$ , we are left with the desired symmetric quantum state  $|X_s\rangle = \mathcal{N}(|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle + |\mathbf{p}_2\rangle|\mathbf{p}_1\rangle)$

#### 4.2 Generalisation to $n$ points

This procedure can be generalised to  $n$  qubits using the technique outlined by Barenco et al. (1996), which involves the iterative application of controlled swap symmetrisation operations using ancilla qubits. In this technique, we group up the ancilla qubits into  $n - 1$  collections. The collection indexed by  $f$  will contain  $f$  ancilla qubits. The unitary operator  $V_f$  prepares  $f$  ancilla qubits into an equal superposition of all states with Hamming weight 0 and 1. This state can be written as

$$\frac{1}{\sqrt{f+1}}(|00\dots 0\rangle + |10\dots 0\rangle + |01\dots 0\rangle + \dots + |00\dots 1\rangle). \quad (10)$$

This is carried out for the collections labelled from  $f = 1$  to  $f = n - 1$ ; hence, in total, there are  $c = \frac{1}{2}n(n - 1)$  ancilla qubits. Construction of the gates that can implement  $V_f$  is discussed in Appendix E.

These collections of ancilla qubits are then used to apply controlled swap gates onto the input states. This is done in an iterative manner. Unitary  $V_1$  will prepare the state  $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ , which will control a swap gate between states  $|\mathbf{p}_1\rangle$  and  $|\mathbf{p}_2\rangle$ . This produces  $\frac{1}{\sqrt{2}}(|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle + |\mathbf{p}_2\rangle|\mathbf{p}_1\rangle)$  as a final outcome. Considering the group elements of  $S_n$ , we could consider the action of this ancilla as acting on the initial state with a combination of  $I + \sigma_{12}$ . Subsequently, unitary  $V_2$  prepares the state  $\frac{1}{\sqrt{3}}(|00\rangle + |01\rangle + |10\rangle)$ , which controls swaps between the the third qubit and the first two in the state  $\frac{1}{\sqrt{2}}(|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle + |\mathbf{p}_2\rangle|\mathbf{p}_1\rangle) \otimes |\mathbf{p}_3\rangle$ . This resorts in the creation of the state

$$\frac{1}{\sqrt{6}}(|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle|\mathbf{p}_3\rangle + |\mathbf{p}_3\rangle|\mathbf{p}_2\rangle|\mathbf{p}_1\rangle + |\mathbf{p}_1\rangle|\mathbf{p}_3\rangle|\mathbf{p}_2\rangle + |\mathbf{p}_2\rangle|\mathbf{p}_1\rangle|\mathbf{p}_3\rangle + |\mathbf{p}_3\rangle|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle + |\mathbf{p}_2\rangle|\mathbf{p}_3\rangle|\mathbf{p}_1\rangle).$$



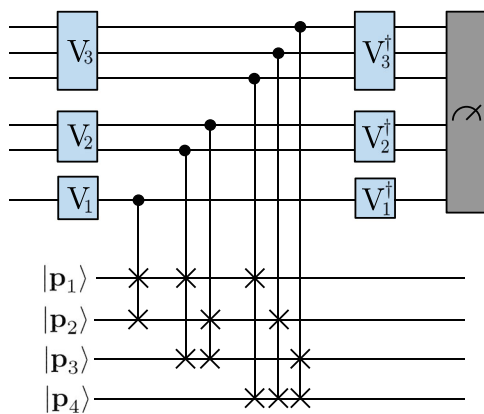
To get a clearer view, one may consider these two steps as applying the permutations

$$(I + \sigma_{12})(I + \sigma_{13} + \sigma_{23}) = I + \sigma_{12} + \sigma_{13} + \sigma_{23} + \sigma_{312} + \sigma_{231}. \tag{11}$$

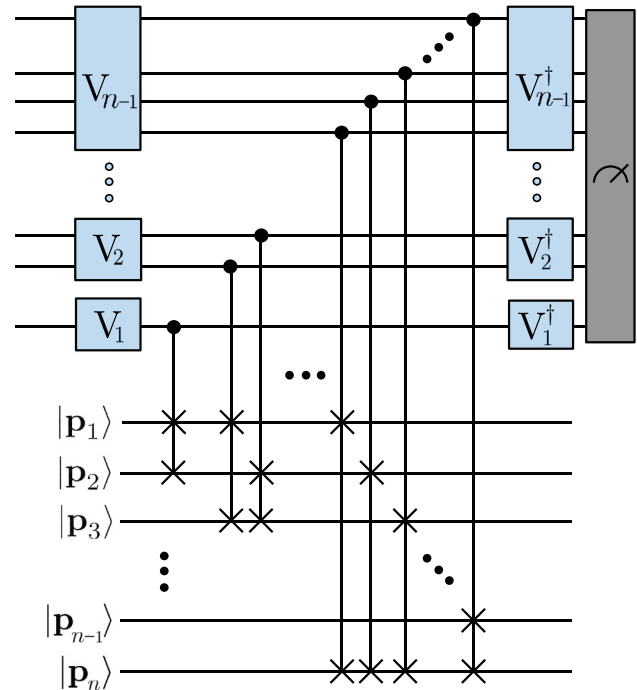
As we iteratively multiply by the sum of all two-point permutations for each new point added in the iteration, the end result is a combination of all possible permutations in  $S_n$ , providing an equal superposition of all permutations.

This process is followed by an application of the inverse of the unitary operations  $V_f^\dagger$  on the ancilla qubits before their measurement. This recursively applies every possible qubit permutation to the input states, resulting in the desired symmetric superposition state when the ancilla qubits are measured in the state  $|0\rangle^{\otimes c}$ . An example circuit for a four-qubit scenario is shown in Fig. 9 and an example of the generalisation to  $n$  qubits is shown in Fig. 10. When each state is composed of multiple qubits, no extra ancilla qubits are necessary. Instead, additional controlled swap gates are applied to the extra qubits in the same manner (Buzek and Hillery 2000). This is demonstrated in Fig. 11, which shows the symmetrisation of a two-dimensional point cloud consisting of only two points.

In general, one will require  $\frac{1}{2}n(n - 1)$  ancilla qubits and  $\frac{1}{2}n(n - 1)$  controlled swap gates to perform the symmetrisation.



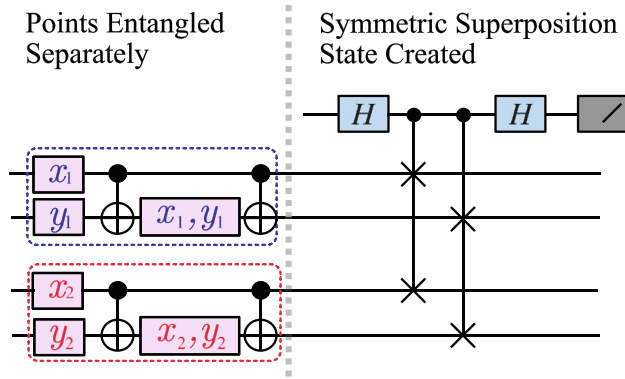
**Fig. 9** Generalisation of the permutation symmetrisation process, as proposed by Barenco et al. (1996), applied to a circuit containing four input states. The unitary operators  $V_3$  perform a transformation on the ancilla qubits such that they are in the state  $\frac{1}{\sqrt{4}}(|000\rangle + |100\rangle + |010\rangle + |001\rangle)$ . Similarly,  $V_2$  prepares  $\frac{1}{\sqrt{3}}(|00\rangle + |10\rangle + |01\rangle)$  and  $V_1$  prepares  $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ . Through implementing controlled swap gates, these ancilla qubits will perform every possible permutation of the input states. This results in an equal superposition of every permutation of the input states after the controlled swap gates are applied and the ancilla qubits, after having  $V^\dagger$  applied, are measured to be in the  $|000000\rangle$  state. This process can be extended to any number of input states



**Fig. 10** Generalisation of the permutation symmetrisation process, as proposed by Barenco et al. (1996), applied to a circuit containing  $n$  input states. The unitary operators  $V_{n-1}$  perform a transformation on the ancilla qubits such that they are in the state with an equal superposition of all basis states with hamming weight zero or one. If the ancilla qubits are measured to be in the zero state, then the input states will have been initialised into a superposition of every possible permutation

tion. We are also required to implement unitary gates  $V_f$  on the ancilla qubits which can be created out of a single qubit gate and  $f - 1$  two qubit gates. Hence counting the  $V_f^\dagger$  gates as well, we would require  $n(n - 1)$  two qubit gates to deal with the ancilla unitaries. Overall the number of gates and qubits for the quantum algorithm scales  $\mathcal{O}(n^2)$ .

One drawback of this probabilistic implementation is the need to discard any states when the ancilla qubits are not in the state  $|0\rangle^{\otimes c}$ . The probability of this happening depends on the states themselves, with the probability being 1 if the input states happen to be all identical, and decreasing as the overlap between states decreases. The probability of measuring the ancilla qubit in the state  $|0\rangle$  for the case of two qubits can be shown to be  $\frac{1}{2}(1 + |\langle \mathbf{p}_1 | \mathbf{p}_2 \rangle|^2)$  (Buzek and Hillery 2000). In this work, we utilised angle encoding for the point encoding circuit  $U$  to produce the input states  $|\mathbf{p}_1\rangle$  and  $|\mathbf{p}_2\rangle$ . This technique results in a relatively low probability for two specific data points to be orthogonal. As a result, the average probability of success remains relatively high even with an increase in the number of data points  $n$ . Additional details and supporting evidence for this claim are presented in Appendix F.



**Fig. 11** Permutation symmetrisation of a 2-dimensional point cloud consisting of two points. On the left side, we first create an entangled state for each point using an encoding function  $U$ . On the right-hand side, we implement the symmetrisation process for a two-state system. If we discard any measurements in which the ancilla qubit is in the state  $|1\rangle$ , we will have prepared a symmetric encoding of this point cloud

Although this method would be viable for small point clouds, the scaling of the probabilistic ancilla-based symmetrisation becomes problematic when a point cloud consists of a large number of points. This is because as the number of ancilla qubits increases, so does the number of gates in the circuit, as well as the number of states that need to be discarded. An alternative approach could be to approximately implement these states by producing states of the form  $|X_e\rangle \approx \mathcal{N}_e(|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle + |\mathbf{p}_2\rangle|\mathbf{p}_1\rangle)$ . This could potentially be achieved using techniques such as genetic state preparation algorithms (Creevey et al. 2023) or quantum generative adversarial networks (Zoufal et al. 2019). Creating approximate symmetric states within some error  $\epsilon$  could even be used to introduce a parameterised symmetry-breaking term that may help fine-tune the model, as has been shown to be useful in some variational quantum eigensolvers (Park 2021). A parameterised method to increase the dimensionality of the problem through symmetry breaking would allow control over the expressibility of the encoding, as well as potentially making the encoding harder to simulate classically.

Alternatively, there is also the possibility of not discarding any states and instead using other superposition states (mixtures of anti-symmetric and symmetric permutation terms with respect to different points). For example, in the two-point case, one could accept the anti-symmetric state  $|X_a\rangle = \mathcal{N}_a(|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle - |\mathbf{p}_2\rangle|\mathbf{p}_1\rangle)$ . This state will exhibit a quasi-symmetry through the fact that permuting points results in a phase shift in the quantum state, but keeps their relative magnitudes intact. The effect that quasi-symmetric or approximately symmetric encodings would have on a QML technique, especially in situations where the model may be able to learn to overcome this peculiarity, is a possible subject for further research.

## 5 Conclusion

This study presents a method for encoding point cloud data into a quantum state that is invariant under point order permutation, using a symmetrisation process to create a quantum superposition of all order permutations of the points. This exponentially reduces the dimensionality of the encoding, leading to an encoding that exhibits better generalisation. This was demonstrated by noting that the permutation invariant IQP encoding accuracy scaled up as the number of points increased, whereas a non-invariant IQP encoding performed worse as the number of points increased. These findings suggest that this method may be a promising solution to the problem of QSVM generalisation performance worsening with increasing qubits (Kübler et al. 2021; Huang et al. 2021), and may have potential applications in future QML algorithms in various fields such as object recognition and particle physics.

This work demonstrates an encoding method to improve generalisation for permutation invariant data; however, it does not necessarily guarantee a quantum advantage exists. Recently there have been results showing that qubit permutation invariant operators may be classically tractable under certain conditions due to their reduction in dimensionality (Anschuetz et al. 2023). A key difference between our work and methods that utilise equivariant variational models (Meyer et al. 2023; Nguyen et al. 2022; Schatzki et al. 2022; Kazi et al. 2023) is that the method presented in this paper implements permutation invariance directly into the encoding step, without considering the classification model. Hence, the variational model does not need to be permutation invariant in order to capture the symmetry. Further investigation into methods of efficient implementation could help improve the technique by overcoming problems such as the  $\mathcal{O}(n^2)$  scaling of ancilla qubits in the probabilistic symmetrisation circuit (Barenco et al. 1996). Additionally, there have recently been efficient methods demonstrated for finding expectation values of symmetric states that could be of use in this technique (Zhang and Tong 2023).

Future work could focus on extending this encoding to other types of symmetries, such as rotational symmetry or translation symmetry, which could be relevant for point clouds and other data types such as images or time series. There is also the possible challenge of finding an efficient implementation on real devices and assessing how the encodings will perform in the presence of noise. The technique suggested here could be used for any data that exhibits permutation invariance, including cases where the input data itself is quantum. Real applications will be dependent on the rate of technological advancement of quantum machines; however, near-term use cases could focus on point clouds with a small number of points, as is often the case in particle physics data.

## Appendix A: Implementation

The methodology employed in this study utilises the quantum support vector machine (QSVM) approach. The QSVM technique is based on the encoding of classical data, denoted by  $X$ , into a higher dimensional space to perform classification tasks. The encoding circuit  $\mathcal{U}$  establishes a mapping  $\mathcal{U} : X \rightarrow |\psi(X)\rangle\langle\psi(X)|$ , which transforms classical data  $X$  into a quantum state represented by a density matrix  $|\psi(X)\rangle\langle\psi(X)|$ . Importantly, it is not necessary to explicitly define the higher dimensional encoding  $|\psi(X)\rangle\langle\psi(X)|$ , only the inner product between data points in the higher dimensional space,  $|\langle\psi(X_i)|\psi(X_j)\rangle|^2$ . The inner product between two points  $i$  and  $j$  becomes entry  $K_{i,j}$  of the kernel matrix  $K$ . Consequently, the actual device implementation consists of calculating the overlap between two quantum states to approximate the kernel matrix, which is determined by measuring the quantity  $K_{i,j} = |\langle 0|\mathcal{U}(X_i)\mathcal{U}^\dagger(X_j)|0\rangle|^2$ . Therefore, an encoding circuit  $\mathcal{U}$  is required, involving the proposed permutation symmetrisation step and its conjugate version.

For the simulation implementation, we employed the Qiskit *statevector\_simulator* to compute the kernel using mathematical methods for determining the exact symmetric states. A point cloud  $X$  contains  $n$  points, with the coordinates of point  $i$  given by  $\mathbf{p}_i$ . In our encoding circuit  $\mathcal{U}$ , each point  $\mathbf{p}_i$  is initially encoded into a quantum state using the point encoding circuit  $U : \mathbf{p}_i \rightarrow |\mathbf{p}_i\rangle$ . As the *statevector\_simulator* was utilised in this process, the amplitudes of all  $|\mathbf{p}_i\rangle$  are known. Thus, we can compute a permutation by finding the tensor product of all point quantum states. We can subsequently sum over all  $n!$  possible permutations to derive the permutation invariant statevector for the entire point cloud  $|X_s\rangle = \mathcal{N} \sum_{\sigma \in S_n} |\mathbf{p}_{\sigma^{-1}(1)}\rangle|\mathbf{p}_{\sigma^{-1}(2)}\rangle \dots |\mathbf{p}_{\sigma^{-1}(n)}\rangle$  and normalising the state. Consequently, it is feasible to directly compute the quantity  $|\langle\psi(X_i)|\psi(X_j)\rangle|^2$  for any two-point clouds. This, in turn, allows for the calculation of the entire kernel matrix for a dataset in a manner that is simpler to calculate classically than simulating entire circuits. This algorithm underpins the theoretical results presented in this paper. The process of generating the permutation invariant state is summarised in Algorithm 1.

In this study, we employed the QSVM method to concentrate solely on the encoding step, without considering any variational ansatz structure. However, in practice, using the real circuit implementation discussed in Sect. 4, the QSVM approach may prove to be rather inefficient due to the necessity of constructing two permutation invariant states simultaneously when computing the kernel entries if using this method. This means that in practice it may be more efficient to use a variational ansatz circuit, placed after the encoding circuit.

## Appendix B: Distribution specifications

The torus distribution was generated using the following

$$x = (1 + \cos(s)) \cos(t) \tag{B.1}$$

$$y = (1 + \cos(s)) \sin(t) \tag{B.2}$$

$$z = \sin(s). \tag{B.3}$$

The average magnitude of the points in this distribution was then calculated to be approximately 1.28. We then scaled down the distribution by this factor so that the average magnitude was 1, matching the radius of the sphere.

## Appendix C: Noise generation specification

A random matrix  $A$  of complex numbers was generated where the real and imaginary components for each entry were drawn from a uniform distribution. This is converted to a Hermitian matrix  $H$  using the process  $H = A^\dagger + A$ . This was then normalised to have trace of zero through  $H \rightarrow H - \text{Tr}(H)I$ . This can then be used to find a unitary matrix through exponentiation  $U_\gamma = e^{i\gamma H}$ .

## Appendix D: Probabilistic symmetric encoding

Here we shall discuss the scaling of the worst-case probability of being able to prepare a symmetric states using the circuit suggested by Barenco et al. (1996). Looking back at the two-qubit example shown in Fig. 8, we start with a control qubit in the state  $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ . This is the control qubit for a controlled swap gate that acts on the two initial states  $|\mathbf{p}_1\rangle$  and  $|\mathbf{p}_2\rangle$  resulting in the following state

$$\frac{1}{\sqrt{2}}(|0\rangle|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle + |1\rangle|\mathbf{p}_2\rangle|\mathbf{p}_1\rangle). \tag{D.4}$$

The procedure then applies a Hadamard gate to the control qubit prior to measurement in the Z basis. This takes the state to the following

$$|\psi\rangle = \frac{1}{2}(|0\rangle(|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle + |\mathbf{p}_2\rangle|\mathbf{p}_1\rangle) + \frac{1}{2}|1\rangle(|\mathbf{p}_1\rangle|\mathbf{p}_2\rangle - |\mathbf{p}_2\rangle|\mathbf{p}_1\rangle) \tag{D.5}$$

From here, the probability of measuring the ancilla qubit in the  $|0\rangle$  state is given by applying the probability operator

$P(0) = \langle \psi | (|0\rangle\langle 0| \otimes I \otimes I) | \psi \rangle$ . This can be written as

$$\frac{1}{4} \left( \langle \mathbf{p}_1 | \langle \mathbf{p}_2 | + \langle \mathbf{p}_2 | \langle \mathbf{p}_1 | \right) \left( | \mathbf{p}_1 \rangle | \mathbf{p}_2 \rangle + | \mathbf{p}_2 \rangle | \mathbf{p}_1 \rangle \right), \tag{D.6}$$

which can then be evaluated to

$$\frac{1}{2} \left( 1 + | \langle \mathbf{p}_1 | \mathbf{p}_2 \rangle |^2 \right). \tag{D.7}$$

Note that the lowest probability this can reach is  $\frac{1}{2}$  in the case where  $| \langle \mathbf{p}_1 | \mathbf{p}_2 \rangle | = 0$ . Once we measure  $|0\rangle$  in the ancilla qubit we will collapse the state into the symmetric superposition, which is invariant under qubit permutation. We can write this state as

$$|X_s\rangle = \mathcal{N} ( | \mathbf{p}_1 \rangle | \mathbf{p}_2 \rangle + | \mathbf{p}_2 \rangle | \mathbf{p}_1 \rangle ).$$

In order to calculate the normalisation constant  $\mathcal{N}$ , just note that  $| \langle X_s | X_s \rangle | = 1$  and from this we can calculate

$$| \mathcal{N} |^2 ( 2 + 2 | \langle \mathbf{p}_1 | \mathbf{p}_2 \rangle |^2 ) = 1, \tag{D.8}$$

$$\mathcal{N} = \frac{1}{\sqrt{2(1 + | \langle \mathbf{p}_1 | \mathbf{p}_2 \rangle |^2)}}. \tag{D.9}$$

### Appendix E: Probabilistic symmetric ancilla preparation

In the method proposed by Barenco et al. (1996), one of the steps involves applying a unitary  $V_f$  that prepares  $f$  ancilla qubits into the state

$$\frac{1}{\sqrt{f+1}} ( |00\dots 0\rangle + |10\dots 0\rangle + |01\dots 0\rangle + \dots + |00\dots 1\rangle ) \tag{E.10}$$

In order to create this state, we need to define two unitary operators

$$R_f = \frac{1}{\sqrt{f+1}} \begin{pmatrix} 1 & -\sqrt{f} \\ \sqrt{f} & 1 \end{pmatrix}, \tag{E.11}$$

$$T_f, j = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{f-j+1}} & \frac{\sqrt{f-j}}{\sqrt{f-j+1}} & 0 \\ 0 & -\frac{\sqrt{f-j}}{\sqrt{f-j+1}} & \frac{1}{\sqrt{f-j+1}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{E.12}$$

If we have  $f$  control qubits initially in the  $|0\rangle^{\otimes f}$  state. Then, we can prepare the desired state by applying  $R_f$  to the first qubit and  $T_f, j$  to adjacent qubits starting from  $j = 1$  to  $j = f - 1$ . This entire procedure creates the unitary operation  $V_f$ . This is repeated for  $f = 1, 2, \dots, n - 1$  where we have  $n$  qubits in total to symmetrise, resulting in a total of  $c = \frac{1}{2}n(n - 1)$  ancilla qubits used.

**Table 3** The average probability of producing permutation symmetrised states when given input states that were randomly initialised using angle encoding via an X rotation gate (Barenco et al. 1996)

Initial states $n$	Mean probability
2	0.842
3	0.752
4	0.630
5	0.530

This corresponds to the probability of measuring the ancilla qubits in the state  $|0\rangle^{\otimes c}$ . Each circuit was run over 10,000 shots using Qiskit *qasm\_simulator* and the proportion of  $|0\rangle^{\otimes c}$  measured. This was then repeated over 1000 separate runs, using randomly initialised input states each time, reporting the average probability found over all runs. Note that the lower bound for the probabilities is much lower, but it is unlikely to be reached with input states randomly initialised via angle encoding

### Appendix F: Success probability for symmetric preparation

We present a small study of the probability of success when using the probabilistic technique for preparing permutation symmetric states described by Barenco et al. (1996). We use angle encoding with random inputs to generate initial states with the results shown in Table 3.

The lowest values of probability are found when initialising input states that are maximally orthogonal to each other. Note that in practice the mean probability was found to be far higher than this when using random initial states, due to the fact that random states are unlikely to be maximally mutually orthogonal. Note that if basis state encoding was used, instead of angle encoding, then the possibility of orthogonal states would potentially be much higher, and thus the chance of successfully symmetrising the states significantly lower.

**Acknowledgements** This work was supported by the University of Melbourne through the establishment of an IBM Quantum Network Hub at the University. This research was supported by the Australian Research Council from grant DP210102831. JH acknowledges the support of the Research Training Program Scholarship and the N.D. Goldsworthy Scholarship. CDH was supported by a research grant from the Laby Foundation.

**Author Contributions** J.H. wrote the main manuscript text. J.H. developed the algorithms described in the manuscript in collaboration with the other authors. All authors reviewed the manuscript.

**Funding** Open Access funding enabled and organized by CAUL and its Member Institutions.

**Data Availability** The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

### Declarations

**Conflict of interest** The authors declare no competing interests.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Abraham H, AduOffei Agarwal R, Akhalwaya IY, Aleksandrowicz G, et al (2019) Čepulkovskis: Qiskit: an open-source framework for quantum computing. <https://doi.org/10.5281/zenodo.2562110>
- Anand A, Lyu M, Baweja PS, Patil V (2022) Quantum image processing. *arXiv*. <https://doi.org/10.48550/arxiv.2203.01831>
- Anschuetz ER, Bauer A, Kiani BT, Lloyd S (2023) Efficient classical algorithms for simulating symmetric quantum systems. <https://doi.org/10.48550/arXiv.2211.16998>. *arXiv:2211.16998* [quant-ph]
- Azevedo V, Silva C, Dutra I (2022) Quantum transfer learning for breast cancer detection. *Quantum Mach Intell* 4(1):5. <https://doi.org/10.1007/s42484-022-00062-4>
- Barenco A, Berthiaume A, Deutsch D, Ekert A, Jozsa R, Macchiavello C (1996) Stabilisation of quantum computations by symmetrisation. *arXiv*. <https://doi.org/10.48550/arxiv.quant-ph/9604028>
- Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N, Lloyd S (2017) Quantum machine learning. *Nature* 549(7671):195–202. <https://doi.org/10.1038/nature23474>
- Bowles J, Wright VJ, Farkas M, Killoran N, Schuld M (2023) Contextuality and inductive bias in quantum machine learning. *arXiv*. [arXiv:2302.01365](https://arxiv.org/abs/2302.01365)
- Buzek V, Hillery M (2000) Optimal manipulations with qubits: universal quantum entanglers. *Phys Rev A* 62(2):022303. <https://doi.org/10.1103/PhysRevA.62.022303>
- Cerezo M, Larocca M, García-Martín D, Diaz NL, Braccia P, Fontana E, Rudolph MS, Bermejo P, Ijaz A, Thanasilp S, Anschuetz ER, Holmes Z (2023) Does provable absence of barren plateaus imply classical simulability? Or, why we need to rethink variational quantum computing. *arXiv:2312.09121*
- Chen S, Liu B, Feng C, Vallespi-Gonzalez C, Wellington C (2021) 3d point cloud processing and learning for autonomous driving: impacting map creation, localization, and perception. *IEEE Signal Process Mag* 38(1):68–86. <https://doi.org/10.1109/MSP.2020.2984780>
- Creevey FM, Hill CD, Hollenberg LCL (2023) GASP – a genetic algorithm for state preparation. <https://doi.org/10.48550/arxiv.2302.11141>
- Fontana E, Herman D, Chakrabarti S, Kumar N, Yalovetzky R, Heredge J, Sureshbabu SH, Pistoia M (2023) The adjoint is all you need: characterizing barren plateaus in quantum Ansatzes. *arXiv:2309.07902*
- Havlíček V, Córcoles AD, Temme K, Harrow AW, Kandala A, Chow JM, Gambetta JM (2019) Supervised learning with quantum-enhanced feature spaces. *Nature* 567(7747):209–212. <https://doi.org/10.1038/s41586-019-0980-2>
- Heredge J, Hill C, Hollenberg L, Seviar M (2021) Quantum support vector machines for continuum suppression in B meson decays. *Comput Softw Big Sci* 5. <https://doi.org/10.1007/s41781-021-00075-x>
- Huang H-Y, Broughton M, Mohseni M, Babbush R, Boixo S, Neven H, McClean JR (2021) Power of data in quantum machine learning. *Nat Commun* 12(1):2631. <https://doi.org/10.1038/s41467-021-22539-9>
- Kazi S, Larocca M, Cerezo M (2023) On the universality of  $S_n$ -equivariant  $k$ -body gates. *arXiv:2303.00728*
- Kübler JM, Buchholz S, Schölkopf B (2021) The inductive bias of quantum kernels. <https://doi.org/10.48550/arxiv.2106.03747>
- Lisnichenko M, Protasov S (2022) Quantum image representation: a review. *Quantum Mach Intell* 5(1):2. <https://doi.org/10.1007/s42484-022-00089-7>
- Liu Y, Arunachalam S, Temme K (2021) A rigorous and robust quantum speed-up in supervised machine learning. *Nat Phys* 17(9):1013–1017. <https://doi.org/10.1038/s41567-021-01287-z>
- Meichanetzidis K, Toumi A, de Felice G, Coecke B (2023) Grammar-aware sentence classification on quantum computers. *Quantum Mach Intell* 5(1):10. <https://doi.org/10.1007/s42484-023-00097-1>
- Meyer JJ, Mularski M, Gil-Fuster E, Mele AA, Arzani F, Wilms A, Eisert J (2023) Exploiting symmetry in variational quantum machine learning. *PRX Quantum* 4:010328. <https://doi.org/10.1103/PRXQuantum.4.010328>
- Mikuni V, Canelli F (2021) Point cloud transformers applied to collider physics. *Mach Learn Sci Technol* 2(3):035027. <https://doi.org/10.1088/2632-2153/ac07f6>
- Nguyen QT, Schatzki L, Braccia P, Ragone M, Coles PJ, Sauvage F, Larocca M, Cerezo M (2022) Theory for equivariant quantum neural networks. *arXiv*. <https://doi.org/10.48550/arxiv.2210.08566>
- Nichol A, Jun H, Dhariwal P, Mishkin P, Chen M (2022) Point-E: a system for generating 3D point clouds from complex prompts. *arXiv*. <https://doi.org/10.48550/arxiv.2212.08751>
- Park C-Y (2021) Efficient ground state preparation in variational quantum eigensolver with symmetry breaking layers. *arXiv*. <https://doi.org/10.48550/arxiv.2106.02509>
- Pregolato M, Zizzi P (2023) SARS-CoV-2 spike and ACE2 entanglement-like binding. *Quantum Mach Intell* 5(1):8. <https://doi.org/10.1007/s42484-023-00098-0>
- Qi CR, Su H, Mo K, Guibas LJ (2016) PointNet: deep learning on point sets for 3D classification and segmentation. *arXiv*. <https://doi.org/10.48550/arxiv.1612.00593>
- Ragone M, Bakalov BN, Sauvage F, Kemper AF, Marrero CO, Larocca M, Cerezo M (2023) A unified theory of barren plateaus for deep parametrized quantum circuits. *arXiv:2309.09342*
- Sajjan M, Li J, Selvarajan R, Sureshbabu SH, Kale SS, Gupta R, Singh V, Kais S (2022) Quantum machine learning for chemistry and physics. *Chem Soc Rev* 51(15):6475–6573. <https://doi.org/10.1039/D2CS00203E>
- Schatzki L, Larocca M, Nguyen QT, Sauvage F, Cerezo M (2022) Theoretical guarantees for permutation-equivariant quantum neural networks. *arXiv:2210.09974*
- Schuld M, Sweke R, Meyer JJ (2021) Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Phys Rev A* 103:032430. <https://doi.org/10.1103/PhysRevA.103.032430>
- Shaydulin R, Wild SM (2022) Importance of kernel bandwidth in quantum machine learning. *Phys Rev A* 106:042407. <https://doi.org/10.1103/PhysRevA.106.042407>
- Shi R, Tang H, Jin X-M (2020) Training a quantum PointNet with Nesterov accelerated gradient estimation by projection. Presented at the first workshop on quantum tensor networks in machine learning, 34th Conference on neural information processing systems (NeurIPS 2020). [https://tensorworkshop.github.io/NeurIPS2020/accepted\\_papers/simptnet\\_cameraready\\_v2.pdf](https://tensorworkshop.github.io/NeurIPS2020/accepted_papers/simptnet_cameraready_v2.pdf)
- Tüysüced C, Rieger C, Novotny K, Demirköz B, Dobos D, Potamianos K, Vallecorsa S, Vlimant J-R, Forster R (2021) Hybrid quantum classical graph neural networks for particle track reconstruction.

- Quantum Mach Intell 3(2):29. <https://doi.org/10.1007/s42484-021-00055-9>
- West MT, Heredge J, Seviar M, Usman M (2024) Provably trainable rotationally equivariant quantum machine learning. [arXiv:2311.05873](https://arxiv.org/abs/2311.05873)
- West M, Seviar M, Usman M (2022) Reflection equivariant quantum neural networks for enhanced image classification. <https://doi.org/10.48550/arxiv.2212.00264>
- Yuan X-J, Chen Z-Q, Liu Y-D, Xie Z, Jin X-M, Liu Y-Z, Wen X, Tang H (2022) Quantum support vector machines for aerodynamic classification. [arXiv:2208.07138](https://arxiv.org/abs/2208.07138)
- Zeguendry A, Jarir Z, Quafafou M (2023) Quantum machine learning: a review and case studies. Entropy 25(2):287. <https://doi.org/10.3390/e25020287>
- Zhang D-J, Tong DM (2023) Inferring physical properties of symmetric states from the fewest copies. [arXiv:2301.10982](https://arxiv.org/abs/2301.10982)
- Zoufal C, Lucchi A, Woerner S (2019) Quantum generative adversarial networks for learning and loading random distributions. NPJ Quantum Inf 5(1):1–9. <https://doi.org/10.1038/s41534-019-0223-2>

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.