Published for SISSA by Springer

Received: *May 3, 2021*
Revised: *June 29, 2021*
Accepted: *August 9, 2021*
Published: *August 31, 2021*

Unsupervised event classification with graphs on classical and photonic quantum computers

Andrew Blance[a,b] **and Michael Spannowsky**[a]

[a] *IPPP, Department of Physics, Durham University, Durham, DH1 3LE, U.K.*
[b] *Institute for Data Science, Durham University, Durham, DH1 3LE, U.K.*

E-mail: andrew.t.blance@durham.ac.uk, michael.spannowsky@durham.ac.uk

Abstract: Photonic Quantum Computers provide several benefits over the discrete qubit-based paradigm of quantum computing. By using the power of continuous-variable computing we build an anomaly detection model to use on searches for New Physics. Our model uses Gaussian Boson Sampling, a #P-hard problem and thus not efficiently accessible to classical devices. This is used to create feature vectors from graph data, a natural format for representing data of high-energy collision events. A simple K-means clustering algorithm is used to provide a baseline method of classification. We then present a novel method of anomaly detection, combining the use of Gaussian Boson Sampling and a quantum extension to K-means known as Q-means. This is found to give equivalent results compared to the classical clustering version while also reducing the $\mathcal{O}$ complexity, with respect to the sample's feature-vector length, from $\mathcal{O}(N)$ to $\mathcal{O}(\log(N))$.

Keywords: Beyond Standard Model, Hadron-Hadron scattering (experiments), Particle correlations and fluctuations

ArXiv ePrint: 2103.03897

https://doi.org/10.1007/JHEP08(2021)170

JHEP08(2021)170

Contents

JHEP08(2021)170

1 Introduction

The search for New Physics depends on our ability to separate Standard Model events from the much rarer and complex signal events from within the data from the LHC. It is desirable to perform data-driven searches, i.e. train on this data directly, without making specific assumptions on the new physics scenario realised by nature. While this allows the search to be widely applicable, it requires the classification method to learn and identify the important features of the background data to discriminate them from rare signal events which do not conform to the same features. Within the realm of classical network methods, autoencoders have been designed for the purpose of anomaly detection and unsupervised event classification [1, 2]. Other, cluster based methods have also been developed [3].

Careful consideration of how the data is represented can improve the performance of searches. Events in high energy physics naturally fit into graph structures. These structures allow us not only to define the features of individual constituents in the event but also how they are related to each other. Graphs have proven to be a powerful representation for LHC data [4–6] and combined with an anomaly detection method could be useful in data-driven searches.

To further boost the performance of an anomaly detection search we propose the use of a quantum device. Quantum computing has shown to be able to solve classically hard problems, such as database searching and factoring [7, 8]. In HEP, quantum computers may also prove useful for solving a range of tasks. Annealers have been used to study field theories [9–11] and optimisation problems [12]. Quantum neural networks to solve classification problems have been built from quantum gates [13]. Calculation of multi-particle interactions [14–23] accomplished with the mapping of field theories onto quantum walks [24–27] or a combination of quantum and classical ideas [28–31] have also all been done using models built from quantum gates.

Previous work focuses on quantum annealers or the discrete, qubit-based paradigm of quantum computing. While these models are very successful, another scheme can be employed. The continuous-variable (CV) model of quantum computer differs from its use of qumodes over qubits [32]. Data is embedded into these qumodes, an infinite-dimensional object. This is typically an electromagnetic field, allowing CV quantum devices to be constructed using quantum photonics hardware. Programming photonic quantum devices proceeds in similarity to qubit-based devices, with both allowing the construction of circuits from quantum gates. As qumodes are represented by an infinite-dimensional uncountable basis, the CV model is a more natural choice to simulate bosonic and continuous systems. In the particular interest of this paper is applications for graph data [33] and machine learning [34].

Photonic devices use single photon emissions, manipulated through squeezers and beamsplitters. A photonic system, even one that uses a non-interacting source of photons, can still exhibit quantum properties such as entanglement. An example of one such system is boson sampling [35]. Boson sampling techniques, such as Gaussian boson sampling (GBS), is an application of continuous-variable quantum computers where there is a clear advantage over classical devices [36]. A GBS device emits photons into an interferometer, generating a sample by counting the photons that exit the device. This setup can be constructed as a circuit in a quantum device. It is difficult to classically simulate the probability distribution of this process as it requires the calculation of the hafnian, a #P problem.

However, when access to GBS devices become more accessible samples can be generated at the rate of 10^5 every second [37]. By embedding graph data into this device we can use it to generate a lower-dimensional representation of the data which is easier to handle when trying to build classifiers. These fast response times give a realised GBS-based anomaly detection method the potential to be suitable for being implemented at the trigger level in future runs of the LHC. While the L1 trigger operates at $1\ \mu s$, the higher-level trigger functions at < 100 ms. Meanwhile, a GBS device could produce around $100,000$ samples in a similar time frame.

The operation of a photonic device also presents some technical advantages over most qubit-based machines. Temperatures of less than 100 millikelvin are required for modern solid-state machines to operate efficiently [38]. Photonic devices are built from photonic hardware that can run at room temperature.

We aim to harness the power of continuous variable quantum computing, specifically Gaussian boson sampling, to survey particle physics events represented as graphs. These

JHEP08(2021)170

samples could then be used as input into various anomaly detection techniques. The aim of our study is to investigate the potential of using boson sampling to embed our graph matrices into a lower dimensional space. The anomaly detection method we use is built around the K-means clustering algorithm. We also present a quantum extension to this. Q-means clustering can be implemented on both a qubit-based, and qumode-based, quantum computer. In its most basic form, Q-means provides a substantial advantage over K-means. The size of the feature vector being used grows the time complexity of K-means linearly, whereas for Q-means it grows logarithmically, an exponential improvement [39, 40].

Specifically, the process of anomaly detection we present has three steps: (i) the creation of data and graphs, (ii) embedding of graphs in a lower dimension representation and (iii) the classification procedure. A classical method is shown as well as the use of quantum equivalents for parts (ii) and (iii). We apply our method to a search for hadronically decaying scalar resonances, produced through a Higgs-portal interaction in the $pp \to HZ$ channel [41, 42]. To allow the process to trigger, we require the Higgs boson to recoil against a boosted leptonically-decaying Z boson. The major Standard Model background for such a signal is $pp \to Z + \text{jets}$.

These events can be represented as a set of graph adjacency matrices, weighted by the event constituent's features (ie. m_{ij}, ΔR). We then find lower-dimensional embeddings for these objects and perform anomaly detection with them. To use as a baseline we create embeddings from the matrix eigenvalues and perform the classification using a K-means clustering algorithm. We then propose a novel method using GBS to create the embeddings and perform anomaly detection using a quantum equivalent of K-means known as Q-means. We find the GBS embedding method has improved performance compared to our classical benchmark.

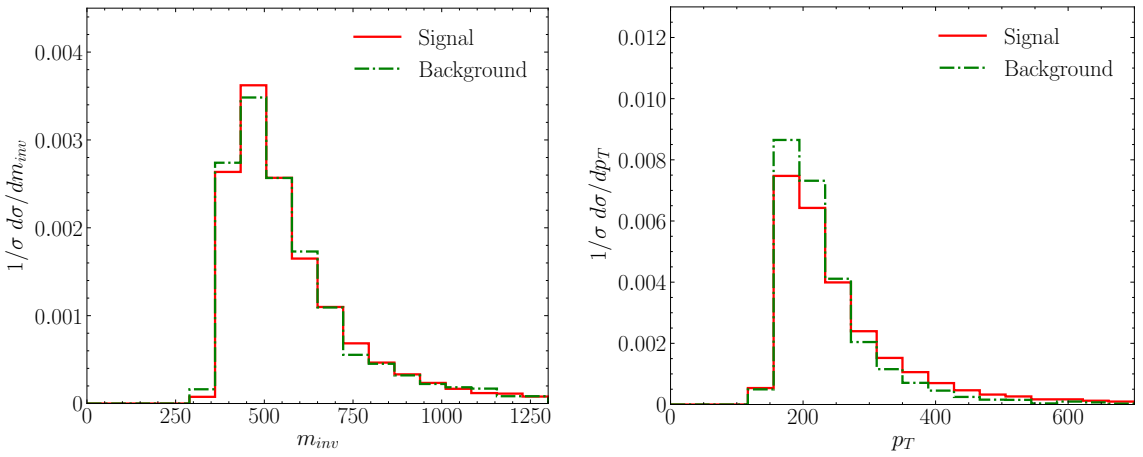
The paper has the following structure: section 2 discusses the event generation and the process of encoding this into a graph structure. In section 3 we introduce our first method of transforming the graph structure into something that can be used to train a classification algorithm. Here, we also present results as to how well the "classical" method of vector embedding performs for anomaly detection. The photonic methodology is discussed in section 4. We detail how a photonic circuit is created, how GBS sampling is performed and the results it gives when its samples are used for classification. Then, in section 5 a quantum equivalent to K-means classification is adopted. Finally, a summary and conclusion is presented in section 6

2 Analysis setup

2.1 Data generation

To use as our background and signal samples we generate $pp \to Z + \text{jets}$ and $pp \to HZ$ events, with subsequent decays $H \to A_1 A_2$, $A_2 \to gg$ and $A_1 \to gg$. Recently an excess over the Standard Model background has been observed by ATLAS in this channel [43]. All events have been generated with a centre of mass energy of 14 TeV and a minimum p_T of the hard process of at least 140 GeV. We force the Z boson to decay leptonically to either e or μ. We have set the Higgs mass to 125 GeV, the A_2 mass to 40 GeV and the mass of A_1

JHEP08(2021)170



(a) **(b)**

Figure 1. (a) shows the invariant mass of the leptons and fat jet while (b) shows the transverse momentum of the fat jet.

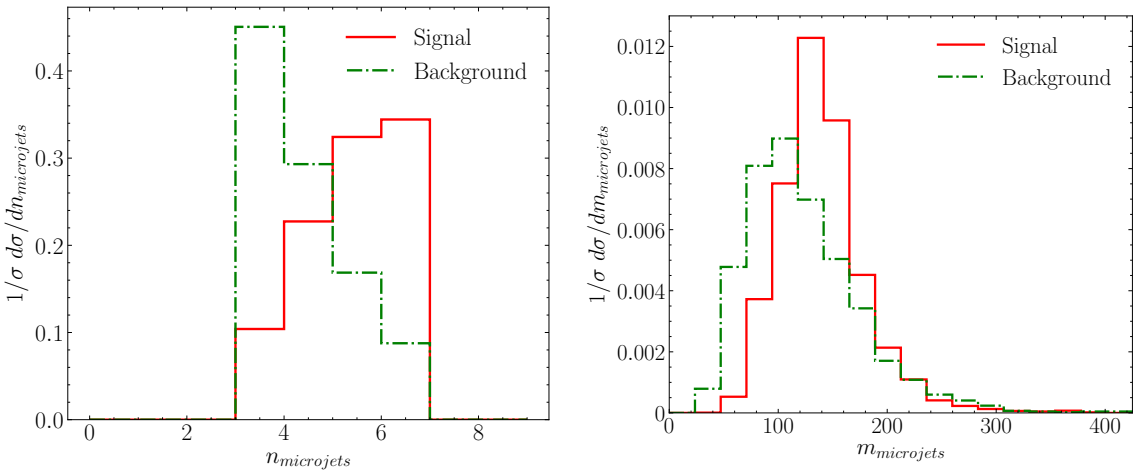
to be 60 GeV. We use Pythia 8.2 to generate events and perform parton showering [44]. Such a scenario could be realised through derivative interactions between the Higgs boson and the pseudoscalars A_1 and A_2, which in turn form an effective, yet highly suppressed, interaction with gluons. Thus, their decay to gluons could still be prompt, whereas their direction production cross section in proton collisions was tiny. In such a scenario the observation of A_1 or A_2 would have to proceed through Higgs decays.

Our analysis follows the methodology used when using jet substructure to find the Higgs for our analysis [45–47]. We will cluster our events into a fat jet, before reclustering its contents into "microjets" [46, 48]. First, though, we impose a rapidity cut of 2.5 and a p_T cut of 10 GeV on all final state leptons. To reconstruct the Z boson we ensure to retain two charged leptons within $|y_l| \leq 2.5$ and require them to have an invariant mass of $80 \leq m_{ll} \leq 100\,\text{GeV}$. To explore the boosted region, we only consider events where the p_T of the Z boson is greater than 150 GeV. Based on the fat-jet properties only, signal events resemble background events very closely.

The remaining objects are subject to a rapidity cut of 5. Using FastJet [49] we cluster these objects into jets using the Cambridge-Aachen algorithm with $R = 1.5$ [50] and demand that there is at least one jet in the event with a transverse momentum of $p_T > 150\,\text{GeV}$. Figure 1 shows the invariant mass of the leptons and fat jet in the event and also the transverse momentum of the fat jet.

The hardest jet is then reclustered into a series of "microjets" using the anti-kt algorithm [51]. Here, we choose $R = 0.2$ and force a transverse momentum of $p_T > 5\,\text{GeV}$. Only events with 3–6 microjets are selected for analysis. The choice of this limit is informed by the runtime of the quantum sampling process used. Larger graphs begin to take inhibitive amounts of time to be sampled on a simulator using our quantum sampling method. The number of jets chosen strikes a balance between the length of time it takes and being able to capture the maximum amount of information on the event. The number of microjets

JHEP08(2021)170



(a) **(b)**

Figure 2. (a) shows the number of microjets in each fat jet. (b) shows the total mass of each fat jet's collection of microjets.

Cuts	background	signal
1 pair of leptons, $\eta < 2.5$, $p_T > 10$ GeV	0.74	0.54
$p_{T_{ll}} > 150$ GeV, 80 GeV $< M_{ll} < 100$ GeV	0.65	0.72
1 fat jet with $p_T > 150$ GeV	0.94	0.97
2 < number of microjets < 7	0.42	0.62

Table 1. Cut efficiencies relative to the previous event reconstruction step for signal and background events.

found, and the total mass of a fat jet's microjets is shown in figure 2. These microjets are the objects used to construct our graphs. The result of the cuts we have applied is shown in table 1. Here, we see the fraction of remaining events after each constraint is applied.

2.2 Constructing the graphs

A method of constructing graphs from our events is to use only the final states [4, 5]. Specifically, we use the microjets found from each event as the nodes to construct a set of graphs. We choose to connect nodes in the final state to every other node. Figure 3 shows an example of a graph we may construct. Signal graphs in our sample contain on average 4.9 nodes, while background graphs contain 3.9. From here we can construct a set of adjacency matrices that can be weighted to give detail on certain features of our event. Adjacency matrices are matrix representations of a graph where the rows and columns are defined by the graph nodes. Entries in the matrix are either 1 or 0, based on whether 2 nodes are connected. However, more information than this can be stored — the matrix entry can be weighted to show how strongly the nodes are connected.

JHEP08(2021)170

Some graph classification methods separate the graph information into feature and adjacency matrices [52]. In these cases, the adjacency matrix will include information on how the nodes in the graph are connected while the node's features are detailed in the feature matrix. However, for many graph sampling techniques, which require as input only symmetric matrices, a feature matrix cannot be readily included. We are limited to how we can manipulate adjacency matrices. Therefore, our aim is to construct a set of symmetric matrices that will include as much information about the node connections, and therefore our event, as possible. Our goal here does not require a model as complex as a neural network. We aim just to find a good representation of the graph, allowing us to use other methods to make classifications.

To begin with, an adjacency matrix can be constructed with connections between nodes weighted by their distance to each other in the (y, ϕ) plane. This distance is given by

$$\Delta R = \sqrt{\Delta y^2 + \Delta \phi^2}, \tag{2.1}$$

where Δy is the difference in rapidity between particle i and j, while $\Delta\phi$ is the difference between the azimuthal angle of particles i and j. The dimensions of the resulting matrix will be determined by the number of nodes in the graph, whereas the matrix entries are found from eq. (2.1). We will eventually want to compare graphs, and hence matrices, of different sizes. Therefore, we pad each matrix with zeros on the right and lower sides such that they all have dimensions of 6×6. To be explicit, we can also construct weighted matrices from $\Delta\phi$ and Δy. Other adjacency matrices can be constructed using a measure of 2 objects energy E_iE_j or invariant mass m_{ij}.

To be able to compare features of individual events with the global properties of the background event sample, each set of adjacency matrices are individually scaled with a constant

$$s = 1/x, \tag{2.2}$$

where x is the maximum value in the set of matrices made from the background samples. For example, the m_{ij} matrices are scaled by a term $s_{m_{ij}}$, while the E_iE_j matrices are scaled by $s_{E_iE_j}$. Thus, the five matrices (ΔR, Δy, $\Delta\phi$, E_iE_j and $m_{i,j}$) are all scaled accordingly.

Figure 3 shows the entire process discussed in this section. To summarise, to create a graph that corresponds to a single event:

1. Use the procedure detailed in section 2.1, we create an event and find a set of final state microjets.

2. The microjets can be thought of as nodes in a fully interconnected graph, which can be written as an adjacency matrix.

3. From this adjacency matrix, five can be created. These five are each weighted by ΔR, Δy, $\Delta\phi$, E_iE_j and $m_{i,j}$.

4. All five are then normalised, then padded to have dimensions 6×6. This is what is shown as the final set of matrices in figure 3 and is what will be embedded in a lower-dimensional space and used as input into a classifier.

These steps are performed for each generated event, such that each event will have 5 corresponding matrices.

JHEP08(2021)170

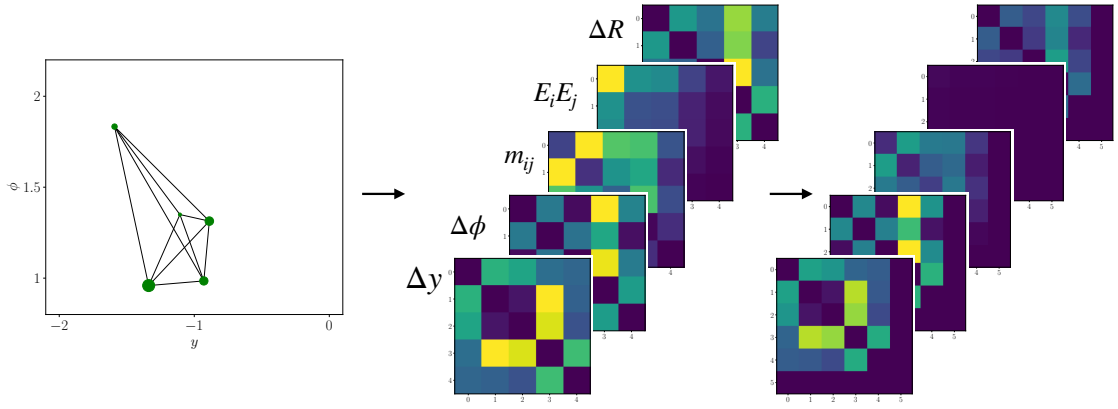


Figure 3. For a graph we find five weighted adjacency matrices from it, each one capturing features of the graph nodes. These five are then scaled and padded, giving the final matrices on the right. These can be embedded and used as input to classification algorithms.

3 Anomaly detection on a classical computer

To make predictions the graph data will be embedded into vectors. By being a 1-dimensional vector, rather than a 2-dimensional matrix, it is simpler to use them as input into a classifier. These vectors will then be used in an unsupervised K-means method of anomaly detection. What is being done here is somewhat different from kernel-based methods of graph classification. If one were to use the kernel framework the entire graph dataset would be embedded into a kernel. These kernels provide a measure of how similar each graph is to the other. This object could then be passed to a kernel-based classification algorithm (ie, a SVM) and from there predictions could be made. While this method is popular and has been shown to have predictive powers it is not without its limitations. By creating a graph-kernel one is required to use a kernel-based algorithm. However, by creating feature vectors from the graphs we can be more versatile when creating a model as the range of algorithms available to use is broader.

The adjacency matrices created in section 2 will be the objects used to create feature vectors. For the classical example we will simply take the eigenvalues of the five matrices. Each of the matrices gives us 6 eigenvalues. When these 5 objects are combined it results in a vector of length 30. Finally, we apply another round of scaling. This is done using StandardScaler from scikit-learn [53].

This method of creating a vector is based on the use of Laplacian Eigenvalues [54]. This method first transforms the adjacency matrix into its Laplacian — another form of graph matrix representation. The eigenvalues are taken of this and ordered. We found that the use of a vector created this way and one created with the more straightforward method described above gives similar results.

We prepare 1000 background samples and 200 signal samples for use. The background set is then split into 800 training examples and 200 test examples. The number of events we use is limited due to the nature of sampling from a simulated GBS device, which is discussed in section 4.

JHEP08(2021)170

K-means clustering is a popular method of unsupervised classification [55, 56]. Its aim is to separate samples into several clusters. Each cluster has a centroid that is defined by the mean μ of the samples inside it. To begin finding these clusters the centroids can be initialised randomly. Then, they are updated by the algorithm by repeating 2 steps: (i) assign every point a label. This label describes what cluster it belongs to and is decided by the centroid closest to the point. (ii) The positions of the centroids are then updated by taking the average of every point in the cluster. These steps repeat until the change in centroid location is less than a selected value. As the centroids are updated the process aims to minimise the within-cluster sum-of-squares

$$\sum_{i=0}^{n} \sum_{x \in C_i} \|x - \mu_i\|^2 , \tag{3.1}$$

where C is the set of clusters $(C_1, \ldots, C_n)$ each with a mean $(\mu_1, \ldots, \mu_n)$. In our method, we only cluster our samples into one cluster. In this scenario, clustering is a somewhat-trivial problem. However, our choice of algorithm here is simply to develop a baseline to allow us to compare our classical sampling technique to a quantum equivalent. Regardless, the cluster is created using the 800 background training samples and results in the centroid c. When testing on a sample, x, we can then define a loss function

$$L = \text{distance}(c, x), \tag{3.2}$$

which measures the distance between a point and the cluster centroid. Intuitively, it can be assumed points closer to the centroid will more likely come from the background sample while points further away will more likely be signal. Therefore, a point with a larger loss value can be classified as an outlier and, thus, as signal.

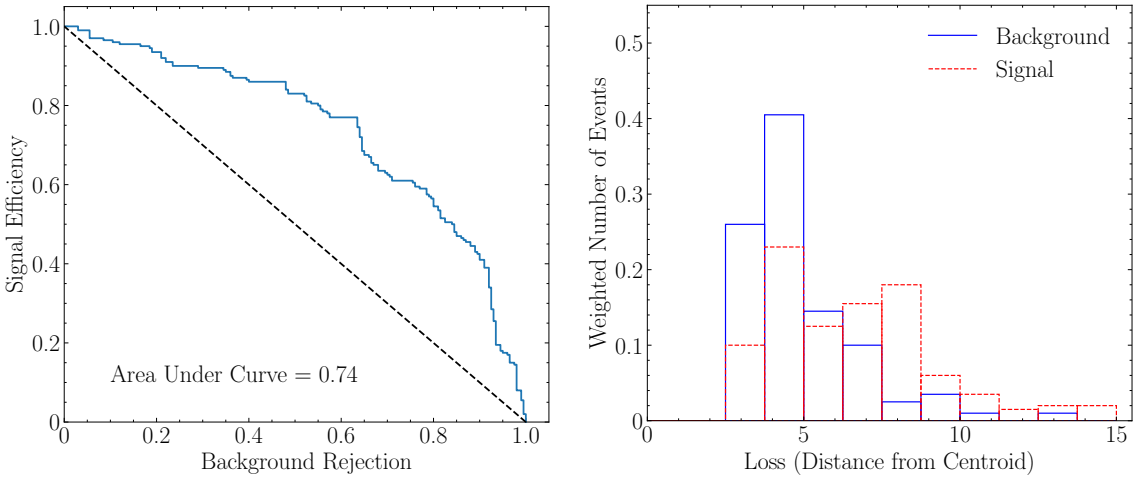
Figure 4(a) shows the ROC curve from testing the fitted K-means algorithm, giving a result of 0.74 AUC. Figure 4(b) shows the distribution of distances for the background and signal test sets.

To give some context for this result we can compare it to another classical method of anomaly detection. This will be done using an autoencoder. Autoencoders are neural networks where the input and out dimensions match while the middle of the network forms a bottleneck. The network's optimisation task is to recreate the input exactly in the output, but this, of course, is impossible due to the smaller number of nodes in the centre of the network. Like the K-means method, the autoencoder is only trained on background data. By comparing the loss associated with background test data and signal data one can create an anomaly detector [2]. We find this method gives the same AUC of 0.74, showing our choice of a benchmark method gives a result comparable to other methods.

4 Anomaly detection on a photonic quantum devices

In section 3 we described a classical method of embedding the matrices we created into a vector and classifying them. Here, we replace the embedding procedure with an equivalent that can be performed on a quantum device.

JHEP08(2021)170



(a) **(b)**

Figure 4. (a) shows the ROC curve for K-means anomaly detection method trained on classically constructed feature vectors. The distribution of the loss for the signal and background test sets is shown in (b).

The continuous-variable quantum computing regime differs from traditional, discrete, qubit-based quantum computing in many key areas. In the qubit system we can describe states as being in the form

$$|\psi\rangle = c_0|0\rangle + c_1|1\rangle. \tag{4.1}$$

However, when we move to the CV form of quantum computing our states are no longer represented as qubits, but rather as qumodes. Qumodes are vectors expressed in an infinite dimensional uncountable basis

$$|\psi\rangle = \int \psi(x)|x\rangle dx. \tag{4.2}$$

While having an infinite dimensional basis is a fundamental change of the configuration space of our quantum system compared to eq. (4.1), the dynamics of the state $|\psi\rangle$ are still governed by unitary operators. Thus, by embedding data into a qumode one can apply gates to evolve the state, which is then measured at the end to obtain the result. States can be described in the Gaussian or Fock basis.

Fock states provide a discrete and countable basis. A quantum device, made up of multiple qumodes, can be described by a state $|X\rangle = |x_1, \ldots, x_N\rangle$, where N is the number of qumodes in the system. Each x_i which forms the state X is an integer number. These are the number of photons found in each detector after exiting the device. If the device emits n total photons, we can define $\Phi_{n,N}$ — the entire set of all possible device output patterns. This leads to a state [35]

$$|\psi_X\rangle = \sum_{X \in \Phi_{n,N}} \alpha_X |X\rangle, \tag{4.3}$$

JHEP08(2021)170

where α is a complex number such that $\sum |\alpha_X|^2 = 1$. The probability of observing a given state is

$$\mathrm{P}(X) = |\alpha_X|^2 = |\langle \psi_X | X \rangle|^2. \tag{4.4}$$

The probability of an input state, $|I\rangle = |i_1, \ldots, i_N\rangle$, transitioning to $|X\rangle$ is shown to be [57]

$$\mathrm{P}(I \to X) = \frac{|\mathrm{Per}(U_{I,X})|^2}{i_1! \ldots i_N! x_1! \ldots x_N!}. \tag{4.5}$$

This calculation depends on the permanent, Per, a matrix manipulation described in further detail in section 4.1. Equation (4.5) also depends on $U_{I,X}$, a representation of the device itself.

To describe a Gaussian state, it is useful to consider the evolution of the vacuum state

$$|\psi\rangle = \exp(-itH)|0\rangle, \tag{4.6}$$

where t is the evolution time and H is a Hamiltonian. A specific qumode k, at its simplest, can be described as a harmonic oscillator [58]

$$\hat{H}_k = \frac{1}{2}(\hat{p}_k^2 + \omega_k^2 \hat{x}_k^2), \tag{4.7}$$

with the position and momentum operators

$$\hat{p}_k = -\sqrt{\frac{\hbar\omega_k}{2}}(\hat{a}_k - \hat{a}_k^\dagger), \tag{4.8}$$

$$\hat{x}_k = \sqrt{\frac{\hbar}{2\omega_k}}(\hat{a}_k + \hat{a}_k^\dagger). \tag{4.9}$$

These operators both depend on the creation and annihilation ladder operators $\hat{a}$ and $\hat{a}^\dagger$. A Gaussian state is one in which the Hamiltonian is, at most, quadratic with respect to the ladder operators. Such states are described by a displacement term α that describes the centre of the $\hat{x}$ and $\hat{p}$ distributions and a squeezing term z that decides the rotation and variance.

As described above, a continuous variable device will be constructed from multiple qumodes, each being evolved by a series of quantum gates. While this is similar to the discrete qubit paradigm of quantum computing, the gate-set available to construct CV circuits largely differs [58, 59]. A gate can take the form of the unitary $\hat{U} = \exp(-\frac{i}{h}t\hat{H})$, where $\hat{H}$ is the generating Hamiltonian with time t. When constructing circuits it is typical to use squeezing, rotation and displacement operations. Here, we consider the term gate and operator as synonymous, with a gate being the physical implementation of the operator.

Squeezing refers to the reduction of one observable in a state, while amplifying another. The effect of squeezing on one qumode can be described with the Hamiltonian [32]

$$\hat{H} = ih\frac{\kappa}{2}(e^{i\phi}\hat{a}^{\dagger^2} - e^{-i\phi}\hat{a}^2), \tag{4.10}$$

where a and $\hat{a}$ are the ladder operators. The amount of squeezing or amplification, and the phase of this change, is parameterised by κ and ϕ, respectively. The squeezing hamiltonian

JHEP08(2021)170

is quadratic, with respect to its ladder operators, and hence will construct a Gaussian gate. Looking at the unitary operator for such a Hamiltonian

$$\hat{U} = \exp\left(-\frac{\kappa}{2}(e^{i\phi}\hat{a}^{\dagger^2} - e^{-i\phi}\hat{a}^2)t\right), \tag{4.11}$$

we can construct a squeezing gate

$$\hat{S}(z) = \exp\left(\frac{1}{2}(z^*\hat{a}^2 - z\hat{a}^{\dagger^2})\right). \tag{4.12}$$

The gates depends on a squeezing parameter $z = -r\text{exp}(i\phi)$, where $r = \kappa t$. The value r controls the squeeze amount and ϕ is the squeeze phase angle. The gate, applied to a mode, performs the transformation

$$\begin{aligned}\hat{S}^\dagger(z)\hat{x}_\phi\hat{S}(z) &= e^{-r}\hat{x}_\phi,\\ \hat{S}^\dagger(z)\hat{p}_\phi\hat{S}(z) &= e^{r}\hat{p}_\phi.\end{aligned} \tag{4.13}$$

For a position-momentum pair, the squeezing gate will amplify one while reducing the other. Similarly, a rotation operator can be constructed that rotates the state

$$\hat{R}(\theta) = \exp\left(i\theta\hat{a}^\dagger\hat{a}\right). \tag{4.14}$$

Here, θ controls the rotation angle. The gates application on a state will rotate a states position and momentum

$$\begin{aligned}\hat{R}^\dagger(\theta)\hat{x}\hat{R}(\theta) &= \hat{x}\text{cos}(\theta) - \hat{p}\text{sin}(\theta),\\ \hat{R}^\dagger(\theta)\hat{p}\hat{R}(\theta) &= \hat{p}\text{cos}(\theta) - \hat{x}\text{sin}(\theta).\end{aligned} \tag{4.15}$$

The final Gaussian gate we will introduce is the beamsplitter gate

$$\hat{B}(\theta,\phi) = \exp\left(\theta(e^{i\phi}\hat{a}_1\hat{a}_2^\dagger - e^{-i\phi}\hat{a}_1^\dagger\hat{a}_2)\right). \tag{4.16}$$

This a 2 mode gate (ie. it takes two qumodes as input) that transforms two qumodes based on its parameters ϕ and θ.

$$\begin{aligned}\hat{B}(\theta,\phi)^\dagger\hat{x_1}\hat{B}(\theta,\phi) &= \hat{x_1}\text{cos}(\theta) - \text{sin}(\theta)[\hat{x_2}\text{cos}(\phi) + \hat{p_2}\text{sin}(\phi)],\\ \hat{B}(\theta,\phi)^\dagger\hat{p_1}\hat{B}(\theta,\phi) &= \hat{p_1}\text{cos}(\theta) - \text{sin}(\theta)[\hat{p_2}\text{cos}(\phi) - \hat{x_2}\text{sin}(\phi)],\\ \hat{B}(\theta,\phi)^\dagger\hat{x_2}\hat{B}(\theta,\phi) &= \hat{x_2}\text{cos}(\theta) + \text{sin}(\theta)[\hat{x_1}\text{cos}(\phi) - \hat{p_1}\text{sin}(\phi)],\\ \hat{B}(\theta,\phi)^\dagger\hat{p_2}\hat{B}(\theta,\phi) &= \hat{p_2}\text{cos}(\theta) + \text{sin}(\theta)[\hat{p_1}\text{cos}(\phi) + \hat{x_1}\text{sin}(\phi)].\end{aligned} \tag{4.17}$$

Beamsplitter gates evolve one qumode based on the properties of another. They are useful for creating devices such as interferometers.

As well as Gaussian gates there are also a set of non-Gaussian gates. These gates differ from Gaussian gates as they will only ever have 1 mode as input, and also through the degree of H. If the hamiltonian in the unitary has as degree of 3 or more (the position,

JHEP08(2021)170

momentum or a ladder operator in the gate is cubed, or more) it is classed as non-Gaussian. An example of this is the cubic phase gate

$$\hat{V}(\gamma) = \exp\left(i\frac{\gamma}{6}\hat{x}^3\right). \tag{4.18}$$

The cubic phase gate evolves a state depending on the parameter γ, such that

$$\begin{aligned}\hat{V}^\dagger(\gamma)\hat{x}\hat{V}(\gamma) &= \hat{x},\\ \hat{V}^\dagger(\gamma)\hat{p}\hat{V}(\gamma) &= \hat{p} + \gamma\hat{x}^2\end{aligned} \tag{4.19}$$

With a set of Gaussian gates it is possible to construct all quadratic unitaries. However, a gate-set combining both Gaussian and non-Gaussian gates allows for universal quantum computation. This is defined by the computers ability to implement in a finite number of steps, with arbitrary precision, a unitary which is polynomial [32].

Continuous-variable photonic quantum devices have been shown to solve some classically difficult problems. One such problem is Gaussian boson sampling. This is the quantum method we will use to embed our matrices. For now, the classification step (K-means) will remain unchanged.

4.1 Gaussian boson sampling

An area where photonic devices demonstrate an advantage is through boson sampling [35, 36]. Boson sampling devices are designed to emit single photons into an interferometer. The photons outputted from the interferometer are counted by detectors. There will be a probability associated with seeing a photon in a specific detector. Therefore, the device output can be represented by a probability distribution. Specifically, a distribution can describe the probability of getting specific photon count patterns from the device. A photon count pattern is a vector describing, for a single run of the device, how often a photon is seen at each detector. A boson sampler can be thought of as probing this probability distribution and acquiring samples from it. Matrices can be embedded into the device's interferometer, deforming the distribution. When the device is running (and the distribution sampled) the output will be related to the information embedded in the interferometer. By probing the device repeatedly one can use the samples created to build vectors to use for classification. This new vector will try to capture some information as to how often photons are seen by each detector. In our scenario, the boson sampler will be a circuit created by the gates described in section 4. This circuit can be run on a photonic device. We will embed graph adjacency matrices into this circuit and find samples. These samples are used to create feature vectors and are fed into our anomaly detection methods [33, 37, 60].

A simple analogue to boson sampling is the Galton board. A Galton board is built from a vertical board with a series of pegs attached. Balls are dropped into the device and make their way down the board, their path being altered by the pegs they hit. At the bottom of the board the balls are collected. In the boson sampling device however bosons can be emitted from multiple locations, instead of just one. Similar to the balls in

JHEP08(2021)170

the Galton board, the bosons do not interact with each other. To encode information into the device one can change the layout of the "pegs". The probability of finding a ball in a specific bin (or a photon at a specific detector) requires the calculation of the permanent. As shown in eq. (4.5), this can also be used to calculate the transition probability between two states in a photonic quantum device. The permanent is given by

$$\mathrm{Per}(A) = \sum_{\pi \in S_N} \prod_{i=1}^{n} A_{i,\pi(i)}. \tag{4.20}$$

The permanent is found from the matrix A, whose entries A_{ij} are the probability that a photon i are found in detector j, and S_N, the entire set of permutations of N photons. The permanent can also be defined by its relation to graphs. Calculating the permanent of the adjacency matrix of a bipartite graph is the equivalent of summing all the graphs perfect matchings. A graph, or subgraph, can be described as a "matching" if every vertex in the graph is connected with, at most, one other vertex. A "perfect matching" is the situation when every vertex is connected to one other vertex. The calculation of the permanent is a #P-problem and hard to solve on classical computers [61].

Boson Sampling requires the generation of deterministic sources of single photons — something not currently available. Extensions to the above boson sampling model have been proposed to get around this. One of these is Gaussian boson sampling (GBS) [36]. GBS, instead of single photon states, uses single mode squeezed states. This setup has the same problem complexity as typical boson sampling. However, since squeezed states can be generated deterministically, it does not have the same scaling issues.

Unlike "standard" boson sampling, which requires the calculation of the permanent, to simulate Gaussian boson sampling requires the calculation of the hafnian. The hafnian and permanent are related through

$$\mathrm{Per}(M) = \mathrm{Haf}\begin{pmatrix} 0 & M \\ M^t & 0 \end{pmatrix} \tag{4.21}$$

where M is a matrix. The hafnian calculates how many perfect matchings are in an arbitrary graph. If the edges are weighted (ie. there is a weighted adjacency matrix) the hafnian calculation sums the weights associated with the vertices of the perfect matchings. The hafnian is calculated as

$$\mathrm{Haf}(M) = \sum_{\pi \in P_n} \prod_{(u,v)\in\pi} M_{(u,v)}, \tag{4.22}$$

where M is a generic symmetric adjacency matrix. Here, P_n is the set of all permutations of pairs from a list of indices n long. If $n = 4$, then $P_n = [\{(1,2),(3,4)\}, \{(1,3),(2,4)\}, \{(1,4),(2,3)\}]$. For a graph containing n nodes, P_n would find every set of perfect matchings. Note, in the $n = 4$ example that the three sets found are the indices of nodes in the three possible perfect matching sets. If n were to equal an odd number, there could not be a perfect matching and the hafnian would equal zero. The hafnian is a more general function than the permanent but like it, there is no known method of classically calculating it efficiently.

JHEP08(2021)170

The state prepared by a GBS device with N modes is fully described by a $2N \times 2N$ covariant matrix σ and a displacement term, d. Here, we focus solely on σ. For such a state the GBS device can be sampled. What will be outputted is an array of photon counts. The array $\bar{n}$ is filled such that $\bar{n} = [n_1, \ldots, n_N]$, where each n_i represents how many photons have been counted at each detector. The probability of seeing a result $\bar{n}$ is

$$P(\bar{n}) = \frac{1}{\sqrt{\det(Q)}} \frac{\text{Haf}(\tilde{A}_{\bar{n}})}{\bar{n}!}. \tag{4.23}$$

Here, $\bar{n}! = n_1! n_2! \ldots n_N!$. The matrix $\tilde{A}_{\bar{n}}$ and Q are both related to the covariance matrix σ. This can be seen through the relations $Q = \sigma + I/2$, $\tilde{A} = X(I - Q^{-1})$ and $X = \left(\begin{smallmatrix} 0 & I \\ I & 0 \end{smallmatrix}\right)$ [36]. In eq. (4.23) the hafnian of $\tilde{A}$ is not being found, but rather a modified version $\tilde{A}_{\bar{n}}$. The new matrix will be the same, except for the removal (or duplication) or certain rows and columns from $\tilde{A}$, depending on the values in $\bar{n}$. If $\bar{n}_i = 0$, then the i-th row and column of $\tilde{A}$ will not be included, if $\bar{n}_i = 1$ the row/ column will not change, and if $\bar{n}_i > 1$ the row/ column will be duplicated. Finally, if all values in $\bar{n}$ are 1 then $\tilde{A} = \tilde{A}_{\bar{n}}$.

For our purposes, we need to be able to embed our graphs into the device to retrieve samples. Therefore, there is a need to relate the adjacency matrices A to the covariance matrix σ. This is done through the double encoding strategy

$$\tilde{A} = A \oplus A^*. \tag{4.24}$$

Eq. (4.23) can be written such to show the probability of sampling a photon event $\bar{n}$ with respect to our adjacency matrices

$$P(\bar{n}) = \frac{1}{\sqrt{\det(Q)}} \frac{|\text{Haf}(A_{\bar{n}})|^2}{\bar{n}!}. \tag{4.25}$$

Again, the hafnian of the entire matrix A is not found, but rather a submatrix $A_{\bar{n}}$. Practically, to sample from this distribution we embed the matrix into the device via parameters in the squeezing gates and interferometer of our photonic circuit.

Overall, the probability of a set of photons $\bar{n}$ being detected is therefore proportional to the weighted number of perfect matchings of the subgraph $A_{\bar{n}}$. By sampling from a GBS device we can create feature vectors to use as input to classifiers.

4.2 Constructing a GBS circuit

As we have seen, to classically simulate the GBS device and calculate the probability of seeing a specific output pattern requires the calculation of the Hafnian. The reliance on the Hafnian calculation makes GBS devices difficult to simulate on a classical machine. However, probing a real photonic device is computationally cheap — a real GBS device can be sampled 10^5 times every second [37].

Using quantum gates one can build a Gaussian boson sampling device that will run on a photonic quantum computer. The GBS circuit is created from squeezing, rotation and beamsplitter gates and is shown in figure 5. It is through these gate parameters we can embed our adjacency matrices. The gate parameters (detailed in eqs. (4.12), (4.14) and (4.16)) are all determined by an adjacency matrix A.

JHEP08(2021)170

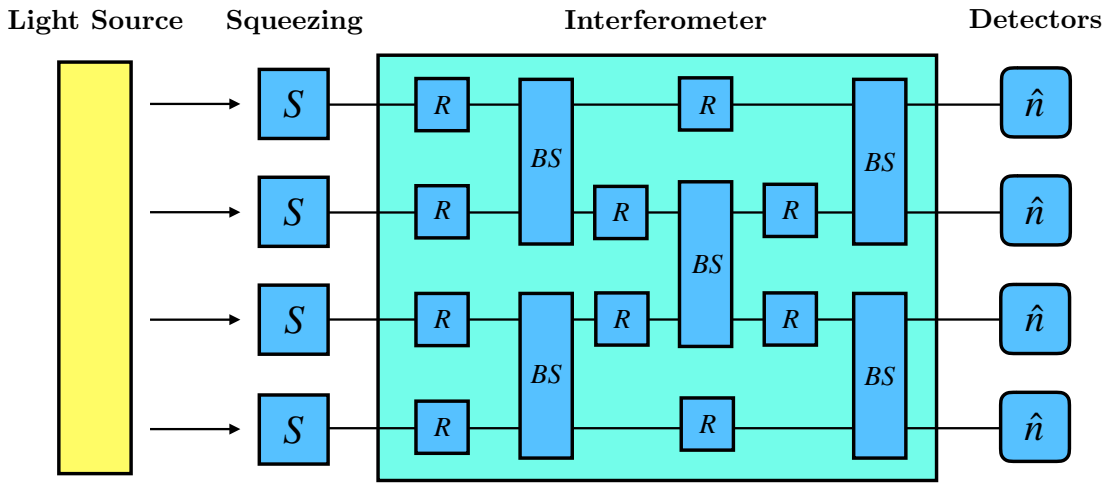


Figure 5. A Gaussian boson sampling device. It consists of a set of squeezing gates, an interferometer and photon detectors. The interferometer itself is constructed from rotation gates and beamsplitters.

Each matrix will be embedded individually. To retrieve the gate parameters from one we must first decompose A using the Takagi-Autonne decomposition, to obtain

$$A = U\text{diag}(\lambda_1, \ldots, \lambda_N)U^T. \tag{4.26}$$

Here, U is a unitary matrix and the set of λ's are the matrix eigenvalues [33].

Squeezing gates require one parameter: z. To find this we must first introduce two new terms: c, a scaling parameter and η, the mean number of photons that will exit the GBS device [33]. This value, η is a parameter we can choose. The scaling term c is chosen such that

$$\eta = \sum_{i=0}^{N} \frac{c\lambda_i^2}{1 - c\lambda_i^2}, \tag{4.27}$$

where N will be the number of qumodes in the device. Using the set value of c and the appropriate λ allows us to find the value for the squeezing gate for a particular qumode i

$$z_i = \tanh^{-1}(c\lambda_i). \tag{4.28}$$

The interferometer parameters on the device will be found using the unitary matrix U found from eq. (4.26). To find the parameters required for the beamsplitter and rotation gates the unitary is decomposed using the procedure detailed in ref. [62].

When the GBS circuit is probed the result is informed by the matrix embedded within it. In a classical scenario, finding information about the output would be time consuming due to the nature of running the simulation. However, this is not the case for a real device. Creating the network (and the embedding of a matrix) is accomplished by using the Python library Strawberry Fields [59]

JHEP08(2021)170

4.3 Creating feature vectors from GBS samples

By first transforming graphs into their adjacency matrix representation (as carried out in section 2) they can then be embedded into the GBS device. When the device is sampled the process will therefore be driven by the information contained in the matrix. The generated samples can be used to create feature vectors to use in a classification procedure. The output from the GBS device will be arrays N long, where N equals the number of nodes in our graph. This vector will be made from information on frequencies of measurements from the device. As we run the device, we can choose from either thresholding the output, or not. In "threshold" mode, the output sample of the device only tracks if a photon has been detected in a specific mode or not, rather than count how many have been seen. On a simulator, not thresholding the results proved very time-intensive. Therefore, the sample results where thresholded. The output of the device will therefore be an array N long, filled with zeroes or ones depending on whether a photon was detected.

Regardless, sampling using the GBS simulator is time-consuming with or without photon thresholds. This constrains us with the amount of data we can use. As mentioned, feature vectors are built from frequencies of certain measurements from within an events set of samples. It follows that we need to sample each graph multiple times to accurately determine these measurements. The amount of samples taken from each event is another factor that influences runtime. We choose to sample each of the 5 matrices in a single event 6500 times, giving a total of 32,500 samples per event.[1]

The generated set of samples can then be used to calculate feature vectors. Here, a few methods exist to construct these [37]. An option is to craft a vector that contains probabilities of a photon being seen at each detector. If we have an event with a device containing k photon detectors we can construct a vector

$$v = (p_{k_1}, p_{k_2}, \ldots, p_{k_N}). \tag{4.29}$$

Here, p_k is the chance a photon will be seen in this position by the detector. For each graph, we have a set of five adjacency matrices. For each matrix a vector is found (just as in eq. (4.29)). The five vectors representing the graph are then combined, making a vector of length 30. This procedure mirrors how we handled the classical eigenvalue scenario. However, here we create the samples from the GBS device.

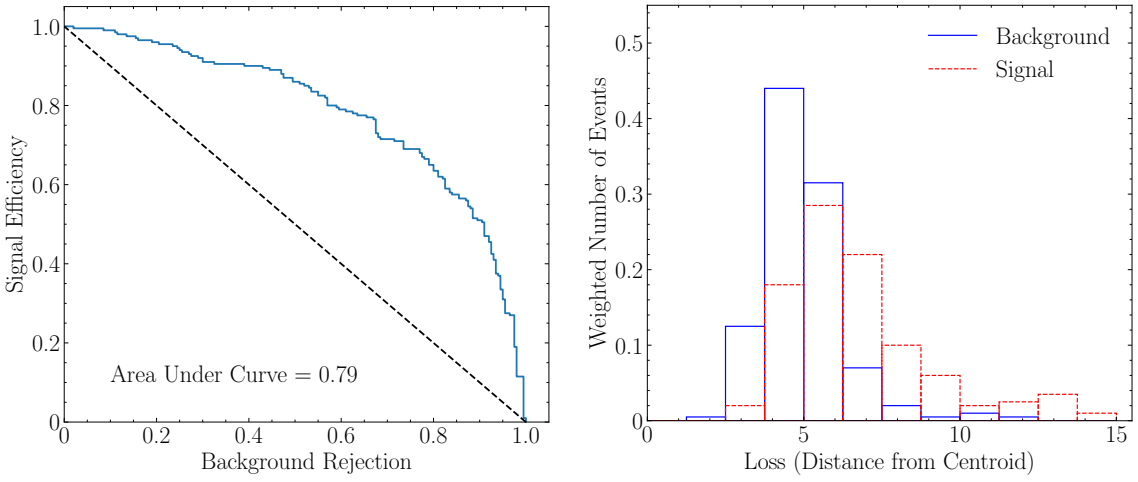
To summarise our procedure:

1. For an event, one of its associated matrices are is embedded into a GBS device. This is done by decomposing the matrix to give values for the gate parameters of a quantum circuit.

2. The GBS device is sampled repeatedly, giving a series of outputs that detail the photon counts seen from the detectors.

[1] We find our choice in the number of samples gives a good result. This choice of samples is guided by the methodology in ref. [63]. Here, it suggests using roughly 10,000. After full sampling of the matrices that make up an event we surpass that value.

JHEP08(2021)170



(a) **(b)**

Figure 6. (a) shows the ROC curve for K-means anomaly detection method trained on feature vectors constructed from GBS samples. The distribution of the loss for the signal and background test sets is shown in (b).

3. With a set of photon counts, we create a vector describing the probability of seeing a photon at a specific detector.[2]

4. Steps 1–3 are repeated for all five matrices associated with a event. The five resulting vectors are combined. This combined vector will be input to the classification method.

4.4 K-means anomaly detection using Gaussian boson sampling

To set a benchmark for anomaly detection with samples from the GBS device we will once again use K-means clustering. The pattern for this will match what was done in section 3. The vectors will, firstly, be scaled using scikit-learn's StandardScaler. The K-means algorithm will then be run on 800 background samples to find a centroid. Distances can be calculated from the centroid to points in our test set of 400 background and signal graphs. This distance will be used as an error value in the fit, as shown in eq. (3.2).

Results for this method are shown in figure 6. We achieve an AUC score of 0.79. This is an improvement over the classical scenario where an AUC of 0.74 was achieved. The GBS sampling method appears to provide a competitive way of creating samples to use for anomaly detection tasks.[3]

[2]While these probabilities may have a statistical uncertainty associated with them, this is not explored in this paper.

[3]In the future, attempting the same methodology with a larger dataset would be a useful exercise. This would ensure the conclusions drawn here are not because of an inadvertently well-chosen sample of data. However, at the moment, we note that creating samples from the GBS simulator is a time-consuming process, which necessitated our decision to use a small dataset. We hope that the results seen here continue to be relevant as it becomes faster to sample the simulator.

JHEP08(2021)170

5 Q-means clustering

As mentioned in section 1 we can consider our anomaly detection problem in three parts: (i) graph creation, (ii) embedding of matrices and (iii) the classification method. In section 4.1 we discussed a quantum method to embed the matrices. In this section, we introduce Q-means clustering, a quantum equivalent of K-means clustering, which we will use for anomaly detection.

K-means can be described as having two steps: (i) assigning points a label and (ii) updating the centroids. To successfully complete part (i) there is a need to be able to deduce what centroid each point is closest to. In this variant of Q-means we focus specifically on this.

As with the GBS sampling device, the Q-means method is applied using a quantum circuit. The circuit constructed aims to provide a measure of how close a point is to a centroid. The circuit itself can be split into three parts: data embedding, distance calculation and readout. After calculating which centroid the point is closest to we can then move onto the second step of the clustering procedure, updating the centroids. The centroids are chosen such that they are the mean of every point in the cluster. These steps will be repeated until the distance between the updated cluster location and the previous location is less than a threshold ϵ [39].

We will first view the task of Q-means classification in the discrete qubit scenario. To embed our data to use in the circuit we will use U_3 gates,

$$U_3(\theta,\phi,\lambda) = \begin{bmatrix} \cos(\frac{\theta}{2}) & -\exp(i\lambda)\sin(\frac{\theta}{2}) \\ \exp(i\phi)\sin(\frac{\theta}{2}) & \exp(i(\phi+\lambda))\cos(\frac{\theta}{2}) \end{bmatrix}. \tag{5.1}$$

The vectors are embedded in the θ, ϕ and λ parameters. This takes the values and embeds them as angles in a qubit. For vectors with more than three features we must use more than one qubit.

After being embedded the states must enter into a circuit designed to measure some metric of distance. This will be done using the SwapTest circuit. See appendix A for more information. For two states $|\alpha\rangle$ and $|\beta\rangle$ the SwapTest routine will measure the overlap between them. If we find a probability of $P(|0\rangle) = 0.5$ then the two states are orthogonal, if $P(|0\rangle) = 1$ then the states are the same. If the centroids are embedded into a state alongside a sample vector we can measure which centroid state overlaps the most with the vector. The centroid with the most overlap will be chosen as the closest and the vector will be assigned to that cluster.

SwapTest circuits are constructed from Hadamard H and SWAP gates, respectively

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{5.2}$$

and

$$\text{SWAP} = \begin{bmatrix} 1&0&0&0 \\ 0&0&1&0 \\ 0&1&0&0 \\ 0&0&0&1 \end{bmatrix}. \tag{5.3}$$

JHEP08(2021)170

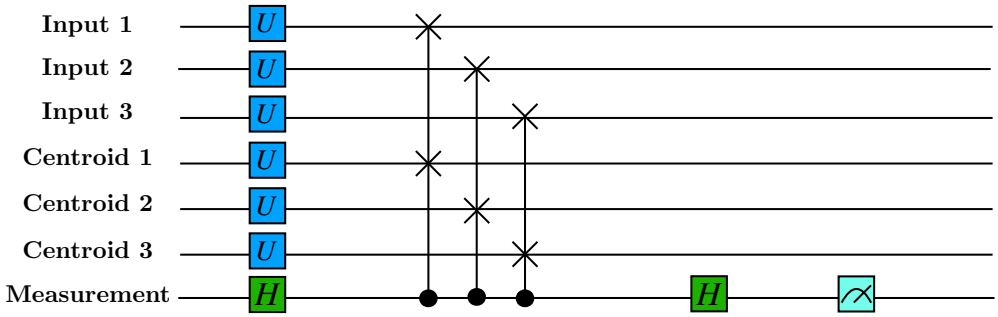


Figure 7. Diagram of the Q-means quantum circuit. The feature vector and centroid is encoded through U_3 gates, while a SwapTest is used to provide a measure of their similarity.

A CSWAP gate is similar to a SWAP gate, eq. (5.3), but depends on the state of a control bit. The Hadamard gate is used regularly when building quantum circuits to introduce superposition into the system. These gates, alongside the U_3 embedding gates are combined to form the circuit shown in figure 7. The circuit shown is designed for a model where vectors contain between 6 and 9 features, and hence need to use 3 qubits to store the information. The vectors created from the GBS method contain 30 elements and hence will require 10 qubits.

We have focused on a single variant of Q-means here. However, the model can be expanded to improve its performance further. Including other quantum algorithms (such as Grovers) can improve the label assigning stage [39], while a deeper reliance on qRAM can allow for an improved scaling complexity with respect to the data-set size itself [64].

What is appealing about the construction of this circuit is that (assuming states are prepared) the SwapTest routine does not depend on the number of points in a vector. To encode a state into quantum memory has the time complexity of $\mathcal{O}(\log(N))$, where N is the number of features in our vector [39, 40]. This matches with what we may intuitively expect — for N features we require $\log_2(N)$ qubits. For regular K-means we expect a time complexity of $\mathcal{O}(N)$. By using a quantum variant of K-means we see an exponential performance increase, with respect to the number of features in our sample.

5.1 Implementation, performance and scalability

The discussion in the previous section is based on the discrete qubit quantum computing regime. However, this methodology is also possible to carry out in the continuous-variable, qumode scenario. The Q-means method presented here is based on using the SwapTest to measure the overlap of two states. This has been shown to be the equivalent of Hong-Ou-Mandel effect [65]. The Hong-Ou-Mandel effect is a way to compare two arbitrary states in quantum photonics. Making use of this would allow one to implement Q-means on a photonic device straightforwardly.

To implement the Q-means quantum circuit shown in figure 7 we use the Python package PennyLane [66]. The Q-means algorithm is trained with the same parameters as the previous two scenarios — using 800 background samples, each with 30 features. We

JHEP08(2021)170

use both methods of generating samples to train the model. In both cases, the classical eigenvalue scenario and the GBS sample variant, we find Q-means and K-means results to be equivalent. If one judges the Q-means performance based on the potential efficiency improvements there is a clear benefit to its use. The power of Q-means lies in its ability to scale to larger feature-vectors. The increase in the availability of quantum devices, and the ability to run these algorithms on their native devices, allows clustering methods to be more viable.

6 Conclusions

Separating rare unspecified signal events from common Standard Model backgrounds is one of the most important tasks multi-purpose experiments at the LHC perform. Data-driven searches, ones that train directly on the data without making assumptions about any features in new physics models, are an appealing option. By learning solely the distributions of the background Standard Model events it is possible to flag events that do not share similar features as signal. When building models and constructing datasets to train on it is worth considering how data is represented. Particle physics events are well-suited to be stored as graph structures, allowing relationships between event constituents to be naturally captured. To use these graph structures for classification one needs to use a model the structure naturally fits (such as a graph neural network) or be able to embed the structure (for example, into a kernel or vector).

Continuous-variable (CV) quantum computing, through the use of Gaussian Boson Sampling, provides a method of embedding a graph into a lower-dimensional representation. CV quantum computers differ from discrete, qubit, models of quantum computing by their reliance on infinite-dimensional qumodes. These devices offer unique quantum advantages, one of which is the ability to perform boson sampling in a computationally efficient manner. Boson sampling creates samples from detecting photons after they travel through an interferometer. Simulating this, due to the matrix operations involved, is a #P problem and therefore a classically hard task.

We propose using Gaussian Boson Sampling to embed generated data into a feature vector and perform anomaly detection with it. Results using GBS to generate features are comparable to when the feature vectors have been classically generated. We hope under a more rigorous comparison this result will remain. Another quantum advantage can be gained from the use of Q-means clustering. We find performing anomaly detection using Q-means clustering gives equivalent results to K-means but has the potential to scale to large feature vectors more efficiently. The variant of Q-means presented here has a $\mathcal{O}(\log(\mathrm{N}))$ complexity, with respect to the size of the feature vector. As devices become larger, and we are able to expand the size of the vectors generated by a GBS device, the advantage of Q-means will become more pronounced. This is compared to the K-means complexity of $\mathcal{O}(N)$. While Q-means is implemented here in the discrete qubit-based model it should be possible to expand this to the CV paradigm.

JHEP08(2021)170

A SwapTest

A SwapTest circuit provides a method of checking the overlap between two states. Following [40], we can construct the circuit from Hadamard and CSWAP gates. We will use the information on the overlap of the states in our implementation of a quantum K-means routine. To understand how SwapTest operates we begin by defining a state

$$|\psi\rangle = |0, \alpha, \beta\rangle \,. \tag{A.1}$$

The state ψ contains the two states we wish to measure (α and β) which can contain multiple qubits and a control qubit. This initial state is evolved by applying a Hadamard gate to the control bit, resulting in the new state

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0, \alpha, \beta\rangle + |1, \alpha, \beta\rangle). \tag{A.2}$$

The next gate applied is CSWAP. The controlled SWAP gate will swap qubits based on the value of the control qubit. The application of a CSWAP will therefore evolve our state to

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}(|0, \alpha, b\rangle + |1, b, \alpha\rangle). \tag{A.3}$$

Finally, another Hadamard gate is applied to the control qubit:

$$|\psi_3\rangle = \frac{1}{2}\,|0\rangle\,(|\alpha, \beta\rangle + |\beta, \alpha\rangle) + \frac{1}{2}\,|1\rangle\,(|\alpha, \beta\rangle - |\beta, \alpha\rangle). \tag{A.4}$$

The probability of measuring the state $|0\rangle$ will give us a measure of the overlap.

$$\begin{aligned} P(|0\rangle) &= \left|\frac{1}{2}\,\langle 0|0\rangle\,(|\alpha, \beta\rangle + |\beta, \alpha\rangle) + \frac{1}{2}\,\langle 0|1\rangle\,(|\alpha, \beta\rangle - |\beta, \alpha\rangle)\right|^2 \\ &= \frac{1}{4}\,|(|\alpha, \beta\rangle + |\beta, \alpha\rangle)|^2 \\ &= \frac{1}{4}(\langle\beta|\beta\rangle\,\langle\alpha|\alpha\rangle + \langle\beta|\alpha\rangle\,\langle\alpha|\beta\rangle + \langle\alpha|\beta\rangle\,\langle\beta|\alpha\rangle + \langle\alpha|\alpha\rangle\,\langle\beta|\beta\rangle) \\ &= \frac{1}{2} + \frac{1}{2}\,|\langle\alpha|\beta\rangle|^2 \end{aligned} \tag{A.5}$$

If states $|\alpha\rangle$ and $|\beta\rangle$ are identical then $P(|0\rangle)$ will equal 0, if they are orthogonal than $P(|0\rangle) = 0.5$

References

[1] T.S. Roy and A.H. Vijay, *A robust anomaly finder based on autoencoders*, arXiv:1903.02032 [INSPIRE].

[2] A. Blance, M. Spannowsky and P. Waite, *Adversarially-trained autoencoders for robust unsupervised new physics searches*, *JHEP* **10** (2019) 047 [arXiv:1905.10384] [INSPIRE].

JHEP08(2021)170

[3] V. Mikuni and F. Canelli, *Unsupervised clustering for collider physics*, *Phys. Rev. D* **103** (2021) 092007 [arXiv:2010.07106] [INSPIRE].

[4] M. Abdughani, J. Ren, L. Wu and J.M. Yang, *Probing stop pair production at the LHC with graph neural networks*, *JHEP* **08** (2019) 055 [arXiv:1807.09088] [INSPIRE].

[5] J. Arjona Martínez, O. Cerri, M. Pierini, M. Spiropulu and J.-R. Vlimant, *Pileup mitigation at the Large Hadron Collider with graph neural networks*, *Eur. Phys. J. Plus* **134** (2019) 333 [arXiv:1810.07988] [INSPIRE].

[6] J. Shlomi, P. Battaglia and J.-R. Vlimant, *Graph neural networks in particle physics*, *Mach. Learn. Sci. Tech.* **2** (2021) 021001.

[7] P.W. Shor, *Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer*, *SIAM J. Sci. Statist. Comput.* **26** (1997) 1484 [quant-ph/9508027] [INSPIRE].

[8] L.K. Grover, *A Fast quantum mechanical algorithm for database search*, quant-ph/9605043.

[9] S. Abel, N. Chancellor and M. Spannowsky, *Quantum computing for quantum tunneling*, *Phys. Rev. D* **103** (2021) 016008 [arXiv:2003.07374] [INSPIRE].

[10] S. Abel and M. Spannowsky, *Observing the fate of the false vacuum with a quantum laboratory*, *P. R. X. Quantum.* **2** (2021) 010349 [arXiv:2006.06003] [INSPIRE].

[11] K.L. Ng, B. Opanchuk, M. Thenabadu, M. Reid and P.D. Drummond, *The fate of the false vacuum: Finite temperature, entropy and topological phase in quantum simulations of the early universe*, *P. R. X. Quantum.* **2** (2021) 010350 [arXiv:2010.08665] [INSPIRE].

[12] A. Mott, J. Job, J.R. Vlimant, D. Lidar and M. Spiropulu, *Solving a Higgs optimization problem with quantum annealing for machine learning*, *Nature* **550** (2017) 375 [INSPIRE].

[13] A. Blance and M. Spannowsky, *Quantum Machine Learning for Particle Physics using a Variational Quantum Classifier*, arXiv:2010.07335 [INSPIRE].

[14] S.P. Jordan, K.S.M. Lee and J. Preskill, *Quantum Computation of Scattering in Scalar Quantum Field Theories*, *Quant. Inf. Comput.* **14** (2014) 1014 [arXiv:1112.4833] [INSPIRE].

[15] L. García-Álvarez et al., *Fermion-Fermion Scattering in Quantum Field Theory with Superconducting Circuits*, *Phys. Rev. Lett.* **114** (2015) 070502 [arXiv:1404.2868] [INSPIRE].

[16] S.P. Jordan, K.S.M. Lee and J. Preskill, *Quantum Algorithms for Fermionic Quantum Field Theories*, arXiv:1404.7115 [INSPIRE].

[17] S.P. Jordan, H. Krovi, K.S.M. Lee and J. Preskill, *BQP-completeness of Scattering in Scalar Quantum Field Theory*, *Quantum* **2** (2018) 44 [arXiv:1703.00454] [INSPIRE].

[18] J. Preskill, *Simulating quantum field theory with a quantum computer*, *PoS* **LATTICE2018** (2018) 024 [arXiv:1811.10085] [INSPIRE].

[19] A.H. Moosavian, J.R. Garrison and S.P. Jordan, *Site-by-site quantum state preparation algorithm for preparing vacua of fermionic lattice field theories*, arXiv:1911.03505 [INSPIRE].

[20] NuQS collaboration, *σ Models on Quantum Computers*, *Phys. Rev. Lett.* **123** (2019) 090501 [arXiv:1903.06577] [INSPIRE].

[21] NuQS collaboration, *Gluon Field Digitization for Quantum Computers*, *Phys. Rev. D* **100** (2019) 114501 [arXiv:1906.11213] [INSPIRE].

JHEP08(2021)170

[22] NuQS collaboration, *Parton physics on a quantum computer*, *Phys. Rev. Res.* **2** (2020) 013272 [arXiv:1908.10439] [INSPIRE].

[23] NuQS collaboration, *Suppressing Coherent Gauge Drift in Quantum Simulations*, arXiv:2005.12688 [INSPIRE].

[24] I. Márquez-Mártin, P. Arnault, G. Di Molfetta and A. Pérez, *Electromagnetic lattice gauge invariance in two-dimensional discrete-time quantum walks*, *Phys. Rev. A* **98** (2018) 032333 [arXiv:1808.04488] [INSPIRE].

[25] P. Arrighi, G. Di Molfetta, I. Márquez-Martín and A. Pérez, *Dirac equation as a quantum walk over the honeycomb and triangular lattices*, *Phys. Rev. A* **97** (2018) 062111 [arXiv:1803.01015] [INSPIRE].

[26] G. Jay, F. Debbasch and J.B. Wang, *Dirac quantum walks on triangular and honeycomb lattices*, *Phys. Rev. A* **99** (2019) 032113 [arXiv:1803.01304] [INSPIRE].

[27] G. Di Molfetta and P. Arrighi, *A quantum walk with both a continuous-time and a continuous-spacetime limit*, arXiv:1906.04483 [INSPIRE].

[28] H. Lamm and S. Lawrence, *Simulation of Nonequilibrium Dynamics on a Quantum Computer*, *Phys. Rev. Lett.* **121** (2018) 170501 [arXiv:1806.06649] [INSPIRE].

[29] NuQS collaboration, *Quantum Simulation of Field Theories Without State Preparation*, arXiv:2001.11490 [INSPIRE].

[30] A.Y. Wei, P. Naik, A.W. Harrow and J. Thaler, *Quantum Algorithms for Jet Clustering*, *Phys. Rev. D* **101** (2020) 094015 [arXiv:1908.08949] [INSPIRE].

[31] K.T. Matchev, P. Shyamsundar and J. Smolinsky, *A quantum algorithm for model independent searches for new physics*, arXiv:2003.02181 [INSPIRE].

[32] S. Lloyd and S.L. Braunstein, *Quantum computation over continuous variables*, *Phys. Rev. Lett.* **82** (1999) 1784 [quant-ph/9810082] [INSPIRE].

[33] T.R. Bromley et al., *Applications of near-term photonic quantum computers: software and algorithms*, *Quantum Sci. Technol.* **5** (2020) 034010.

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

[35] S. Aaronson and A. Arkhipov, *The computational complexity of linear optics*, arXiv:1011.3245.

[36] C.S. Hamilton, R. Kruse, L. Sansoni, S. Barkhofen, C. Silberhorn and I. Jex, *Gaussian boson sampling*, *Phys. Rev. Lett.* **119** (2017) 170501.

[37] M. Schuld, K. Brádler, R. Israel, D. Su and B. Gupt, *Measuring the similarity of graphs with a gaussian boson sampler*, *Phys. Rev. A* **101** (2020) 032314.

[38] L. Petit et al., *Universal quantum logic in hot silicon qubits*, *Nature* **580** (2020) 355.

[39] S. Lloyd, M. Mohseni and P. Rebentrost, *Quantum algorithms for supervised and unsupervised machine learning*, arXiv:1307.0411.

[40] D. Kopczyk, *Quantum machine learning for data scientists*, arXiv:1804.10068.

[41] A. Falkowski, D. Krohn, L.-T. Wang, J. Shelton and A. Thalapillil, *Unburied Higgs boson: Jet substructure techniques for searching for Higgs' decay into gluons*, *Phys. Rev. D* **84** (2011) 074022 [arXiv:1006.1650] [INSPIRE].

JHEP08(2021)170

[42] C.-R. Chen, M.M. Nojiri and W. Sreethawong, *Search for the Elusive Higgs Boson Using Jet Structure at LHC*, *JHEP* **11** (2010) 012 [arXiv:1006.1151] [INSPIRE].

[43] ATLAS collaboration, *Search for Higgs boson decays into two spin-0 particles in the bbμμ final state with the ATLAS detector in pp collisions at* $\sqrt{s} = 13$ *TeV*, ATLAS-CONF-2021-009 (2021).

[44] T. Sjöstrand et al., *An introduction to PYTHIA 8.2*, *Comput. Phys. Commun.* **191** (2015) 159 [arXiv:1410.3012] [INSPIRE].

[45] J.M. Butterworth, A.R. Davison, M. Rubin and G.P. Salam, *Jet substructure as a new Higgs search channel at the LHC*, *AIP Conf. Proc.* **1078** (2009) 189 [arXiv:0809.2530] [INSPIRE].

[46] D.E. Soper and M. Spannowsky, *Combining subjet algorithms to enhance ZH detection at the LHC*, *JHEP* **08** (2010) 029 [arXiv:1005.0417] [INSPIRE].

[47] S. Marzani, G. Soyez and M. Spannowsky, *Looking inside jets: an introduction to jet substructure and boosted-object phenomenology*, vol. 958, Springer (2019) [DOI] [arXiv:1901.10342] [INSPIRE].

[48] D.E. Soper and M. Spannowsky, *Finding top quarks with shower deconstruction*, *Phys. Rev. D* **87** (2013) 054012 [arXiv:1211.3140] [INSPIRE].

[49] M. Cacciari, G.P. Salam and G. Soyez, *FastJet User Manual*, *Eur. Phys. J. C* **72** (2012) 1896 [arXiv:1111.6097] [INSPIRE].

[50] Y.L. Dokshitzer, G.D. Leder, S. Moretti and B.R. Webber, *Better jet clustering algorithms*, *JHEP* **08** (1997) 001 [hep-ph/9707323] [INSPIRE].

[51] M. Cacciari, G.P. Salam and G. Soyez, *The anti-k_t jet clustering algorithm*, *JHEP* **04** (2008) 063 [arXiv:0802.1189] [INSPIRE].

[52] T.N. Kipf and M. Welling, *Semi-Supervised Classification with Graph Convolutional Networks*, arXiv:1609.02907 [INSPIRE].

[53] F. Pedregosa et al., *Scikit-learn: Machine learning in Python*, *J. Mach. Learn. Res.* **12** (2011) 2825.

[54] N. de Lara and E. Pineau, *A simple baseline algorithm for graph classification*, arXiv:1810.09155.

[55] S. Lloyd, *Least squares quantization in pcm*, *IEEE Trans. Inform. Theory* **28** (1982) 129.

[56] M.E. Celebi, H.A. Kingravi and P.A. Vela, *A comparative study of efficient initialization methods for the k-means clustering algorithm*, *Expert Syst. Appl.* **40** (2013) 200.

[57] D.J. Brod, E.F. Galvão, A. Crespi, R. Osellame, N. Spagnolo and F. Sciarrino, *Photonic implementation of boson sampling: a review*, *Adv. Photonics* **1** (2019) 034001.

[58] S.L. Braunstein and P. van Loock, *Quantum information with continuous variables*, *Rev. Mod. Phys.* **77** (2005) 513 [quant-ph/0410100] [INSPIRE].

[59] N. Killoran, J. Izaac, N. Quesada, V. Bergholm, M. Amy and C. Weedbrook, *Strawberry fields: A software platform for photonic quantum computing*, *Quantum* **3** (2019) 129.

[60] K. Brádler, P.-L. Dallaire-Demers, P. Rebentrost, D. Su and C. Weedbrook, *Gaussian boson sampling for perfect matchings of arbitrary graphs*, *Phys. Rev. A* **98** (2018) 032310.

[61] L. Valiant, *The complexity of computing the permanent*, *Theor. Comput. Sci.* **8** (1979) 189.

JHEP08(2021)170

[62] W.R. Clements, P.C. Humphreys, B.J. Metcalf, W.S. Kolthammer and I.A. Walmsley, *Optimal design for universal multiport interferometers*, *Optica* **3** (2016) 1460.

[63] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn and K. Borgwardt, *Efficient graphlet kernels for large graph comparison*, in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, D. van Dyk and M. Welling, eds., vol. 5 of *Proc. Mach. Learn. Res.*, pp. 488–495, PMLR, Hilton Clearwater Beach Resort, Clearwater Beach, Florida, U.S.A., 16–18 April 2009.

[64] I. Kerenidis, J. Landman, A. Luongo and A. Prakash, *q-means: A quantum algorithm for unsupervised machine learning*, arXiv:1812.03584.

[65] J.C. Garcia-Escartin and P. Chamorro-Posada, *SWAP test and Hong-Ou-Mandel effect are equivalent*, *Phys. Rev. A* **87** (2013) 052330.

[66] V. Bergholm et al., *Pennylane: Automatic differentiation of hybrid quantum-classical computations*, arXiv:1811.04968.

JHEP08(2021)170