Searching for New Physics using Classical and Quantum Machine Learning

BLANCE, ANDREW,TULLOCH

Searching for New Physics using Classical and Quantum Machine Learning

Andrew Blance

A Thesis presented for the degree of
Doctor of Philosophy

Institute for Particle Physics Phenomenology
Department of Physics
Durham University
United Kingdom

December 2021

Searching for New Physics using Classical and Quantum Machine Learning

Andrew Blance

December 2021

Abstract: The development of machine learning (ML) has provided the High Energy Physics (HEP) community with new methods of analysing collider and Monte-Carlo generated data. As experiments are upgraded to generate an increasing number of events, classical techniques can be supplemented with ML to increase our ability to find signs of New Physics in the high-dimensional event data. This thesis presents three methods of performing supervised and unsupervised searches using novel ML methods.

The first depends on the use of an autoencoder to perform an unsupervised anomaly detection search. We demonstrate that this method allows you to carry out a data-driven, model-independent search for New Physics. Furthermore, we show that by extending the model with an adversary we can account for systematic errors that may arise from experiments. The second method develops a form of quantum machine learning to be applied to a supervised search. Using a variational quantum classifier (a neural network style model built from quantum information principles) we demonstrate a quantum advantage arises when compared to a classical network. Finally, we make use of the continuous-variable (CV) paradigm of quantum computing to build an unsupervised method of classifying events stored as graph data. Gaussian boson sampling provides an example of a quantum advantage unique to the CV method of quantum computing and allows our events to be used in an

anomaly detector model built using the Q-means clustering algorithm.

Contents

Declaration

The work in this thesis is based on research carried out in the Department of Physics at Durham University. No part of this thesis has been submitted elsewhere for any degree or qualification.

- Chapter 2 is based on A. Blance, M. Spannowsky and P. Waite, *Adversarially-trained autoencoders for robust unsupervised new physics searches*, *Journal of High Energy Physics* **2019** (Oct, 2019) , [1905.10384]

- Chapter 3 is based on A. Blance and M. Spannowsky, *Quantum machine learning for particle physics using a variational quantum classifier*, *Journal of High Energy Physics* **2021** (Feb, 2021) , [2010.07335]

- Chapter 4 is based on A. Blance and M. Spannowsky, *Unsupervised Event Classification with Graphs on Classical and Photonic Quantum Computers*, 2103.03897

During my studies I also contributed to :

- A. Blance, M. Chala, M. Ramos and M. Spannowsky, *Novel b -decay signatures of light scalars at high energy facilities*, *Physical Review D* **100** (Dec, 2019) , [1907.13151]

- S. Abel, A. Blance and M. Spannowsky, *Quantum Optimisation of Complex Systems with a Quantum Annealer*, 2105.13945

Acknowledgements

First, I would like to thank my supervisor Michael Spannowsky. Over the last four years Michael has provided fantastic and invaluable guidance in all of our projects together. His insights and ideas have taught me an enormous amount, and without him this work would not have been possible.

The IPPP has provided a brilliant place to carry out my research over the course of my PhD. There are far too many people to thank individually, but everyone in the community has always been so friendly and welcoming and are all so important in making the experience as good as it has been. I will give a special mention though to Parisa, Dorian and Joe, the other three people who occupied OC321. Our office's wine time, book club and large plant collection were all PhD highlights.

I'd also like to thank Dorian, Elliott, Hitham, Jack, Jonny, Joe Aylett-Bullock, Joe Walker, Khadeejah, Lucy, Mia, Parisa and Ryan who generously offered to proofread this thesis, and whose comments were very helpful in making this document as readable as it is.

During the last 3 years I have had two flatmates/bandmates, Balthazar (who was perfect) and Elliott (who was great, but compared to Balthazar was only ok tbh), thank you both. I would also like to thank my girlfriend Parisa for her love and encouragement!

Finally, thank you to my family (mam and dad and Catherine and Matthew) for all their support over the entire time I have been in academia.

Chapter 1

Introduction

Machine learning methods have become a key part of the high energy physics data analysis pipeline. From the use of binary decision trees (BDTs) at trigger level at the Large Hadron Collider (LHC) or using neural networks to help analyse and classify event data, machine learning is widely used. A natural property of particle physics data - namely, the high number of dimensions that are seen when analysing collider (or Monte-Carlo generated) events - means machine learning methods have the potential of being useful in the field. As the dimensions of a dataset grow many classical methods become less viable to use. In contrast, a well trained neural network can be used to find correlations in high-dimensional data with high accuracy. While many of the methods discussed in this thesis focus on this complexity (for example, how quantum machine learning may provide better optimisations on datasets with large number of features), it is worth noting that machine learning models can benefit from being trained on large datasets. As the high luminosity upgrade will increase the amount of data the LHC can gather the value of machine learning methods are likely to grow.

This thesis will discuss three applications of machine learning, on both classical and quantum hardware. We aim to introduce methods to help boost the sensitivity in searches of New Physics by showing how machine learning methods can be used to perform supervised, and unsupervised, searches.

In Chapter 2 we begin with a neural network architecture known as an autoencoder. Autoencoders can be trained to perform unsupervised anomaly detection. These types of searches are appealing for particle physics, as we would only need to train our models on background data. This, of course, is useful as if we were to perform a true data-driven search (i.e. using only real data) we cannot expect to be able to gather a training set of beyond the Standard Model events. However, performing a search using only LHC data will introduce the problem of the classifier becoming biased by any systematic experimental error. We address this here by extending the autoencoder model with an adversary. We show that by doing this we can create an anomaly detection method that is robust against experimental error.

To look for novel methods of applying machine learning we can turn to quantum computing. Typically, quantum computers use what is known as a qubit as its fundamental object. Qubits exist in either state $|0\rangle$, $|1\rangle$, or a linear superposition of both. In a classical computer, which uses a binary bit, the state can only ever be 0 or 1. By applying operations onto the qubit its state will evolve and one can then measure it. Powerful algorithms can be embedded into a quantum computer and be shown to provide a large advantage over classical machines. Quantum machine learning is therefore an emerging field aiming to apply quantum advantages to machine learning methods. This is the topic of Chapter 3 and 4. Firstly, in Chapter 3 we introduce the concept of a variational quantum classifier. These models are neural network style structures built using quantum information theory concepts.

While the predominant method of quantum computing is the qubit-based paradigm we explore another method in Chapter 4. The continuous-variable (CV) paradigm differs by using a qumode as its fundamental object. Qumodes are infinite-dimensional objects, implemented using quantum photonics hardware. We use Gaussian boson sampling, a process with a quantum advantage unique to CV quantum computing, to help build an anomaly detection method.

First, though, we will discuss the Standard Model of particle physics, perhaps the most successful theory in Physics. The Standard Model describes the fundamental

u	c	t	γ	h
d	s	b	g	
ν_e	ν_μ	ν_τ	W	
e	μ	τ	Z	

Table 1.1: The Standard model of particle physics. In the top and bottom left of the table are the three generations of quarks and leptons. These make up the fermionic matter of the universe. To the right of those are the gauge bosons. These mediate the interactions. The top right contains the Higgs boson.

building blocks of nature and how they interact. Following this, we discuss collider phenomenology and some of the observables used to describe final states in event data. We then introduce some applications of machine learning in HEP and give some insight into the construction and theory of neural networks. This chapter then ends with a brief introduction to the qubit model of quantum computing.

1.1 The Standard Model

The Standard Model of particle physics (SM) is our current best understanding of the universe. The model details the most fundamental constituents of nature and their interactions. These consist of fermions, the matter particles of the universe, and bosons, the particles which mediate their interactions. The model has remarkably predictive powers and is a testament to the development of fundamental physics over the last century. Even with the enormous energy scales that are reached with modern colliders we find that the Standard Model predictions, for the most part, remain consistent with experimental results.

Quantum field theories (QFTs) are a framework that allows one to combine the ideas of quantum mechanics and special relativity. In this framework, particles are seen as excitations in a quantum field. The Standard Model is itself a QFT, with a particle content shown in Table 1.1, and belonging to the gauge group

$$SU(3)_c \times SU(2)_L \times U(1)_Y. \tag{1.1.1}$$

This group contains two significant parts: the quantum chromodynamic (QCD) and the electroweak (EW) sectors. The group $SU(3)_c$ is the gauge group of QCD, with c referring to the colour charge [6–8]. QCD is the theory that describes the strong force. This is mediated by the 8 gluons, which interact based on an objects colour. Meanwhile, $SU(2)_L \times U(1)_Y$ describes the electroweak sector [9–11]. Here, the quantum number Y represents the Hypercharge and L informs us that only left-handed objects couple in the $SU(2)$ field interactions. This group represents the unification of the weak and electromagnetic forces. This leads to the production of the $W^{\pm}$ and Z^0 vector bosons, the mediators of this force. The $SU(2)_L \times U(1)_Y$ electroweak interaction, via the Higgs mechanism, is spontaneously broken [12–14]. The group breaks down to a $U(1)$ group describing quantum electrodynamics (QED) [15–18]. QED is the field theory that describes the interaction between matter and photons.

We note that $U(1)_Y$ is an abelian group, while $SU(3)_c$ and $SU(2)_L$ are not. Instead, they are non-commuting and follow the Lie algebra

$$[t^a, t^b] = if^{abc}t^c. \tag{1.1.2}$$

Here, t is some generator and f^{abc} is a structure constant. What follows from this, is that gauge theories derived from the $SU(3)_c$ and $SU(2)_L$ groups will not commute either. This results in the self-interaction of the bosons associated with these groups.

The dynamics of a QFT can be described by its Lagrangian density $\mathcal{L}$. The Standard Model Lagrangian will therefore describe the interactions within the model and can be written as [19–21]

$$\begin{aligned}\mathcal{L}_{SM} &= -\frac{1}{4}F^{\mu\nu}F_{\mu\nu} \\ &+ i\bar{\Psi}\not{D}\Psi \\ &+ |D_\mu H|^2 - V(H) \\ &- y_{ij}\bar{\Psi}_i H \Psi_j + \text{h.c.}\end{aligned} \tag{1.1.3}$$

Here, $F^{\mu\nu}$ is the gauge field strength tensor, Ψ is the spinor (a two component vector-like quantity) of a matter field, H is the Higgs doublet and y_{ij} are the Yukawa couplings. Alongside these are $\not{D}$, the covariant derivative, and $V(H)$, the Higgs potential. The rest of this section will be devoted to the further discussion of all these objects. To do this, we will treat each line of Eq. (1.1.3) individually, thus splitting the Lagrangian into four distinct parts - $\mathcal{L}_{Gauge}$, $\mathcal{L}_{Fermion}$, $\mathcal{L}_{Higgs}$ and $\mathcal{L}_{Yukawa}$.

1.1.1 The Gauge Terms

The first part of the Lagrangian we will investigate is responsible for the dynamics of the gauge fields. It will be used to describe the self-interactions in the gauge bosons used in the strong and electroweak force. The terms for these fields are

$$\begin{aligned}\mathcal{L}_{Gauge} &= -\frac{1}{4}F^{\mu\nu}F_{\mu\nu} \\ &= -\frac{1}{4}G^A_{\mu\nu}G^{A,\mu\nu} - \frac{1}{4}W^I_{\mu\nu}W^{I,\mu\nu} - \frac{1}{4}B^{\mu\nu}B_{\mu\nu},\end{aligned} \tag{1.1.4}$$

where $G^A_{\mu\nu}$, $W^I_{\mu\nu}$ and $B^{\mu\nu}$ are field strength tensors. These fields correspond to the $SU(3)$, $SU(2)$ and $U(1)$ groups, respectively. For a $SU(N)$ group, the indices can run from 1 up to $N^2 - 1$. From this, we can see that the $G^A_{\mu\nu}$, belonging to $SU(3)$, runs up to 8 and thus accounts for the 8 different gluons of the strong force. The $SU(2)$ group field $W^I_{\mu\nu}$ runs up to 3, giving us three bosons W^1, W^2 and W^3. We will see in Section 1.1.3 that W^1 and W^2 can mix to give $W^\pm$ while W^3 can mix

Field	$SU(3)_c$	$SU(2)_L$	$U(1)_Y$
G_μ^A	**8**	**1**	0
W_μ^I	**1**	**3**	0
B^μ	**1**	**1**	0

Table 1.2: How the Gauge field are represented in the SM. Where **1** indicates it is a trivial representation for the $SU(N)$ groups. A value aside from **1** represents the fundamental representation of the group.

with the $U(1)$ boson B to give Z^0, A. The full gauge group representations for the field strength tensors are shown in Table 1.2. The three tensors can are used to define the field strength tensors

$$\begin{aligned} G_{\mu\nu}^A &= \partial_\mu G_\nu^A - \partial_\nu G_\mu^A + g_s f^{ABC} G_\mu^B G_\nu^C, \\ W_{\mu\nu}^I &= \partial_\mu W_\nu^I - \partial_\nu W_\mu^I + g\epsilon^{IJK} W_\mu^J W_\nu^K, \\ B_{\mu\nu} &= \partial_\mu B_\nu - \partial_\nu B_\mu. \end{aligned} \tag{1.1.5}$$

For $SU(3)_c$ and $SU(2)_L$ there are associated structure constants, f^{ABC} and the Levi-Civita tensor ϵ^{IJK}. The fields also have two parameters, g and g_s, which are coupling constants. An important distinction between B_μ and the other fields is the lack of terms resembling $B_\mu B^\mu$. The result is that, unlike G_μ and W_μ that contain $g_s f^{ABC} G_\mu^B G_\nu^C$ and $g\epsilon^{IJK} W_\mu^J W_\nu^K$, there will be no self-interaction between the bosons associated with a $U(1)_Y$ group. If we perform an infinitesimal gauge transform on these fields they transform as

$$\begin{aligned} G_\mu^A &\to G_\mu^A - \frac{1}{g_s}\partial_\mu \alpha^A + \epsilon^{ABC}\alpha^B G_\mu^C, \\ W_\mu^I &\to W_\mu^I - \frac{1}{g}\partial_\mu \alpha^I + \epsilon^{IJK}\alpha^J W_\mu^K, \\ B_\mu &\to B_\mu - \frac{1}{g'}\partial_\mu \alpha. \end{aligned} \tag{1.1.6}$$

In the final line of Eq. (1.1.6) the coupling constant of the $U(1)_Y$ gauge group is introduced as g'. In all three cases, α acts as a term that parameterises the transformations. The fields transforming in this manner ensures the Standard Model lagrangian has a local symmetry. For all three lines of Eq. (1.1.6) we can propose a term like $G^A_\mu G^{A\mu}$, $W^I_\mu W^{I\mu}$ or $B_\mu B^\mu$. Each of these however is not invariant. The consequence of this is that a mass term resembling $m^2 B_\mu B^\mu$ would be unwelcome in the Standard Model Lagrangian. Unless we introduce a method of breaking the symmetry this forces the gauge bosons to have a mass of zero.

1.1.2 The Fermionic Terms

We now consider the second line of Eq. (1.1.3). This considers the interactions of the fermions, the spin$-\frac{1}{2}$ matter fields of the Standard Model. These fields all couple to the gauge fields described in Section 1.1.1 through the covariant derivative

$$D_\mu = \partial_\mu - ig'YB_\mu - ig\frac{\tau^I}{2}W^I_\mu - ig_s\frac{T^A}{2}G^A_\mu. \tag{1.1.7}$$

This covariant derivative depends on the three coupling constants g', g and g_S and a set of generators. These are Y for the $U(1)$ field, τ^I (proportional to the Pauli matrices) for the $SU(2)_L$ group and T^A for the $SU(3)_c$ group. Depending on the field the covariant derivative is acting on, its definition will change. For example, if applied to an $SU(2)_L$ field, the derivative would only contain the relevant terms.

The fermion fields can be described as spinors, Ψ. As with the bosons in the previous section, we can define these fields through their representation under $SU(3)_c$, $SU(2)_L$ and $U(1)_Y$. Fermion fields are further split into three generations, each with the same set of charges but differing amounts of mass and mixing values (discussed in Section 1.1.4). Furthermore, fermions can be categorised as either quarks or leptons. Looking only at the first generation we can consider q_L, a $SU(2)_L$ doublet, that contains the left-handed quark fields

$$q_L = \begin{pmatrix} u_L \\ d_L \end{pmatrix}. \tag{1.1.8}$$

Field	$SU(3)_c$	$SU(2)_L$	$U(1)_Y$
q_L	$\mathbf{3}$	$\mathbf{2}$	$\frac{1}{3}$
u_R	$\mathbf{3}$	$\mathbf{1}$	$\frac{4}{3}$
d_R	$\mathbf{3}$	$\mathbf{1}$	$-\frac{2}{3}$
l_L	$\mathbf{1}$	$\mathbf{2}$	-1
e_R	$\mathbf{1}$	$\mathbf{1}$	-2

Table 1.3: How the fermion fields are represented in the Standard Model. Where **1** indicates it is a trivial representation for the $SU(N)$ groups. A value aside from **1** represents the fundamental representation of the group.

Individual quarks (such as u_L) exist as a triplet representation of $SU(3)_c$. These fields are represented as

$$u_L = \begin{pmatrix} u_L^r \\ u_L^b \\ u_L^g \end{pmatrix}, \tag{1.1.9}$$

where r, b and g are the three QCD colours. The left-handed lepton field doublet contains a generation's charged lepton and neutrino

$$l_L = \begin{pmatrix} \nu_L \\ e_L \end{pmatrix}. \tag{1.1.10}$$

Unlike the quarks, individual leptons have a trivial representation in $SU(3)_c$ and therefore have no colour charge. Another fundamental feature of the lepton sector is the difference between how left and right-handed objects are treated. Left-handed leptons exist as a $SU(2)_L$ doublet. This is contrasted by right-handed particles where, u_R, d_R and e_R are only singlets and transform under a trivial representation in $SU(2)_L$. Note, there is no right-handed neutrino in our model.

All these fields can then also be described by how they are represented under $SU(3)_c$, $SU(2)_L$ and $U(1)_Y$. Table 1.3 contains the quantum numbers for the left and right-handed objects discussed. Using these fields, we can define a term for the Lagrangian

$$\begin{aligned}\mathcal{L}_{Fermion} \supset \bar{q}_L i\gamma^\mu D_\mu q_L + \bar{u}_R i\gamma^\mu D_\mu u_R + \bar{d}_R i\gamma^\mu D_\mu d_R \\ + \bar{l}_L i\gamma^\mu D_\mu l_L + \bar{e}_R i\gamma^\mu D_\mu e_R.\end{aligned} \tag{1.1.11}$$

Other Lagrangian terms will exist, so that all three generations are included. We can see then that the Standard Model really treats left and right handed objects differently. Right-handed leptons transform differently to left-handed leptons. For a field ψ we can project out the handedness of the field using

$$\psi_{R,L} = P_{R,L}\psi, \tag{1.1.12}$$

where $P_{R,L}$ are the projection operators

$$P_R = \frac{1}{2}(1+\gamma^5), \tag{1.1.13}$$

and

$$P_L = \frac{1}{2}(1-\gamma^5). \tag{1.1.14}$$

Here, L and R indicate the handedness of the field, and $\gamma^5 = i\gamma^0\gamma^1\gamma^2\gamma^3$ (which each γ being one of the gamma matrices). Using these we can derive the mass term we would expect to find in the Standard Model Lagrangian. However, this term will rely on fields of both chiralities, such that

$$-m\bar{\Psi}\Psi = -m\bar{\Psi}_R\Psi_L - m\bar{\Psi}_L\Psi_R. \tag{1.1.15}$$

As left-handed fermions contain different charges to their right-handed equivalents these terms will not be gauge invariant. This means we cannot add explicit mass terms related to fermions into the Lagrangian. Similar to what we saw in the gauge sector, if we cannot find a way to break this symmetry, fermions in the Standard Model will be massless.

Field	$SU(3)_c$	$SU(2)_L$	$U(1)_Y$
H	**1**	**2**	1

Table 1.4: How the Higgs is represented in the Standard Model. Where **1** indicates it is a trivial representation for the $SU(N)$ groups. A value aside from **1** represents the fundamental representation of the group.

1.1.3 The Higgs Terms

As we have seen in Sections 1.1.1 and 1.1.2 it is difficult to add mass terms for the fermions and gauge bosons into the Standard Model Lagrangian. To do this we will begin by introducing the Higgs boson [12–14]. The Standard Model Lagrangian has terms (shown in line 3 of Eq. (1.1.3)) that detail the self-interactions of the Higgs and its interaction with the gauge bosons. The inclusion of these terms, as we will see, introduces spontaneous symmetry breaking. This is the method we will use to generate masses for the gauge bosons and fermions.

The Higgs field can be described by the $SU(2)$ complex doublet

$$H = \begin{pmatrix} \phi^+ \\ \phi^0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \phi_1 + i\phi_2 \\ \phi_3 + i\phi_4 \end{pmatrix}. \tag{1.1.16}$$

The doublet includes four degrees of freedom, described by the four real scalar fields ϕ_1, ϕ_2, ϕ_3 and ϕ_4. The full gauge group representation is given in Table 1.4. The Higgs potential takes the form

$$V(H) = -\mu^2 H^\dagger H + \lambda (H^\dagger H)^2, \tag{1.1.17}$$

where λ is positive and describes the self-interactions of the field. For $\mu^2 < 0$ the potential has a minimum at zero. Values of μ^2 greater than zero lead to what is known as symmetry breaking. A symmetry is spontaneously broken if the Lagrangian has a symmetry not visible in the ground state. Here, we are about to break the $SU(2)_L \times U(1)_Y$ symmetry to $U(1)_{EM}$. In scenarios where $\mu^2 > 0$ the Higgs potential shifts to a non-zero minimum at $|H| = \sqrt{H^\dagger H}$. The potential is minimised when the field takes the form

$$\langle H \rangle = \frac{\nu}{\sqrt{2}} = \sqrt{\frac{\mu^2}{2\lambda}}, \tag{1.1.18}$$

where ν is known as the vacuum expectation value (VEV). From here we have a choice of which of the four fields from Eq. (1.1.16) will contain the VEV. If we wish to conserve electric charge a VEV can only be acquired from a neutral field. This means the VEV must be chosen as either ϕ_3, ϕ_4 or a mixture of the two, as they are electrically neutral. This leads to the traditional choice of

$$\phi_3 = v + h, \tag{1.1.19}$$

where h, a pertubation around the VEV, is a real scalar field that we interpret as the physical Higgs boson. Using this, the Higgs field can be written as

$$H = \begin{pmatrix} \phi^+ \\ \phi^0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \phi_1 + i\phi_2 \\ v + h + i\phi_4 \end{pmatrix}. \tag{1.1.20}$$

The three fields ϕ_1, ϕ_2 and ϕ_3 are massless Goldstone bosons, each with a VEV of 0. These can be gauged away [22], removing them from the Lagrangian, and leaving the Higgs scalar doublet to be written as

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ \nu + h \end{pmatrix}. \tag{1.1.21}$$

By considering a non-zero VEV we find the Lagrangian is invariant under the group $SU(2)_L \times U(1)_Y$. However, it will still conserve electromagnetic symmetry. Therefore, we can say the symmetry group is broken as $SU(2)_L \times U(1)_Y \rightarrow U(1)_{EM}$. The $U(1)$ group corresponds to electromagnetism, and means in the vacuum we still conserve the symmetry. As we will see, this will lead to the boson fields associated with the broken generators acquiring mass (the $SU(2)$ fields associated with τ), while the unbroken field (EM) does not.

The physical Higgs boson mass can be derived using Eq. (1.1.17) and Eq. (1.1.18) to be

$$m_h^2 = 2\lambda\nu^2 = -2\mu^2. \tag{1.1.22}$$

Experimentally, the mass of the Higgs has been found to be $m_h \sim 125$ GeV [23]. Aside from $V(H)$, the Higgs part of the Standard Model Lagrangian also contains a term $|D_\mu H|^2$. By investigating this we will see how the Higgs couples to the gauge bosons and the origins of their mass. Applied to the Higgs the covariant derivative takes the form

$$D_\mu = \partial_\mu - ig'YB_\mu - ig\frac{\tau^I}{2}W_\mu^I. \tag{1.1.23}$$

By applying this to Higgs doublet shown in Eq. (1.1.21) we arrive at

$$D_\mu H = \frac{1}{\sqrt{2}}\begin{pmatrix} -i\frac{g}{2}(W_\mu^1 - iW_\mu^2)(\nu + h) \\ \partial h + \frac{i}{2}(-g'B_\mu + gW_\mu^3)(\nu + h) \end{pmatrix}. \tag{1.1.24}$$

We can then see that the Lagrangian will contain terms that look like

$$\begin{aligned} |D_\mu H|^2 \supset &\frac{1}{8}g^2(\nu + h)^2(W_\mu^1 - iW_\mu^2)(W^{1\mu} + iW^{2\mu}) \\ &+ \frac{1}{8}(\nu + h)^2(-g'B_\mu + gW_\mu^3)^2. \end{aligned} \tag{1.1.25}$$

In the first term we note the combinations of W_μ^1 and W_μ^2. These are related to the physical $W^\pm$ bosons as

$$W_\mu^\pm = \frac{1}{\sqrt{2}}(W_\mu^1 \mp iW_\mu^2). \tag{1.1.26}$$

By reading from the Lagrangian (taking the terms associated with ν^2), we can define the mass of the W bosons as

$$m_W^2 = \frac{g^2\nu^2}{4}, \tag{1.1.27}$$

where m_W has been found to be 80.38 GeV [24]. Just as we determine $W^{\pm}$ to be a combination of W^1_μ and W^2_μ we can define Z_μ in a similar way. From the second term of Eq. (1.1.25) we take Z_μ to be

$$Z_\mu = \frac{-g' B_\mu + g W^3_\mu}{\sqrt{g^2 + g'^2}}. \tag{1.1.28}$$

Using this definition of Z_μ allows us to extract the mass from the Lagrangian as

$$m_Z^2 = \frac{(g^2 + g'^2)\nu^2}{4}, \tag{1.1.29}$$

which has been measured as 91.18 GeV [24]. To find the mass of A_μ we will briefly consider Z_μ as

$$Z_\mu = c_W W^3_\mu - s_W B_\mu, \tag{1.1.30}$$

where c_W and s_W are combinations of g and g' [1]. We then take A_μ to be

$$A_\mu = c_W W^3_\mu + s_W B_\mu, \tag{1.1.31}$$

the state orthogonal to Z_μ. This does not appear in Eq. (1.1.25) and does not get a coupling to the Higgs. Therefore, $m_A^2 = 0$ and we identify this boson as the photon.

In summary, the gauge bosons are typically unable to have mass terms in the Standard Model Lagrangian. Instead, they gain their mass through a coupling with the Higgs boson. The Higgs potential takes a non-zero VEV, leading to spontaneous symmetry breaking. This takes us from the $SU(2)_L \times U(1)_Y$ to $U(1)_{EM}$. As we have seen, this leads to the gauge bosons associated with the broken generators (those of $SU(2)$) to attain a mass, while the one associated with the unbroken generator remains massless.

[1] c_W and s_W are defined such that $c_W = \frac{g}{\sqrt{g^2+g'^2}}$ and $s_W = \frac{g'}{\sqrt{g^2+g'^2}}$

1.1.4 The Yukawa Terms

In the previous section, we were able to successfully give the gauge bosons mass. Now, we turn our attention to the fermions. Fermion mass is generated from the interactions within the $\mathcal{L}_{Yukawa}$ part of the Standard Model Lagrangian. This part of the Lagrangian includes terms like

$$\mathcal{L}_{Yukawa} \supset -(y_e)_{ij}\bar{l}^i H e^j - (y_u)_{ij}\bar{q}^i \tilde{H} u^j - (y_d)_{ij}\bar{q}^i H d^j + h.c., \tag{1.1.32}$$

where y_e, y_u and y_d are the Yukawa coupling factors and $\tilde{H} = i\sigma^2 \bar{H}$. Focusing in on a single fermion in the first generation we can find the generated mass terms. We begin with the up quark:

$$\mathcal{L}_{Yukawa} \supset -[(y_u)_{11}\bar{u}_R \tilde{H}^\dagger q_L + (y_u^*)_{11}\bar{q}_L \tilde{H} u_R] \tag{1.1.33}$$

Eq 1.1.33 is the Lagrangian term for the up fermions of the first generation. This is equivalent to

$$\mathcal{L}_{Yukawa} \supset -\frac{(y_u)_{11}\nu}{\sqrt{2}}\bar{u}u - \frac{(y_u)_{11}}{\sqrt{2}}h\bar{u}u, \tag{1.1.34}$$

leading us to mass terms such as

$$m_u = \frac{y_u \nu}{\sqrt{2}}. \tag{1.1.35}$$

A similar process allows us to find the down fermion and electron masses. Note, these terms give the neutrino no mass. As there are multiple generations of quark, the Yukawa coupling factors are in fact matrices.

Recall, $W^\pm$ and Z did not align with the gauge bosons from the original $\mathcal{L}_{Gauge}$ and were instead mixtures. Now, we must ensure the fermion states derived here can be related with what we saw in $\mathcal{L}_{Fermion}$. If we assume the fermions discussed above are not physical, we must find a way to mix them into the objects we observe. We will do this by first considering the Yukawa matrices. While we do not know the form these matrices may take, we will diagonalise them such that

$$\begin{aligned} M_e &= E_L^\dagger y_e E_R, \\ M_u &= U_L^\dagger y_u U_R, \\ M_d &= D_L^\dagger y_d D_R, \end{aligned} \tag{1.1.36}$$

where $E_{L,R}$, $U_{L,R}$ and $D_{L,R}$ are unitary matrices. By using the mass matrices in Eq. (1.1.36) a fermion can be rotated into the mass basis. These transformations will go like $e_L \to E_L e_L$, taking us from an unphysical basis to a physical one.

For a neutral Z interaction (where up and down fields are not mixed) these terms will cancel, such that $D_L^\dagger D_L = 1$. However, in a charged $W^\pm$ interaction it is possible to see terms that resemble $\bar{u}_L \gamma^\mu d_L$. When this is transformed to the mass we end up with terms like $\bar{u}_L \gamma^\mu (U_L^\dagger D_L) d_L$. Here, we could have an interaction between fermions of different families and generations. We can define a matrix

$$V = U_L^\dagger D_L. \tag{1.1.37}$$

This is known as the Cabibbo-Kobayashi-Maskawa (CKM) matrix. The CKM matrix allows us to parameterise how the states mix in these currents. The 3 × 3 matrix contains 9 complex numbers, resulting in 18 parameters. Since this is a unitary matrix, we are constrained to 9 parameters. Of these 9 free parameters, 6 are phases and 3 are angles. Then, by redefining the quark fields we can reduce the number of relative phases. This allows us to describe the CKM matrix with only 4 values - 3 rotations and a single phase.

1.1.5 Going Beyond the Standard Model

As we have seen, the Standard Model provides a successful description of nature. It explains the interactions of the strong, weak and electromagnetic forces and dynamics of our fundamental particles. However, while it is accurate to very small scales, the Standard Model does not explain all phenomena and has some theoretical problems. The physics that must exist to describe these phenomena make up a part of what is known as Beyond the Standard Model (BSM) physics. As we worked through the Standard Model Lagrangian we began to see where some problems may occur. Some of the hints that physics must exist beyond what we account for in the Standard Model are:

- From Eq. (1.1.32) we saw that fermion masses are generated from the Yukawa matrices and the following mass matrices. However, the construction of these required the fermions to have right-handed components. As neutrinos have no right-handed component it is implied they are massless. This is in contradictions to recent observations of neutrino oscillations, which suggest massive neutrinos [25].

- From cosmology measurements we can infer the existence of Dark Matter. This form of matter could make up to 85% of the matter of the universe but currently is not included in the Standard Model [26].

- To explain the nature of how the universe is expanding it is necessary to introduce a new form of energy. Dark Energy has been proposed for this purpose [27].

- A naive assumption would be that the amount of matter and anti-matter in the universe is equal. This does not align with observation. In fact, there is an asymmetry - most of the universe is made from matter [28] While there are some CP-violating processes in the Standard Model, these cannot explain the asymmetry seen.

- The Standard Model does not contain an explanation of gravity. There is no bosonic field to mediate the force, as you may expect. Attempts to add a "graviton" into the theory have not been successful. Einstein's theory of relativity still remains the best explanation of gravity.

This list is no by no means exhaustive and other problems exist. For example, the hierarchy problem relates to the large separation between the Higgs mass and the Planck scale. Then, the flavour problem relates to the large separation of masses between the fermions themselves and also the nature of constructing and using the Yukawa matrices. While this thesis does not attempt to provide solutions to these problems, it is important to understand that while the Standard Model is very successful, it does not provide a full explanation of our universe.

1.2 Collider Phenomenology

To test Standard Model predictions, and look for answers to BSM mysteries, we need a tool that is capable of probing the behaviour of particles. The tool we use for this purpose is a particle collider. Currently, the largest and most powerful one is the Large Hadron Collider (LHC). The LHC is a 27 km ring built beneath the border of France and Switzerland. Inside the ring, two beams of protons circulate in opposite directions at relativistic speeds. As they journey through the collider, thousands of magnets, super-cooled to around -271 °C, are used to guide them. The protons travel through the beams in bunches and can be made to collide at numerous points. Two of these points are inside the ATLAS [29] and CMS [30] detectors. These detectors are used to record the outcomes of the collisions. From here, the recorded data can be catalogued and sent across the world for analysis.

As two protons collide, their partons can interact, each carrying a fraction of their parent proton's energy. The initial objects can radiate energy in the form of gluon emissions - a form of radiation known as initial state radiation. The colliding partons can decay and undergo hadronisation. These hadronised objects can themselves then decay. By looking for collimated decay objects we can construct what are known as

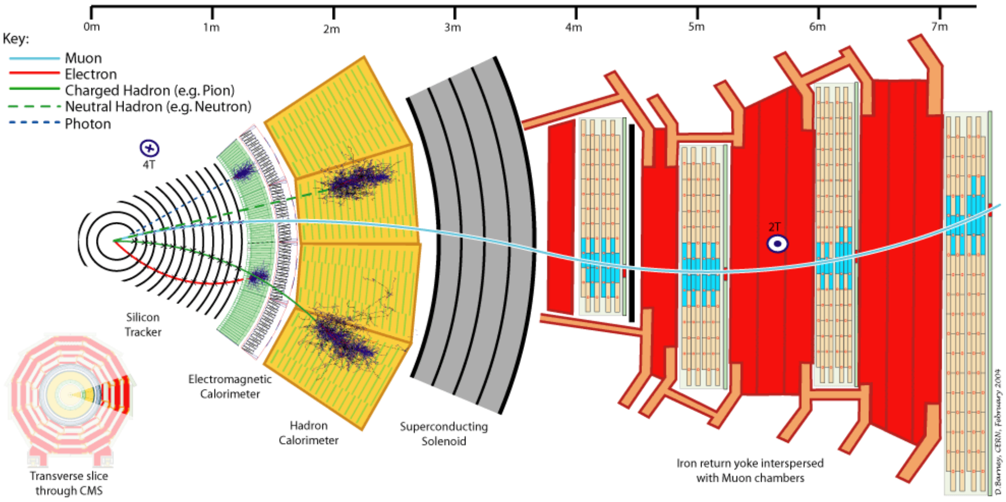

0m
1m
2m
3m
4m
5m
6m
7m
Key:
Muon
Electron
Charged Hadron (e.g. Pion)
Neutral Hadron (e.g. Neutron)
Photon
4T
2T
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
Iron return yoke interspersed with Muon chambers
Transverse slice through CMS
D.Barney, CERN, February 2004

Figure 1.1: A cross-section of the CMS detector system at the LHC. Taken from Ref. [31].

jets, which we can use to help us analyse the decay. By viewing the final states we can imperfectly reconstruct what happened in the collision. We hope to be able to separate any interesting events from Standard Model background.

To look at how these collisions are handled, we will focus on the CMS detector - a cross-section of which is shown in Fig. 1.1. The detector is made up of layers, each having a different purpose. The four layers are:

- (i) The Tracking Chamber. Here, the trajectory of a charged particle can be tracked. This section also houses the vertex detectors. This is of importance for b-tagging jets. A hadron containing a b quark has a relatively long lifespan before decaying. By tracking these decay products one can accurately tag a jet originating from a b quark. The tracking chamber is also useful for reducing Pile Up. Each bunch of protons will lead to multiple collisions, with each of these producing new particles. We need to be able to separate these events. This is what is known as Pile Up. Careful tracking of the vertices can help mitigate this.

- (ii) The Electromagnetic Calorimeter (ELAC). The ELAC measures the energy for electrons, positrons and photons

- (iii) The Hadron Calorimeter (HLAC). This layer is used to detect the high energy hadrons. These decay into further products, which can be clustered into jets.

- (iv) The Muon chambers. Muons pass through the first three layers and into the Muon chambers. These chambers, like the first layer, curve the path of the particle, from which we can infer its properties.

With information from the detector, we can build a picture of the event. Having access to observables such as a particle's energy and mass allows us to begin analysing the process. Another observable we note is the object's momentum. To define an invariant quantity, we choose a coordinate system such that the z-axis follows the beam direction. Using this, we can measure the momentum associated with the (x, y) plane. Thus, the transverse momentum p_T, should sum to zero for an event. Missing p_T is, therefore, an indicator that we have not accounted for something, such as an undetected neutrino or a new physics object.

There are a series of angles, measured from the beam axis, which are also of interest to us. Namely, the azimuthal angle ϕ and the polar angle θ. We can also define quantities known as rapidity and pseudorapidity:

$$\begin{aligned} y &= \frac{1}{2}\text{ln}\frac{E+p_z}{E-p_z}, \\ \eta &= -\text{ln } \tan\frac{\theta}{2}, \end{aligned} \tag{1.2.1}$$

where E is the energy and p_z is the momentum in the z direction. The rapidity y, for a massless particle, is equivalent to η. For the final state objects in the event, $y = \eta$ gives a good approximation. Using the pseudorapidity, we can construct a measure of the angular difference between two particles

$$\Delta R = \sqrt{\Delta\phi^2 + \Delta\eta^2}. \tag{1.2.2}$$

The amount of information we collect (i.e. the number of events generated) is dependent on the experiment. An important value associated with this is a collider's luminosity. Over time, at the LHC, it has been upgraded. There are two relavant values to consider: instantaneous luminosity and integrated luminosity. Instantaneous luminosity describes how many collisions can occur per unit time, over an area. For a rate of events being detected in a period of time dN/dt, we expect a luminosity

$$\mathcal{L} = \frac{1}{\sigma(s)}\frac{dN}{dt}, \tag{1.2.3}$$

where $\sigma(s)$ is the cross-section (the probability that an event will occur) at a given centre-of-mass energy s. By finding the integrated luminosity, one can find the total number of events that will be produced in the time t, for a cross-section $\sigma(s)$ [32]. We find

$$N = \int \mathcal{L}dt\ \sigma(s), \tag{1.2.4}$$

By increasing the luminosity of an experiment over its lifespan the amount of data it gathers also increases. Run 1 of the LHC operated at an integrated luminosity of 20 fb^{-1}, Run 2 increased this to 150 fb^{-1}, and Run 3 will operate at 300 fb^{-1}. This trend will continue with the development of the High-Luminosity Large Hadron Collider (HL-LHC). Here, it is hoped that we will be able to gather data at an integrated luminosity of 3000 fb^{-1}. Since the number of gathered events grows as $N \sim \int \mathcal{L}dt$ it is clear that the upgrade will have a profound effect on data collection and analysis efforts [33].

Once data is collected we wish to use it to make a discovery. This can be done through resonance searches. In a collision, a high-mass state known as a resonance may be produced. These will be unstable and have short lifespans. However, the existence of such a resonance can be inferred from the event data. A process that includes a resonance will follow the Breit-Wigner probability distribution

$$f(E) = \frac{k}{(E^2 - M^2)^2 + M^2\Gamma^2}, \tag{1.2.5}$$

where M is the mass of the resonance and Γ is the decay width. The factor k is a constant dependant on M and Γ. If we were to plot this function, we would find a peak around M, with a width proportional to Γ.

By plotting how the number of events varies with the energy we can, firstly, tell if a resonance really exists and, secondly, infer its mass and width. A good channel for searching for a resonance will have a high production rate and final state topologies that are easy to decipher.

The discovery of the Higgs follows this method. The search was dominated by two channels - the diphoton channel and the four lepton channel. These two channels both have topologies that are relatively simple to identify. In Fig. 1.2 we see results from ATLAS for such a search. There is a clear resonance at $m_h \sim 125$ GeV, which belongs to the Higgs boson. Predicting resonances, then finding them, is how many discoveries in high energy physics occur.

While searching for resonances is a successful method of making discoveries, a collider is inherently limited in what resonances it can produce. There is a limit to how massive a particle it can find, and therefore what models it can confirm or theories it can test. If we want to see heavier resonances than the LHC can generate, we must build a collider capable of reaching higher energies. Reaching higher energies can be accomplished by building a larger collider. The reasoning for this can be seen by considering the amount of synchrotron radiation an experiment will produce. As a particle travels through a collider it loses energy, requiring the magnets to

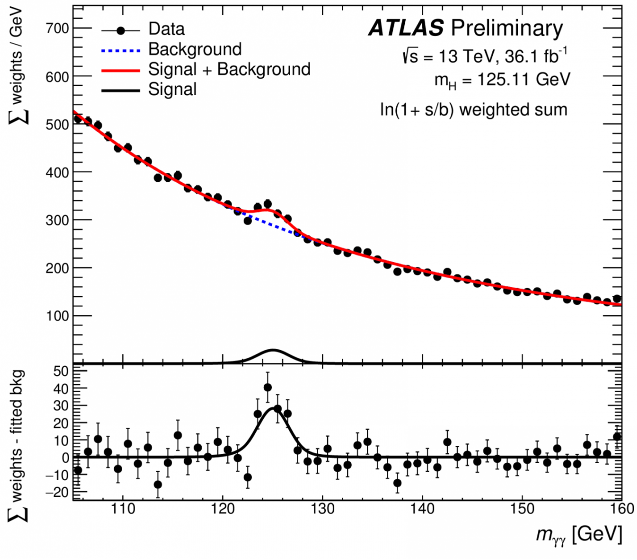


Figure 1.2: A resonance search, in the diphoton channel, showing a bump at $m_h \sim 125$ GeV corresponding to the Higgs Boson [34].

compensate. The amount it loses in one rotation is

$$\Delta E \propto \frac{1}{R}\left(\frac{E}{m}\right)^4, \tag{1.2.6}$$

where R is the collider radius, E the beam energy and m is the particle's mass. In a collider, an amount of energy has to be spent to counteract the effect of the synchrotron radiation. Effectively, this will limit the maximum centre-of-mass energy achievable to a collider. However, as we increase the radius of a collider (or increase the particle mass), we can reduce the effects of the radiation, allowing us to access higher energies. The precursor to the LHC, the Large Electron–Positron Collider (LEP), accelerated electron and positrons. Here, a centre-of-mass collision energy of 200 GeV was achievable. At the LHC, around 14 TeV can be achieved. The proposed Future Circular Collider, with a radius of 100 kM, will likely be able to reach an enormous 100 TeV [35]. By increasing the energies we can probe, we can test a wider range of BSM theories and continue expanding the Standard Model.

1.3 Machine Learning in HEP

In the previous Section, we noted that the increasing luminosity of experiments has a large impact on our ability to gather data for analysis. Currently, one event generated at the LHC requires roughly 1 Mb to store. The task of parsing and storing this data in real-time is enormous. Since around 40 Tb is being generated per second it is impossible to permanently store every event. A series of real-time cuts are applied, resulting in around only 0.01% to 0.001% of events being chosen for storage. What is left is tens of petabytes of data, per year, needing to be stored and analysed [36]. Finding patterns in this high-dimensional data is important for continuing the search for new physics. With this goal, one can look for opportunities to use novel methods to analyse this information.

Traditionally, analysis of event information is performed by applying cuts on the event properties. Then, using this reduced dataset, statistical methods are applied [37]. However, performing these techniques can become problematic as the number of dimensions in the dataset increases (for example, visualising a high-dimensional histogram or the computational complexity of fitting a high-dimensional probabiity distribution) [38]. Machine learning, which has the potential to work well on large-dimensional datasets, can be used to supplement and hopefully improve these methods.

Machine learning is a wide set of algorithms that through being exposed to data learn to make predictions. Generally, these techniques form a subset of artificial intelligence (AI) and have the capacity of becoming extremely accurate. Unlike traditional modelling methods, which may require the user to directly input the behaviour of a system, ML algorithms aim to learn these behaviours themselves. These models can improve their predictive abilities as they are trained on larger datasets. Fields in which dataset sizes are extremely large are therefore incentivised to explore techniques based on machine learning theory. Throughout industry, whether in social media companies, or finance and health, rapidly increasing dataset sizes have led to machine learning becoming ubiquitous. Furthermore, improvements in the algorithms themselves, and the availability of powerful machines that are capable of running them in a reasonable time, have also lead to their increased usage.

Machine learning techniques generally undergo a training procedure where internal parameters are adjusted to improve the quality of the model. At this time, the algorithm is passed data and an output is produced. The output will then be used to help update the parameters of the model, with the goal of improving the quality of the prediction. By performing this step repeatedly, the internal state of the model can be tuned such that the output improves. The range of models and potential uses are vast. This section will therefore focus only on applications in HEP and the architecture of neural networks, a popular ML method.

A range of ML techniques have been adopted by the HEP community. Two popular algorithms are Binary Decision Trees (BDTs) and Neural Networks (NN). A BDT is an ML method that forms a branching graph structure. At each node a binary decision is made (i.e. is $p_T > 10$?, is $\theta < 0.3$?) and the dataset is split. Depending on how an event traverses the tree, based on the rules set at each node, it can be classified. This structure is popular as it provides a fast, interpretable and reliable way to classify objects. BDTs are commonly used at trigger level at the LHC, to quickly decide which events to keep [39].

A neural network is a powerful algorithm, made from a complex system able to approximate a function (we will go into more detail in Section 1.3.1). NNs are extremely versatile, suitable for use in a wide range of problems. Generally, in HEP, they will be used for classification and regression. Classification techniques will focus on separating a signal (or multiple signals) from a background. In regression tasks, the aim is to fit continuous values. Neural networks have been used for jet tagging [39], solving ordinary differential equations [40], determining parton distribution functions [41] and evaluating amplitudes [42, 43].

We can further categorise machine learning methods based on whether they are supervised or unsupervised. A supervised algorithm has access to the labels of the dataset during training time (for a classification problem this may be the class the data belongs to). As a model is being trained, it is shown whether its prediction is correct, and can use this information to help improve itself. The model will aim to ensure its output matches the provided label. If the ML method is unsupervised, the algorithm is unaware of the "truth" and its predictions must be based on patterns in the data alone. This can lead to models with less predictive power than a supervised equivalent, however, this method can be very useful in cases where labelled datasets are not available. For this reason, unsupervised searches for new physics are appealing, as creating a labelled dataset of BSM events from LHC data would be unrealistic. Techniques such as K-means classification, or an autoencoder, can be used to perform anomaly detection (as will be shown in Chapters 2 and 4). Since an unsupervised method does not need to be trained with any signal data, this powerful technique allows us to perform model independent searches for new physics, as we shall see in Chapter 2.

1.3.1 Neural Network Architecture

An important concept in both the quantum, and classical, computing chapters in this thesis is the training and architecture of neural networks. Neural networks are powerful algorithms that allow for complex relationships to be found in high-dimensional datasets. Building a NN allows its designer a lot of flexibility when choosing the structure, leading to a wide range of use-cases. These models were originally inspired by the biology of the brain - where information from one neuron "flows" into another, resulting in a highly interconnected structure. While this is an imperfect comparison (a trained network lacks the adaptibility of the human brain) it does share some conceptual and aesthetic similarities. It may be more accurate to think of a network as learning abstract representations of the data it is being trained on. Earlier layers in the network typically learn very general features of the dataset (shapes, edges), while deeper layers will learn its more nuanced features.

Regardless, models that emulate the structure of the brain have been discussed for decades [44]. However, due to their reliance on large matrix multiplications it has only recently been practical to create and train them. The increase in computing power and availability of GPUs designed to perform matrix calculations quickly, have allowed NN research to further advance.

As a framework to help us understand a NN, we can describe them as being formed by three pillars. These are:

i. An adaptable complex system able to approximate a function.

ii. The calculation of a loss function based on the output of the network.

iii. A method to update the network's parameters based on the result of the loss function.

Pillar (i) is accomplished by forming layers out of interconnected nodes. The first layer of nodes takes an input vector x and transforms it, such that

$$\sigma(W^{(0)}x + b^{(0)}) = h^{(0)}, \tag{1.3.1}$$

where W and b are matrices known as the weights and biases. During training, the weights and biases will be altered to improve the quality of the model. The function σ is known as the activation function, and is included to break the linearity of the node. Without this, the network could learn only linear functions. Finally, $h^{(0)}$ is the node output from the first layer. These nodes (a singlular node in a layer is shown in Fig. 1.3, with weights w that would be a component of a weight matrix, and an output y that would be passed forward) will be stacked upon, and alongside, each other. The next layer of the network will then look like

$$\sigma(W^{(1)}h^{(0)} + b^{(1)}) = h^{(1)}. \tag{1.3.2}$$

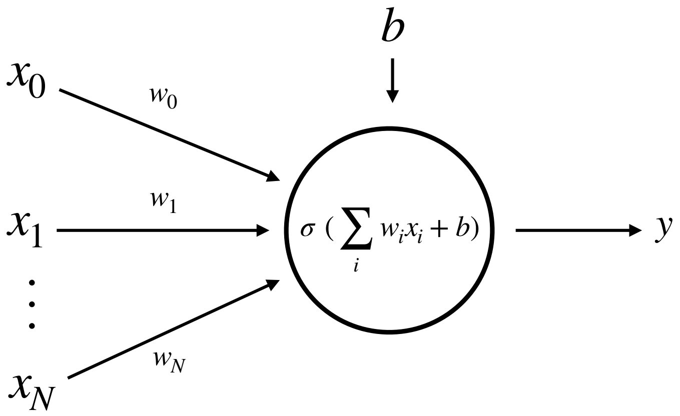


Figure 1.3: A node of a neural network. It will take a vector x of data (or information passed from a previous node) and weight the input accordingly using a value w and a weight b. An activation function, σ, is applied to this and the output is passed forward.

As we have moved to the next layer of the network, the input to each subsequent node will not be x, but rather the outputs from the previous layer of nodes. This will be repeated, until we reach an output layer. Depending on the number of layers, and the quantity of nodes in each layer, the network's ability to successfully model the data will be affected. These choices also determine the dimensions of W and b. The resulting network will have a structure similar to Fig. 1.4.

Mentioned above were the activation functions, designed to break the linearity of the network. These have a direct impact on the network performance, guiding what sort of functions the network can learn to emulate. The internal, hidden, layers of the network typically use ReLU or tanh activations, while the output layers tend to make use of something such as a softmax layer (shown in Figure 1.5). Included in Figure 1.5 is the linear function, which could be used instead of a activation function. However, as the name suggests, this will not break linearity. ReLU and tanh are both widely used, but do have some common issues. Using the ReLU function can lead to "dead neurons" - a situation where a neuron output gets set to

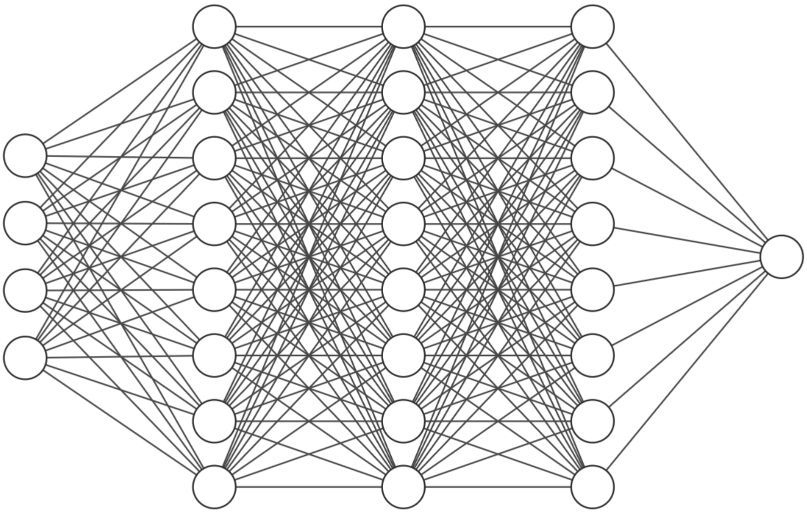
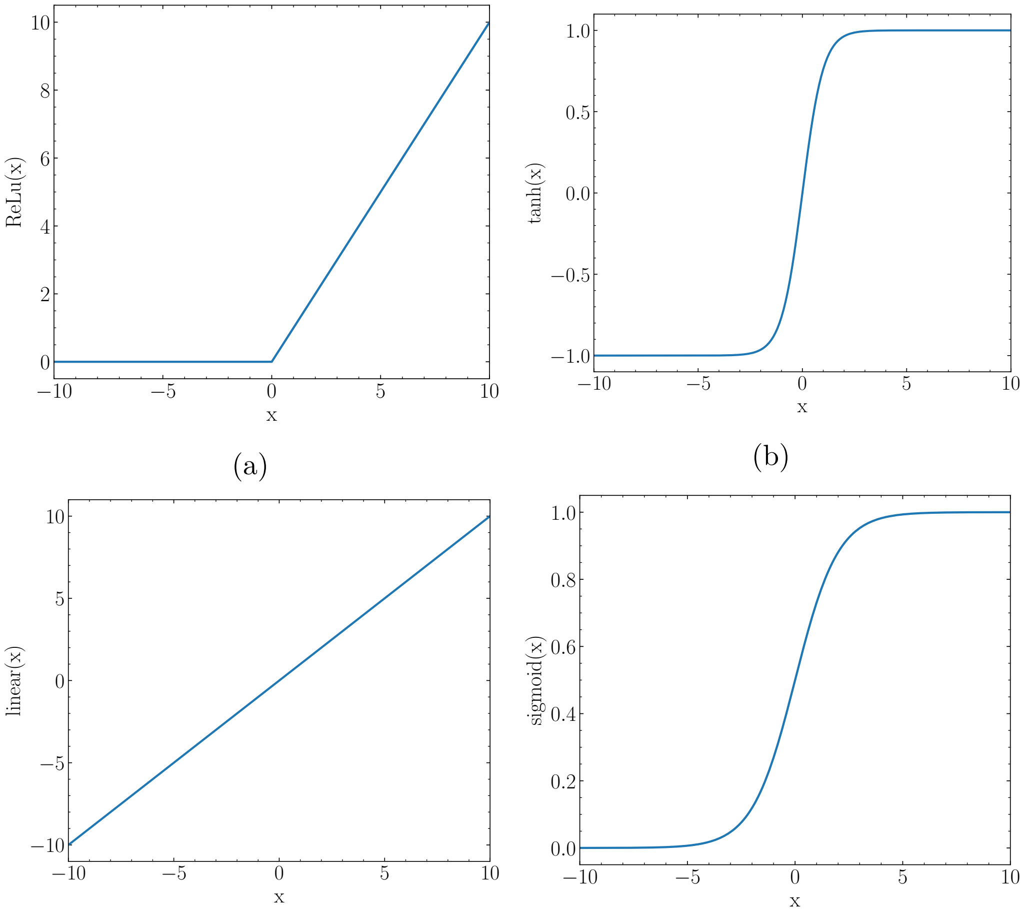


Figure 1.4: The structure of a dense neural network made from a series of interconnected nodes.

0 and essentially cannot contribute to the network. The tanh function is susceptible to what is known as the “vanishing gradient problem” - here, we note that certain values of the gradient of tanh can be very close to 0, disturbing the optimisation process.

The number of layers, the quantity of nodes within each of them and the activation functions used are all hyperparameters of the network. These are parameters which can be tuned to increase model performance. This leads to a versatility that allows neural networks to be able to perform a wide range of tasks.

Pillar (ii) is the calculation of a loss function, L. The loss function will compare the output of the network against the truth in order to give a metric for how well the network has performed. The goal of training a network is, typically, to minimise a loss function. Generally, as the output of L decreases we can say that the network is improving its ability to carry out its training objective. The choice of loss function is decided by the task the network is designed to carry out. A network designed to classify objects would typically use a binary cross-entropy function, while a regression problem would use a mean squared error function. The binary cross-entropy function



(a) (b)

(c) (d)

Figure 1.5: Representations of functions that can be used as, or instead of, neural network activation functions. Shown is (a) ReLU, (b) tanh, (c) linear and (d) sigmoid.

typically takes the form

$$L = -\sum_{i=0}^{n} y_i \log(p(x_i)), \tag{1.3.3}$$

where $p(x_i)$ is the probability of seeing a given result, and y_i is the label, for the ith object in a dataset. Mean squared error has the form

$$L = \frac{1}{n}\sum_{i=0}^{n}(f(x_i) - y_i)^2, \tag{1.3.4}$$

with $f(x_i)$ being the model output and y_i being the true value.

Pillar (iii) makes use of the result of the loss function to update the trainable parameters in the network (i.e. the weights and biases of the nodes). This is done by performing backpropagation. This process finds the gradient over the network, for a given set of parameters. Using this information, new parameters can be chosen. The simplest option for this is gradient descent:

$$\phi_{i+1} = \phi_i - \alpha \nabla_\phi L(\phi_i), \tag{1.3.5}$$

where ϕ is the network's parameters and α is the learning rate. The learning rate decides how large a step you take when you change the parameters of the network. In parameter space, this step moves the network parameters down the slope of steepest descent. The best values for the parameters will lie at the bottom of this valley. Choosing a learning rate too large will result in the algorithm always stepping over the true minimum, choosing a value to small will result in you never getting there. A common technique is to set a learning rate schedule. Here, you would begin with a large learning rate (to quickly explore the parameter space) but as the loss plateaued the learning rate would reduce. The aim is to help the model settle into the correct result.

These three pillars make up the basic training loop for a neural network. During training, it is standard practise to split the dataset the network will be applied to into two (and sometimes three) different sets. A training set will be used only at training time, and a test set will be used only for evaluation. A concern during training is that your model will learn to overfit. Overfitting is the scenario where your model learns your training data precisely, but performs very poorly on a never-before-seen sample. By having a set of samples that is hidden from your model during training it is easier to tell if your model has overfit.

As the network is being trained, the training data can be batched. If the number of batches is one, then the gradient calculated during backpropagation is found using every sample in your training dataset. The number of batches can be increased, decreasing how much data goes through the network at once. This results in the network's trainable parameters being updated using multiple subsets of your dataset.

After the entire training dataset has passed through the network, it is said an epoch has passed. Overall, the process looks like:

i. Perform a forward pass of the network and calculate the loss function.

ii. Using the output of the loss function, backpropagate and update the parameters.

iii. Perform steps (i) and (ii) for every batch.

iv. The first three steps are repeated until a set number of epochs has passed, or the loss value has accomplished some set criteria.

These steps provide a large hyperparameter space to tune. The batch size, epoch, number of nodes and layers, activation functions, loss function, learning rate and method of optimisation can all be tuned to improve the networks performance.

Constructing a model in this manner, with its inheritent customisability, allows it to be very powerful. Here, we have discussed the simplest form a neural network can take - an artificial feed-forward network. However, a range of other architectures exist. Of importance is the convolutional neural network structure (popular for image-based problems) and the recursive neural network (useful for tackling time series data). Even without these more complex structures though, a network can be trained to be a universal function approximator, i.e. a neural network could learn to emulate any function [45]. Networks have been designed to play complex games like Chess [46] or Starcraft [47] and learn to drive cars [48]. They have been taught to perform tasks like classification, image recognition [49, 50] and natural language processing [51, 52]. Other uses include generating new data such as images [53] and text [54]. The following chapters of this thesis apply the techniques discussed here to both supervised and unsupervised searches for new physics.

1.4 Quantum Computing

As well as machine learning, another field that has been proposed to boost the performance of particle physics methods is quantum computing. A quantum computer is a fundamentally different device from a classical computer. Classical devices store information in bits. These fundamental objects are binary systems - they are either in state 0 or 1. Instructions and data are stored as binary numbers and executed and interpreted by the computer's CPU.

Quantum computers do not depend on bits, instead, they are built up from qubits [55, 56]. A qubit, like a bit, also contains two states $|0\rangle$ and $|1\rangle$. What separates the qubit and bit though is the qubit can exist as a linear combination of states,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle. \tag{1.4.1}$$

The state $|\psi\rangle$ depends on α and β, complex numbers that tell us how "mixed" the qubit is. In a classical computer, we can easily probe a bit to find its state. In this case, it will always either be 0 or 1. However, if we measure a qubit we must deal with probabilities. Performing a measurement will result in observing state $|0\rangle$ with a probability $|\alpha|^2$, or state $|1\rangle$ with probability $|\beta|^2$.

We can also write a qubit in the form

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle. \tag{1.4.2}$$

This is a useful interpretation, as the angles θ and φ can be used to define a point on a sphere. This sphere, known as the Bloch sphere, is shown in Fig. 1.6.

While we can only measure a qubit as being in state $|0\rangle$ or $|1\rangle$, the qubit itself contains a lot of information. There is an infinite number of potential states that can exist on the Bloch sphere. We can apply operators onto these states, evolving them. These manipulations will then also change the outcomes of the measurements

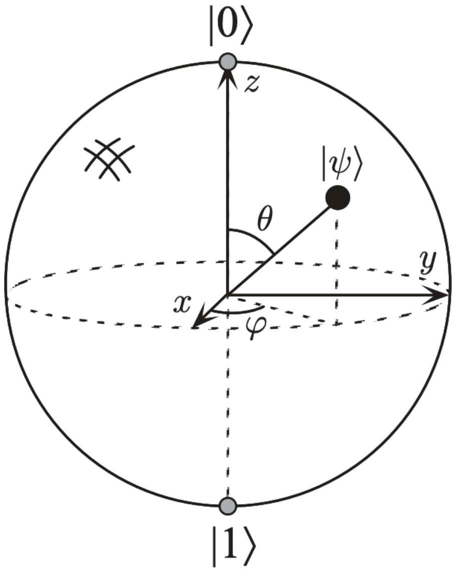


Figure 1.6: Representation of a 1 qubit state $|\psi\rangle$ on a Bloch sphere. Taken from Ref. [55].

we make. As we shall see, by performing a series of operators we can implement algorithms that can be run on a quantum computer.

By expanding to include two qubits, we can write a state

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \ . \tag{1.4.3}$$

Now, there are four possible results that can be attained after measurement. Each result has a corresponding probability α. These coefficients are known as amplitudes. As we expand the number of qubits, n, in a system, the number of amplitudes increases as 2^n.

To gain insight into why using qubits to perform computations may be useful, we shall consider a specific two-qubit system. This system, the Einstein, Podolsky, and Rosen (EPR) pair, displays an interesting property. An EPR pair, or Bell state, has the form

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle. \tag{1.4.4}$$

Two physicists, Alice and Bob, could each have a qubit from this pair. Now, if the pair physically separated the qubits, they would still remain entangled. A measurement of Alice's qubit would reveal an observation of $|0\rangle$ with probability 0.5 or $|1\rangle$ with probability 0.5. Now, if Bob measures his qubit he will always get the same result as Alice. This simple state is highly correlated and shows the potential of achieving non-local effects in the quantum computing systems we design. For example, we can use EPR pairs to design a system to teleport a state.

To design programs on a quantum computer we must have a method of evolving the state of a qubit. This is done by the application of a unitary operator known as a gate. A series of gates applied to qubit (or multiple qubits) is known as a circuit. Creating circuits is the typical method of implementing programs on a quantum computer. There are a large number of gates available. They will be introduced, as needed, in Chapters 3 and 4. For now, we will limit our discussion to the single qubit phase gate

$$R_\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}. \tag{1.4.5}$$

If the R_ϕ gate is applied to our original state, described in Eq. 1.4.1, it will be evolved to

$$R_\phi|\psi\rangle = \alpha|0\rangle + \beta e^{i\phi}|1\rangle. \tag{1.4.6}$$

While this is a simple evolution, it still demonstrates that by applying a gate to a qubit, we change its state. This manipulation, in turn, will affect the result found from a measurement.

Quantum computing presents then a unique way of performing computations. The use of a qubit, and the quantum properties it inherently has, is fundamentally different than the classical bit. Using a qubit as the fundamental object for a computation, however, is not a purely theoretical exercise. There are a set of processes that can be performed on a quantum computer that demonstrate what is known as a quantum advantage. These advantages arise when the quantum device can perform a task that no classical computer could solve in a reasonable time. Broadly speaking, algorithms displaying an advantage fall into one of three categories:

- Search Algorithms. A standard quantum computing algorithm is Grover's search algorithm [57]. Search algorithms on a quantum computer can give a potentially quadratic speedup compared to what is possible on classical machines. Typically, to search through N objects using classical techniques require $\sim N$ operations - quantum equivalents can do this in $\sim \sqrt{N}$.
- Algorithms based on Fourier transforms. Performing Fourier transforms make up a large part of many computational algorithms. Compared to a classical machine, there is a large efficiency increase associated with using a quantum device. Classically, to transform $N = 2^n$ numbers, where n is the number of bits (or qubits), will take $\sim n2^n$ steps. On a quantum device, this can be designed to take $\sim n^2$ steps. Shor's algorithm makes use of this to perform fast factoring of numbers [58].
- Simulation of systems. Using a quantum computer for simulating a quantum system also appears to be an avenue to achieving an advantage. On a classical machine, storing a state with n components takes roughly 2^n bits (as you need to be able to account for entanglement between states). This can be accomplished using a quantum computer with n qubits, as the quantum properties of the system are naturally described by qubits. This reduction in required memory may allow us to efficiently simulate systems that we previously, on a classical device, could not.

Alongside these applications, quantum computers have become a popular method to solve a range of tasks. Quantum annealers have proven to be a robust way to optimise a potential. Using a quantum method allows you to more consistently find the true minimum compared to classical techniques such as Nelder-Mead [5, 59–61]. Building circuits from quantum gates have been used to write parton showering algorithms [62], jet clustering algorithms [63] and mapping field theories to quantum walks [64].

So far, we have focused on the qubit paradigm of quantum computing. Now, we shall consider the continuous-variable (CV) paradigm [65]. In this other paradigm, instead of the qubit, the fundamental object is the qumode. The qumode is an object with an infinite-dimensional Hilbert space associated with it, leading to states of the form

$$|\psi\rangle = \int \psi(x)|x\rangle dx. \tag{1.4.7}$$

By considering a quantum computer constructed from qumodes, one can take advantage of advantages associated with the CV paradigm [66]. While this branch of computing is fundamentally different to the discrete qubit form, we still use gates and circuits to implement algorithms. In Chapter 4 we return to these concepts and use them to perform anomaly detection.

Directly relevant to this thesis is quantum machine learning. Quantum machine learning is an emerging subfield, which aims to find quantum advantages associated with building ML models that operate based on quantum information principles. One option is to build a quantum neural network [67]. Here, the trainable parameters of the network are the parameters of the quantum gates used to create a circuit. The use of quantum machine learning techniques will be discussed in more detail in Chapter 3 and 4.

Chapter 2

Adversarial Autoencoders for Anomaly Detection

The separation of anomalous signal events from Standard Model background events is an important task for data searches at the LHC. It is in cases like this that performing a data-driven search can be beneficial. This involves training a model to identify features in the background events without making assumptions about what your signal may look like. A method, as we will discuss, would still leave you open to an experiment's systematic uncertainties. This motivation is expanded upon in Section 2.1. To reduce the impact of these errors, we propose combining an autoencoder designed for anomaly detection with an adversarial network to detect smearing in an event's final state objects. The generated data and smearing process are shown in Section 2.2. Section 2.3 details a supervised classifier that we use as a baseline. The autoencoder and its adversarial extension are introduced and explained in Section 2.4. The chapter ends with a summary and conclusion.

2.1 Motivation

As discussed in Chapter 1, machine learning in particle physics has become a popular method of performing both classification and regression tasks. These techniques can search through high-dimensional parameter spaces, analysing large amounts of observables simultaneously. These searches allow one to find areas in the parameter space where there is an excess of signal events compared to the background. The application of these algorithms to new physics searches at the LHC allows for greater sensitivity.

To classify HEP data, a supervised search can be performed. In a supervised search, the label of the event is used as information while training the classifier. To train the classifier, one must generate pseudo-data of the background and signal samples using a Monte-Carlo event generator. As the label of the sample is given to the algorithm during training time a powerful classifier can be created. However, models trained this way are limited in their scope - they only can have predictive power on models they have been trained on. Furthermore, samples created by event generators will be subject to their own theoretical uncertainties. Any model trained on generated data will therefore also be biased by these uncertainties. Training a model using generated data could result in classification being made based on the uncertainties involved, rather than on features that could be seen in a collider. To protect against this, or to help stop models becoming biased towards certain features in a dataset, adversarial models have been proposed [68–70]. We will design our model based on this architecture.

One method of removing model dependence on generator uncertainties is to not train on generated data at all. Instead, one could try a data-driven search. Here, the model is trained on experimental data. However, this can introduce a new problem. To train a supervised classifier the data must contain a suitable amount of well separated background and signal events. For a rare new physics signal this could be an incredibly difficult, even impossible, task. Regardless, performing a data-driven search remains an appealing option.

To perform a data-driven search, an unsupervised model could be trained using only background samples. The model would learn the kinematic features of these events. If shown an event that did not possess these traits it could be flagged as signal.

Unlike a supervised search, where you train on a specific set of signal models, this method is independent of your choice of signal events. Since generator errors have been removed from model training, what remains is systematic experimental error.

The method of unsupervised classification presented in this chapter is based on the use of an autoencoder [71,72] . Autoencoders are neural network structures which can perform denoising [73], generative [74] and anomaly detection tasks [70, 75–77]. The training goal of the autoencoder is to “copy” the input data to the output. This task is made non-trivial by forcing an information bottleneck in the centre of the network. As the input and output of the network will match, the bottleneck is implemented by reducing the number of nodes in the centre of the network. As it is unable to produce a perfect reproduction of the input, the network is forced to learn the intrinsic features of the training set in order to minimise the loss function. To minimise the loss function, the output must contain as much information about the input as possible. When the trained model is shown a signal event, it can be expected then to have a lower ability to reproduce it compared to the background. This is because the kinematic features of the background and signal will be different. Therefore, the resulting values of the loss function will differ from that of the background set. An anomaly detector can be built by thresholding these values and deciding everything above this value will be counted as signal.

Adversarial networks can be used to remove a model’s dependence on features within the training data. Generally, an adversary aims to make a prediction based on the original model’s output. The adversary and original model are in a zero-sum game. If the adversary can correctly make a prediction, the base model will be penalised. However, if the base model can perform its training goal then the adversary is penalised. This setup is discussed more in Section 2.4. To account for systematic experimental uncertainties we extend our autoencoder with an adversary. We show that by training this extended model, the classification is desensitised to some uncertainties while still retaining classification ability. To perform the training we use Monte-Carlo generated pseudo-data, with experimental uncertainty being represented by the systematic smearing of the original events.

2.2 Data Generation and Smearing Procedure

To carry out the anomaly detection we focus on heavy resonance searches. The events generated are based on work using fat jets to reconstruct highly boosted top quarks [78–81]. The background and signal samples used here consist of $pp \to t\bar{t}$ events and $pp \to Z' \to t\bar{t}$ events, respectively. The background events have been generated with a centre-of-mass energy of 14 TeV. In each event, we force one of the final state top quarks to decay hadronically and the other top quark to decay leptonically. A heavy new boson, Z' [82], is used as signal, with a mass of 2 TeV and a decay width of 89.6 GeV. Consistent with the background events, one top quark decays hadronically and the other leptonically. For all events, a cut of $p_T > 500$ GeV is placed on the transverse momentum of the top quarks to ensure we are in the boosted region. All events are generated using MadGraph5_aMC@NLO [83] while the parton showering and hadronisation is performed with Pythia 8.2 [84].

The hadrons and the non-isolated leptons are clustered into fat jets (jets from massive, boosted particle decays) with a radius $R = 1.0$, using the Cambridge-Aachen algorithm [85]. With such a large radius, we expect to capture the full collimated decays. The k_T algorithm is implemented using FastJet [86] to recluster the hardest two fat jets into jets with radius $R = 0.2$. Jets are b-tagged based on proximity to a B-meson, whilst requiring them to have a transverse momentum $p_T > 30$ GeV. We also demand any isolated leptons to have a transverse momentum $p_T > 10$ GeV. These cuts aim to remove the softer emissions and radiation we are uninterested in.

The selection of these events is then based on numerous criteria. One fat jet per event must contain at least one b-jet whilst the other fat jet must contain at least two light jets and one b-jet. The events must also contain a minimum of one isolated lepton and are required to have a scalar-summed transverse momentum of $H_T > 1$ TeV.

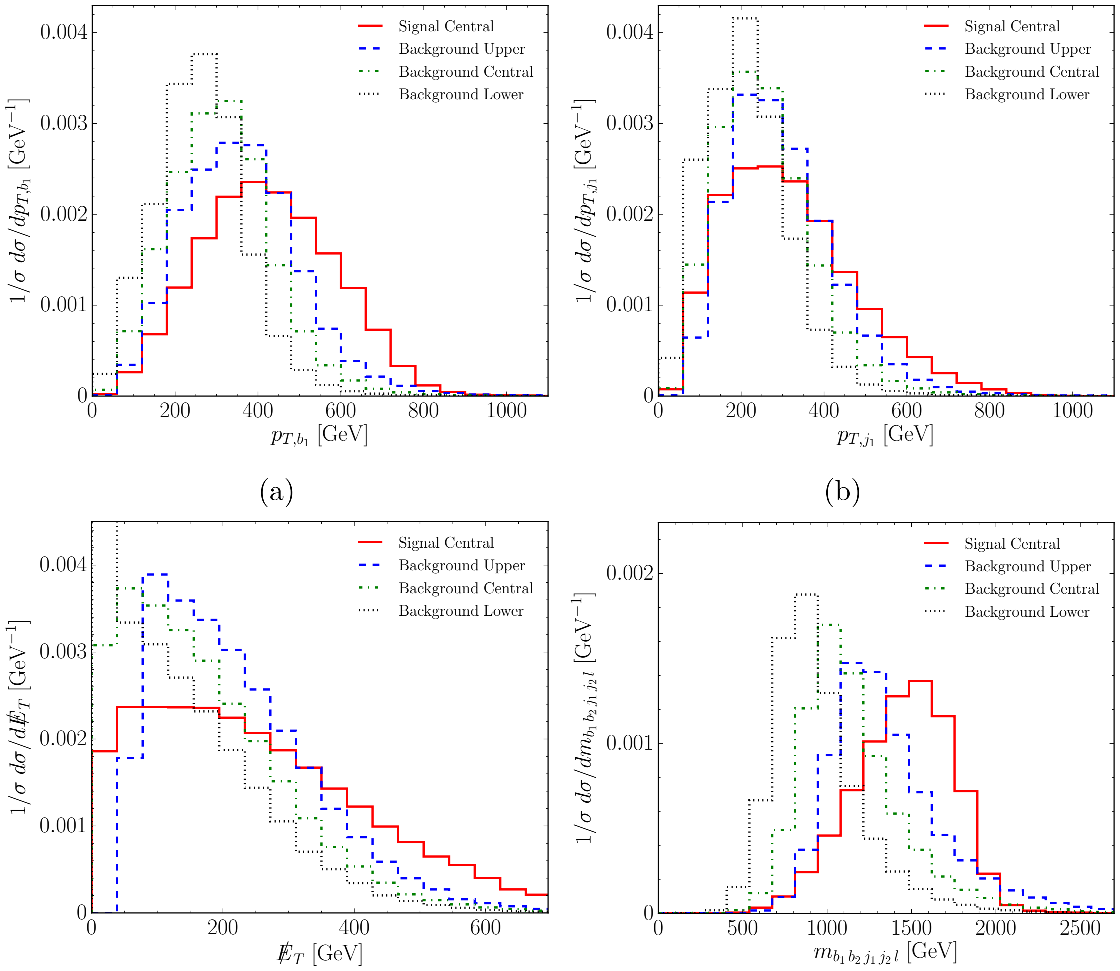
The following analysis is carried out using the four-momenta of the two b-jets, two light jets and isolated lepton, as well as the missing energy ($\not{E}_T$) in the event. This results in 21 total observables.

To create data to represent systematic uncertainties we apply a smearing procedure to the properties of each event's jets and missing energy. By smearing an event upwards (or downwards) we are acting as if all the systematic error is going in one direction (either all error is positive or all negative). For the event's jets and leptons, the procedure is based on Refs. [87, 88] where the object's three-momentum is smeared. The values chosen to smear the bins of the histogram are based on the resolution of the jet energy [89]. The relative amount of the smearing is larger for smaller values of p_T. For missing energy, the smearing is proportional to $\sqrt{H_T}$ [90]. In all cases, smearing is always performed in the same direction, either moving the values up or down. The value chosen to smear by is increased by a factor of three, thus increasing the effect of the procedure. This ensures that we account for the systematic uncertainty, but will also demonstrate our setup's effectiveness.

The smearing procedure is applied to the sets of background samples. This results in a total of three samples based on background events: (i) an original, unsmeared set, (ii) a set smeared in the upward direction and (iii) a set smeared in the downward direction. The signal set is left unsmeared[1]. Each set of samples comes from a statistically independently generated sample. After performing the described cuts, 100,000 events from each set of the four samples remain. Of the 400,000 events, 20% are used for a test set. Before being used for model training, the data is scaled using the StandardScaler method from the scikit-learn package [91].

Fig. 2.1 shows the generated data. The effect of smearing on the p_T of the hardest b-jet and light jet, the missing energy of the event and the invariant mass of the jets and lepton in the background samples is demonstrated. By comparing the smeared

[1] It would be possible to perform this with smearing signal set too, however, this is not carried out here. We aim instead to demonstrate resilience against background smearing. In a worst case scenario, we would have to compare an upwardly smeared background set against a downwardly smeared signal set - we hope to have covered a similarly challenging situation by factoring the applied smearing to the background by 3.



(a)

(b)

(c)

(d)

Figure 2.1: The unsmeared background and the signal samples compared to the smeared sets for (a) the p_T of the hardest b-jet, (b) the p_T of the hardest light jet, (c) the missing energy and (d) the invariant mass of the jets and lepton.

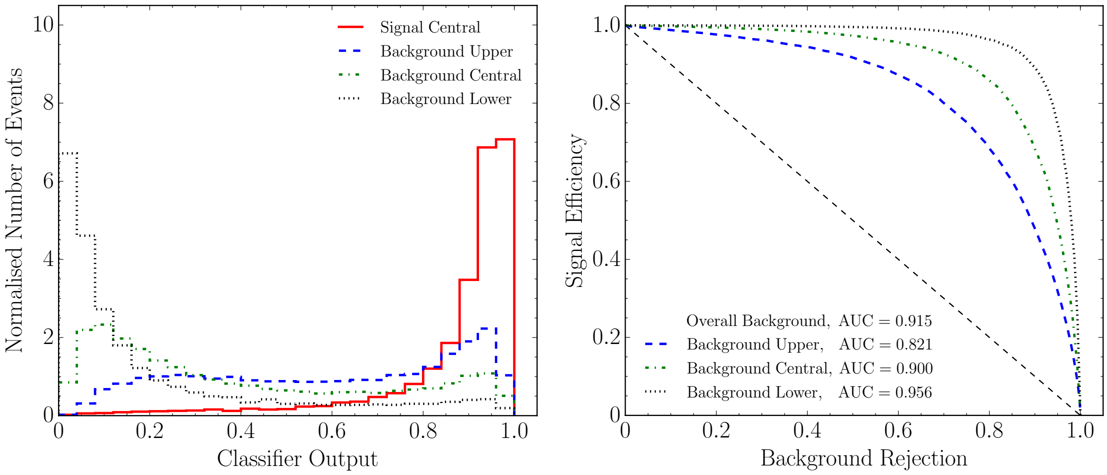
background events to the signal distribution it is clear that some background sets (depending on the direction of the smear) will be easier to discriminate from signal than others.

2.3 Supervised Classification of Jets

Before training an autoencoder to perform unsupervised anomaly detection we first train a supervised model. This sets a signal-background discrimination benchmark. The supervised model was trained on both the signal and background events. As the model is trained on both sets of samples and shown the sample set labels, we expect this method to outperform any unsupervised methods.

The supervised approach uses a standard neural network. This network is constructed from two hidden layers, each with 20 nodes and ReLU activations. The final layer consists of one node with a sigmoid activation, a standard choice for classification tasks. Since there are only two classes (signal and background) a binary cross-entropy loss function is chosen. The loss function is weighted based on the class to account for the higher frequency of background samples in the training data (for each signal event there are now three background events). While training we use the Adam optimiser [92] with a learning rate of 0.01 and a batch size of 500. Training lasts for 500 epochs. Throughout this paper the networks are implemented and trained using Keras [93] with a TensorFlow [94] backend.

The results from the trained model are shown in Fig. 2.2. The results are presented in two forms: (a) shows the distribution of the network output while (b) presents the receiver operating characteristic (ROC) diagram. These curves show a measure of how well the network performs with respect to true and false positives. A perfect result, where everything is correctly classified, would be represented in a ROC plot by a curve reaching entirely up to the upper-right hand corner. A curve for a classifier with no predictive power would be a diagonal line (i.e. the classifier randomly guessing). The results in the ROC plot are created by comparing each background set (no smearing, smeared up and smeared down) against the signal samples. Also provided are the area-under-curve (AUC) scores. Also provided is an overall AUC score calculated from passing all samples through the model at once. While this would not naturally happen, this value is presented simply to give an overall sense of the model's performance. The figures show that the network performance is strongly dependent on how the smearing has been applied. In Fig. 2.2 (a) the upward smeared background set overlaps the most with the signal set. Looking at Fig. 2.2 (b) shows that the classification performance is negatively affected because of this.



(a)

(b)

Figure 2.2: The output of the supervised neural network classifier (a) and ROC plots (b) for a classifier trained to classify the signal and background events. The different background distributions result from the different directions of smearing.

2.3.1 Supervised Adversarial Anomaly Detection

To mitigate the effects of smearing on the network's performance we will extend the classifier with an adversarial network. This adversary is designed to take the network output and discriminate which style of smearing (smeared upwards, downwards, or left unchanged) was originally applied to it. The aim for the anomaly detector is to remove its performance being dependent on the smearing. This will result in the adversary being unable to make a prediction [68, 69]. The classifier and adversary are in a zero-sum game. The classifier predictions must be made without using information derived from the smearing such that the adversary cannot discriminate the origin of the background samples. This is done by forcing the two networks to have opposite optimisation goals, meaning the classifier will be penalised as the performance of the adversary increases.

The adversarial network is a neural network made up of two hidden layers of 20 nodes with ReLU activations. The input for the adversary comes from the output of the classifier. The final layer of the adversary has three nodes (one for each smearing category) with a softmax activation function and categorical cross-entropy loss, customary for multi-class classification problems. The model is trained using the following procedure:

1. The classifier network is trained for three epochs, with the same parameters as discussed in the previous section.

2. The classifier is then frozen while the adversary is trained for three epochs. This is done using mini-batch gradient descent with a learning rate of 0.01 and a batch size of 500. This training is carried out only using background events.

3. The classifier is then trained for one epoch. This is done with a mini-batch gradient descent with a batch size of 500. The loss function is described by

$$\mathcal{L}_{\text{tot}} = \mathcal{L}_{\text{class}} - \alpha \mathcal{L}_{\text{adv}} \ . \tag{2.3.1}$$

   This new loss function accounts not just for performance of the classifier, but also for the adversaries. In this loss, two class weightings are applied. The first accounts for the higher frequency of background events in the training set, while the second accounts for the higher frequency of unsmeared events (half of all events are unsmeared).

4. The adversary is trained on background events for one epoch. This uses a mini-batch gradient descent optimisation with a batch size of 500.

5. Steps 3 and 4 are repeated 1000 times. A learning rate decay schedule is implemented. Initially, a value of 0.01 is set while the schedule lowers this every 100 epochs by a factor of 0.75.

The loss function shown in Eq. (2.3.1) is weighted by a factor α. This determines the relative importance of the two objectives during optimisation. If set to zero, the adversary does not affect training. If set too large, then the adversary's objective will dominate and the classifier performance is negatively affected. Here, we set α to be 100. In our scenario, during training, we update the two network's weights

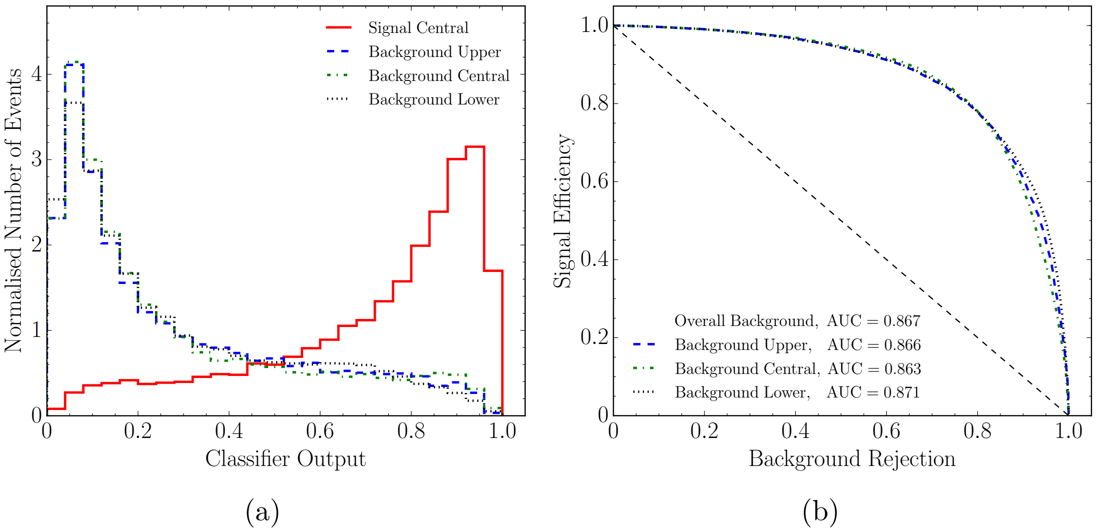


Figure 2.3: Output of an adversarially trained supervised neural network classifier trained to classify signal and background events (a) and ROC plot (b). The different background distributions results from the different directions of smearing.

separately. While the adversary weights are being updated, the classifier weights are frozen, and vice versa. It is also possible to update the model weights simultaneously, however we find our method results in more stable training.

Fig. 2.3 shows the results of training the adversarial classifier model. Unlike in Fig. 2.2, where it was clear that the performance was heavily dependant of the direction on smearing, here, the network performance is almost entirely independent of it. Whether the background is smeared up, or down, the classifier output distribution mirrors the non-smeared distribution. This results in the ROC and AUC scores also becoming very similar. The inclusion of the adversary has reduced the overall classification ability of the network. This is a trade-off required to decrease the model's dependence on the direction of smearing.

2.4 Anomaly Detection with an Autoencoder

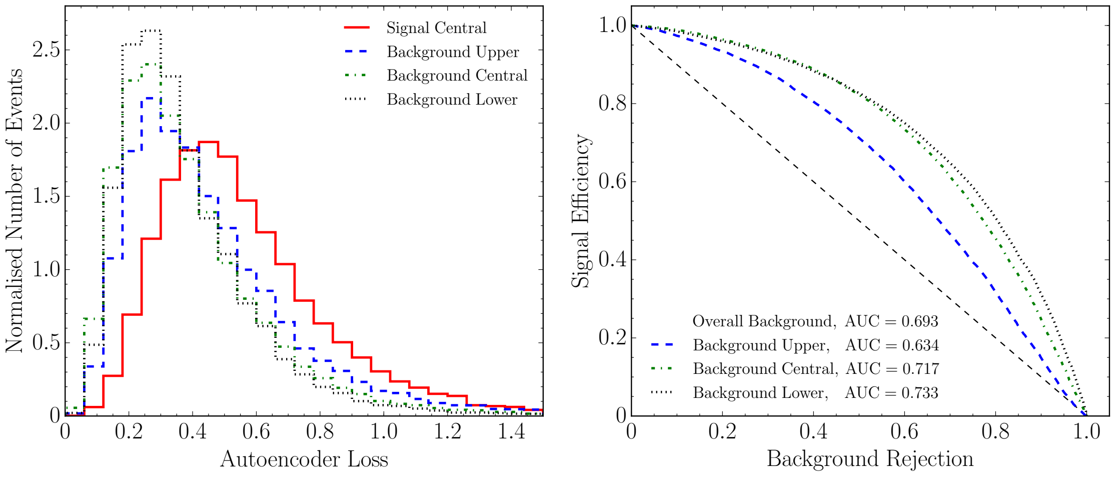
Supervised learning requires a label for each entry in your dataset. In our case, performing a data-driven supervised search is an unrealistic option. This would require creating a labelled dataset of both standard model background and rare BSM signal. Instead, an unsupervised anomaly search can be performed. This would use a model only trained exclusively on standard model background samples. In this section, we propose the use of an autoencoder to perform such an unsupervised discrimination task.

2.4.1 Unsupervised Anomaly Detection

An autoencoder can be used as a method of unsupervised anomaly detection. We construct our model with three hidden layers of 10, 3 and 10 nodes, each with sigmoid activation functions. The input layer and final output layer each have dimensions that match the number of features in the input. In our case, this is 21. The loss is the mean squared error between the input of the network and the output. This results in the autoencoder having the training goal of reconstructing the input as fully as possible. However, this task is made non-trivial by the encoding and decoding process. Specifically, this is caused by the information bottleneck in the centre of the network forcing the information into a lower-dimensional space.

The network is trained on background events using the Adam optimiser for 500 epochs with a learning rate of 0.01. Since the autoencoder sees only background events during training one expects that the model will be able to reconstruct them with greater accuracy. Fig. 2.4 (a) shows the reconstructed background events consistently have a lower associated loss value than the signal samples. This information can be used to create a classifier. By placing a cut on the loss a classification can be made. Events below this cut are classified as background and events above are classified as signal. Varying the classification threshold changes the number of events classified as signal, resulting in the ROC plot drawn in Fig. 2.4 (b).

The ROC diagram presented for this scenario follow a similar pattern to the supervised example in Section 2.3. The smearing of the background samples has a profound effect on the autoencoder's ability to classify events. Events which are



(a)

(b)

Figure 2.4: Autoencoder loss (a) and ROC plot (b) for an autoencoder trained only on background events. The different background distributions results from the different directions of smearing.

smeared upward are more likely to be misclassified as signal than events smeared down. The autoencoder generally performs worse than the supervised classifiers, visible in Fig. 2.2. However, this is to be expected as the autoencoder is not exposed to any of the signal events during training. Furthermore, when training the classifier the training objective is to perform a strong classification. This is not the case for the autoencoder.

2.4.2 Adversarial Extension to Autoencoder

Just as we combined the supervised classifier with an adversary we now extend the autoencoder with an adversarial network. As we saw, samples presented to the autoencoder are classified based on their loss. To ensure the model can make predictions without dependence on the smearing, the loss of the autoencoder is used as input into the adversary. This is analogous to our previous model, where the classifier network's output is used as the adversary input. There too, a cut is placed on the output of the network to discriminate signal from the background.

The adversarial network has two hidden layers, each with 20 nodes and ReLU

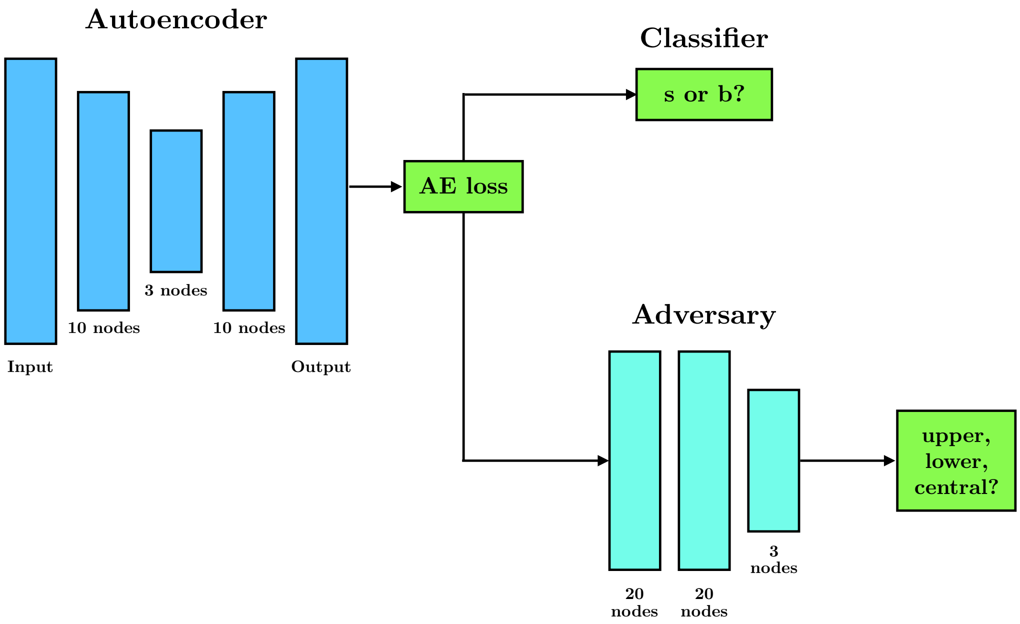


Figure 2.5: Architecture of the adversarial autoencoder. The loss function of the autoencoder is used as an input to the adversary for it to discriminate the smeared background samples.

activation functions. The final layer has three nodes with softmax activations and uses a categorical cross-entropy loss. The network architecture is shown in Fig. 2.5. The training procedure follows a similar pattern to the previous adversarial network, however here we use only background samples:

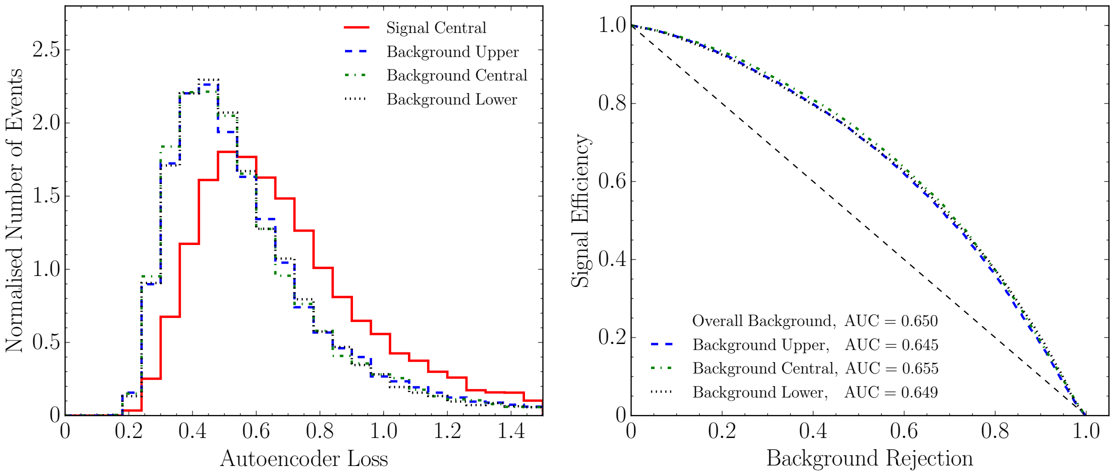
1. The autoencoder is trained for three epochs using an Adam optimiser set with a learning rate of 0.01 and the batch size of 500.

2. The adversary is trained for three epochs. This is done using mini-batch gradient descent with a learning rate of 0.01 and a batch size of 500.

3. The autoencoder is then trained for one epoch. This is done with a mini-batch gradient descent with a batch size of 500. The loss function is described by

$$\mathcal{L}_{\text{tot}} = \mathcal{L}_{\text{auto}} - \alpha \mathcal{L}_{\text{adv}} \ . \tag{2.4.1}$$

4. The adversary is trained for one epoch. This uses a mini-batch gradient descent optimisation with a batch size of 500.



(a) (b)

Figure 2.6: The loss (a) for an adversarial autoencoder trained only on background events and the ROC plots (b). The different background distributions results from the different directions of smearing.

5. Steps 3 and 4 are repeated 1500 times. A learning rate decay schedule is implemented. Initially, a value of 0.01 is set while the schedule lowers this every 100 epochs by a factor of 0.75.

Using these settings provides us with a stable method of training our model. As before, we set the relative weight between the autoencoder and its adversary to $\alpha = 100$. Fig. 2.6 shows the network's performance after training. By applying an adversarial extension the background distributions have been shaped such that they are independent of the smearing direction, as shown in Fig. 2.6 (a). This results in the ROC plots presented being almost identical. This shows our method of using an adversarial autoencoder for anomaly detection can allow for signal-background discrimination independent of the inherent uncertainties of final-state reconstruction at the LHC.

We note that the model also performs well when tested on background sets with lower amounts of smearing. While we trained on samples smeared by a very large amount the network can interpolate to smaller quantities. Furthermore, testing on samples that have lower amounts of smearing increases the AUC score, giving similar results to a network trained using less smearing.

By combining an adversary with the autoencoder network we have managed to perform unsupervised anomaly detection without dependence on systematic error. We began by generating three background sets - a default one, one smeared upwards and one smeared downwards. The smearing was applied to the jets, leptons and missing energy of events. A set of signal events was also generated and left unsmeared. The effect of the smearing was to shift the kinematic features of each event, as shown in Fig. 2.1. Practically, this meant that some background events became harder to distinguish from signal.

2.4.3 Corrupted Training and Applications to Other Models

So far, training has been carried out using sets of pure background samples (i.e. there are no signal events in the set). However, in a true data-driven search the analysis carried out will not look like this. Realistically, background samples gathered using real data would also contain, if it exists, new physics events. It is therefore important to account for this. This can be done by injecting a number of signal events into the background event samples. By training and carrying out analysis on these corrupted background sets we can view how sensitive the autoencoder is to corrupted training sets.

Fig. 2.7 shows the results from training our autoencoder model, using training sets "corrupted" by signal samples. The band shows the difference in the AUC for the upward and downward smeared sets, while the green line shows the overall AUC. During the training of these models all hyperparameters are held constant, with only the amount of signal injected into the training set changed. From Fig. 2.7 we see that corrupting the training set has little effect on our classification ability even when the amount of corruption is very large. This demonstrates our adversarial autoencoder method's potential to be applied to real data.

As well as testing our model's performance while using corrupted datasets we can

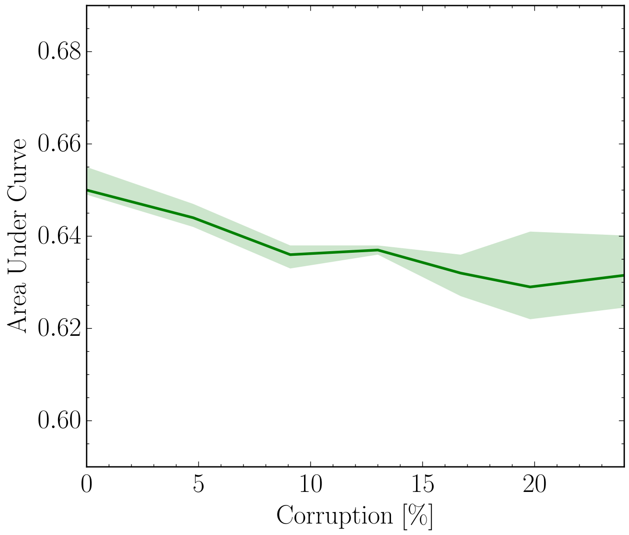


Figure 2.7: Effect of contaminating the training sample with an increasing fraction of signal events. The central line shows the overall AUC score while the band represents the difference between the upper and lower AUC scores.

expand our tests to include other new physics signals. A benefit of using the autoencoder over supervised techniques is that training is done using only background events. This allows it to be independent of the signal we wish to test on. We now introduce other models and test the autoencoder with them. Our choices are motivated by our aim to quantify the effects that a resonance may have on performance. The three models used are:

- Two other Z' cases. These have widths of 10 GeV and 200 GeV. In both cases the masses are held at 2 TeV.
- A scalar colour-octet [95]. This has a mass of 2 TeV. The scalar and axial parameters fixed to ensure the width is $\sim$ 89.6 GeV.
- A scalar colour-singlet with a mass of 2 TeV and a width of 89.6 GeV.

In all cases the autoencoder is trained on the same background samples as before. Table 2.1 shows the results of testing the model on these new signals. In all cases, the overall AUC is similar to the previous signal models. We see that in all scenarios the adversary can perform well, resulting in a low difference between the upper and lower

Signal	Overall AUC	Upper-Lower	Cross Section Limit (pb)
$Z'_{w=10\text{ GeV}}$	0.662	0.009	0.0101
$Z'_{w=89.6\text{ GeV}}$	0.656	0.009	0.0098
$Z'_{w=200\text{ GeV}}$	0.650	0.009	0.0105
Scalar	0.654	0.010	0.0104
Octet	0.659	0.010	0.0102

Table 2.1: The overall AUC score, difference between the largest and smallest AUC scores and the cross section limits found from using the adversarial autoencoder trained only on background events and tested on five signals.

curves. The table also provides estimates of potential limits. These are found from the classification performance of the autoencoder. These values are found by finding the point on the ROC plot curve that maximises $S/\sqrt{B}$. This is then compared to the background cross-section assuming an integrated luminosity of 100 fb^{-1}. The value of $S/\sqrt{B}$ is required to be greater than 2 to ensure a 95% confidence limit. We see that the limits are insensitive to our choice of signal and are comparable to limits found by ATLAS [96].

2.5 Conclusions

Ideally, machine learning algorithms would be used as part of a data-driven search. This allows for the use of experimental data rather than generated pseudo-data and hence removes the inclusion of any theoretical uncertainties from a Monte-Carlo generator. However, even in these scenarios, residual uncertainties from the reconstruction of the final-state objects can remain. These uncertainties in the data can bias attempts to make classification models. This chapter proposed a method of performing data-driven searches that are robust against systematic uncertainty. The proposed solution involves the use of an autoencoder. Autoencoders can be used to perform unsupervised anomaly detection. To incorporate the effect of uncertainties in our data we create extra training sets by smearing the features of our background set upwards and downwards. We then extend our autoencoder with an adversary, designed to remove any dependency the classification has on information found in the smearing.

We apply our model to semileptonic $t\bar{t}$ final state resonance searches. We show that an autoencoder (without an adversarial extension) can discriminate these events from Standard Model background and therefore perform this search. However, these classifications are strongly dependent on the smearing of the samples, showing the need to extend the model. When using the adversarial autoencoder model we find that the dependence on the smearing to make classifications is significantly decreased, at the cost of the model's predictive power. Our model is only slightly affected by corrupting the training set with signal, suggesting the method is appropriate to use with real data. Compared to a supervised model, the autoencoder has a weaker classification performance. However, the supervised method requires access to signal samples at training time while the autoencoder needs only to be trained on background data. This allows for data-driven searches using only experimental data to be carried out.

While we applied our method to Monte-Carlo generated pseudo-data, we believe the same process could be applied to experimental data also. Analogous to our process, labelled smeared datasets can also be created from real data, allowing an adversary to be trained. Our setup proves to be a robust and data-driven way to perform an anomaly detection procedure for new physics searches. The results are independent of the smearing of the reconstructed final state objects and can even be extended to corrupted backgrounds.

Chapter 3

Classification Using a Variational Quantum Classifier

Quantum computers have been demonstrated to have numerous advantages over classical computers. A branch of this research is quantum machine learning (QML). QML is a nascent field aiming to apply these quantum advantages to machine learning. In this chapter, we build a quantum computing model, based on the structure of a neural network, and use it to classify particle physics events. We develop the motivations to build such a model in Section 3.1. The model used in this chapter is known as a Variational Quantum Classifier (VQC) and is detailed in Section 3.2. Like a classical neural network, we have a choice in how to optimise our quantum neural network. One option is to construct a quantum equivalent of gradient descent, known as quantum gradient descent. This extension is introduced in Section 3.3. We describe the events we generate for classification in Section 3.4 and the results we find after training our model in Section 3.5. Finally, we end with a summary of the model and conclusions in Section 3.6.

3.1 Motivations

Chapter 2 introduced our first application of machine learning. We saw how neural networks, such as autoencoders, are powerful tools for classification and anomaly detection. In this chapter, we introduce how quantum computing can extend the performance of these networks. Techniques in HEP often rely on machine learning algorithms that show an ability to find correlations in high-dimensional parameter space. These can be used to discriminate signal from background processes. Algorithms can be trained to classify events based on the physical observables of reconstructed objects, e.g. the transverse momentum of a jet $p_{T,j}$ or the total missing transverse energy $\not{E}_T$. Just like traditional machine learning algorithms, quantum machine learning (QML) can be applied to these searches as well. Here, we will develop a method of QML to apply to new physics searches.

The construction of a quantum neural network will follow many of the same principles of a traditional artificial neural network (NN). As described in Chapter 1.3.1, these systems are built on three pillars:

i. An adaptable complex system that allows the approximation of a function. In the NN, this is the interconnected collection of nodes that form the network's layers.

ii. The calculation of a loss function from the output of the final layer which is used to define the objective of the NN algorithm. This function is minimised during the training of the network.

iii. A way to update the network during training, based on the result of the loss function e.g. through backpropagation. This aims to minimise the loss function.

Quantum machine learning is an emerging field aiming to use the prowess of quantum computing to improve machine learning methods. A quantum neural network combines quantum information processing principles and the pillars used to create artificial neural networks. With present and near-term devices, quantum algorithms can recreate individual pillars. This allows us to use hybrid quantum-classical machine learning approaches.

Novel techniques are being developed that can be used to support each of the pillars above. Pillar (i) can be performed on a quantum device by connecting trainable quantum nodes together. Usually, this is in the form of quantum gates that can evolve a state. These can form a variational circuit [97–99] or be combined with a classical neural network in a hybrid approach [100,101]. For pillar (ii), the calculation of a loss function, we can use a quantum algorithm such as a variational quantum algorithm approach [102,103]. Finally, to support pillar (iii), optimisation of the loss function can be done using a quantum annealer [104,105] or a quantum optimisation algorithm [106,107].

This chapter will therefore focus on combining quantum computing techniques with classical neural network principles. Specifically, we will focus on using the discrete, qubit-based, model of quantum computing. By applying our methodology to problems in particle physics, we aim to get a performance increase seen when sometimes using quantum computing algorithms. Overall, this will hopefully lead to improved sensitivity in searches for new physics.

An important task for machine learning algorithms in particle physics is classification. To aid this we will construct a novel hybrid neural network, based on a variational quantum classifier (VQC). These systems are known to have an advantage over a traditional neural network with respect to the model size [99]. Variational quantum classifiers have a similar structure to a traditional neural network. Therefore, they provide a good framework to discuss how the three pillars of classical neural networks can be supported using quantum information techniques. Finally, we extend our model with a quantum optimisation algorithm known as quantum gradient descent.

Our model will be applied to a Z' resonance search, which decays to a pair of top quarks [96,108,109]. We use the same events that were generated in the Chapter 2.

However, here, we limit ourselves to only using two features as input to our network - the transverse momentum of the hardest b jet p_{T,b_1} and the missing transverse energy in the event $\not{E}_T$. While it is possible to extend our feature space and potentially increase the networks predictive power, we are limited by the number of qubits available. Increasing the model size too far would prevent us from running the hybrid quantum neural network on a real device.

We compare the results from the VQC to a classical network with a similar number of trainable parameters. By comparing to a classical NN, we can investigate whether a quantum advantage arises from using a quantum neural network and quantum gradient descent. Since current devices are prone to noise errors and are limited in their number of qubits, we are prohibited from creating larger models. State-of-the-art networks that use a large number of input features could easily outperform the smaller VQC setups we present. However, we are optimistic that the advantages associated with using a VQC will still be relevant when larger quantum devices become more readily available.

3.2 Structure of a Variational Quantum Classifier

Variational quantum classifiers are a form of quantum neural network that can be used for supervised classification. This is achieved by designing a quantum circuit that behaves similarly to a traditional machine learning algorithm. Quantum circuits are built from quantum gates designed to evolve the state of a qubit. This evolution will be determined by the gate itself and the gate's parameters. The quantum circuit will therefore also depend on the quantum gate's parameters. We can treat the network similarly to a neural network and apply a training procedure to it. During training, the circuit parameters can be optimised to reduce the value of a loss function. This is analogous to the weights in a neural network being optimised at training time. The trained circuit is described by the form

$$f(w, b, x) = y, \tag{3.2.1}$$

where f is the network with trainable parameters w and b. Then, there is the input data x, and y is the network output used to calculate the loss function L. Thus, the structure of a VQC shares many similarities with a traditional neural network. In both cases, the network f is built from discrete modular blocks (nodes in the classical neural network and quantum gates in a quantum neural network). Furthermore, both the classical and quantum models learn by minimising a loss function.

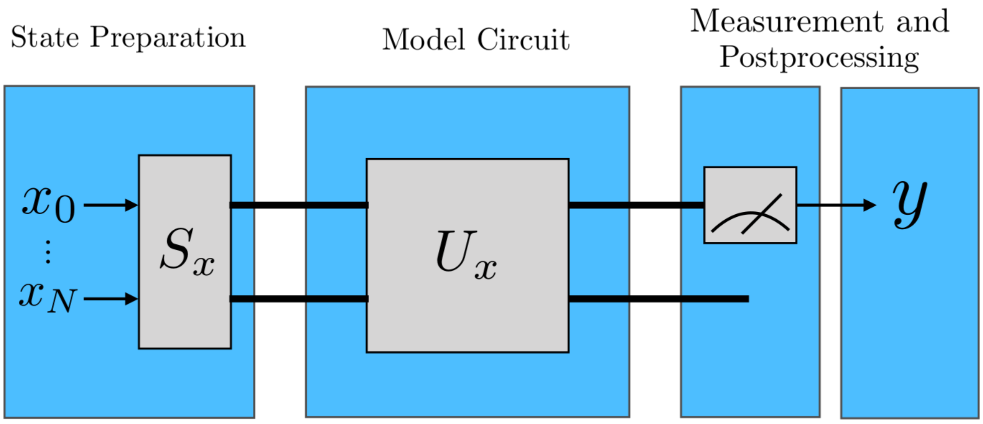


Figure 3.1: A variation classifier described by 3 parts. The state preparation circuit is desgined to take our input, x, and encode it on a N-qubit quantum state. The model circuit will apply trainable and non-trainable gates to this state. In the final steps we measure the states and apply any postprocessing necessary.

Our classifier is designed as a circuit-centric quantum classifier [99]. It is structurally depicted in Fig. 3.1 and consists of three parts: (1) the *state preparation circuit*, (2) the *model circuit* and (3) *measurement and postprocessing*. These three parts can all be related to the three pillars of machine learning, discussed in Section 3.1. Our classifier corresponds to (i) a complex adaptable system that calculates input to the (ii) loss function. The optimisation of the trainable parameters w and b relates to (iii).

More specifically, the state preparation step encodes the input data to an N-qubit quantum state. Then, the model circuit is constructed from gates that evolve the input state. The circuit is based on unitary operations and depends on external parameters that will be adjusted during training.

Finally, the postprocessing step measures the state. Traditionally, we measure the output of the first qubit. This step will also include any classical postprocessing we may wish to include.

3.2.1 State Preparation

Before applying the model circuit of our classifier (the main trainable part of our model), we use a state preparation circuit S_x to encode the input data into a quantum state. S_x acts on the initial state $|\phi\rangle$ as

$$x \mapsto S_x|\phi\rangle = \ S_x|0\rangle^{\otimes n} = |x\rangle \ , \tag{3.2.2}$$

where $|\phi\rangle = |0\rangle^{\otimes n}$. The number of qubits n is defined by the number of features in our dataset (i.e. since we have two features in the dataset we will require only two qubits).

There are many options on how to encode data into a quantum circuit. The choice of parametrisation of the encoding can potentially influence the decision boundaries of the classifier [110]. Here, we use the angle encoding

$$|x\rangle = \bigotimes_{i=1}^{n} \cos(x_i)|0\rangle + \sin(x_i)|1\rangle \ , \tag{3.2.3}$$

where $x = (x_0, ...x_N)^T$. Practically, this amounts to using the input data x as angles in a unitary quantum gate. We take the state preparation circuit as the rotation unitary gate

$$R_y(\theta) = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \ . \tag{3.2.4}$$

3.2.2 Model Circuit

Given a prepared state, $|x\rangle$, the model circuit, $U(w)$, maps $|x\rangle$ to a vector $|\psi\rangle = U(w)|x\rangle$. In turn, $U(w)$ consists of a series of unitary gates and can be decomposed as

$$U(w) = U_{l_{\max}}(w_{l_{\max}})...U_l(w_l)...U_1(w_1) \ , \tag{3.2.5}$$

where every $U_l(w_l)$ is a layer in the circuit, with its corresponding weight parameters, and $l_{\max}$ is the maximum number of layers. These are constructed from a set of single and two-qubit gates which will evolve the state $|x\rangle$. The gates include parameters that will be trained during the optimisation of the network. A single qubit gate can be written as a 2×2 unitary matrix with the form

$$G(\alpha, \beta, \gamma, \phi) = e^{i\phi} \begin{pmatrix} e^{i\beta}\cos(\alpha) & e^{i\gamma}\sin(\alpha) \\ -e^{-i\gamma}\sin(\alpha) & e^{-i\beta}\cos(\alpha) \end{pmatrix} . \tag{3.2.6}$$

We can neglect $e^{i\phi}$ as it only gives rise to a global phase that has no measurable effect. Thus, the parameters α, β, and γ are all that is needed to parametrise a single qubit gate.

The circuit we use in our model in shown is Fig. 3.2. This is constructed using a rotation gate, R, and CNOT[1]. The rotation gate is a single qubit gate that is applied to both qubits in our system. This gate is designed to rotate our state based on a

[1] The controlled-NOT (CNOT) gate is a quantum register that can be used to entangle and disentangle quantum states. The matrix representation of a CNOT gate is

$$\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} .$$

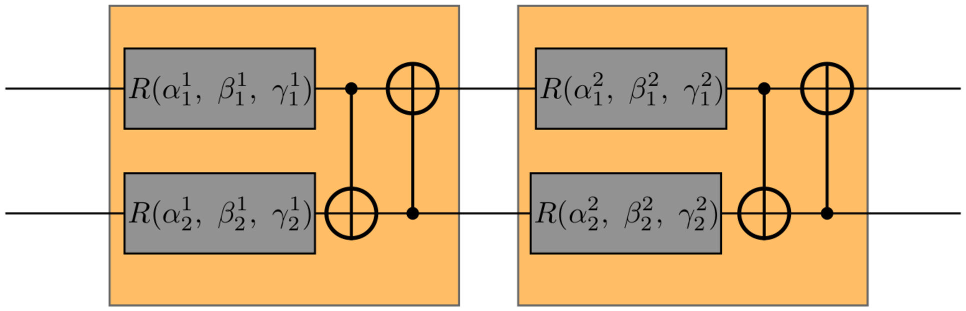

$R(\alpha_1^1,\ \beta_1^1,\ \gamma_1^1)$
$R(\alpha_2^1,\ \beta_2^1,\ \gamma_2^1)$
$R(\alpha_1^2,\ \beta_1^2,\ \gamma_1^2)$
$R(\alpha_2^2,\ \beta_2^2,\ \gamma_2^2)$

Figure 3.2: Circuit diagram for our variational quantum classifier model made of two qubits in each of the two layers.

set of learnable parameters $w = (\alpha,\ \beta,\ \gamma)$

$$\begin{aligned} R(\alpha,\ \beta,\ \gamma) &= R_Z(\gamma)R_Y(\beta)R_Z(\alpha) \\ &= \begin{pmatrix} e^{-i(\alpha+\gamma)}\cos(\beta/2) & -e^{-i(\alpha-\gamma)}\sin(\beta/2) \\ e^{-i(\alpha-\gamma)}\sin(\beta/2) & e^{i(\alpha+\gamma)}\cos(\beta/2) \end{pmatrix} \end{aligned} \tag{3.2.7}$$

The angles of Eq. (3.2.7) are a subset of all trainable parameters of the model and make up the parameters in the weight vector $w \in \mathbb{R}^{n\times 3\times l}$, where n is the number of qubits and l is the number of layers in our network. This object, w, will contain some of the parameters that will be learned during training time. While the number of qubits will mirror the number of features in our dataset, the number of layers in the network, l, is a hyperparameter we can tune. In the circuit centric design we are using, the number of qubits is held constant, however, the model could be extended for a more flexible network design [98].

Each layer in our model contains two CNOT gates - a standard 2-qubit gate in quantum computing with no learnable parameters. These gates flip the state of a qubit based on the value of another control bit. Each gate in the layer uses a different qubit as the control bit.

3.2.3 Measurement and Postprocessing

After applying $U(w)$ to the initial state we need to measure its output. We do this by applying the Pauli Z operator on the first qubit and taking the expectation value

$$\mathbb{E}(\sigma_z) = \langle 0|S_x(x)^\dagger U(w)^\dagger \hat{O} U(w) S_x(x)|0\rangle = \pi(w,x) \;, \tag{3.2.8}$$

where $\hat{O} = \sigma_z \otimes \mathbb{I}^{\otimes(n-1)}$. To obtain an estimate, we run the circuit repeatedly. The number of repetitions we do is known as the number of shots S.

Classical postprocessing is applied to the expectation value of the circuit before returning a final classifier output. Like in a classical neural network approach, the postprocessing step gives a great deal of flexibility to the user to tackle the problem how they see fit. Generally, it will include the addition of any bias terms, the drawing of a classification decision boundary, the calculation of a loss function and the optimisation procedure.

The bias term b will also be a trainable parameter. Its introduction increases model flexibility. We can write the output of our model, before drawing a decision boundary, by combining the expectation value of the model circuit $\pi(w,x)$ and the bias term b

$$f(w,b,x) = \pi(w,x) + b \;. \tag{3.2.9}$$

A decision boundary is drawn to seperate the value of $f(w,b,x)$ into the two classes. The binary classification result, $\mathrm{cls}(w,b,x)$, is calculated as

$$\mathrm{cls}(w,b,x) = \begin{cases} 1 & \text{if } f(w,b,x) > 0 \;, \\ \\ -1 & \text{else} \;. \end{cases} \tag{3.2.10}$$

Following this, the loss function is calculated and the optimisation procedure is carried out. This will be discussed in Section 3.3.

3.3 Optimisation

As alluded to above, during training we aim to find values of w and b to optimise a given loss function. This is analogous to a traditional neural network. In both cases, the methods of optimisation you can perform are similar. For a quantum neural network and a traditional neural network, we perform a forward pass of the model and calculate a loss function. Then, we can backpropagate through the network and update the trainable parameters. This is the equivalent of the third pillar of machine learning, mentioned in Section 3.1.

During training we have chosen to use mean squared error (MSE) as the loss function[1]. The allows us to measure a distance between the truth and our model's predictions, represented by the value of the function

$$L = \frac{1}{n} \sum_{i=1}^{n} \left[y_i^{\text{truth}} - f(w, b, x_i) \right]^2 \ . \tag{3.3.1}$$

We train our model using vanilla gradient descent and quantum gradient descent [107]. The latter is a quantum optimisation algorithm designed to be performed on a hybrid network such as the model we have proposed.

3.3.1 Backpropagation

To perform backpropagation for a network with adjustable parameters $\theta = (w, b)$ we must compute the gradient $\frac{\partial}{\partial \theta} f$. This is equivalent to computing the change of the output of the network when varying θ. The gradient over a quantum circuit can be calculated using the parameter-shift rules [112, 113]. Being able to calculate gradients for a quantum circuit opens up the possibility of using gradient descent methods to train our variational quantum circuit. The methodology is identical to how optimisation and training techniques are performed on classical neural networks.

[1] As discussed in Chapter 1.1, the binary cross-entropy is a preferred measure for the loss function. In this case, we find that the choice of BCE or MSE leads to similar results. As a result, we choose to follow the choice for the loss function of Refs. [99, 111]. On testing the difference, we find that either loss function results in a model performance of around 70% accuracy.

The parameter-shift rules provide the relation

$$\frac{\partial}{\partial\theta} f = r \left[f(\theta + s) - f(\theta - s) \right] \,, \tag{3.3.2}$$

where the shift $s = \pi/4r$. The value of r is an arbitrary normalisation factor which we choose in our implementation to be $r = 1/2$[1]. From Eq. (3.3.2) we can calculate gradients over quantum gates by shifting their parameters. As the difficulty of calculating $\frac{\partial}{\partial\theta} f$ has been reduced to simply probing the quantum circuit at different parameter points, it is now possible to evaluate the gradient on a quantum device.

3.3.2 From Classical to Quantum Gradient Descent

The geometry of the parameter space has a direct impact on the reliability and efficiency of an optimisation algorithm [114]. Therefore, a suitable choice of optimisation strategy is a key performance factor for a variational quantum circuit. It is an open question as to what is the best form of parameter space to use and whether the use of a traditional Euclidean geometry is appropriate for variational models [115].

For our problem, we propose to augment the vanilla gradient descent method, often used in classical neural networks, with a quantum gradient descent method [107].

In vanilla gradient descent, the network parameters θ_t, at each iteration step t, are updated to θ_{t+1}. The goal is to choose the parameters θ_{t+1} such that the loss function $L(\theta)$ is minimised. One approach is to update θ_t in the direction of the steepest decline, $-\nabla L(\theta)$, weighted by a learning rate η

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta). \tag{3.3.3}$$

While all parameters are updated with the same step size, the rate at which the loss function changes for each model parameter can vary by large amounts. By using this form of gradient descent it is possible to miss the global minimum in the space

[1] The rule is similar to the traditional finite differences (FD) method of finding a derivative. However, unlike the parameter-shift rules, FD is an approximation. Also, parameter-shift requires a shift of $\pi/2$ while the shift in a FD setup must be $<< 1$.

of the loss function.

An improvement would be to change the coordinate system to ensure the loss function changes consistently with each step, for each parameter. One way to address this problem is to use natural gradient descent, which makes use of the Fisher Information Matrix [116, 117]. This is a classical extension to vanilla gradient descent method.

When a new set of parameters are proposed it is possible to compare the new output distribution from the model to what it previously was. A distance between these two distributions can be found using the KL-divergence. The goal of this form of gradient descent then is to limit how far a new distribution can be from the previous one, using this distance. The Fisher Information Matrix (which can be derived from the KL-divergence) is used to weight the gradient, ensuring the output distribution of the network always shifts by the same amount as the parameters are updated. We have gone from the parameters of the network always being updated by the same step size to the distance of the network's output always being shifted the same amount.

Algorithmically, natural gradient descent can be written as

$$\theta_{t+1} = \theta_t - \eta F^{-1} \nabla L(\theta) \ , \tag{3.3.4}$$

where F is the Fisher Information Matrix. In each optimisation step, the parameters are updated in the direction of steepest descent of the information geometry rather than the Euclidean geometry. The inclusion of F^{-1} in Eq. (3.3.4) generally improves the performance of the optimisation algorithm. In most classical deep neural networks, calculating the inverse of a large matrix becomes prohibitively expensive because of the computations involved. However, in our hybrid network, which benefits from a small model size, it follows that the parameter space will also be small. Thus, our aim is to use a quantum optimisation equivalent of this method that we can use on variational circuits.

The parameter space of quantum states has can be described by an metric. Similar to how the Fisher Information Matrix is used to promote the gradient descent method to the natural gradient descent method, the Fubini-Study metric g (derived and elaborated on in Appendix A) exploits the geometric structure of the variational quantum classifier's parameter space to establish the quantum gradient descent method. Here, the optimisation algorithm reads [107]

$$\theta_{t+1} = \theta_t - \eta g^{+} \nabla L(\theta) \ , \tag{3.3.5}$$

where g^{+} is the pseudo-inverse of the Fubini-Study metric. We implement this using the Python package PennyLane [118]. This allows us to find the steepest descent in the parameter space of the quantum states. The approach of Eq. (3.3.5) is designed to optimise the parameters of the quantum variational circuit only, i.e. the quantum gates with trainable parameters $w = (\alpha, \beta, \gamma)$. To perform a full optimisation of our hybrid model, we need to consider the classical components of our model - the bias term. Thus, we propose to optimise our weights using quantum gradient descent (3.3.6) while using vanilla gradient descent for the classical bias term b. By calculating both gradients at each optimisation step,

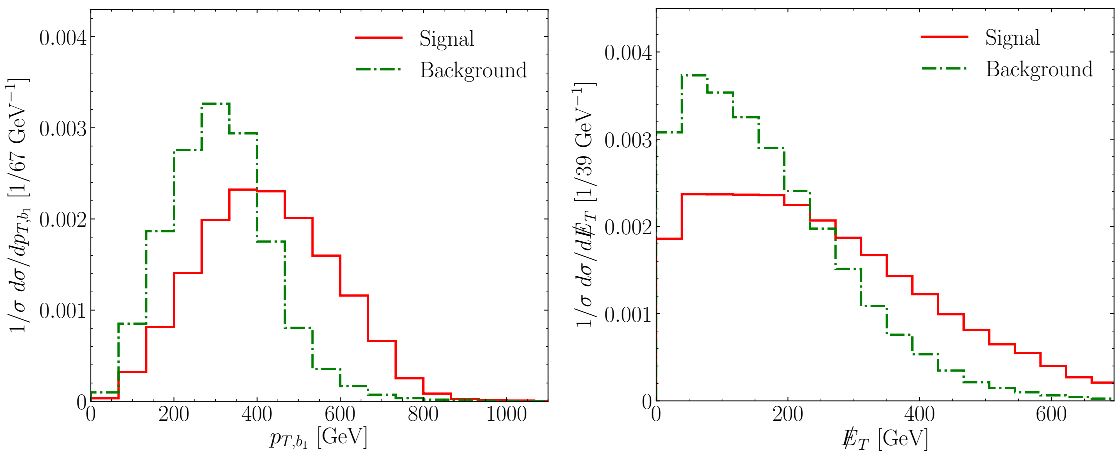
$$\begin{aligned} \theta^w_{t+1} &= \theta^w_t - \eta g^{+} \nabla^w L(\theta) \ , \\ \theta^b_{t+1} &= \theta^b_t - \eta \nabla^b L(\theta) \ , \end{aligned} \tag{3.3.6}$$

we can be sure that our entire range of parameters is optimised simultaneously.

3.4 Analysis Setup

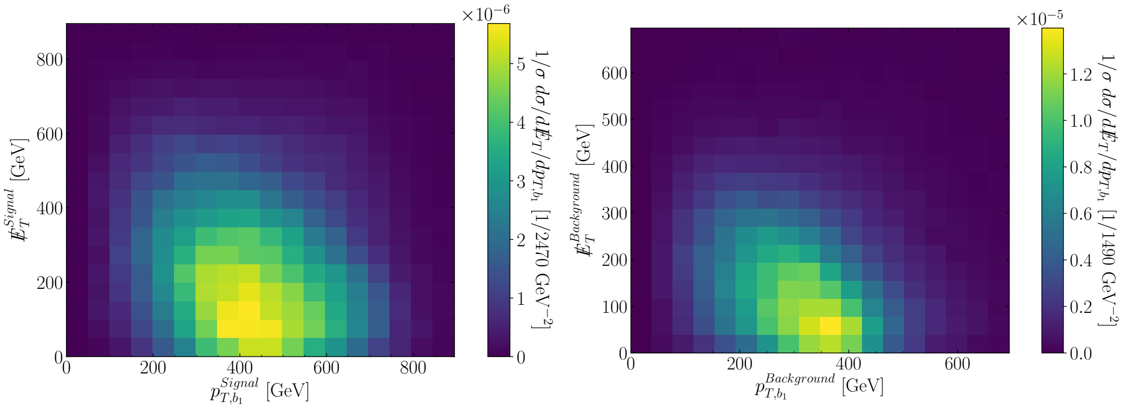
Our analysis will be performed using background and signal samples consisting of $pp \to t\bar{t}$ events and $pp \to Z' \to t\bar{t}$ events, respectively. These events are generated using the same method as found in Chapter 2.2.

The analysis is based exclusively on the transverse momentum of one b-jet, p_{T,b_1}, and the event's missing energy, $\not{E}_T$. We show the distributions of these observables in Fig. 3.3 and their heatmaps in Fig. 3.4.



(a) (b)

Figure 3.3: Distribution of signal and background of the (a) p_T of the hardest b-jet and the (b) missing energy.



(a) (b)

Figure 3.4: Heatmaps of signal and background of the (a) p_T of the hardest b-jet and the (b) missing energy.

Our data x is normalised using min-max scaling such that $x_{\text{scaled}} \in [0, \pi]$. This allows our features to be encoded as an angle in a qubit rotation when we begin training. The target labels are defined as -1 for the background set and 1 for the signal set.

3.5 Network Performance

We compare three models: a classic neural network with vanilla gradient descent (NN-GD), a VQC with vanilla gradient descent (VQC-GD) and a VQC trained using our quantum gradient descent implementation (VQC-QGD).

The VQC is built using two qubits, each corresponding to a feature in the samples (p_{T,b_1} and $\not{E}_T$) and two layers. Each layer consists of a rotation gate for each qubit followed by two CNOT gates. This model, depicted in Fig. 3.2, is implemented using PennyLane [118]. It was trained for 30 epochs using a batch size of 32 events and an initial learning rate of $\eta = 0.01$. A learning rate reduction schedule is implemented throughout training, where the learning rate is reduced whenever the loss plateaus. However, in this instance, it appears to have little effect on the performance of the trained network. The networks poor capacity to discriminate signal from the background is reflective of the similarities of the two sets. Fig. 3.4 shows the probability density for the events to populate areas in the feature space (p_T, $\not{E}_T$).

We anticipate that a significant advantage of the variational quantum classifier lies in its smaller network structure. This allows us to employ computationally more expensive optimisation algorithms (as detailed in Section 3.3), which in turn will give rise to a faster training time (in terms of the number of iterations required). Such a method would be particularly advantageous in cases where one has to train directly on a limited amount of data, e.g. rare decays.

To compare the quantum and classical networks ability to learn quickly, we use a total of 2500 events for the signal and background samples respectively. We impose a 60-20-20 split between training-validation-test sets, i.e. we train on 1500 events. To get an understanding of the effect that the size of training samples have on the model performance, we train another set of models using only 500 events each. While training is carried out on PennyLane's inbuilt simulator, we test the model's performance on the PennyLane simulator, the IBM Q simulator[1] and IBM Q Yorktown[2]. Accessing the IBM hardware was done through PennyLane's Qiskit plugin [119, 120]. For all backends, in training and testing, we use a total of 8192

[1] 32-qubit backend: IBM Q team, "IBM Q simulator backend specification V0.1.547," (2020). Retrieved from https://quantum-computing.ibm.com

[2] 5-qubit backend: IBM Q team, "IBM Q 5 Yorktown (ibmqx2) backend specification V2.1.0," (2020). Retrieved from https://quantum-computing.ibm.com

shots.

For a baseline, a classical neural network is trained using vanilla gradient descent for optimisation. To provide as fair a comparison as possible, the network is constructed to have a similar number of trainable parameters as the VQC model. The classical network has one hidden layer with three nodes with a ReLU activation function. The remaining hyperparameters match those that were chosen for the VQC model. This was implemented using Keras [121] with a TensorFlow backend [122].

Training with the classical neural network was found to be unstable, sometimes resulting in the loss plateauing at around 1. To attempt to account for this, we trained 15 separate instances of the model. The results presented in Fig. 3.5 show the average loss from these runs for each model. From Fig. 3.5, we see that optimising using the quantum gradient method leads to a faster convergence than when using the traditional gradient descent optimisation. Furthermore, the VQC outperforms the classical neural network trained with traditional gradient descent.

Out of each of the three sets of 15 trained models, one was chosen that had a loss value that had converged to a point during training that was similar to the average. These models were used for testing. In Fig. 3.6 we see an example of the variational classifier output before the decision boundary is applied (a) and the ROC curve for the chosen VQC-QGD, VQC-GD and NN-GD models (b). While the model was trained on a simulator, we can test it on a variety of devices. Table 3.1 shows the performance of the quantum gradient descent method when the test data is applied to it for different devices. We see that the model, trained on the simulator, still performs well on the real hardware.

Device	**Accuracy (%)**
PennyLane default.qubit	72.6
ibmq_qasm_simulator	72.6
ibmqx2	71.4

Table 3.1: Test set results from model trained with quantum gradient descent sent to PennyLanes in-built simulator, IBM Q simulator and IBM Q Yorktown (ibmqx2).

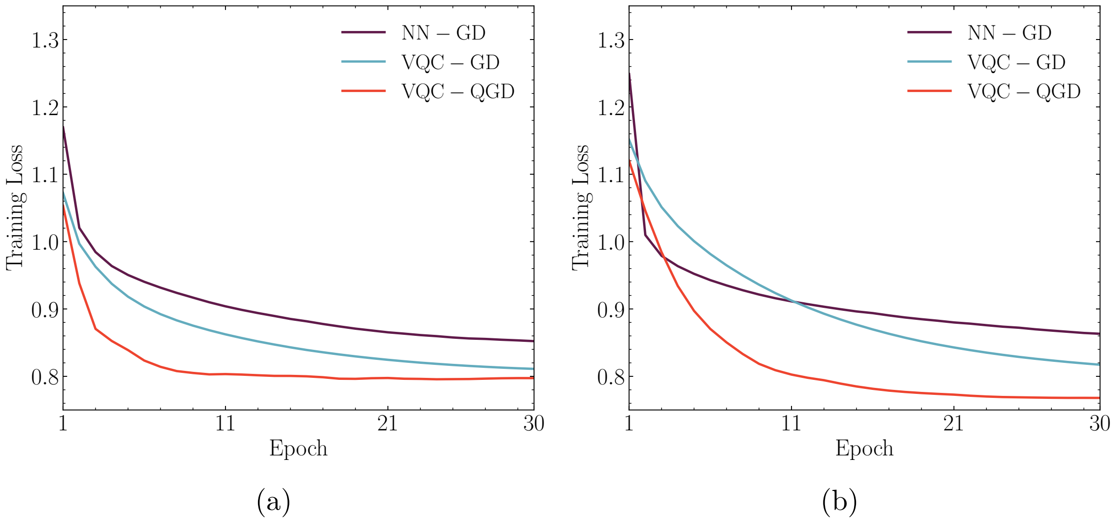


Figure 3.5: Comparison of the averaged training history for 15 runs of the QVC models trained with quantum gradient descent, QVC models trained using vanilla gradient descent and the classical NN models. Figure (a) show models trained with 1500 samples and Figure (b) shows models trained with 500 samples.

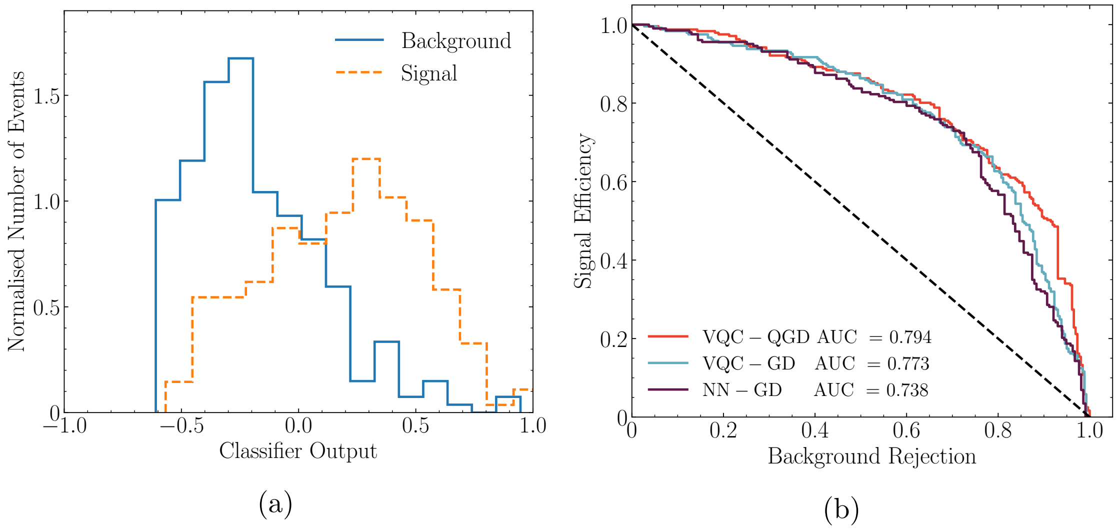


Figure 3.6: (a) Output of a QVC model trained with quantum gradient descent and (b) ROC curve for a QVC model trained with quantum gradient descent, a QVC model trained with vanilla gradient descent and the classical NN.

3.6 Conclusions

Classification of rare signal events from standard model background is an important part of machine learning algorithms in collider phenomenology. Recently, more effort has been dedicated to the development of novel techniques to find correlations in high-dimensional parameter spaces. In this chapter, we present a novel quantum-classical hybrid neural network. Models such as the one developed make up part of what is known as quantum machine learning (QML). This is the emerging field aimed at applying quantum computing benefits to machine learning. By applying the power of quantum computing to machine learning, it is hoped that one can create classification techniques which will increase sensitivity in new physics searches.

The model proposed here is based on a variational quantum classifier. Variational quantum classifiers are in many ways analogous to classical neural networks. An advantage that a VQC classifier provides over a classical neural network is its small model size. The model shown here uses a quantum algorithm equivalent to natural gradient descent. Typically, due to the need to invert large matrices, natural gradient descent is computationally prohibitive when training neural networks. However, thanks to the model-size advantage of the VQC, we can make use of quantum gradient descent to optimise our network.

We combine the use of quantum gradient descent to optimise the quantum gate parameters in the model with classical gradient descent to optimise the classical bias term. This model was used to perform a Z' resonance search. We compared the performance of this model against a purely classical neural network and a VQC optimised with standard gradient descent. The hybrid approach proved successful in maximising the learning outcome. The hybrid approach learns faster than an equivalent classical neural network or the classically trained VQC. Even on small data samples the hybrid VQC still retains a high classification ability. While we applied this methodology to generated data, we believe this approach can prove useful in data-driven classification problems where there is a small amount of data available.

Chapter 4

Graph Classification on Photonic Quantum Computers

In Chapter 3 we discussed using the discrete, qubit-based, model of quantum computing to perform event classification. In this chapter, we will branch out to a separate quantum computing paradigm. While the continuous variable (CV) paradigm is distinctly different to the discrete model it still maintains a set of advantages over classical devices. One such advantage is demonstrated through Gaussian Boson Sampling (GBS) - a sampling method difficult to perform classically. We propose using a novel combination of Gaussian Boson Sampling and Q-means classification (a quantum equivalent of K-means) to perform anomaly detection on events stored as graph data. We motivate these decisions and give a more in-depth introduction to these ideas in Section 4.1. The events we generate, and their subsequent transformation into graphs, are detailed in Section 4.2. In Section 4.3 we present a classical anomaly detection methodology we can use as a baseline. A discussion on CV quantum computing and how GBS can be used to generate samples for use in anomaly detection models is given in Section 4.4. A second anomaly detection method is based on the use of Q-means and is detailed in Section 4.5. Finally, a summary and conclusion is given in Section 4.6

4.1 Motivations

Chapter 3 demonstrated how to use a quantum computer to aid new physics searches. We showed that the use of a quantum device can allow a performance boost to be achieved when training a machine learning model. Furthermore, in Chapter 2 we discussed the importance and potential benefits of performing anomaly detection. Unsupervised anomaly detection methods have the advantage over supervised techniques by not requiring any signal data during training. In our case, this means that the method requires only Standard Model background data, which can be easily generated. Using these unsupervised anomaly detection techniques also opens up the possibility of performing a data-driven search, i.e. training directly on LHC data. In this chapter, we combine these two concepts and perform unsupervised anomaly detection using quantum information concepts.

We begin by careful consideration of how our data is represented. Data in HEP very naturally fits into a graph structure, i.e. particle decays, or final states. The benefit of a graph structure is in its ability to give a natural method of defining not only individual constituents of an event but also their relationship to each other. Graph structures have proven to be a powerful way to encode particle physics events [123–125]. We will make use of this here.

Chapter 3, and previous work in general, focuses largely 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 computing differs by its use of qumodes over qubits [65]. Here, data is embedded into a qumode, an infinite-dimensional object. This is typically an electromagnetic field, allowing CV devices to be constructed using quantum photonics hardware. While the discrete and continuous paradigms have different theoretical foundations, programming photonic devices proceeds in similarity to qubit-based models. Both methods allow for the construction of a quantum circuit out of a series of gates. As qumodes are represented by an infinite-dimensional uncountable basis, the CV model can be seen as a more natural choice to simulate continuous systems. Of particular interest to this chapter are CV applications using graph data [126] and machine learning [127].

A photonic device uses single photon emissions, manipulated through squeezers and beamsplitters. Even a device that uses a non-interacting source of photons can still exhibit quantum properties such as entanglement. An example of this is boson sampling [66]. Boson sampling techniques, such as Gaussian boson sampling (GBS), is an application of continuous-variable quantum computers where there is a demonstrated quantum advantage [128]. A GBS device emits photons into an interferometer and generates a sample by counting the photons that exit the device. This can be constructed as a quantum circuit for a photonic device. The advantage comes from the difficulty of classically simulating this process. To simulate a Gaussian boson sampling probability distribution requires the calculation of the hafnian, a #P problem[1].

However, when access to GBS devices become more accessible, samples can be generated at the rate of 10^5 every second [67]. These fast response times could allow GBS-based anomaly detection methods to be suitable to implement 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. A GBS device could produce around $100,000$ samples in a similar time frame.

We aim to harness the power of continuous-variable quantum computing, specifically Gaussian boson sampling, to survey particle physics events represented as graphs. By embedding graph data into such a device we can generate a lower-dimensional representation of the data. Matrices representing the graphs are embedded into the GBS devices interferometer, thus guiding the probability distribution associated with detecting photons when they exit the device. As the photon counts will now be

[1] This refers to a class of complexity problem. A P problem is one which an algorithm can provide a solution in polynomial time. A related category is a NP problem - one which can be verified in polynomial time. The category of #P relates to counting problems, asking how many solutions a problem may have.

influenced by the graph data these samples can be used to create a lower-dimensional representation of the graph. These samples could then be used as input into various anomaly detection techniques.

To show these samples allow for predictions to be made, and that GBS sampling is a viable technique for embedding particle physics events, we first present a simple anomaly detection method built using the K-means clustering algorithm. We then expand this to a quantum extension of K-means known as Q-means. This method can be implemented on both a qubit-based, and qumode-based, quantum device. In its most basic form, Q-means presents a substantial advantage over K-means. Typically, in K-means, the time complexity of the algorithm grows linearly with the size of feature vector. In Q-means it grows logarithmically, an exponential improvement [129, 130].

The method of anomaly detection presented here has three steps: (i) the creation of the data and graphs, (ii) embedding the graphs into a lower-dimensional representation using the GBS sampling device and (iii) the classification procedure. The feature vectors created in step (ii) can be used in any standard machine learning anomaly detection approach (such as the one presented in Chapter 2). Here, however, we will use a method based on K-means and Q-means.

These steps are applied to a search for hadronically decaying pseudoscalar resonances produced via $H \to 2A \to 4X$ (where A is a pseudoscalar and X a SM state) in the $pp \to HZ$ channel [131, 132]. We shall consider the scenario where the pseudoscalars decay into gluons, described as a "buried Higgs" model [133]. Our analysis will involve investigating jet substructure to uncover this decay in the data. 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 an event's constituent's features (i.e. invariant mass $m_{i,j}$ or the distance ΔR). Using GBS sampling, we find lower-dimensional representations of this data that we will use to perform anomaly detection. As a baseline, we create lower-dimensional feature vectors using the matrices' eigenvalues and classify them using the K-means clustering algorithm. Then, we propose a novel method using GBS to create the embeddings and perform anomaly detection using Q-means. We find the GBS embedding method performs favourably compared to the classical version. To be clear, the sections introducing the concepts of adjacency matrices, CV quantum computing and GBS act as a short literature review. Using GBS in the context of particle physics, and having the generated embeddings used as input to K-means and Q-means anomaly detection are novel methods that we are proposing.

4.2 Analysis Setup

4.2.1 Data Generation

To use as 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 $A_1 \to gg$. 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 to be 60 GeV. Events were generated, and parton showering performed, using Pythia 8.2 [134]. 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. Their decay to gluons could still be prompt, whereas their direct production cross-section in proton collisions is tiny. In such scenarios, the observation of A_1 or A_2 would have to proceed through Higgs decays.

Our analysis follows the methodology of using the jet substructure to find the Higgs [135–137]. We will cluster our events into a fat jet, before reclustering its contents into "microjets" [80, 136]. First, though, we impose a rapidity cut of 2.5

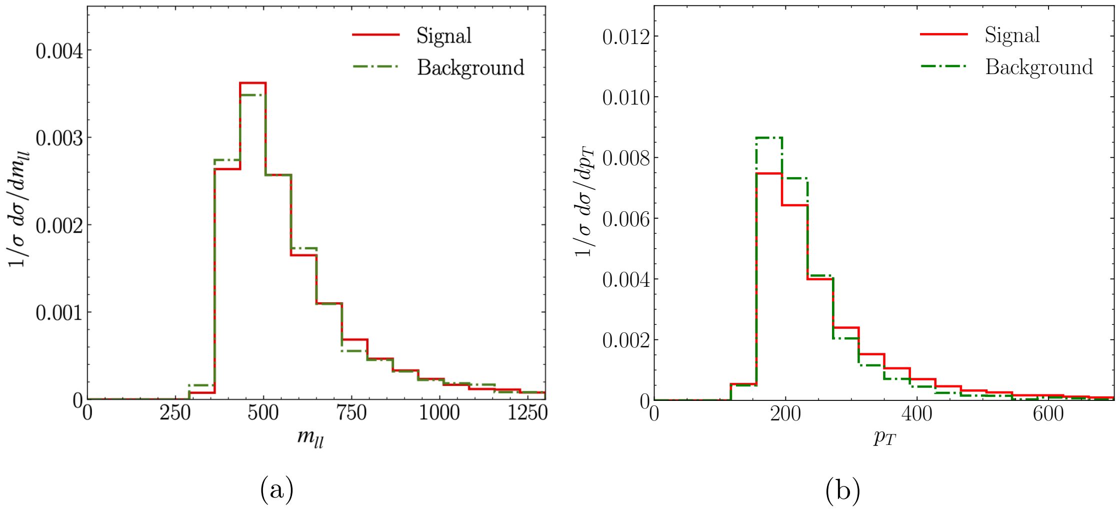
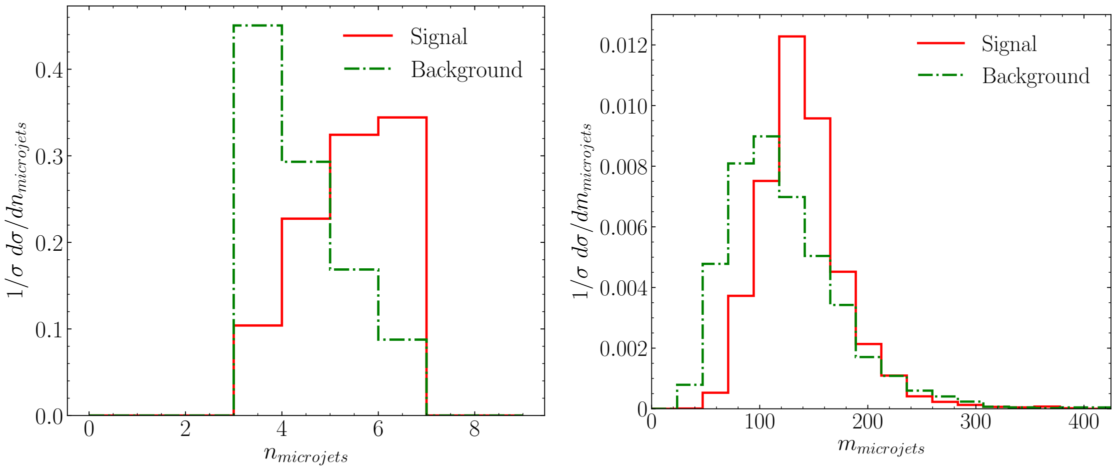


Figure 4.1: (a) shows the invariant mass of the leptons and fat jet while (b) shows the transverse momentum of the fat jet.

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 $80 \leq m_{ll} \leq 100$ 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 [138] we cluster these objects into jets using the Cambridge-Aachen algorithm with $R = 1.5$ [139]. We demand that there is at least one jet in the event with a transverse momentum of $p_T > 150$ GeV. Fig. 4.1 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 [140]. Here, we choose $R = 0.2$ and force a transverse momentum of $p_T > 5$ GeV. Of what remains, events with 3-6 microjets are selected for analysis. This is motivated by constraints we will face when trying to embed larger graphs into a GBS device: larger graphs begin to take inhibitive amounts of time to be sampled on a simulator. The number of jets chosen strikes a balance between the length of time taken and being able to capture the maximum amount of information on the event. The number of microjets found, and the total mass of a fat jet's microjets is



(a) (b)

Figure 4.2: (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 4.1: Cut efficiencies relative to the previous event reconstruction step for signal and background events.

shown in Fig. 4.2. These microjets are the objects used to construct our graphs. The result of the cuts we have applied is shown in Table 4.1. Here, we see the fraction of remaining events after each constraint is applied.

4.2.2 Constructing the Graphs

Graphs can be constructed from our events using only final states [123, 124]. Specifically, we use the microjets found from the procedure detailed in section 4.2.1 as the nodes in a graph. We choose to connect these nodes found from the final state to every other node.

These graphs can be used to generate weighted adjacency matrices that capture the information found in the graph and hence the event's final state. Adjacency matrices

	0	1	2	3	4
0	1	1	1	1	0
1	1	1	1	0	0
2	1	1	1	0	0
3	1	0	0	1	1
4	0	0	0	1	1

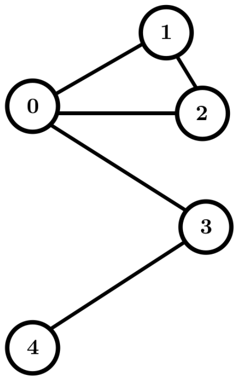


Figure 4.3: An example of an adjacency matrix and its corresponding graph.

are matrix representations of a graph where the rows and columns are defined by the graph nodes. Usually, 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. An example is shown in Fig. 4.3.

Fig. 4.4 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.

In some graph classification methods, the graph information is separated into feature and adjacency matrices [141]. 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, many graph sampling techniques require symmetric matrices as input and therefore a feature matrix cannot be easily included. We are limited to how we can manipulate adjacency matrices. 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. This does not require the use of a model as complex as a neural network. We aim to simply find a representation of the graph that will be useful for anomaly detection tasks.

To produce the first matrix in our set, 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

$$\Delta R = \sqrt{\Delta y^2 + \Delta\phi^2}, \tag{4.2.1}$$

where Δy is the difference in rapidity between particle i and j, while $\Delta\phi$ is the difference between their azimuthal angles. The dimensions of the resulting matrix will be determined by the number of nodes in the graph. The entries of the matrix will be weighted using Eq. (4.2.1). Eventually, the matrices will need to be compared to others derived from different events. To do this reliably it will be useful to ensure all matrices have the same dimensions. Therefore, we pad each matrix with zeros on the right and lower sides such that they all have dimensions of 6×6. To be thorough, we can also construct weighted matrices from $\Delta\phi$ and Δy. Other adjacency matrices can be constructed using a measure of 2 object's energy E_iE_j or invariant mass $m_{i,j}$.

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

$$s = 1/x, \tag{4.2.2}$$

where x is the maximum value in the set of matrices made from the background samples. For example, the $m_{i,j}$ matrices are scaled by a term $s_{m_{i,j}}$, 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. Of course, we cannot take x to be the maximum value in the signal set, as this information would be hidden from us. In data-driven anomaly detection methods, only background data is used during training.

Fig. 4.4 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 4.2.1 to 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.

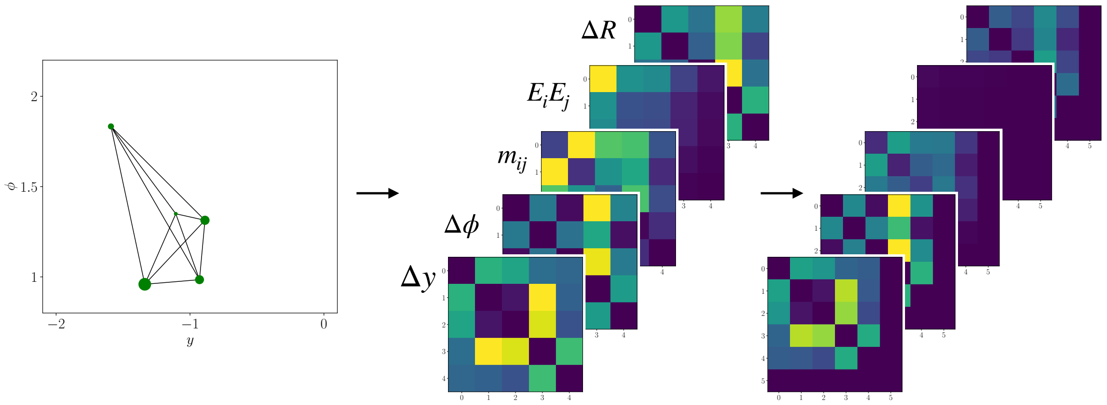


Figure 4.4: For a graph we find five weighted adjacency matrices, each one capturing features of the graph nodes. These five are then scaled and padded, giving the final matrices on the right. These can be embedded and used as input to classification algorithms.

3. 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 set is what is shown shown as the final set of matrices in Fig. 4.4 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.

4.3 Anomaly Detection on a Classical Computer

To make predictions using our graph adjacency matrices they will be embedded into vectors. By being 1-dimensional vectors, rather than a 2-dimensional matrices, it is simpler to use them as input for a classifier. The vectors will then be used as input to an anomaly detection method based on the K-means clustering algorithm. Here, we aim to see if the vectors have enough information included in them to be predictive and if our methods of embedding are viable for anomaly detection.

Another option would be to use a kernel-based method of graph classification. Doing this would involve embedding the entire graph dataset into a kernel. These kernels measure how similar each graph is to the other. These objects can then be passed to a kernel-based classification method (such as an SVM) and from there a classification can be made. While this is a popular method 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 a feature vector from the graphs we are free to be more versatile when creating a model. Instead of being limited to kernel-based algorithms the range of algorithms is broader - we can now choose any algorithm that accepts a vector as input.

The adjacency matrices created in Section 4.2 will be the objects used to create feature vectors. In this classical example, we will embed the matrices into a vector by simply taking the matrix eigenvalues[1]. Each of the five matrices will return six eigenvalues. These five vectors are combined to give a vector of length 30. Another round of scaling is then applied using StandardScaler from scikit-learn [91].

We prepare 1000 background samples and 200 signal samples to use. The background set is split into 800 training events and 200 test events. We limit the number of events used here due to the nature of sampling from a simulated GBS device. Simulating a device is computationally expensive and necessitates the use of small datasets. The GBS device is discussed in Section 4.4.

K-means clustering is a popular method of unsupervised classification [143, 144]. 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, the centroids are updated by the algorithm by repeating 2 steps: (i) assign every point a label. This label describes which cluster it belongs to and is decided by the centroid closest to the point. In step

[1] This method of creating a vector from a graph is based on the use of Laplacian Eigenvalues [142]. This method first transforms the adjacency matrix into its Laplacian - another form of graph matrix representation. The eigenvalues are taken from this and ordered. We found that the use of a vector created with this method gives similar results compared to our simpler method.

(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

$$\sum_{i=0}^{n} \sum_{x \in C_i} \|x - \mu_i\|^2, \tag{4.3.1}$$

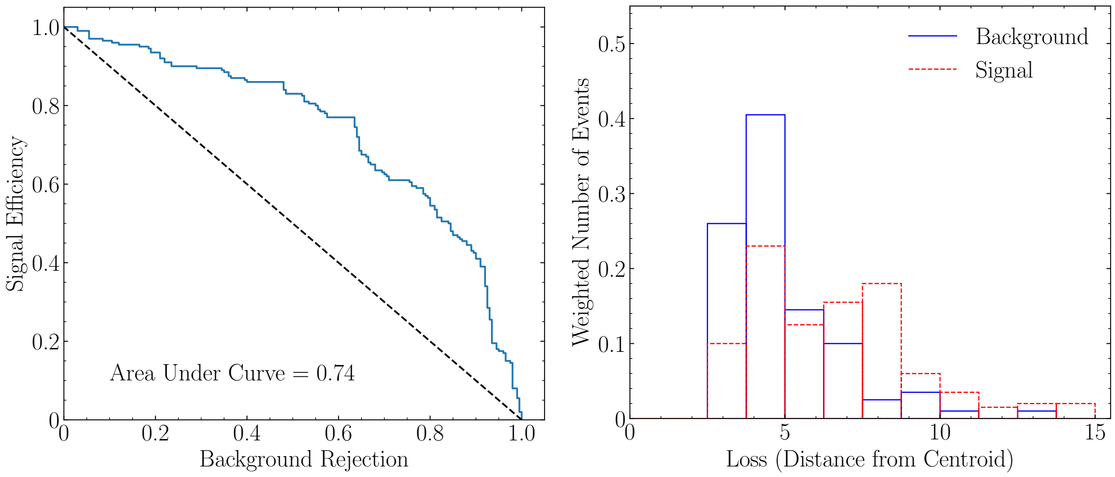
where C is the set of clusters $(C_1, ..., C_n)$ each with a mean $(\mu_1, ..., \mu_n)$. In our method, we only cluster our samples into one cluster. In this scenario, clustering is a somewhat trivial problem. However, our choice of algorithm here is simply to develop a baseline to allow us to compare 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

$$L = \text{distance}(c, x), \tag{4.3.2}$$

which measures the distance between a point and the cluster centroid. Intuitively, it can be assumed that points closer to the centroid will be more likely to come from the background sample while points further away will more likely be signal. Therefore, a threshold can be applied to the loss such that a point above this limit can be classified as an outlier and, thus, as signal.

Fig. 4.5 (a) shows the ROC curve from testing the fitted K-means algorithm, giving a result of 0.74 AUC. Fig. 4.5 (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. As we saw in Chapter 2, autoencoders are neural networks whose input and output 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



(a) (b)

Figure 4.5: (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).

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 [1]. We find this method gives the same AUC of 0.74.

4.4 Anomaly Detection on a Photonic Quantum Device

In Section 4.3 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.

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

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle. \tag{4.4.1}$$

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

$$|\psi\rangle = \int \psi(x)|x\rangle dx. \tag{4.4.2}$$

While having an infinite dimensional basis is a fundamental change of the configuration space of our quantum system compared to Eq. (4.4.1), 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.

For certain problems, it can be useful to introduce a countable basis (instead of measure x or p, instead you may wish to count discrete single photons entering a detector). Fock states provide such a discrete and countable basis. A quantum device, made up of multiple qumodes, can be described by a state $|X\rangle = |x_1, ..., x_N\rangle$, 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 [66]

$$|\psi\rangle = \sum_{X\in\Phi_{n,N}} \alpha_X|X\rangle, \tag{4.4.3}$$

where α_X are complex numbers such that $\sum|\alpha_X|^2 = 1$. The probability of observing a given state, $|\psi\rangle$, is

$$\mathrm{P}(X) = |\alpha_X|^2 = |\langle\psi|X\rangle|^2. \tag{4.4.4}$$

The probability of an input state, $|I\rangle = |i_1, ..., i_N\rangle$, transitioning to $|X\rangle$ is shown to be [145]

$$\mathrm{P}(I \to X) = \frac{|\mathrm{Per}(U_{I,X})|^2}{i_1!...i_N!x_1!...x_N!}. \tag{4.4.5}$$

This calculation depends on the permanent, Per, a matrix manipulation described in further detail in Section 4.4.1. Eq. (4.4.5) 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

$$|\psi\rangle = \exp(-itH)|0\rangle, \tag{4.4.6}$$

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 [146]

$$\hat{H}_k = \frac{1}{2}(\hat{p}_k^2 + \omega_k^2 \hat{x}_k^2), \tag{4.4.7}$$

with the position and momentum operators

$$\hat{p}_k = -\sqrt{\frac{\hbar\omega_k}{2}}(\hat{a}_k - \hat{a}_k^\dagger), \tag{4.4.8}$$

$$\hat{x}_k = \sqrt{\frac{\hbar}{2\omega_k}}(\hat{a}_k + \hat{a}_k^\dagger). \tag{4.4.9}$$

These operators both depend on the creation and annihilation ladder operators $\hat{a}^\dagger$ and $\hat{a}$. 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 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 a CV circuit differs [98,146]. 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 [65]

$$\hat{H} = ih\frac{\kappa}{2}(e^{i\phi}\hat{a}^{\dagger 2} - e^{-i\phi}\hat{a}^2), \tag{4.4.10}$$

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

$$\hat{U} = \exp\left(-\frac{\kappa}{2}(e^{i\phi}\hat{a}^{\dagger^2} - e^{-i\phi}\hat{a}^2)t\right), \tag{4.4.11}$$

we can construct a squeezing gate

$$\hat{S}(z) = \exp\left(\frac{1}{2}(z^*\hat{a}^2 - z\hat{a}^{\dagger^2})\right). \tag{4.4.12}$$

The gates depends on a 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

$$\begin{aligned}\hat{S}^\dagger(z)\hat{x}_\phi\hat{S}(z) &= e^{-r}\hat{x}_\phi,\\ \hat{S}^\dagger(z)\hat{p}_\phi\hat{S}(z) &= e^{r}\hat{p}_\phi.\end{aligned} \tag{4.4.13}$$

For a position-momentum pair, the squeezing gate will amplify one while reducing the other. Similarly, to rotate the state, a rotation operator

$$\hat{R}(\theta) = \exp\left(i\theta\hat{a}^\dagger\hat{a}\right), \tag{4.4.14}$$

can be constructed. Here, θ controls the rotation angle. The gate's application on a state will rotate a state's position and momentum

$$\begin{aligned}\hat{R}^\dagger(\theta)\hat{x}\hat{R}(\theta) &= \hat{x}\text{cos}(\theta) - \hat{p}\text{sin}(\theta),\\ \hat{R}^\dagger(\theta)\hat{p}\hat{R}(\theta) &= \hat{p}\text{cos}(\theta) - \hat{x}\text{sin}(\theta).\end{aligned} \tag{4.4.15}$$

The final Gaussian gate we will introduce is the beamsplitter gate

$$\hat{B}(\theta,\phi) = \exp\left(\theta(e^{i\phi}\hat{a}_1\hat{a}_2^\dagger - e^{-i\phi}\hat{a}_1^\dagger\hat{a}_2)\right). \tag{4.4.16}$$

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

$$\begin{aligned}
\hat{B}(\theta,\phi)^{\dagger}\hat{x_1}\hat{B}(\theta,\phi) &= \hat{x_1}\text{cos}(\theta) - \sin(\theta)[\hat{x_2}\text{cos}(\phi) + \hat{p_2}\text{sin}(\phi)], \\
\hat{B}(\theta,\phi)^{\dagger}\hat{p_1}\hat{B}(\theta,\phi) &= \hat{p_1}\text{cos}(\theta) - \sin(\theta)[\hat{p_2}\text{cos}(\phi) - \hat{x_2}\text{sin}(\phi)], \\
\hat{B}(\theta,\phi)^{\dagger}\hat{x_2}\hat{B}(\theta,\phi) &= \hat{x_2}\text{cos}(\theta) + \sin(\theta)[\hat{x_1}\text{cos}(\phi) - \hat{p_1}\text{sin}(\phi)], \\
\hat{B}(\theta,\phi)^{\dagger}\hat{p_2}\hat{B}(\theta,\phi) &= \hat{p_2}\text{cos}(\theta) + \sin(\theta)[\hat{p_1}\text{cos}(\phi) + \hat{x_1}\text{sin}(\phi)]. \qquad (4.4.17)
\end{aligned}$$

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 has as degree of 3 or more (i.e. the position, 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

$$\hat{V}(\gamma) = \exp\left(i\frac{\gamma}{6}\hat{x}^3\right). \qquad (4.4.18)$$

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

$$\begin{aligned}
\hat{V}^{\dagger}(\gamma)\hat{x}\hat{V}(\gamma) &= \hat{x}, \\
\hat{V}^{\dagger}(\gamma)\hat{p}\hat{V}(\gamma) &= \hat{p} + \gamma\hat{x}^2 \qquad (4.4.19)
\end{aligned}$$

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 computer's ability to implement, in a finite number of steps, with arbitrary precision, a unitary which is polynomial [65].

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.

4.4.1 Gaussian Boson Sampling

An area in which photonic devices demonstrate a quantum advantage is through boson sampling [66, 128]. Our goal is to use such a device to create an embedding of our graph adjacency matrices. Boson sampling devices are designed to emit single photons into an interferometer. The photons output from the interferometer are counted by detectors. There will be a probability associated with observing 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 observing 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. These new vectors 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 4.4. 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 [67, 126, 147].

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 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. (4.4.5), this can also be used to calculate the transition probability between two states in a photonic quantum device. The permanent is given by

$$\mathrm{Per}(A) = \sum_{\pi \in S_N} \prod_{i=1}^{n} A_{i,\pi(i)}. \tag{4.4.20}$$

The permanent is found from a matrix A (which, in our example, will contain information about the setup of the interferometer) and S_N is the entire set of permutations of N photons. As shown in Eq. (4.4.5), the permanent can be used to calculate the transition probability between two states. 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 of the graph's perfect self-matchings. A single graph, or subgraph, can be described as a "matching" if every vertex in the graph is connected with, at most, one other internal vertex. A "perfect matching" is the situation in which every vertex in a graph is connected to one other internal vertex. This can be an intensive, combinatoric, calculation. For this reason, finding the permanent is hard to solve on classical computers [148].

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) [128]. GBS, instead of single photon states, uses single mode squeezed states. This setup has the same problem complexity as typical boson sampling. However, since photon states can be difficult to generate compared to squeezed states, it is easier to generate larger systems of states.

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

$$\mathrm{Per}(M) = \mathrm{Haf}\begin{pmatrix} 0 & M \\ M^t & 0 \end{pmatrix} \tag{4.4.21}$$

where M is a matrix. The hafnian calculates how many perfect self-matchings are in an arbitrary graph. If the edges are weighted (i.e. 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

$$\mathrm{Haf}(M) = \sum_{\pi \in P_n} \prod_{(u,v)\in\pi} M_{(u,v)}, \tag{4.4.22}$$

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. For example, 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 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, similarly, there is no known method of classically calculating it efficiently.

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 output is an array of photon counts. The array $\bar{n}$ is filled such that $\bar{n} = [n_1, ..., n_N]$, where each n_i represents how many photons have been counted at each detector. The probability of observing a result $\bar{n}$ is

$$P(\bar{n}) = \frac{1}{\sqrt{\det(Q)}} \frac{\mathrm{Haf}(\tilde{A}_{\bar{n}})}{\bar{n}!}. \tag{4.4.23}$$

Here, $\bar{n}! = n_1!n_2!...n_N!$. The matrix $\tilde{A}_{\bar{n}}$ (a modified version of $A_{\bar{n}}$) 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 $X = \left(\begin{smallmatrix} 0 & I \\ I & 0 \end{smallmatrix}\right)$ [128]. In Eq. (4.4.23) the hafnian of $\tilde{A}$ is not being found, but rather that of a modified version $\tilde{A}_{\bar{n}}$. This 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

$$\tilde{A} = A \oplus A^*. \tag{4.4.24}$$

Eq (4.4.23) can be written such to show the probability of sampling a photon event $\bar{n}$ with respect to our adjacency matrices

$$P(\bar{n}) = \frac{1}{\sqrt{\det(Q)}} \frac{|\mathrm{Haf}(A_{\bar{n}})|^2}{\bar{n}!}. \tag{4.4.25}$$

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 $A_{\bar{n}}$.

To find the probabilities associated with GBS is a classically hard problem. By building a real device, which is fast to run, you can experimentally find these probabilities and avoid having to do the classical computation. It is through generating samples from a GBS device we can create feature vectors to use as input to classifiers.

4.4.2 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. It is this reliance on the hafnian calculation that makes the process difficult to simulate on a classical machine. However, probing a real photonic device is computationally cheap

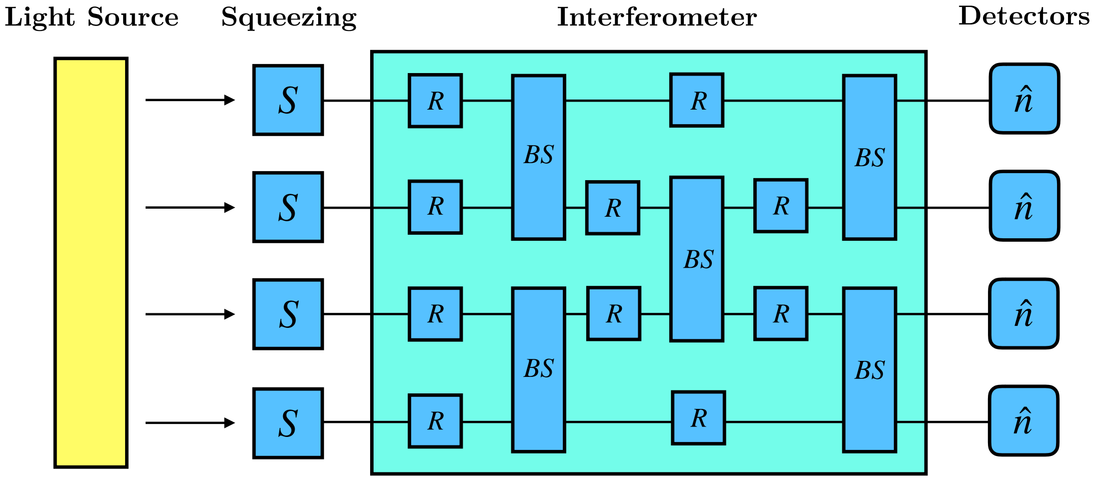


Figure 4.6: A Gaussian boson sampling device. It consists of a set of squeezing gates, an interferometer and photon detectors. In this implementation, the interferometer is constructed from rotation gates and beamsplitters.

- a real GBS device can be sampled 10^5 times every second [67]. Through running the device repeatedly the probabilities of seeing specific patterns could be found (i.e. experimentally finding the results of Eq. (4.4.25)).

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 4.6. It is through these gate parameters we can embed our adjacency matrices. The gate parameters (detailed in Eqs. (4.4.12), (4.4.14) and (4.4.16)) are all determined by an adjacency matrix A.

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

$$A = U\text{diag}(\lambda_1, ..., \lambda_N)U^T. \tag{4.4.26}$$

Here, U is a unitary matrix and the set of λ's are the matrix eigenvalues [126].

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 [126]. This value, η, is a parameter we can choose. The scaling term

c is chosen such that

$$\eta = \sum_{i=0}^{N} \frac{c\lambda_i^2}{1 - c\lambda_i^2}, \tag{4.4.27}$$

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 parameter for the squeezing gate for a particular qumode i

$$z_i = \tanh^{-1}(c\lambda_i). \tag{4.4.28}$$

The interferometer of the device will be found using the unitary matrix U found from Eq. (4.4.26). The process of retrieving the interferometer gate parameters from U is discussed in Refs. [149–151].

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 circuit (and the embedding of a matrix) is accomplished by using the Python library Strawberry Fields [152].

4.4.3 Creating Feature Vectors from GBS Samples

Previously, we used an eigenvalue method to embed graph adjacency matrices into a feature vector. This one-dimensional form then made it simple to use the data as input into a variety of classification algorithms. Here, we will do something similar. However, the feature vectors will be created from the GBS device.

By representing the graphs as adjacency matrices they can be embedded into the GBS device. As discussed in Section 4.4.2 of this chapter, this is done by using properties of the matrix as parameters in the quantum gates that make up the GBS circuit. 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 of length N, where N

is 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 are gathered using the thresholding method. The output of the device will therefore be an array N long, filled with zeroes or ones depending on whether one or more photons were detected.

With or without thresholding, sampling using a simulated GBS device is time-consuming. This constrains the amount of data we can use. As mentioned, feature vectors are built from frequencies of certain measurements from an event's 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. This value gives us a good result.

The generated set of samples can then be used to calculate feature vectors. Here, a few methods exist to construct these [67]. An option is to craft a vector that contains probabilities of a photon being observed at each detector. If we have an event with a device containing k photon detectors we can construct a vector

$$v = (p_{k_1}, p_{k_2}, ..., p_{k_N}). \tag{4.4.29}$$

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 feature vector is found (just as in Eq. (4.4.29)). 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 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.

3. With a set of photon counts, we create a vector describing the probability of observing a photon at a specific detector.

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.

4.4.4 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 4.3. 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. (4.3.2).

Results for this method are shown in Fig. 4.7. 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 an advantage in creating samples to use for anomaly detection tasks.

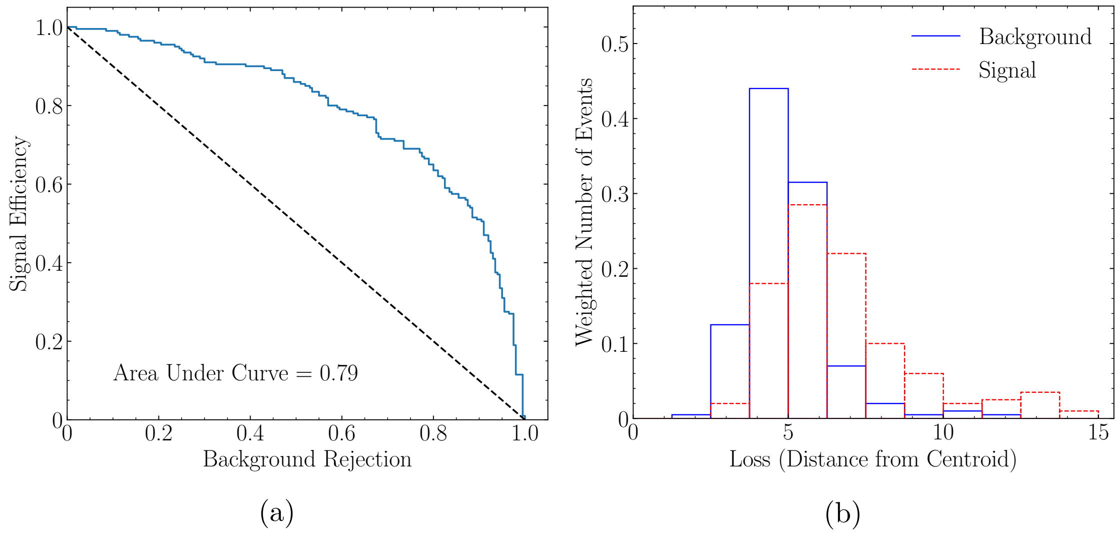


Figure 4.7: a) shows the ROC curve for K-means anomaly detection method trained on feature vectors constructed from GBS samples. The distribution of the loss for the signal and background test sets is shown in b)

4.5 Q-means Clustering

As discussed in Section 4.1 we can consider our anomaly detection problem in three parts: (i) the creation of our graphs, (ii) the embedding of these graphs into a lower-dimensional feature vector and (iii) the classification method. In Section 4.4.3 we showed a quantum method for step (ii), the embedding of the graphs. Here, we introduce Q-means clustering, a quantum equivalent of K-means clustering. This will be our quantum version of step (iii), and will be the classification method we use in this section.

The K-means algorithm can be described as having two steps: (i) assigning points a label and (ii) updating the centroids. To successfully complete part (i) you must deduce which node is closest to each point in your sample set. In this variant of Q-means we focus specifically on this.

As with the GBS device, we can use quantum gate computing methods to implement the Q-means clustering algorithm. The circuit that is constructed to perform Q-means is designed to give a measure of how close a point is to a centroid. Just as in Chapter 3 (wherein a circuit was designed to use a quantum neural network), the circuit itself can be considered in parts. These will be the data embedding, distance calculation, and readout sections.

After using this circuit to find the closest centroids to the points, we can 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 [129].

To build the quantum circuit, we will first consider the discrete qubit scenario. First, to embed the samples into a quantum circuit we will use U_3 gates,

$$U_3(\theta, \phi, \lambda) = \begin{bmatrix} \cos(\frac{\theta}{2}) & -\exp(i\lambda)\sin(\frac{\theta}{2}) \\ \exp(i\phi)\sin(\frac{\theta}{2}) & \exp(i(\phi+\lambda))\cos(\frac{\theta}{2}) \end{bmatrix}. \tag{4.5.1}$$

The samples 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. Both the centroids and the samples can be embedded into quantum states for comparison.

After being embedded, the states must enter a circuit designed to measure some metric of distance. This measure of distance will be used to determine how similar two states are. This will be done using the SwapTest circuit. See Appendix B 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. By embedding the centroids into the circuit alongside the sample vectors, we can find the centroid which overlaps 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 CSWAP gates [153], respectively

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{4.5.2}$$

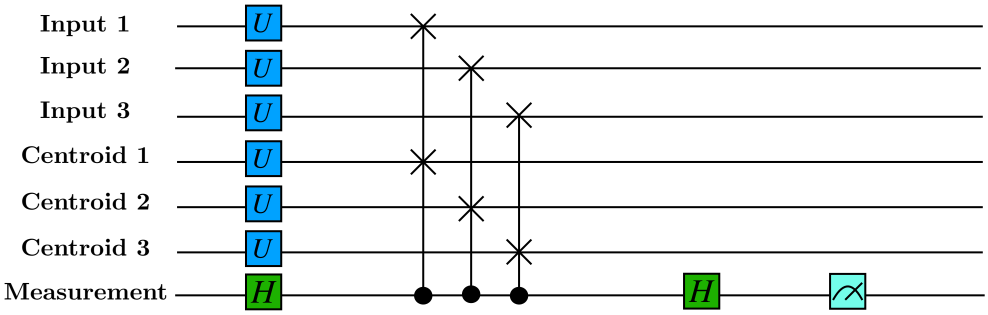


Figure 4.8: Diagram of the Q-means quantum circuit. The feature vector and centroid is encoded through U_3 gates, while a SwapTest is used to provide a measure of their similarity.

and

$$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (4.5.3)$$

A CSWAP gate is similar to a SWAP gate, Eq. (4.5.3), but depends on the state of a control bit. The Hadamard gate is used regularly when building quantum circuits to introduce entanglement into the system. These gates, alongside the U_3 embedding gates are combined to form the circuit shown in Fig. 4.8. 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 [129], while a deeper reliance on qRAM can allow for an improved scaling complexity with respect to the data-set size itself [154].

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 (i.e. the number of components in the training vectors) [129, 130]. 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.

4.5.1 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 [155]. The Hong-Ou-Mandel effect presents 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 Fig. 4.8 we use the Python package PennyLane [156]. The Q-means algorithm is trained with the same parameters as the previous two scenarios - using 800 background samples, each with 30 features. We use both methods of generating samples to train the model (i.e. the classical eigenvalue method and the GBS sampling method). In both cases, 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.

4.6 Conclusions

Unsupervised, data-driven techniques of event classification are useful for separating rare signal events from the abundant Standard Model background in LHC searches.

When building models used to perform these searches it is important to consider how these events are being represented. Graphs provide a suitable structure to store HEP events, allowing for the relationships between event constituents to be naturally captured. To classify these graph structures one either has to use a model that this structure naturally fits (such as a graph neural network) or embed them.

In this chapter, we propose the use of a continuous variable (CV) quantum computing technique, known as Gaussian boson sampling, to embed these graphs into a lower-dimensional structure and perform classification on them. CV quantum computing differs from the discrete qubit paradigm due to its reliance on infinite-dimensional qumodes. This paradigm offers a unique set of quantum advantages, one of which is demonstrated through boson sampling. Boson sampling can be used to create samples by detecting photons exiting an interferometer. This is a classically difficult task, requiring complicated matrix manipulations.

Gaussian boson sampling, on a near-term quantum device, can be performed extremely quickly and thus provides a continuous-variable photonic device with a quantum advantage. Using GBS samples to create feature vectors which are used as input to an anomaly detection method performs favourably when compared to methods using classically generated feature vectors. It would be useful to expand our analysis to include more classical techniques, allowing us to further the claim of a quantum advantage arising that also increases classification performance. A further quantum advantage can be found by using the Q-means clustering algorithm. This quantum extension of K-means gives equivalent results but can 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 N. This is compared to the K-means complexity of $\mathcal{O}(N)$. As the development of quantum devices moves forward, and we are able to use the GBS to create larger vectors, the power of a method similar to Q-means will become more apparent. While Q-means is implemented here in the discrete qubit-based model it should be possible to expand this to the CV paradigm.

Chapter 5

Summary and Conclusions

Over the course of this thesis we have introduced several machine learning methods that can be applied to searches for New Physics. While the Standard Model provides a descriptive account of the fundamental constituents of nature, it is clear that it cannot describe every aspect of the universe. Using colliders to probe the high energy frontier to search for hints of what lies beyond the Standard Model generates enormous, high-dimensional datasets. Gathering this data uses some of the most advanced machines ever devised and is a process vital for us to expand our knowledge of fundamental physics. To ensure we make the full use of this data it is therefore also important to search for novel analysis thechniques.

We began in Chapter 2 by introducing a neural network structure known as an autoencoder. Autoencoders can be used for unsupervised anomaly detection, which we applied to a $t\bar{t}$ resonance search. While the performance of this model was lower than a supervised search the use of the autoencoder provides a large benefit. The autoencoder, by being an unsupervised approach, needed only to be trained on background data. This had two benefits of note - it would be possible to perform a data-driven search using only real LHC data, and that the anomaly detection method is model independent. While training on real data avoids any error associated with Monte-Carlo generators it did leave the possibility of the model's classification ability being influenced by experimental error. We saw that by applying an adversarial extension to the model that one can create a robust method of anomaly detection, that is not dependant on systematic errors.

Quantum machine learning, a field of quantum computing aimed at applying quantum advantages to ML, is used in Chapter 3. A quantum neural network (QNN) can be constructed from a quantum circuit. This is analogous to a traditional neural network, where the quantum gate parameters act similarly to the weights and biases in the network's nodes. These QNNs allow one to use smaller networks when compared to classical NNs. We used this to our advantage to perform optimisation techniques that would be computationally expensive to do on a larger network. We saw that a quantum neural network, trained using a quantum equivalent to gradient descent, can outperform a classical network of a similar size. Furthermore, these networks can be used on near-term quantum hardware.

In Chapter 4 our focus shifted away from qubits and onto qumodes - the fundamental object used in the continuous-variable paradigm of quantum computing. The CV paradigm makes use of quantum photonic hardware to build device and has a selection of quantum advantages unique to the method. Here, we made use of Gaussian boson sampling (GBS) to sample events stored in a graph structure. Graph structures are well suited to store particle physics data, as they can capture relationships between the event's constituents. The output of the GBS device was then used as input to a Q-means (a quantum equivalent of K-means) anomaly detection method. The act of running a GBS device shows a quantum advantage, as it is a difficult process to simulate classically. However, the results also suggest that it is possible for the anomaly detector to achieve a performance boost using this method.

As the LHC has been upgraded the amount of data it has gathered has steadily increased. This is set to continue with the development of the HL-LHC. To be able to efficiently work with this large and high-dimensional dataset many approaches can be taken. Machine learning, and extensions using quantum information theory, provide powerful methods of allowing us to continue the search for new physics.

Appendix A

The Quantum Geometric Tensor

Geometric quantum mechanics states that the traits of a quantum system can be described by geometric features on a complex projective space. In this space, there is an invariant metric tensor, the Fubini-Study tensor (FST), that can be used to describe distances between quantum states [157–159]. The FST can be found by taking the real part of the Quantum Geometric Tensor (QGT). This tensor is used for the optimisation of quantum neural networks.

We will give a general introduction to this tensor before briefly discussing how it can be approximated on real hardware and how it relates to our VQC. We will construct the QGT by considering the distance between the states $|\psi_0(\theta)\rangle$ and $|\psi_0(\theta + d\theta)\rangle$, where ψ is a general wave function state. We can write the probability to excite the parameter from θ to $\theta + d\theta$ as

$$ds^2 = 1 - \left|\langle\psi_0(\theta)|\psi_0(\theta + d\theta)\rangle\right|^2 \ . \tag{A.0.1}$$

The amplitude of a state being excited from $|\psi_0(\theta)\rangle$ to $|\psi_n(\theta)\rangle$ can be written as

$$a_n = \langle\psi_n(\theta + d\theta)|\psi_0(\theta)\rangle \ , \tag{A.0.2}$$

whereas the probability for a transition between states to occur can be found by evaluating

$$ds^2 = \sum_{n \neq 0} \left| a_n^2 \right| = d\theta_i d\theta_j G_{ij} + O(|d\theta|^3) \ , \tag{A.0.3}$$

where G_{ij} is the Quantum Geometric Tensor, defined as

$$G_{ij} = \langle \partial_i \psi_0 | \partial_j \psi_0 \rangle - \langle \partial_i \psi_0 | \psi_0 \rangle \langle \psi_0 | \partial_j \psi_0 \rangle \ . \tag{A.0.4}$$

This tensor therefore signifies the distance between the two quantum states [160]. The Fubini-Study metric is the real part of this tensor, $g_{ij}(\theta) = \mathrm{Re}[G_{ij}(\theta)]$. We will use this as a distance measure between wave functions, or transition probability between the states [158].

Importantly, G_{ij} can be calculated on quantum hardware [107]. We consider a variational circuit where each layer l is parametrised by θ_l and includes gates $U(\theta_l)$. We can represent a unitary gate in the form

$$U(\theta) = e^{i\theta_l V}, \tag{A.0.5}$$

where V is the Hermitian generator of $U(\theta_l)$. From these gates result in the relations

$$\begin{aligned} \partial_i U_l(\theta_l) &= -iV_i U_l(\theta_l) \ , \\ \partial_j U_l(\theta_l) &= -iV_j U_l(\theta_l) \ , \end{aligned} \tag{A.0.6}$$

where V_i and V_j are Hermitian generator matrices. From Eq. (A.0.6) we can find

$$\left\langle \partial_i \psi_\theta | \partial_j \psi_\theta \right\rangle = \left\langle \psi_l | V_i V_j | \psi_l \right\rangle \ , \tag{A.0.7}$$

$$i \left\langle \psi_\theta | \partial_j \psi_\theta \right\rangle = \left\langle \psi_l | V_j | \psi_l \right\rangle \ . \tag{A.0.8}$$

By considering both A.0.7 and A.0.8, a representation of the Quantum Geometric Tensor can be formed for a block of parameters that exist in layer l

$$G_{ij}^l = \langle \psi_l | V_i V_j | \psi_l \rangle - \langle \psi_l | V_i | \psi_l \rangle \langle \psi_l | V_j | \psi_l \rangle \ . \tag{A.0.9}$$

The quantum states ψ_l can be determined experimentally from the variational quantum classifier. This approximation of the QGT also allows us to find the Fubini-Study metric by taking the real part, such that $g^l_{ij} = \mathrm{Re}[G^l_{ij}]$.

For use in quantum neural network optimisation techniques, such as quantum gradient descent, it is desireable to find the inverse. To do this, we use the Moore-Penrose pseudo-inverse

$$g^+ = (g^T g)^{-1} g^T \ . \tag{A.0.10}$$

This method allows us to find an inverse matrix even for matrices that cannot be inverted, as shown in Eq A.0.10. In cases where the matrix is invertible, the matrix pseudo-inverse and matrix inverse are identical.

Appendix B

SwapTest

A SwapTest circuit provides a method of checking the overlap between two states. Following [130], 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

$$|\psi\rangle = |0, \alpha, \beta\rangle \,. \tag{B.0.1}$$

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

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0, \alpha, \beta\rangle + |1, \alpha, \beta\rangle). \tag{B.0.2}$$

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

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}(|0, \alpha, b\rangle + |1, b, \alpha\rangle). \tag{B.0.3}$$

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

$$|\psi_3\rangle = \frac{1}{2}|0\rangle(|\alpha,\beta\rangle + |\beta,\alpha\rangle) + \frac{1}{2}|1\rangle(|\alpha,\beta\rangle - |\beta,\alpha\rangle). \tag{B.0.4}$$

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

$$\begin{align}
P(|0\rangle) &= \left|\frac{1}{2}\langle 0|0\rangle(|\alpha,\beta\rangle + |\beta,\alpha\rangle) + \frac{1}{2}\langle 0|1\rangle(|\alpha,\beta\rangle - |\beta,\alpha\rangle)\right|^2 \tag{B.0.5}\\
&= \frac{1}{4}|(|\alpha,\beta\rangle + |\beta,\alpha\rangle)|^2 \tag{B.0.6}\\
&= \frac{1}{4}(\langle\beta|\beta\rangle\langle\alpha|\alpha\rangle + \langle\beta|\alpha\rangle\langle\alpha|\beta\rangle + \langle\alpha|\beta\rangle\langle\beta|\alpha\rangle + \langle\alpha|\alpha\rangle\langle\beta|\beta\rangle) \tag{B.0.7}\\
&= \frac{1}{2} + \frac{1}{2}|\langle\alpha|\beta\rangle|^2 \tag{B.0.8}
\end{align}$$

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

Bibliography

[1] A. Blance, M. Spannowsky and P. Waite, *Adversarially-trained autoencoders for robust unsupervised new physics searches*, *Journal of High Energy Physics* **2019** (Oct, 2019) , [1905.10384].

[2] A. Blance and M. Spannowsky, *Quantum machine learning for particle physics using a variational quantum classifier*, *Journal of High Energy Physics* **2021** (Feb, 2021) , [2010.07335].

[3] A. Blance and M. Spannowsky, *Unsupervised Event Classification with Graphs on Classical and Photonic Quantum Computers*, 2103.03897.

[4] A. Blance, M. Chala, M. Ramos and M. Spannowsky, *Novel b -decay signatures of light scalars at high energy facilities*, *Physical Review D* **100** (Dec, 2019) , [1907.13151].

[5] S. Abel, A. Blance and M. Spannowsky, *Quantum Optimisation of Complex Systems with a Quantum Annealer*, 2105.13945.

[6] D. J. Gross and F. Wilczek, *Asymptotically free gauge theories. i*, *Phys. Rev. D* **8** (Nov, 1973) 3633–3652.

[7] S. Weinberg, *Non-abelian gauge theories of the strong interactions*, *Phys. Rev. Lett.* **31** (Aug, 1973) 494–497.

[8] H. Fritzsch, M. Gell-Mann and H. Leutwyler, *Advantages of the color octet gluon picture*, *Physics Letters B* **47** (1973) 365–368.

[9] S. L. Glashow, *Partial-symmetries of weak interactions*, *Nuclear Physics* **22** (1961) 579–588.

[10] S. Weinberg, *A model of leptons*, *Phys. Rev. Lett.* **19** (Nov, 1967) 1264–1266.

[11] A. Salam, *Weak and electromagnetic interactions*, pp. 244–254. 10.1142/9789812795915-0034.

[12] G. S. Guralnik, C. R. Hagen and T. W. B. Kibble, *Global conservation laws and massless particles*, *Phys. Rev. Lett.* **13** (Nov, 1964) 585–587.

[13] P. W. Higgs, *Broken symmetries and the masses of gauge bosons*, *Phys. Rev. Lett.* **13** (Oct, 1964) 508–509.

[14] F. Englert and R. Brout, *Broken symmetry and the mass of gauge vector mesons*, *Phys. Rev. Lett.* **13** (Aug, 1964) 321–323.

[15] J. Schwinger, *Quantum electrodynamics. i. a covariant formulation*, *Phys. Rev.* **74** (Nov, 1948) 1439–1461.

[16] Z. Koba, T. Tati and S.-i. Tomonaga, *On a Relativistically Invariant Formulation of the Quantum Theory of Wave Fields. II: Case of Interacting Electromagnetic and Electron Fields*, *Progress of Theoretical Physics* **2** (10, 1947) 101–116, [`https://academic.oup.com/ptp/article-pdf/2/3/101/5285399/2-3-101.pdf`].

[17] J. Schwinger, *On quantum-electrodynamics and the magnetic moment of the electron*, *Phys. Rev.* **73** (Feb, 1948) 416–417.

[18] R. P. Feynman, *Mathematical formulation of the quantum theory of electromagnetic interaction*, *Phys. Rev.* **80** (Nov, 1950) 440–457.

[19] M. E. Peskin and D. V. Schroeder, *An Introduction to quantum field theory.* Addison-Wesley, Reading, USA, 1995.

[20] S. F. Novaes, *Standard model: An Introduction*, in *10th Jorge Andre Swieca Summer School: Particle and Fields*, 1, 1999, `hep-ph/0001283`.

[21] H. E. Logan, *Tasi 2013 lectures on higgs physics within and beyond the standard model*, 2017.

[22] M. E. Peskin and D. V. Schroeder, *An introduction to quantum field theory.* Westview, Boulder, CO, 1995.

[23] G. Aad, B. Abbott, J. Abdallah, O. Abdinov, R. Aben, M. Abolins et al., *Combined measurement of the higgs boson mass in pp collisions at sqrt[s]=7 and 8 tev with the atlas and cms experiments*, *Phys Rev Lett* **114** (05, 2015) 191803.

[24] P. D. Group, P. A. Zyla, R. M. Barnett, J. Beringer, O. Dahl, D. A. Dwyer et al., *Review of Particle Physics*, *Progress of Theoretical and Experimental Physics* **2020** (08, 2020) .

[25] Super-Kamiokande Collaboration collaboration, Y. Fukuda, T. Hayakawa, E. Ichihara, K. Inoue, K. Ishihara, H. Ishino et al., *Evidence for oscillation of atmospheric neutrinos*, *Phys. Rev. Lett.* **81** (Aug, 1998) 1562–1567.

[26] E. Corbelli and P. Salucci, *The extended rotation curve and the dark matter halo of m33*, *Monthly Notices of the Royal Astronomical Society* **311** (Jan, 2000) 441–447.

[27] P. J. E. Peebles and B. Ratra, *The cosmological constant and dark energy*, *Rev. Mod. Phys.* **75** (Apr, 2003) 559–606.

[28] L. Canetti, M. Drewes and M. Shaposhnikov, *Matter and antimatter in the universe*, *New Journal of Physics* **14** (Sep, 2012) 095012.

[29] ATLAS collaboration, G. Aad et al., *The ATLAS Experiment at the CERN Large Hadron Collider*, *JINST* **3** (2008) S08003.

[30] CMS collaboration, S. Chatrchyan et al., *The CMS Experiment at the CERN LHC*, *JINST* **3** (2008) S08004.

[31] CMS Collaboration collaboration, S. R. Davis, *Interactive Slice of the CMS detector*, .

[32] W. Herr and B. Muratori, *Concept of luminosity*, .

[33] G. Apollinari, O. Brüning, T. Nakamoto and L. Rossi, *High Luminosity Large Hadron Collider HL-LHC*, 1705.08830.

[34] ATLAS collaboration, M. Aaboud et al., *Measurement of the Higgs boson mass in the $H \to ZZ^* \to 4\ell$ and $H \to \gamma\gamma$ channels with $\sqrt{s} = 13$ TeV pp collisions using the ATLAS detector*, *Phys. Lett. B* **784** (2018) 345–366, [1806.00242].

[35] M. Benedikt, A. Blondel, P. Janot, M. Mangano and F. Zimmermann, *Future Circular Colliders succeeding the LHC*, *Nature Phys.* **16** (2020) 402–407.

[36] V. V. Gligorov, *Real-time data analysis at the lhc: present and future*, 2015.

[37] K. Cranmer, *Practical statistics for the lhc*, 2015.

[38] D. Guest, K. Cranmer and D. Whiteson, *Deep learning and its application to lhc physics*, *Annual Review of Nuclear and Particle Science* **68** (Oct, 2018) 161–181.

[39] A. R. M. and, *The run-2 ATLAS trigger system*, *Journal of Physics: Conference Series* **762** (oct, 2016) 012003.

[40] M. L. Piscopo, M. Spannowsky and P. Waite, *Solving differential equations with neural networks: Applications to the calculation of cosmological phase transitions*, *Phys. Rev. D* **100** (2019) 016002, [1902.05563].

[41] T. N. Collaboration, L. D. Debbio, S. Forte, J. I. Latorre, A. Piccione and J. Rojo, *Neural network determination of parton distributions: the nonsinglet case*, *Journal of High Energy Physics* **2007** (mar, 2007) 039–039.

[42] S. Badger and J. Bullock, *Using neural networks for efficient evaluation of high multiplicity scattering amplitudes*, *Journal of High Energy Physics* **2020** (Jun, 2020) .

[43] D. Maitre and H. Truong, *A factorisation-aware matrix element emulator*, 2021. 10.1007/JHEP11(2021)066.

[44] H. Wang and B. Raj, *On the origin of deep learning*, 2017.

[45] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning.* The MIT Press, 2016.

[46] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez et al., *A general reinforcement learning algorithm that masters chess, shogi, and go through self-play*, *Science* **362** (12, 2018) 1140–1144.

[47] K. Arulkumaran, A. Cully and J. Togelius, *Alphastar*, *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Jul, 2019) .

[48] Y. Huang and Y. Chen, *Autonomous driving with deep learning: A survey of state-of-art technologies*, 2020.

[49] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, *arXiv 1409.1556* (09, 2014) .

[50] A. Butter et al., *The Machine Learning Landscape of Top Taggers*, *SciPost Phys.* **7** (2019) 014, [1902.09914].

[51] I. Sutskever, O. Vinyals and Q. Le, *Sequence to sequence learning with neural networks*, *Advances in Neural Information Processing Systems* **4** (09, 2014) .

[52] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal et al., *Language models are few-shot learners*, 2020.

[53] M.-Y. Liu, X. Huang, J. Yu, T.-C. Wang and A. Mallya, *Generative adversarial networks for image and video synthesis: Algorithms and applications*, 2020.

[54] J. Bullock and M. Luengo-Oroz, *Automated speech generation from un general assembly statements: Mapping risks in ai generated texts*, 2019.

[55] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition.* Cambridge University Press, USA, 10th ed., 2011.

[56] R. de Wolf, *Quantum computing: Lecture notes*, 2021.

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

[58] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, *SIAM Journal on Computing* **26** (Oct, 1997) 1484–1509.

[59] S. Abel, N. Chancellor and M. Spannowsky, *Quantum Computing for Quantum Tunnelling*, `2003.07374`.

[60] S. Abel and M. Spannowsky, *Observing the fate of the false vacuum with a quantum laboratory*, `2006.06003`.

[61] 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) 375–379.

[62] K. Bepari, S. Malik, M. Spannowsky and S. Williams, *Towards a quantum computing algorithm for helicity amplitudes and parton showers*, *Physical Review D* **103** (Apr, 2021) .

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

[64] I. Márquez-Mártin, P. Arnault, G. D. Molfetta and A. Pérez, *Electromagnetic lattice gauge invariance in two-dimensional discrete-time quantum walks*, *Physical Review A* **98** (2018) 032333, [1808.04488].

[65] S. Lloyd and S. L. Braunstein, *Quantum computation over continuous variables*, *Physical Review Letters* **82** (Feb, 1999) 1784–1787.

[66] S. Aaronson and A. Arkhipov, *The computational complexity of linear optics*, 2010.

[67] M. Schuld, K. Brádler, R. Israel, D. Su and B. Gupt, *Measuring the similarity of graphs with a gaussian boson sampler*, *Physical Review A* **101** (Mar, 2020) .

[68] C. Englert, P. Galler, P. Harris and M. Spannowsky, *Machine Learning Uncertainties with Adversarial Neural Networks*, *Eur. Phys. J.* **C79** (2019) 4, [1807.08763].

[69] G. Louppe, M. Kagan and K. Cranmer, *Learning to Pivot with Adversarial Networks*, 1611.01046.

[70] T. Heimel, G. Kasieczka, T. Plehn and J. M. Thompson, *QCD or What?*, *SciPost Phys.* **6** (2019) 030, [1808.08979].

[71] B. Kiran, D. Mathew Thomas and R. Parakkal, *An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos*, *Journal of Imaging* **4** (01, 2018) , [1801.03149].

[72] D. P. Kingma and M. Welling, *Auto-Encoding Variational Bayes*, 1312.6114.

[73] P. Vincent, H. Larochelle, Y. Bengio and P.-A. Manzagol, *Extracting and composing robust features with denoising autoencoders*, in *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, (New York, NY, USA), pp. 1096–1103, ACM, 2008.

[74] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen et al., *Event Generation and Statistical Sampling with Deep Generative Models and a Density Information Buffer*, 1901.00875.

[75] M. Farina, Y. Nakai and D. Shih, *Searching for New Physics with Deep Autoencoders*, 1808.08992.

[76] J. Hajer, Y.-Y. Li, T. Liu and H. Wang, *Novelty Detection Meets Collider Physics*, 1807.10261.

[77] T. S. Roy and A. H. Vijay, *A robust anomaly finder based on autoencoder*, 1903.02032.

[78] T. Plehn and M. Spannowsky, *Top Tagging*, *J. Phys.* **G39** (2012) 083001, [1112.4441].

[79] T. Plehn, M. Spannowsky and M. Takeuchi, *How to Improve Top Tagging*, *Phys. Rev.* **D85** (2012) 034029, [1111.5034].

[80] D. E. Soper and M. Spannowsky, *Finding top quarks with shower deconstruction*, *Phys. Rev.* **D87** (2013) 054012, [1211.3140].

[81] D. E. Soper and M. Spannowsky, *Finding physics signals with event deconstruction*, *Phys. Rev.* **D89** (2014) 094005, [1402.1189].

[82] G. Altarelli, B. Mele and M. Ruiz-Altaba, *Searching for New Heavy Vector Bosons in $p\bar{p}$ Colliders*, *Z. Phys.* **C45** (1989) 109.

[83] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer et al., *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*, *JHEP* **07** (2014) 079, [1405.0301].

[84] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten et al., *An Introduction to PYTHIA 8.2*, *Comput. Phys. Commun.* **191** (2015) 159–177, [1410.3012].

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

[86] M. Cacciari, G. P. Salam and G. Soyez, *FastJet User Manual*, *Eur. Phys. J.* **C72** (2012) 1896, [1111.6097].

[87] A. Buckley, J. Butterworth, L. Lonnblad, D. Grellscheid, H. Hoeth, J. Monk et al., *Rivet user manual*, *Comput. Phys. Commun.* **184** (2013) 2803–2819, [1003.0694].

[88] ATLAS collaboration, *Data-driven determination of the energy scale and resolution of jets reconstructed in the ATLAS calorimeters using dijet and multijet events at $\sqrt{s} = 8$ TeV*, 2015.

[89] ATLAS Collaboration collaboration, G. Aad and Abbott, *Jet energy scale and resolution measured in proton–proton collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*, *Eur. Phys. J. C* **81** (Jul, 2020) 689. 73 p, [2007.02645].

[90] ATLAS collaboration, G. Aad et al., *Performance of Missing Transverse Momentum Reconstruction in Proton-Proton Collisions at 7 TeV with ATLAS*, *Eur. Phys. J.* **C72** (2012) 1844, [1108.5602].

[91] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel et al., *Scikit-learn: Machine learning in Python*, *Journal of Machine Learning Research* **12** (2011) 2825–2830.

[92] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, 1412.6980.

[93] F. Chollet et al., "Keras." https://github.com/fchollet/keras, 2015.

[94] M. Abadi et al., *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*, 1603.04467.

[95] R. Frederix and F. Maltoni, *Top pair invariant mass distribution: A Window on new physics*, *JHEP* **01** (2009) 047, [0712.2355].

[96] ATLAS collaboration, M. Aaboud et al., *Search for heavy particles decaying into top-quark pairs using lepton-plus-jets events in proton–proton collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*, *Eur. Phys. J.* **C78** (2018) 565, [1804.10823].

[97] E. Farhi and H. Neven, *Classification with quantum neural networks on near term processors*, 2018.

[98] N. Killoran, T. R. Bromley, J. M. Arrazola, M. Schuld, N. Quesada and S. Lloyd, *Continuous-variable quantum neural networks*, *Physical Review Research* **1** (Oct, 2019) .

[99] M. Schuld, A. Bocharov, K. M. Svore and N. Wiebe, *Circuit-centric quantum classifiers*, *Physical Review A* **101** (Mar, 2020) .

[100] J. R. McClean, J. Romero, R. Babbush and A. Aspuru-Guzik, *The theory of variational hybrid quantum-classical algorithms*, *New Journal of Physics* **18** (Feb, 2016) 023023.

[101] A. Mari, T. R. Bromley, J. Izaac, M. Schuld and N. Killoran, *Transfer learning in hybrid classical-quantum neural networks*, 2019.

[102] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love et al., *A variational eigenvalue solver on a photonic quantum processor*, *Nature Communications* **5** (Jul, 2014) .

[103] E. Farhi, J. Goldstone and S. Gutmann, *A quantum approximate optimization algorithm*, 2014.

[104] H. Neven, Google, V. Denchev, M. Drew-Brook, J. Zhang and W. Macready, *Nips 2009 demonstration: Binary classification using hardware implementation of quantum annealing*, .

[105] E. Farhi, J. Goldstone, S. Gutmann and M. Sipser, *Quantum computation by adiabatic evolution*, .

[106] Y. Aharonov, L. Davidovich and N. Zagury, *Quantum random walks*, *Phys. Rev. A* **48** (Aug, 1993) 1687–1690.

[107] J. Stokes, J. Izaac, N. Killoran and G. Carleo, *Quantum natural gradient*, *Quantum* **4** (May, 2020) 269.

[108] CMS collaboration, S. Chatrchyan et al., *Search for Anomalous $t\bar{t}$ Production in the Highly-Boosted All-Hadronic Final State*, *JHEP* **09** (2012) 029, [`1204.2488`].

[109] ATLAS collaboration, M. Aaboud et al., *Search for heavy particles decaying into a top-quark pair in the fully hadronic final state in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*, *Phys. Rev.* **D99** (2019) 092004, [`1902.10077`].

[110] R. LaRose and B. Coyle, *Robust data encodings for quantum classifiers*, 2020.

[111] A. Macaluso and Others, *A variational algorithm for quantum neural networks*, in *Computational Science – ICCS 2020*, (Cham), pp. 591–604, Springer International Publishing, 2020.

[112] K. Mitarai, M. Negoro, M. Kitagawa and K. Fujii, *Quantum circuit learning*, *Physical Review A* **98** (Sep, 2018) .

[113] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac and N. Killoran, *Evaluating analytic gradients on quantum hardware*, *Physical Review A* **99** (Mar, 2019) .

[114] B. Neyshabur, R. Salakhutdinov and N. Srebro, *Path-sgd: Path-normalized optimization in deep neural networks*, 2015.

[115] A. Harrow and J. Napp, *Low-depth gradient measurements can improve convergence in variational hybrid quantum-classical algorithms*, 2019.

[116] S.-i. Amari, *Natural gradient works efficiently in learning*, *Neural Computation* **10** (1998) 251–276, [`https://doi.org/10.1162/089976698300017746`].

[117] S. ichi Amari, R. Karakida and M. Oizumi, *Fisher information and natural gradient learning of random deep networks*, 2018.

[118] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, M. S. Alam, S. Ahmed et al., *Pennylane: Automatic differentiation of hybrid quantum-classical computations*, 2018.

[119] D. C. McKay, T. Alexander, L. Bello, M. J. Biercuk, L. Bishop, J. Chen et al., *Qiskit backend specifications for openqasm and openpulse experiments*, 2018.

[120] H. Abraham et al., *Qiskit: An open-source framework for quantum computing*, 2019. 10.5281/zenodo.2562110.

[121] F. Chollet et al., "Keras." `https://keras.io`, 2015.

[122] M. Abadi et al., *Tensorflow: Large-scale machine learning on heterogeneous distributed systems*, 2016.

[123] M. Abdughani, J. Ren, L. Wu and J. M. Yang, *Probing stop pair production at the lhc with graph neural networks*, *Journal of High Energy Physics* **2019** (Aug, 2019) .

[124] J. A. Martinez, O. Cerri, M. Pierini, M. Spiropulu and J.-R. Vlimant, *Pileup mitigation at the large hadron collider with graph neural networks*, 2019.

[125] J. Shlomi, P. Battaglia and J.-R. Vlimant, *Graph neural networks in particle physics*, *Machine Learning: Science and Technology* **2** (Jan, 2021) 021001.

[126] T. R. Bromley, J. M. Arrazola, S. Jahangiri, J. Izaac, N. Quesada, A. D. Gran et al., *Applications of near-term photonic quantum computers: software and algorithms*, *Quantum Science and Technology* **5** (May, 2020) 034010.

[127] N. Killoran, T. R. Bromley, J. M. Arrazola, M. Schuld, N. Quesada and S. Lloyd, *Continuous-variable quantum neural networks*, *Physical Review Research* **1** (Oct, 2019) .

[128] C. S. Hamilton, R. Kruse, L. Sansoni, S. Barkhofen, C. Silberhorn and I. Jex, *Gaussian boson sampling*, *Physical Review Letters* **119** (Oct, 2017) .

[129] S. Lloyd, M. Mohseni and P. Rebentrost, *Quantum algorithms for supervised and unsupervised machine learning*, 2013.

[130] D. Kopczyk, *Quantum machine learning for data scientists*, 2018.

[131] 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, [1006.1650].

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

[133] B. Bellazzini, C. Csáki, A. Falkowski and A. Weiler, *Buried higgs boson*, *Physical Review D* **80** (Oct, 2009) .

[134] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten et al., *An introduction to pythia 8.2*, *Computer Physics Communications* **191** (Jun, 2015) 159–177.

[135] J. M. Butterworth, A. R. Davison, M. Rubin, G. P. Salam, P. Ko and D. Ki Hong, *Jet substructure as a new higgs search channel at the lhc*, *AIP Conference Proceedings* (2008) .

[136] D. E. Soper and M. Spannowsky, *Combining subjet algorithms to enhance zh detection at the lhc*, *Journal of High Energy Physics* **2010** (Aug, 2010) .

[137] S. Marzani, G. Soyez and M. Spannowsky, *Looking inside jets: an introduction to jet substructure and boosted-object phenomenology*, vol. 958. Springer, 2019, 10.1007/978-3-030-15709-8.

[138] M. Cacciari, G. P. Salam and G. Soyez, *Fastjet user manual*, *The European Physical Journal C* **72** (Mar, 2012) .

[139] Y. Dokshitzer, G. Leder, S. Moretti and B. Webber, *Better jet clustering algorithms*, *Journal of High Energy Physics* **1997** (Aug, 1997) 001–001.

[140] M. Cacciari, G. P. Salam and G. Soyez, *The anti-ktjet clustering algorithm*, *Journal of High Energy Physics* **2008** (Apr, 2008) 063–063.

[141] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, 2017.

[142] N. de Lara and E. Pineau, *A simple baseline algorithm for graph classification*, 2018.

[143] S. Lloyd, *Least squares quantization in pcm*, *IEEE Transactions on Information Theory* **28** (1982) 129–137.

[144] M. E. Celebi, H. A. Kingravi and P. A. Vela, *A comparative study of efficient initialization methods for the k-means clustering algorithm*, *Expert Systems with Applications* **40** (Jan, 2013) 200–210.

[145] D. J. Brod, E. F. Galvão, A. Crespi, R. Osellame, N. Spagnolo and F. Sciarrino, *Photonic implementation of boson sampling: a review*, *Advanced Photonics* **1** (2019) 1 – 14.

[146] S. L. Braunstein and P. van Loock, *Quantum information with continuous variables*, *Reviews of Modern Physics* **77** (Jun, 2005) 513–577.

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

[148] L. Valiant, *The complexity of computing the permanent*, *Theoretical Computer Science* **8** (1979) 189–201.

[149] W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer and I. A. Walmsley, *Optimal design for universal multiport interferometers*, *Optica* **3** (Dec, 2016) 1460–1465.

[150] M. Reck, A. Zeilinger, H. J. Bernstein and P. Bertani, *Experimental realization of any discrete unitary operator*, *Phys. Rev. Lett.* **73** (Jul, 1994) 58–61.

[151] W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer and I. A. Walmsley, *Supplement 1: Optimal design for universal multiport interferometers*, Apr, 2017. 10.1364/OPTICA.3.001460.s001.

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

[153] H. Buhrman, R. Cleve, J. Watrous and R. de Wolf, *Quantum fingerprinting*, *Physical Review Letters* **87** (Sep, 2001) .

[154] I. Kerenidis, J. Landman, A. Luongo and A. Prakash, *q-means: A quantum algorithm for unsupervised machine learning*, 2018.

[155] J. C. Garcia-Escartin and P. Chamorro-Posada, *swap test and hong-ou-mandel effect are equivalent*, *Physical Review A* **87** (May, 2013) .

[156] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, M. S. Alam, S. Ahmed et al., *Pennylane: Automatic differentiation of hybrid quantum-classical computations*, 2020.

[157] T. W. B. Kibble, *Geometrization of quantum mechanics*, *Comm. Math. Phys.* **65** (1979) 189–201.

[158] D. C. Brody and L. P. Hughston, *Geometric quantum mechanics*, *Journal of Geometry and Physics* **38** (Apr, 2001) 19–53.

[159] R. Cheng, *Quantum geometric tensor (fubini-study metric) in simple quantum system: A pedagogical introduction*, 2013.

[160] M. Kolodrubetz, D. Sels, P. Mehta and A. Polkovnikov, *Geometry and non-adiabatic response in quantum and classical systems*, *Physics Reports* **697** (2017) 1 – 87.