# Quantum One-Class Classification

Group 3 - Remco Spelthan, Naveen Paul, Wenxiong Wu, Ioannis Grepsios, Alexander Prochnow

***Abstract*—In this paper, we propose and evaluate three different quantum one-class classification (QOCC) algorithms for anomaly detection and outlier detection. The first algorithm is an One-Class Support Vector Machine (OC-SVM) with enhanced feature spaces using a ZZFeatureMap, the second algorithm is a Variational quantum one-class classifier, and the third algorithm is a normal Variational quantum circuit. We evaluate the performance of the algorithms using the evaluation metric Macro F1-score, since this gives us the best balance between correctly identifying outliers and inliers. We also compare the results with the classical one-class classification methods to highlight the advantages of the proposed quantum one-class classification methods. Our results show that the proposed QOCC algorithms achieve a slight improvement in performance over the classical methods in specific datasets, but generally seems to perform as well or worse than classical methods. We also discuss the current challenges and open problems and identify potential directions for future research.**

CONTENTS



## I. INTRODUCTION

One-class classification is a machine learning problem that aims to identify novel or abnormal data points in a given dataset. This technique has a wide range of applications, such as fraud detection, medical diagnoses and data cleaning. In recent years, quantum computing has emerged as a powerful tool for solving complex optimization problems. As a result, quantum one-class classification (QOCC) methods have been proposed, which leverage the unique properties of quantum systems to improve the performance of one-class classification. In this paper, we provide an overview of quantum one-class classification methods, and compare them to classical methods. We then discuss the advantages and limitations of these models, and identify areas in where QOCC may offer improvements over classical one-class classification methods. This leads to the following research questions this paper attempts to answer:

- How do quantum one-class classification methods compare to classical one-class classification methods in terms of performance?
- How do quantum one-class classification methods compare to classical one-class classification methods in terms of computational complexity?
- What are the advantages of using quantum computing in one-class classification, and in which specific applications can these advantages be observed?

Additionally, we discuss the current challenges for our project and potential future problems as well as identify potential directions for future research. We published our code and results on GitHub at https://github.com/welremco/Group3QuantumOneClassSVMs

## II. STATE OF THE ART

The field of quantum one-class classification (QOCC) is still in its early stages of development, but it has already

shown promise as a powerful tool for anomaly detection and outlier detection. A number of recent papers have proposed and evaluated different QOCC methods.

One such method is the Hybrid classical-quantum autoencoder for anomaly detection (Alona Sakhnenko et al., 2020) [1]. This paper combines classical autoencoder with a parametrized quantum circuit that is inserted into its bottleneck, to improve the performance of anomaly detection. The method was tested on various datasets and showed improved performance compared to classical autoencoder.

Another recent paper, "Anomaly detection in high-energy physics using a quantum autoencoder" (Alona Sakhnenko et al., 2022) [2], proposed the use of quantum autoencoder based on variational quantum circuits for anomaly detection in high-energy physics data. The method was tested on a high-energy physics dataset and outperforms its classical counterpart for the same inputs. The results were also reproducible on real quantum devices.

A Variational quantum one-class classifier (Park et al., 2023) [3] was proposed to improve the performance of one-class classification by training a fully-parameterized quantum autoencoder with a normal dataset that does not require decoding. This method is then tested on image datasets. The method is comparable to classical OC-SVM and PCA, but the complexity of the quantum method has a slight advantage, since the number of model parameters grows logarithmically with the data size. The quantum method did outperform the deep convolutional autoencoder, in most cases.

In Quantum machine learning for quantum anomaly detection (Liu et al., 2018) [4], the authors proposed a quantum version of support vector machines and kernel principal component analysis. They found corresponding quantum algorithms to detect anomalies in quantum states. They showed that these two quantum algorithms can be performed using logarithmic resources in terms of the dimensionality of quantum states for mixed state, and also logarithmic in the number of quantum states for training the machine learning algorithm for pure state. A pure state is a state vector that corresponds to a single and definite state of the system. It can be represented by a unit vector in a complex Hilbert space, and it is completely determined by its wave function. A pure state is represented by a single point in the Hilbert space. A mixed state represents a quantum system that is in a state of uncertainty or a state of being in multiple states simultaneously. A mixed state is described by a density matrix, which is a positive semidefinite matrix with trace 1 that represents the probabilities of different pure states[4].

In Supervised learning with quantum-enhanced feature spaces (Havlíček et al., 2019) [5], the authors propose a quantum-enhanced feature map to improve the performance of supervised learning tasks, by using a quantum kernel estimator and optimizing the classifier directly. The method was tested on a dataset generated by using the quantum kernel, and shows that, using the quantum kernel estimation function, this method does perform better than classical counterparts on this specific dataset.

In Classification with Quantum Neural Networks on Near Term Processors (Farhi et al., 2018) [6], a quantum neural network is introduced, that can respresent labeled data and be trained by supervised learning. This method relies on the classical simulation of small quantum systems, is designed with near-term quantum processors in mind, and it will be therefore possible to run this quantum neural network on a near term gate model quantum computer.

In Circuit-centric quantum classifiers (Schuld et al., 2018) [7], low-depth variational quantum algorithms for supervised learning are proposed. This is done by encoding input feature vectors onto the amplitudes of a quantum system, and using the parameters of that system to classify the input. It is shown that these quantum classifiers perform well on standard classical benchmark datasets, while requiring fewer parameters that other methods.

These papers demonstrate the potential of QOCC methods for improving the performance of anomaly detection and outlier detection, and they highlight the need for further research in this area in order to fully realize the potential of QOCC methods. In particular, further research is needed to develop QOCC methods that can effectively handle high-dimensional data and big data, as well as to evaluate the scalability of these methods. Additionally, it is important to investigate how QOCC methods can be integrated with other quantum machine learning techniques for improved performance.

## III. Methods

In this paper, we propose and evaluate three different quantum one-class classification (QOCC) algorithms. The first algorithm is an One-Class Support Vector Machine (OC-SVM) with enhanced feature spaces using a ZZFeatureMap. The second algorithm is a Variational quantum one-class classifier, and the third algorithm is a Variational Quantum Circuit.

### *A. OC-SVM With Enhanced Feature Spaces*

The One-Class Support Vector Machine (OC-SVM) is a popular method for one-class classification that separates the data from the origin in the feature space. The idea behind this method is to map the original data into a higher-dimensional feature space where it is easier to separate the normal data points from the abnormal ones. In this method, we propose to enhance the feature space using a ZZFeatureMap, which is a quantum feature map that encodes the data into a higher-dimensional space. The ZZFeatureMap can be represented as a unitary matrix, where each element is parameterized by a set of parameters. These parameters can be optimized using a quantum circuit to achieve the optimal feature space for one-class classification. Once the feature space is enhanced, the OC-SVM algorithm is trained by solving a quadratic programming problem. The objective of the problem is to find the hyperplane that maximizes the distance between the normal data points and the origin. Once the hyperplane is found, it can be used to classify new data points as normal or abnormal. The key advantage of this method is that it can

handle high-dimensional data, which is a common issue in one-class classification problems.

*1) ZZFeatureMap:* To achieve a possible quantum advantage over a conventional support vector machine, the feature vector kernel has to be complex enough that it is hard to classically simulate. To obtain this advantage, we need a map based on circuits that are hard to simulate classically. The paper that introduced the ZZFeatureMap chose to base this on a limited version of the Pauli Expansion circuit. With this method, all qubits are initialized as $|0\rangle$. We use the coefficients generated by our data mapping $\phi_S(\vec{x})$ to encode data $\vec{x}$. The feature map is then defined by the full circuit $\mathfrak{U}_\Phi(\vec{x}) = U_{\Phi(\vec{x})}H^{\otimes n}U_{\Phi(\vec{x})}H^{\otimes n}$, where H is the Hadamard gate. Equation 1 is the diagonal gate in the Pauli Z-basis. Here $U_{\Phi(\vec{x})}$ refers to equation 1. In our experiments, we will also be using Z Feature Map and Pauli feature map. Here we'll quickly explain what the difference is between these methods. A Pauli feature map is a quantum feature map that encodes classical data into the amplitudes of a quantum state using a combination of Pauli matrices. The Pauli matrices are a set of three 2x2 matrices, the identity matrix (I), the Pauli-X matrix (X), and the Pauli-Y matrix (Y). The Pauli feature map can be used to map classical data into a higher-dimensional space, where it is easier to separate the different classes. A Z feature map is a quantum feature map that encodes classical data into the amplitudes of a quantum state using a combination of the Pauli-Z matrix (Z). The Z feature map is a simpler version of the Pauli feature map and can be used for similar tasks as the Pauli feature map. A ZZ feature map is a quantum feature map that encodes classical data into the amplitudes of a quantum state using a combination of both Pauli-Z matrix (Z) and Pauli-X matrix (X).

$$U_{\Phi(\vec{x})} = exp(i \sum_{S\subseteq[n]} \phi_S(\vec{x}) \prod_{i\in S} Z_i) \tag{1}$$

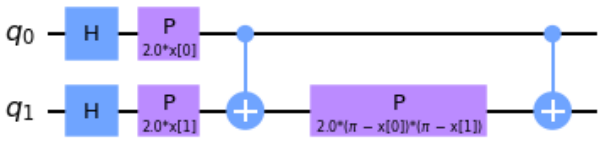



Fig. 1. A ZZFeatureMap circuit with 2 qubits and 1 repetition

*2) Data Mapping:* The data mapping function $\phi_S(\vec{x})$ allows us to experiment with different data mappings. We have three data mappings. The goal of these mappings are to transform the data in such a way that the features can be used more effectively on the ZZFeatureMap circuit.

The first data mapping, seen in equation 2, is the self product data map. This mapping function maps classical data into higher-dimensional space by taking the self-product of the data. If $S$ contains a single index $i$, this means that the value of the input data point at that index is passed through the function unchanged.

The second case in the function maps the input data point $(\pi - x_i)(\pi - x_j)$ if $S = i, j$. This means that if $S$ contains two indices $i$ and $j$, the values of the input data point at those indices are subtracted from $\pi$ and then multiplied together. This creates a new feature that is a combination of the values of the input data point at those two indices.

This mapping function allows the ZZFeatureMap to capture the interactions between different features of the input data, which can lead to a more accurate one-class classifier. However, it is important to mention that this mapping function is specific and it may not be suitable for all datasets or applications, as seen later.

$$\phi_S : \mathbf{x} \mapsto \begin{cases} x_i & \text{if } S = i \\ (\pi - x_i)(\pi - x_j) & \text{if } S = i, j \end{cases} \tag{2}$$

In this data mapping, equation 3, we remove the constant $\pi$ from the data mapping function, and change the negative sign to a plus sign. This changes the data mapping function behaviour. The first case of the function will stay the same. This means that the data mapping function will no longer subtract values from the input data point from $\pi$, but instead will make a feature by multiplying these 2 values. This changes the scale of the resulting features, which can affect the classification boundary and threshold, resulting in different classifications.

$$\phi_S : \mathbf{x} \mapsto \begin{cases} x_i & \text{if } S = i \\ (x_i)(x_j) & \text{if } S = i, j \end{cases} \tag{3}$$

The following equation was mentioned in the "Quantum machine learning for quantum anomaly detection" (Liu et al., 2018)[4] paper. The goal of this mapping is to have a valid positive kernel matrix resulting from this mapping. This mapping is otherwise known as superfidelity. In this mapping equation, equation 4, we see that the mapping operations are different. We also see that it is much more complex than previously mentioned data mappings. The reason this mapping is more complex, is because it normalizes the values of $x_i$ and $x_j$ when $S$ contains two indices $i$ and $j$.

$$\phi_S : \mathbf{x} \mapsto \begin{cases} x_i & \text{if } S = i \\ (\frac{x_i}{\sqrt{1-tr(x_i)^2}})(\frac{x_j}{\sqrt{1-tr(x_j)^2}}) & \text{if } S = i, j \end{cases} \tag{4}$$

### *B. Variational Quantum One-Class Classifier*

The Variational quantum one-class classifier (VQOCC) is a quantum machine learning algorithm that uses a quantum circuit to learn a decision boundary separating the normal data points from the abnormal data points. The parameterized quantum circuit is trained by minimizing the cost function defined by taking the Hamming distance between the trash qubit system and the target state. The target state here is just $|0\rangle$. The key advantage of this method is that it can leverage the unique properties of quantum systems to learn complex decision boundaries that are not possible with classical methods. The algorithm is based on Variational Quantum Circuit (VQC) which is a family of quantum algorithms that use a

parametric quantum circuit to approximate a target function or probability distribution. The parameters of the circuit are optimized to minimize a given cost function.

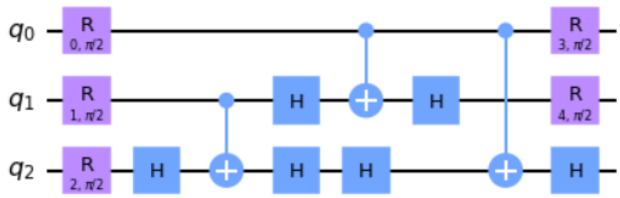



Fig. 2. A Variational Quantum One-Class Classifier with 2 latent qubits and 1 trash qubit

*1) Encodings:* For this method, we used 2 encodings, namely Amplitude encoding and Flexible Representation of Quantum Images (FRQI) encoding.

*a) Amplitude encoding:* Amplitude encoding loads classical data into the amplitude of a quantum state. To encode a classical data point into the quantum state, we first need to normalize the datapoints, such that the normalised vector $\sum_i |x_i|^2 = 1$. This vector can then be represented by the amplitudes of a quantum state as noted in equation 5

$$|\psi_x\rangle = \sum_{i=0}^{2^n-1} x_i |i\rangle \tag{5}$$

*b) Flexible Representation of Quantum Images (FRQI):* This encoding is designed for encoding image data into quantum state, but any 2 dimensional input in part of the image can be done. This method maps each value to the amplitude, but additionally gets the corresponding positions in a datapoint and maps that into the quantum state[8]. We see the formula used to do this mapping in equation 6.

$$|\psi_x\rangle = \frac{1}{2^n} \sum_{i=0}^{2^n-1} (sin(\theta_i)|0\rangle + cos(\theta_i)|1\rangle)|i\rangle \tag{6}$$

*2) Cost function:* Our quantum circuit is trained by minimizing a cost function, using the Hamming distance between the trash qubit and the target state $|0\rangle^{\otimes n_t}$. If the trash qubit state is 0, this means that there is only normal data inputted into the circuit. Only normal data can generate the state $|0\rangle^{\otimes n_t}$, and therefore when this state is further away from 0, we can call this an anomaly. We use the local cost function based on the Hamming distance as follows:

$$C = \frac{1}{2} \sum_{i=1}^{n_t} (1 - \langle Z_i \rangle) \tag{7}$$

In this equation, $\langle Z_i \rangle$ is an expectation value of the after applying the Pauli-Z operator on the $i$th trash qubit.

### C. Variational Quantum Circuit

The Variational quantum circuit is a quantum circuit which uses a set of free parameters that influence the output, and a measurement with a certain output to which the parameters can be updated accordingly. The circuit is trained by minimizing a certain cost function, which can measure a difference between predicted and true output labels, or can use a internal distance metric. In this method, we use a normal quantum circuit which means that it can be efficiently implemented on current and near-term quantum devices. The circuit is trained using Nesterov Momentum optimization algorithm to minimize the square loss function. The key advantage of this method is that it can be implemented on current quantum devices, which makes it more accessible for real-world applications.


Fig. 3. A Variational Quantum Circuit with parameters and 2 qubits

### D. Classical OC-SVM

We provide the classical One-Class Support Vector Machine[9] method as a baseline to compare our other Quantum One-Class methods against. In this method it is possible to use the kernel trick to translate the data into different higher dimensions, making it in some cases easier to draw a more correct decision boundary.

## IV. Data and Experiments

In order to assess the performance of our developed methods and to compare them to classical equivalents, we use two types of data: Artificially generated data (section IV-A) and data from a real anomaly detection application, namely the identification of ghost tracks in the CERN particle accelerator (section IV-B).

### A. Artificial data

We generated artificial datasets for two reasons. The first is to get familiar with the translation to a quantum domain on a dataset, which is easier if we know exactly what the properties and labels of the data are. The second reason is that the generated data can be simpler, meaning that the algorithms will also be easier to run and check on a classical computer.

*1) Datasets overview:* The four data sets we generated are:

- A dataset with three almost linearly separable blobs, each generated from a different Gaussian distribution. The largest one of the blobs is the inlier cluster (190 data points), and the two smaller ones are the outlier clusters (9 and 11 data points, respectively). See Dataset 1 in Figure 14.
- The Ad Hoc dataset (see Dataset 2 in Figure 15). This dataset stems from the paper that introduced the ZZ Feature Map [10] and was generated using the ZZ Feature Map, therefore being easily solvable by it, while at the same time being difficult for classical feature maps to solve adequately. We used the built-in method in Qiskit, the Python library of IBM Quantum, to generate Ad Hoc

data with 160 inliers (both inlier class A and B seen in Figure 15 were condensed to one class) and 20 outliers.
- The third dataset contains one Gaussian blob with 190 data points as the inliers and 20 Ad Hoc data points as outliers (all Ad Hoc generated data points were labeled outliers). See Dataset 3 in Figure 16.
- Lastly, we created a dataset with 80 Ad Hoc-generated inliers and 20 outliers coming from three Gaussian blobs. See Dataset 4 in Figure 17.

For more information on the data generation process, please visit our GitHub page.

*2) Experimental setup:* The experiments we carried out on the artificial data served two main purposes: To learn about how our One-Class SVM, VQOCC and VQC perform on different types of data and to learn about the effects on performance that different parameters of our methods have.

First, for the experiments on our One-Class SVM: We varied one parameter of our One-Class SVM, while leaving the other parameter settings constant, then trained and tested each model on all four artificial datasets. Explanations of the parameters were covered in section III-A). Here, we used all data, unlabeled, as training data, then tested using all data and compared against the labels. The five experiments were as follows:

- Experiment 1 varies the number of circuit repetitions, which means it varies how many times the exact same circuit is repeated. The more repeats, the harder it becomes to simulate classically.
- Experiment 2 varies the feature map between the ZZ feature map, Z feature map and Pauli feature map.
- Experiment 3 varies the data map, between a self product data map, a superfidelity data map and our custom data map.
- Experiment 4 tests four different classical SVM kernels: The RBF (radial basis function) kernel, the linear kernel, the poly kernel and the sigmoid kernel. These are the four kernels available by default in Scikit-Learn, which we used to run this experiment.
- Experiment 5 combines the data from previous experiments and compares the best performing classical and quantum method settings for each dataset.

In each experiment, the other parameters for our One-Class SVMs were left at their defaults, which were chosen to be two circuit repetitions, the ZZ feature map and a self product data map.

Next, the experimental setup for our VQOCC, which contains two parameters: The data encoding, either FRQI (Flexible Representation of Quantum Images) or amplitude encoding, see section III-B1, and the number of layers, namely 2, 4, 6 or 8 layers. The two experiments we performed therefore test all setting combinations on all four datasets:

- Experiment 6 is performed using the FRQI encoding and varying the number of layers.
- Experiment 7 is performed analogously using the amplitude encoding.

We measure both test performance and runtime.

Lastly, we tested our VQC in Experiment 8 on the four datasets using different numbers of layers ranging from 1 to 6.

### *B. CERN ghost track data*

The final dataset we will use to assess the performance of our developed methods is a dataset from a real application of anomaly detection, and moreover a problem that is know to be hard.

*1) Dataset overview:* In order to describe the dataset, we will briefly cover the necessary background on ghost track detection. The experimental setup that generated the data is as follows: The particle accelerators at the European Organization for Nuclear Research (CERN) propel a particle, such as a proton, at high speeds towards a target, often another particle. The point at which they collide, known as the primary vertex, is the point at which we start measuring. At this point, the energy of the collision is transformed into matter and creates many new particles. These particles fly away from the collision point at different angles, usually in an almost straight trajectory. Their flight paths are known as tracks. There are four points (called vertices) at which we measure for particles mid-flight, and the measurements collected at those points are used to reconstruct the track of the particle. The results of these reconstructions are stored in the dataset that is available to us, however the reconstruction also contains errors. This happens when the reconstruction procedure predicts a certain particle track, even though there was no physical particle that flew along that track. These reconstruction errors are called ghosts tracks.

Intuitively, ghost tracks can be identified by checking the sum of the energy/speed and direction of all the measured particles/particle tracks that were created from the collision, and see if those are expected given the original energy/speed and direction at which the collision happened. If the difference between the reconstructed and expected values is too large, we are likely dealing with a reconstruction error, a ghost track. The ghost track classification dataset contains features that measure exactly these differences between reconstruction and expectation, in the form of Chi-squared ($\chi^2$) statistics.

The features we use for the ghost track classification task are as follows:

- `chi2/dof`: The normalized $\chi^2$ statistic of the reconstruction, which represents the goodness of fit of the reconstruction. For normalization, the $\chi^2$ statistic is divided by the number of degrees of freedom, which are roughly related to the heat each particle track leaves behind.
- `kalman_ip_chi2`: For reconstruction, measurements are combined over time using a Kalman filter to predict an Impact Parameter (IP) (which is the transverse distance of the closest approach between a particle trajectory and a vertex). Our dataset feature then represents the $\chi^2$ statistic used to compare this prediction with the observation. The better the $\chi^2$, the more likely a track is to be real, however

this statistic is only meaningful if the `chi2/dof` fit is good.

- `nb_hits`: The number of hits in the detector. If the number of hits is low, the chance of a ghost track is higher.

The other features contained in the dataset were not used for the classification task in order to reduce the runtime of our methods and excluded since they carry less meaning for the identification of ghost tracks. Details about these can be found in the Appendix VIII-B.

*2) Experimental setup:* We conducted our experiments on the dataset as follows: We define the ghost tracks as outliers, and the regular tracks as inliers. The dataset contains around $3.5\%$ ghost tracks, and the sample of 1000 reconstructed tracks we took also roughly contains this fraction of outliers. We choose the number of samples to be 1000 in order to limit the runtime. For these experiments, we used our defined default settings for our quantum OC-SVM (see section IV-A2). For our VQOCC we performed the experiments as with the artificial data, i.e. we experimented with four different numbers of layers, namely 2, 4, 6 or 8 layers as well as two different data encodings, namely FRQI (Flexible Representation of Quantum Images) and amplitude encoding. Additionally, as a preprocessing step, we standardize the data, i.e. subtracting the mean and scaling every feature to unit variance.

## V. Results

We assessed our developed quantum methods under two aspects and compared these to classical equivalents: First, we report how well our methods were able to solve the anomaly detection task, and second, we report theoretical time and space complexities, i.e. memory and runtime requirements.

### A. Performance

The metric we chose to assess the performance of our model is the macro F1 score, which is the average of both the individual F1 scores, i.e. the F1 score with inliers as positive class and the F1 score with outliers as positive class. This gives an accurate representation of the performance of our model, because both the averaging of the F1 scores and the F1 score itself account for class imbalance, which is usually present in anomaly detection.

*1) Experiments on artificial datasets:*

*a) Experiment 1:* In figure 4, we see the effect of circuit repetitions in our Feature Map on the final F1 score our algorithm produces. In this experiment, we see that for Dataset 1, the best performing number of circuit repetitions is 2, but 1 repetition is not far behind. In Dataset 2, we actually see an increase of performance from 1 to 3 repetitions. In dataset 3, the optimal number of repetitions again seems to be 2. In dataset 4, we also see a slow increase in performance from 1 to 3 repetitions, with 2 repetitions only having slightly lower performance than 1 repetition.

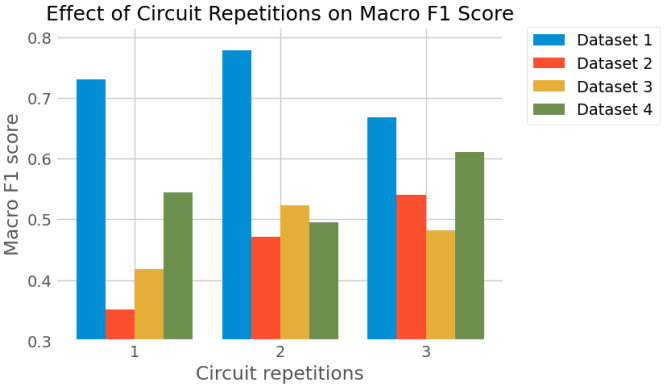



Fig. 4. Results of Experiment 1: Number of Circuit Repetitions.

*b) Experiment 2:* In figure 5, we see the effect of using the ZZFeatureMap, the ZFeatureMap and the Pauli Feature map. In this experiment, we see that for Dataset 1 and 3, the performance is both the highest for the ZZ Feature Map and the Pauli Feature Map. We see that for Dataset 2, all the methods have about the same Macro F1 score. For Dataset 4, the Z Feature Map scores the highest.

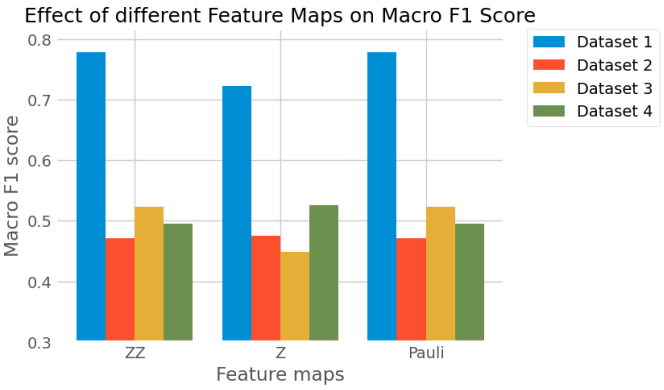



Fig. 5. Results of Experiment 2: Feature maps.

*c) Experiment 3:* In figure 6, we see the effect of using different Data Maps in our One Class SVM with enhanced feature spaces. We see that for Dataset 1, 2 and 3, the self product data mapping performs the best. For dataset 4, we see that the superfidelity data mapping performs the best.

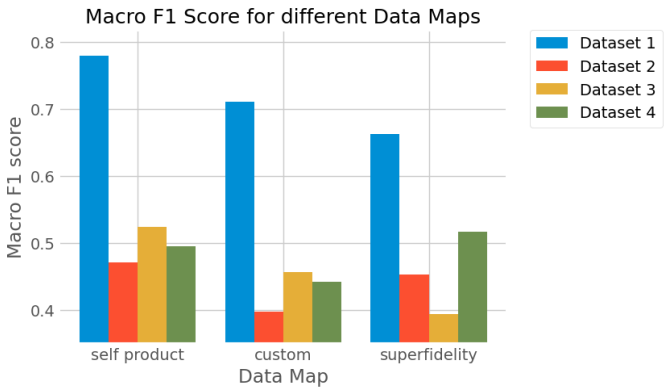



Fig. 6. Results of Experiment 3: Data maps.

*d) Experiment 4:* In figures 7 and 8, we see the effect of using different kernels in our classical OC-SVM. We see especially in figure 8 that for Dataset 1, 2 and 3, the linear and poly method perform the best. On Dataset 4, we actually see that the RBF kernel performs slightly better.

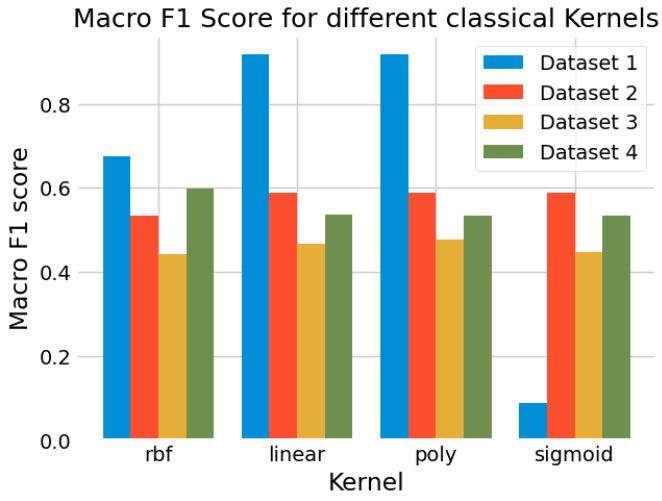



Fig. 7. Results of Experiment 4: Results of classical kernels.

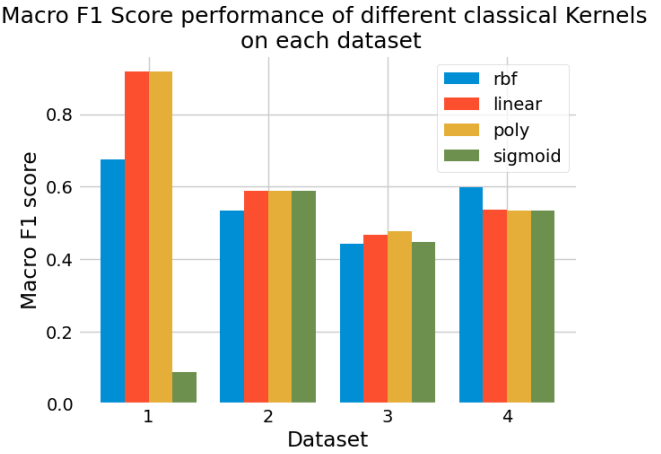



Fig. 8. Comparison of classical kernel performance on each dataset. This contains the same data as Figure 7, i.e. results from Experiment 4, visualized for better comparison of kernels on the same dataset.

*e) Experiment 5:* In figure 9, we see the effect of different numbers of layers when using FRQI encoding in the VQOCC. We see that for every dataset, less layers gives us a better performance.

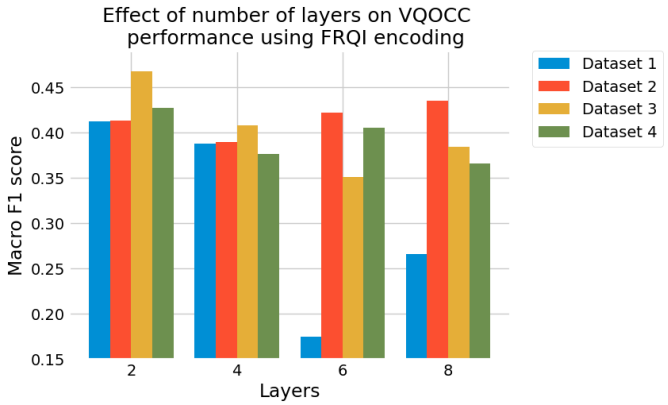



Fig. 9. Results of Experiment 5: VQOCC with FRQI encoding on artificial datasets.

*f) Experiment 6:* In figure 10, we see the effect of different numbers of layers when using Amplitude encoding in the VQOCC. We see generally, for every dataset 2 layers seems to be the best. There is one exception, for dataset 1, 6 layers performs slightly better than its 2 layer variant.

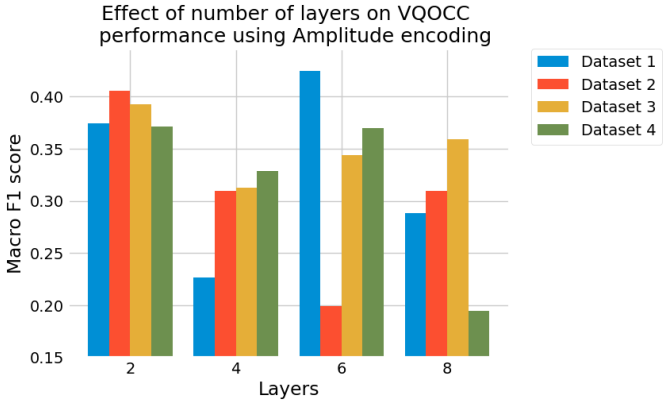



Fig. 10. Results of Experiment 6: VQOCC with Amplitude encoding on artificial datasets.

*g) Experiment 7:* When testing our VQC on the artificial datasets, we noticed that varying the number of layers between 1 and 6 had no impact on the resulting F1 score. Our VQC achieved 0.4700 macro F1 on datasets 1 and 3, and 0.4767 macro F1 on datasets 2 and 4.

*h) Experiment 8:* In figure 11, we see the comparisons of the best performing quantum settings identified in the previous experiments versus all classical OC-SVM kernels. We see that for all datasets, our OC-SVM with Enhanced Feature Spaces performs the best out of the quantum methods. For the first dataset, the classical linear and poly kernels are outperforming all our quantum methods, while the sigmoid kernel has the worst performance on this dataset. For dataset 2, all classical kernels outperform all quantum methods, with the linear, poly and sigmoid kernel having a very similar performance. For dataset 3, our quantum OC-SVM was able to outperform the best classical kernel by around 7%. For dataset 4, our quantum OC-SVM outperforms the best classical kernel, the RBF kernel, by around 1%.

In general we can notice that the performance on datasets 3 and 4, is overall lower than on datasets 1 and 2, which are less complex than datasets 3 and 4.

The best settings identified in the previous experiments for each method on each dataset are as follows:

Dataset 1:

- Classical: Linear kernel
- Quantum OC-SVM: Defaults
- VQOCC: Amplitude encoding, 6 layers

Dataset 2:

- Classical: Poly kernel
- Quantum: 3 circuit repetitions, defaults otherwise
- VQOCC: FRQI encoding, 8 layers

Dataset 3:

- Classical: Sigmoid kernel
- Quantum: Defaults
- VQOCC: FRQI encoding, 2 layers

Dataset 4:

- Classical: RBF kernel
- Quantum: 3 circuit repetitions, defaults otherwise
- VQOCC: FRQI encoding, 2 layers

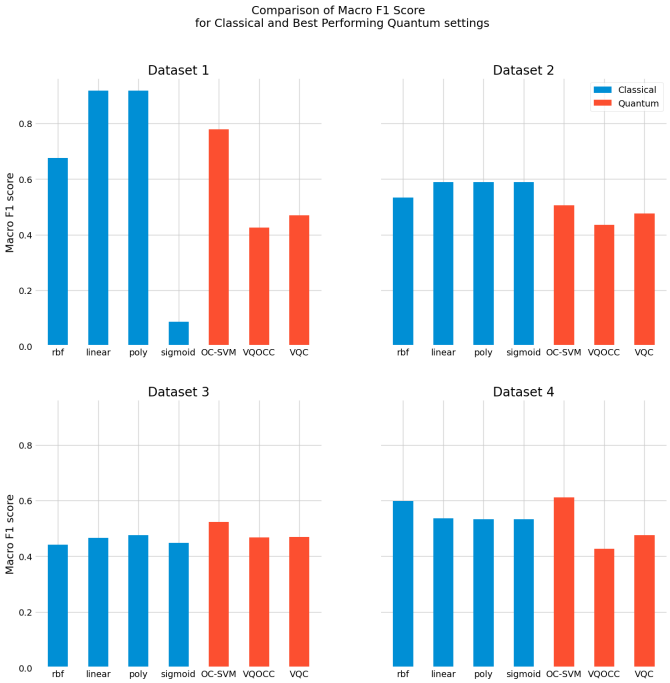



Fig. 11. Results of Experiment 8: Comparing the best performing classical and quantum method for each dataset.

*2) Experiments on CERN ghost track dataset:* For figure 12, we see most classical OC-SVM kernels perform about the same, with the exception of the poly kernel, which performs much worse on this data. Importantly, our OC-SVM With Enhanced Feature Spaces performs as well as the best performing classical OC-SVM methods, while our VQOCC performs slightly worst.

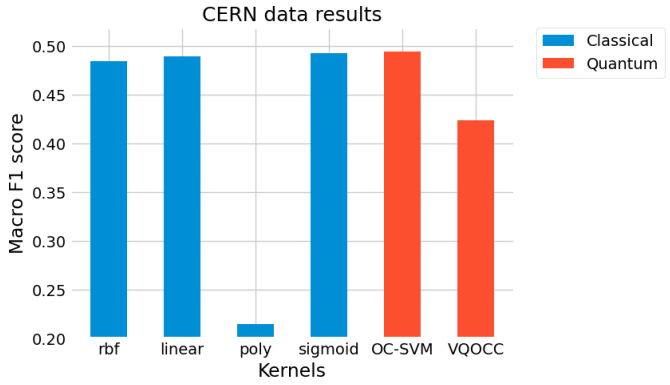



Fig. 12. CERN data results. Trained with 1000 samples, tested on 100 samples

In figure 13, we experiment with the number of layers for the VQOCC and compare between using the FRQI encoding and the Amplitude encoding. Here we see that the best performing method is the FRQI encoding with 2 layers. We also notice that for 4 and 6 layers, the Amplitude encoding

method is performing better than its FRQI counterpart. On 8 layers, the FRQI encoding method is performing better again.

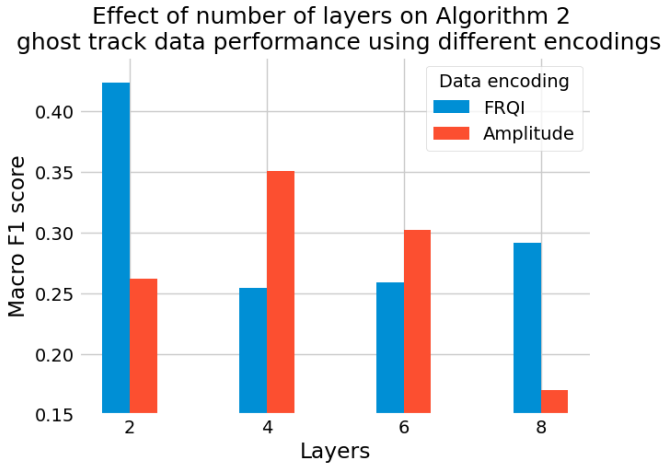



Fig. 13. Results of different encodings on VQOCC performance on the CERN dataset.

### B. Theoretical time and space complexity

Additionally, we compared the theoretical time and space complexities of our methods to their classical counterparts. To calculate time complexity for our Quantum methods, we calculated how many operations were needed on the quantum circuit depending on the hyperparameters. After this, we divided by the number of qubits needed so we could get an average on how many gates a qubit uses. This results in a complicated time complexity calculation, which we then simplify by only keeping the parts that have the biggest influence on this time complexity. For the quantum space complexity, which is the number of qubits needed, we were able to retrieve it from the papers that originally implemented variants of these methods.

*1) OC-SVM With Enhanced Feature Spaces:* Here, given a $d$-dimensional dataset, the number of qubits required for the model scales linearly with the dimension of the data, so $O(d)$. Given the number of qubits $n$ and the number of repetitions $r$ we want to know what the time complexity is of performing the ZZFeatureMap. This complexity is about $O(\frac{r(n^2+\frac{n}{4}^2+2n)}{n})$, which we can simplify to $O(rn)$

*2) Variational Quantum One-Class Classifier:* Given a $d$-dimensional dataset, the number of qubits and model parameters required in VQOCC is $O(log(d))$. Given this $d$-dimensional dataset, and given that the number of layers in the VQOCC is $l$, that the number of trash qubits is denoted as $t$ and the number of qubits is $n$. We calculate the time complexity by counting the number of gates and operations done on the qubits. We then calculate the complexity of how the gates and operations were put on that circuit. We denote this as the time complexity. Our time complexity than becomes: $O(\frac{t(n^2+t^2+-2n+t+t(log(\frac{n-t}{t})))}{n+t})$ if the number of trash qubits is less than or equal to the number of qubits divided by 2, and $O(\frac{lt(n^2+t^2+-2n+t+(n-t)(log(\frac{n}{n-t})))}{n+t})$ otherwise.

Note here that $t < n$ , and we can simplify to: $O(\frac{lt(n^2+t^2)}{n+t})$ for both respectively.

If we assume that $t = n-1$ and $l = 1$, then the complexity becomes about $O(n)$.

*3) Variational Quantum Circuit:* The number of qubits needed for this method scales linearly with the size of the data, so $O(d)$. For this circuit, on every qubit a arbitrary rotation is performed, and a CNOT gate is applied that entangle the qubit with its neighbour. This results in a time complexity of $O(2n)$, which can be simplified to $O(n)$.

*4) Classical OC-SVM:* The training complexity for a classical One-Class Support Vector Machine is $O(max(n,d)min(n,d)^2)$ [11]. With a space complexity of $O(dn)$, where d is the dimension and n the number of examples.

### C. Real runtimes - simulation

Training on 1000 samples of the CERN data and testing on 100 sample took roughly 28min for our quantum OC-SVM (25min training, 3min inference), and for our VQOCC 2h 15min with FRQI encoding and 2h with amplitude encoding, including inference/testing runtime, on our local machines simulating qubits. In comparison, running a classical OC-SVM with any kernel took less than a second for both training and testing on the same samples.

## VI. Discussion

In this paper, we proposed and evaluated three different quantum one-class classification (QOCC) algorithms for anomaly detection and outlier detection. The first algorithm is an One-Class Support Vector Machine (OC-SVM) with enhanced feature spaces using a ZZFeatureMap, the second algorithm is a Variational quantum one-class classifier and the third is a Variational quantum circuit. The main goal of this project was to see if there was a performance, computational complexity, or application benefit to using quantum computing in one-class classification.

The results of our first experiment indicate that the increasing of circuit repetitions can potentially increase the expressiveness of the feature space, but we see that in our results this is not necessarily the case. The variations between the performance of experimenting on these datasets indicate that increase or decreasing the number of circuit repetitions has no definite measurable benefit for our one-class classification use case. There is a general benefit to using 2 circuit repetitions instead of 1, with an exception for dataset 4, indicating that at 2 circuit repetitions the expressiveness of the circuit has reached a certain plateau.

The results of our second experiment indicate that the expressiveness of the ZZ Feature Map is generally the best, with the exception for dataset 4. This effect is fairly small however, and there maybe is an indication that different expanded versions of the Pauli Circuit could give a better performance.

The results of our third experiment indicate that the standard self-product data mapping is considered to be the best for the

first, second and third dataset. An Exception is dataset 4 with the superfidelity data mapping.

The results of our fourth experiment indicate that the classical linear and poly kernel are the best performing classical one-class SVM kernels.

The results of our fifth and sixth experiment indicate that on the FRQI and Amplitude encoding, for the VQOCC, that less layers performs generally better on our artificial datasets. The simplicity of using less parameters and less operations is probably causing the algorithm to find the seperations between these simpler datasets easier.

The results of our eight experiment indicate that the quantum advantage of OC-SVM With Enhanced Feature Spaces remains questionable. On the first 2 datasets, we see clearly that the classical methods are performing way better. On the 3th and 4th dataset however, our quantum OC-SVM With Enhanced Feature Spaces is able to perform better than our classical variations.

The results of our 9th experiment, comparing all algorithms on the CERN dataset, indicate that the quantum algorithms can, in the best case, perform as well as the classical methods. Even though the VQOCC performs less then the other classification methods, the quantum OC-SVM With Enhanced Feature Spaces performs as well on this data. This could be due to the added complexity of the data itself, and that therefore the quantum methods were able to perform better than in previous cases. It could also be the case that in this data, the higher number of features (three features versus only two in the artificial datasets) played a role. This resulted in the quantum OC-SVM With Enhanced Feature Spaces using more qubits, increasing the potential for a quantum advantage. We also see that the number of layers is generally good at 2 layers on the CERN datasets as well. The FRQI encoding seems to be performing better than the Amplitude encoding, this is probably due to the fact that more information is encoded using FRQI encoding in contrast to Amplitude encoding. The reason FRQI encodes more information is that FRQI also gets the corresponding positions of the encoded datapoint, and not just the values of the datapoint itself.

The implications for these results regarding our research questions are as follows. For the first research question, which described if there is an advantage in terms of performance for quantum one-class classification, the answer is rather ambiguous. We have some datasets in which the classical methods far outperform the quantum methods, and others were the quantum methods only slightly outperform the classical ones. Since our scope for the classical methods was only limited to OC-SVM with different kernels, we don't have a full picture until there are comparisons with other one-class classification methods. We also saw that for data where we used more features, as seen in the section describing the results for the application of our methods to the CERN ghost track data, our quantum methods tend to perform as well as the classical methods if we have more than 2 features.

The implications for our second research question regarding the quantum advantage in terms of time complexity for our algorithms is as follows. The computational complexity calculation for quantum methods is somewhat ambiguous. For our case, we denoted the space complexity as how many qubits are needed for the algorithm to perform, and the number of operations performed on the qubits is counted as the time complexity. For our space complexities, we can postulate that since the classical OC-SVM needs $O(dn)$ space, and VQOCC needs $O(log(d))$ space, and OC-SVM With Enhanced Feature Spaces needs $O(d)$ space, we see that there is an advantage for our quantum algorithms in terms of space complexity. In terms of our time complexity, we see that our classical method and our Quantum method seem to have about the same complexity. We are still skeptical as to whether these time complexity numbers are indicative of the real time it will take to solve these problems.

The implications for our third research question are a bit more delicate. There does seem to be an advantage for our quantum methods, in space needed, and for data with more features. There seems to be an advantage in space needed, since we need less qubits to load our dataset for a quantum method then we need space to load our dataset in a classical OC-SVM. We also see a potential benefit for data with a large number of features, since these can be more easily dealt with in a $O(n)$ or $O(log(n))$ way on a quantum computer.

The generalizability of the paper is limited by a few things. Since we generated data for a few of our experiments in order to have smaller datasets to test our quantum algorithms with, we created an inherent bias for this generated data. Since this data is not representative of real world data, this poses a limitation for our results. We of course did test our methods on the CERN ghost track data, but these couldn't be experimented on as much as we hoped. Since we weren't able to compare more one-class classification methods, there still is a possibility for some one-class classification algorithm to be better in both time and space complexity, as well as performance. One limitation of this study is that we didn't experiment with more than a handful of qubits at the same time. This is of course a limitation of simulating on classical data, but to really take advantage of quantum methods, entanglement with multiple qubits is usually needed. This also leads to our next limitation, which is that the methods we used usually take advantage of more then 2 features, and that when we used 3 features, our OC-SVM with enhanced feature spaces suddenly had the same performance as the classical methods. Future research should therefore experiment with multiple features, or improving quantum algorithms for use with fewer features. It should also use datasets which are fairly small, but still representative of real data. In future research, more circuits should be taken into account, and compared such that more conclusive evidence can be given against or for these research questions.

## VII. Conclusion

As quantum computing becomes more prevalent, we need to know if Quantum One-Class Classification methods have certain advantages over classical One-Class Classification

methods. In our paper, we found that for our methods there are some small benefits to quantum methods regarding space complexity, usage of features and performance in some cases. But we also saw that cases where classical methods were performing reasonably, the quantum methods did rather poorly in comparison. Future research should focus on improving quantum methods for a smaller amount of features, increasing the performance with more qubits, a comparison should be made against more quantum methods and the data on which it is tested should be representative of real world data.

## REFERENCES


[1] Alona Sakhnenko et al. "Hybrid classical-quantum autoencoder for anomaly detection". In: *Quantum Machine Intelligence* 4.2 (2022), pp. 1–17.

[2] Vishal S Ngairangbam, Michael Spannowsky, and Michihisa Takeuchi. "Anomaly detection in high-energy physics using a quantum autoencoder". In: *Physical Review D* 105.9 (2022), p. 095004.

[3] Gunhee Park, Joonsuk Huh, and Daniel K Park. "Variational quantum one-class classifier". In: *Machine Learning: Science and Technology* 4.1 (2023), p. 015006. DOI: 10.1088/2632-2153/acafd5. URL: https://dx.doi.org/10.1088/2632-2153/acafd5.

[4] Nana Liu and Patrick Rebentrost. "Quantum machine learning for quantum anomaly detection". In: *Physical Review A* 97.4 (2018), p. 042315.

[5] Vojtěch Havlíček et al. "Supervised learning with quantum-enhanced feature spaces". en. In: *Nature* 567.7747 (2019), pp. 209–212. DOI: 10.1038/s41586-019-0980-2. URL: http://dx.doi.org/10.1038/s41586-019-0980-2.

[6] Edward Farhi and Hartmut Neven. "Classification with quantum neural networks on near term processors". In: *arXiv preprint arXiv:1802.06002* (2018).

[7] Maria Schuld et al. "Circuit-centric quantum classifiers". In: *Phys. Rev. A* 101 (3 2020), p. 032308. DOI: 10.1103/PhysRevA.101.032308. URL: https://link.aps.org/doi/10.1103/PhysRevA.101.032308.

[8] Phuc Q. Le, Fangyan Dong, and Kaoru Hirota. "A flexible representation of quantum images for polynomial preparation, image compression, and processing operations". In: *Quantum Information Processing* 10.1 (Apr. 2010), pp. 63–84. DOI: 10.1007/s11128-010-0177-y. URL: https://doi.org/10.1007/s11128-010-0177-y.

[9] Bernhard Schölkopf et al. "Support Vector Method for Novelty Detection". In: *Advances in Neural Information Processing Systems*. Ed. by S. Solla, T. Leen, and K. Müller. Vol. 12. MIT Press, 1999. URL: https://proceedings.neurips.cc/paper/1999/file/8725fb777f25776ffa9076e44fcfd776-Paper.pdf.

[10] Vojtech Havlicek et al. "Supervised learning with quantum-enhanced feature spaces". In: *Nature* 567.7747 (2019), pp. 209–212. DOI: 10.1038/s41586-019-0980-2.

[11] Olivier Chapelle. "Training a Support Vector Machine in the Primal". In: *Neural Computation* 19 (2007), pp. 1155–1178.

## VIII. Appendix

### A. Artificial dataset plots

Fig. 14. Artificial dataset 1

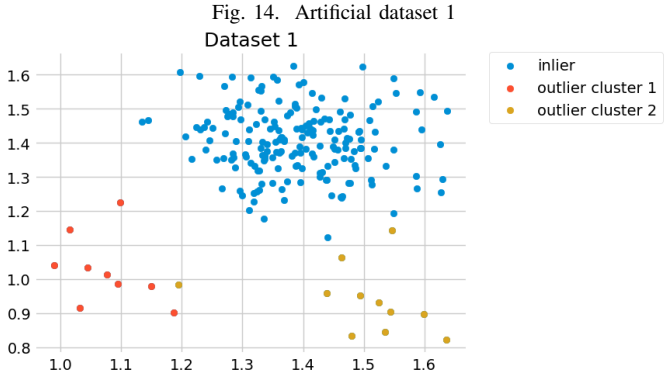



Fig. 15. Artificial dataset 2

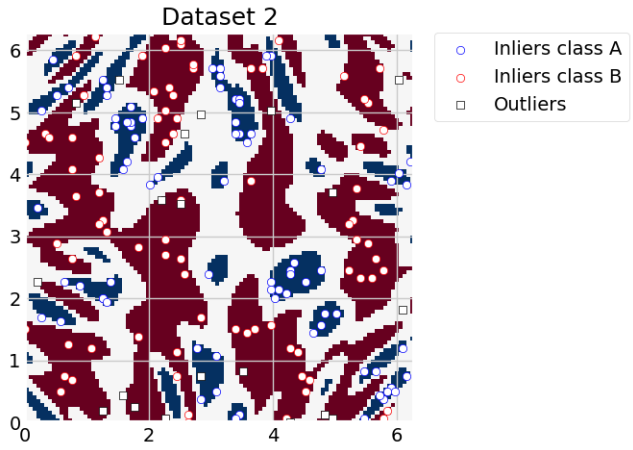



Fig. 16. Artificial dataset 3

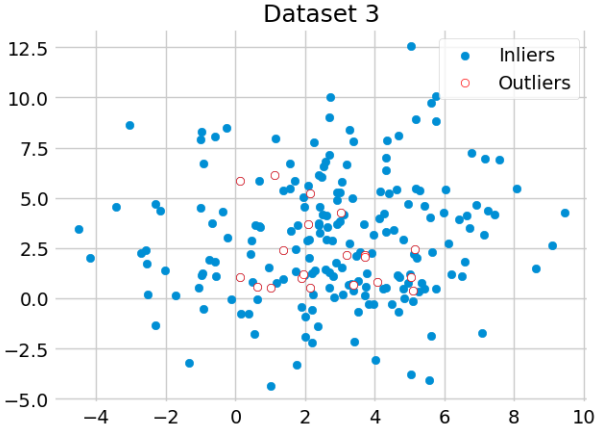
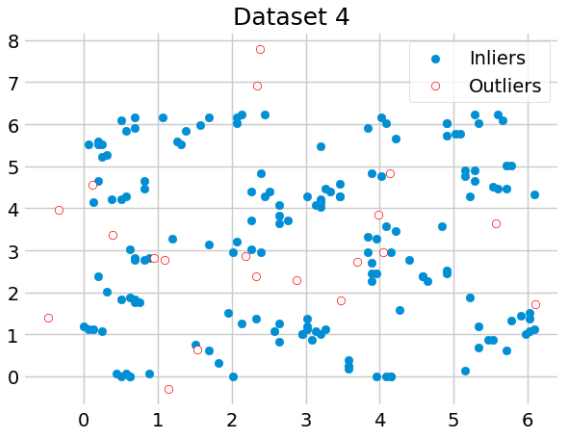



Fig. 17. Artificial dataset 4




### B. CERN ghost track data - excluded features

Here we describe the features that we excluded for the classification task as well as our reasoning for doing so.

- `qop`: Q/P, with Q being the charge of the particle and P its power, i.e. its mass times its speed. This is measured using a large magnet placed after the collision, which causes a curvature in the particle's track when passing through its magnetic field. This feature may aid in the ghost track classification, but was excluded in favor of faster runtimes.
- `x,y,z`: The Cartesian coordinates of a track, measured at inconsistent points along the track. Z is the main axis, Y is the vertical axis and X is the horizontal axis, along which the magnetic field is weak. These features were excluded because the measurements were taken at different positions, and this inconsistency would likely only worsen performance (as well as resulting in longer runtimes).
- `tx, ty`: Slopes of the particle's track w.r.t. the X and Y direction. These features serve as an indication of the bend of the particle's track and may aid in the classification task, but were excluded in favor of faster runtimes. Additionally, since the magnetic field is weak in X direction, we also see `tx` values all being close to 0. Thus, including this feature would likely not impact the performance much, but only increase the runtimes.