Published for SISSA by Springer

Received: *November 14, 2020*
Accepted: *January 12, 2021*
Published: *February 24, 2021*

Quantum machine learning for particle physics using a variational quantum classifier

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

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

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

Abstract: Quantum machine learning aims to release the prowess of quantum computing to improve machine learning methods. By combining quantum computing methods with classical neural network techniques we aim to foster an increase of performance in solving classification problems. Our algorithm is designed for existing and near-term quantum devices. We propose a novel hybrid variational quantum classifier that combines the quantum gradient descent method with steepest gradient descent to optimise the parameters of the network. By applying this algorithm to a resonance search in di-top final states, we find that this method has a better learning outcome than a classical neural network or a quantum machine learning method trained with a non-quantum optimisation method. The classifiers ability to be trained on small amounts of data indicates its benefits in data-driven classification problems.

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

ArXiv ePrint: 2010.07335

https://doi.org/10.1007/JHEP02(2021)212

Contents

1 Introduction

To discover new physics at the LHC, highly complex rare signal events have to be separated from a large number of Standard Model background events. Novel reconstruction techniques often rely on machine learning algorithms which show an outstanding ability to find correlations in high-dimensional parameter spaces to discriminate signal from background processes. In collider phenomenology, the feature space on which the machine-learning methods are trained to classify events into signal and background consists usually of physical observables of reconstructed objects, e.g. the transverse momentum of a jet $p_{T,j}$ or the total amount of missing transverse energy $\not{E}_T$.

The most popular machine learning techniques in recent years are artificial neural networks (NN), which are built on three pillars:

i. an adaptable complex system that allows approximating a complicated function,

ii. the calculation of a loss function in the output layer which is used to define the task the NN algorithm should perform by minimising this function, and

iii. a way to update the network continuously while minimising the loss function, e.g. through backpropagation.

In NNs the adaptable system in (i) consists of a variable number of layers made of interconnected neurons. The neurons receive inputs from previous layers in terms of weights and a bias, which are then processed as arguments of an activation function. Due to its modular setup and variable complexity an NN can be trained to perform a large spectrum of tasks.[1]

Quantum machine learning is an emergent research field which aims to release the prowess of quantum computing to improve machine learning methods. At the moment a full quantum neural network, where all three pillars are combined in an algorithm that is entirely built on the principles of quantum information processing, is not attainable. However, with present or at least with near-term quantum devices dedicated quantum algorithms can support pillars (i)-(iii) individually in form of a hybrid quantum machine learning approach.

In relation to NNs novel techniques are being developed and applied in a beneficial way for each of the pillars (i)-(iii) above. For example and concreteness, to support (i) quantum nodes can be connected with each other to form a variational circuit [8–10] or added to a classical neural network in a hybrid approach [11, 12]. For (ii) the loss function can be calculated using a quantum algorithm, e.g. in variational quantum algorithm approaches [13, 14]. Whereas for (iii) one can minimise the loss function using a quantum annealer [15, 16] or a quantum optimisation algorithm [17, 18].

Thus, by combining quantum computing methods with a NN and applying this approach to tackle challenges in particle physics we aim to foster an increase of performance associated with quantum computing algorithms[2] which can then translate into an improved sensitivity in searches for novel physical phenomena. One of the most popular applications of machine learning in particle physics is classification. To pursue this task using quantum machine learning, we construct a novel hybrid neural network, based on a quantum variational classifier. Quantum variational classifiers are known to have an advantage in model size compared to classical neural networks [10]. This allows us to augment the optimisation process of our hybrid network using the quantum gradient descent (QGD) method, which is inspired by the natural gradient descent method. Such complex optimisation methods are often computationally prohibitive for deep neural networks. Variational quantum classifiers are structurally very similar to classical neural networks and provide therefore an instructional framework to discuss in how far (i)-(iii) of classical NNs can be augmented using quantum computing elements.

Specifically, we use a quantum neural network approach, i.e. trainable quantum nodes connected in a circuit, and also include classic neural network elements, i.e. a bias term.

[1]Applications range from playing Go [1] or Chess [2], over classification and image recognition [3, 4] to natural language processing [5] and generative algorithms [6]. Beyond classification and the regression of data points NN can also be used to find solutions to functionals and integro-differential equations [7].

[2]Relevant to particle physics, a recent surge in proposals has emerged in how quantum computing can provide benefits for a variety of tasks. Quantum annealers, for example, perform continuous time quantum computations and are therefore well-suited to study the dynamics of quantum systems, even quantum field theories [19, 20], and in solving optimisation problems [21]. Quantum gate computers are in particular a popular choice to calculate multi-particle processes [22–32], often with field theories mapped onto a discrete quantum walk [33–36] or a combined hybrid classical/quantum approach [37–40].

During training, we use a modified quantum optimisation algorithm, based on quantum gradient descent [18], designed to account for the classic elements of our model. We apply this method to a Z' resonance search, which decays to a pair of top quarks [41–43]. This provides a timely and realistic playground for a phenomenologically relevant classification problem. Samples of top quark pairs where one top quark decays hadronically and the other leptonically can be purified to a very high degree, i.e. the confidence that one trains on a pure $t\bar{t}$ sample is very high. Although $t\bar{t}$ production results in jet and lepton-rich final states, for the purpose of a transparent discussion of how variational quantum classifiers can be used to support searches of new physics, we limit ourselves to only two feature variables as input to the NNs, i.e. the transverse momentum of the hardest bottom quark p_{T,b_1} and the amount of missing transverse energy in the event $\not{E}_T$. Extending the feature space is conceptually straightforward and will improve the networks ability to discriminate between signal and background, however, it will impact on the size of the network and the number of qubits, which would prevent us from running our hybrid quantum neural network on a real quantum device.

We will compare our VQC model against a classical neural network model of similar complexity, i.e. with a similar number of trainable parameters. This will allow us to answer the question of whether there can be a quantum advantage due to a more efficient training of the VQC or the exploitation of quantum gradient descent. Current quantum devices are limited in their numbers of qubits and are prone to noise errors. Thus, the implementation of larger models is currently prohibitive. State-of-the-art classical neural networks that include more features in the dataset and a larger number of trainable parameters would easily outperform a small-scale VQC. Conversely, we hope that the quantum advantage we observe in such a VQC over small-scale classical neural networks prevails when quantum devices that can accommodate more complex models become accessible.

The paper is structured as follows: section 2 is dedicated to a pedagogical overview to variational quantum classifiers (VQC) and to how VQC contribute to pillars (i) and (ii) in the context quantum machine learning algorithms. Section 3 addresses pilar (iii), where we introduce various optimisation methods applicable to the training of the NN during backpropagation. In section 4 we outline the technical setup for the analysis on pseudo-data. Subsequently, as described in section 5, we train and test two different quantum machine learning models. One will use an entirely classical approach of gradient descent while the other is trained using a quantum optimisation method. To provide a baseline we compare to a classical neural network. Finally, we provide a summary and concluding remarks in section 6.

2 Structure of a Variational quantum classifier

Variational quantum classifiers are a form of quantum neural network that can be used for supervised learning. This is achieved by designing a quantum circuit that behaves similarly to a traditional machine learning algorithm. The quantum machine learning algorithm contains a circuit which depends on a set of parameters that, through training, will be optimised to reduce the value of a loss function. This trained circuit is described

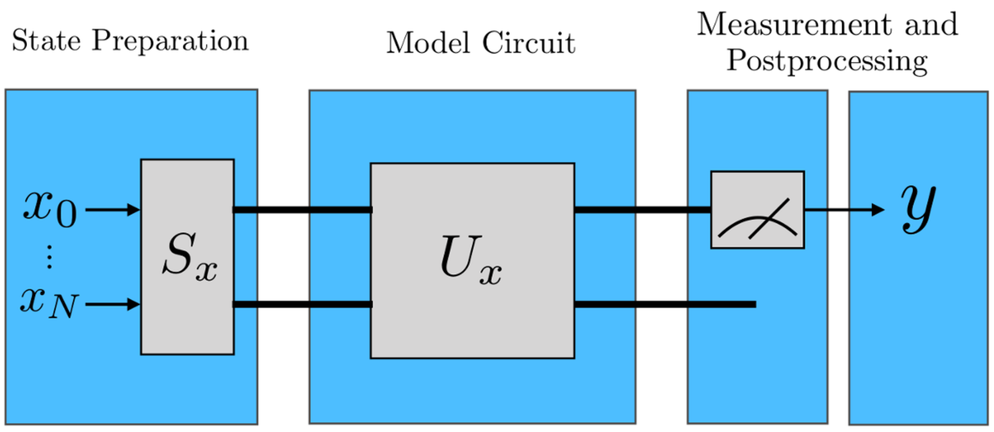


Figure 1. A variation classifier described by 3 parts. The state preparation circuit is desgined to take our input, $x \in \mathbb{R}^n$, and encode it in a N-qubit quantum state. The model circuit will apply trainable and non-trainable gates to this state. In the final steps we measure the states and apply any postprocessing necessary. This model is inspired by circuit-centric quantum classifiers [10].

in functional form by

$$f(w, b, x) = y, \tag{2.1}$$

where f is the network, y is the network output used to calculate the loss function L, the network has trainable parameter w, b and input data x. 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, i.e. nodes in the classical neural network while a quantum circuit is composed of quantum gates, and share techniques used for training.

Our classifier, is designed as a circuit-centric quantum classifier [10]. It is structurally depicted in figure 1 and consists of three parts: (1) the *state preparation circuit*, (2) the *model circuit* and (3) the *measurement and postprocessing*. These three parts of our model can be related in turn to the three pillars of machine learning, discussed in section 1. Our classifier corresponds to (i) a complex adaptable system that (ii) calculates the value of a loss function. The continuous adaptation of the parameters w and b, after obtaining y through a measurement with the aim to continually reduce the loss function L, directly relates to the network optimisation of (iii).

More specifically, the state preparation step, shown in figure 2, encodes the input data to an N-qubit quantum state. In classical computer algorithm this is carried out with bits, whereas on a quantum computer this is performed using qubits. A qubit is a 2-state quantum system which can be parametrised by

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle = \begin{pmatrix} \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2}e^{i\phi} \end{pmatrix}. \tag{2.2}$$

The state of eq. (2.2) can be visualised as a vector on the Bloch sphere. By performing operations on a qubit one rotates the vector on the Bloch sphere. Circuits can be constructed to act on numerous qubits, where a 2-qubit state can be described as a tensor product of two 1-qubit states

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle\,. \tag{2.3}$$

The model circuit is constructed from gates that evolve the input state. The circuit is based on unitary operations and depends on external parameters which 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.

2.1 State preparation

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

$$x \mapsto S_x|\phi\rangle = S_x|0\rangle^{\otimes n} = |x\rangle\,, \tag{2.4}$$

where $|\phi\rangle = |0\rangle^{\otimes n}$. The number of qubits n is defined by the number of features in our dataset.

The parametrisation of the encoding can affect the decision boundaries of the classifier and can therefore be chosen in a form that suits the problem at hand [44]. Here, we use the so-called angle encoding

$$|x\rangle = \bigotimes_{i=1}^{n} \cos(x_i)|0\rangle + \sin(x_i)|1\rangle\,, \tag{2.5}$$

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

$$R_y(\theta) = \begin{pmatrix} \cos(\theta/2) & \text{-}\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix}\,. \tag{2.6}$$

2.2 Model circuit

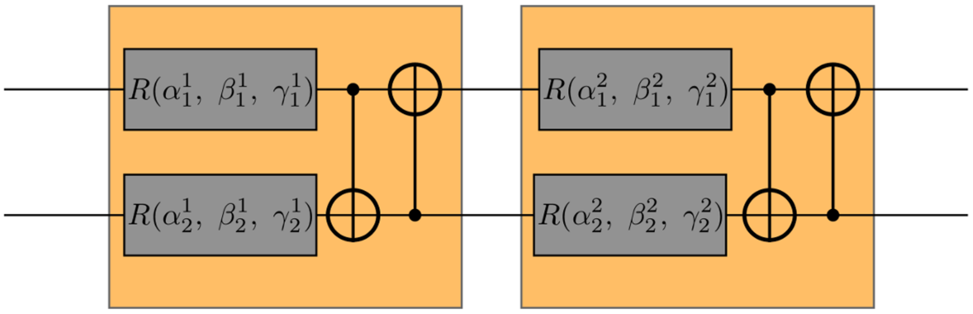
Given a prepared state, $|x\rangle$, the model circuit, $U(w)$, maps $|x\rangle$ to another vector $|\psi\rangle = U(w)|x\rangle$. In turn $U(w)$ consists of a series of unitary gates and can be decomposed as

$$U(w) = U_{l_{\max}}(w_{l_{\max}}) \ldots U_l(w_l) \ldots U_1(w_1)\,, \tag{2.7}$$

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

$$G(\alpha,\beta,\gamma,\phi) = e^{i\phi} \begin{pmatrix} e^{i\beta}\cos\alpha & e^{i\gamma}\sin\alpha \\ -e^{-i\gamma}\sin\alpha & e^{-i\beta}\cos\alpha \end{pmatrix}\,. \tag{2.8}$$

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 suffice to parametrise a single qubit gate.



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

Figure 2. Circuit diagram for our variational quantum classifier model, made of two qubits in each of the two layers.

We use a rotation gate, R, and CNOT in our model. 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 set of learnable parameters $w = (\alpha,\ \beta,\ \gamma)$

$$R(\alpha,\ \beta,\ \gamma) = R_Z(\gamma)R_Y(\beta)R_Z(\alpha) = \begin{pmatrix} e^{-i(\alpha+\gamma)}\cos(\beta/2) & -e^{-i(\alpha-\gamma)}\sin(\beta/2) \\ e^{-i(\alpha-\gamma)}\sin(\beta/2) & e^{i(\alpha+\gamma)}\cos(\beta/2) \end{pmatrix} \tag{2.9}$$

The angles of eq. (2.9) are a subset of 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. The number of qubits will mirror the number of features in our dataset whereas 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 or other frameworks used for a more flexible network design [9].

Each layer in our model contains two CNOT gates, a standard 2-qubit gate in quantum computing with no learnable parameters. A CNOT, if used alongside a Hadamard gate, could be used to introduce entanglement into our circuit. These gates flip the state of a qubit based on the value of another control bit.[3] Each gate in the layer uses a different qubit as the control bit. The model circuit of the VQC used here is shown in figure 2.

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

$$\mathbb{E}(\sigma_z) = \langle 0|S_x(x)^\dagger U(w)^\dagger O U(w) S_x(x)|0\rangle = \pi(w,x)\,, \tag{2.10}$$

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

[3]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

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

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 be a trainable parameter. Its introduction increases model flexibility and ensures the classifier output is continuous. We can write the output of our model, before thresholding, by combining the expectation value of the model circuit $\pi(w, x)$ and the bias term b:

$$f(w, b, x) = \pi(w, x) + b\,. \tag{2.11}$$

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

$$\mathrm{cls}(w, b, x) = \begin{cases} 1 & \text{if } f(w, b, x) > 0\,, \\ -1 & \text{else}\,. \end{cases} \tag{2.12}$$

The final steps, the calculation of the loss function and carrying out the optimisation procedure will be discussed in section 3.

3 Optimisation

As alluded to above, during training we aim to find the values of w and b to optimise a given loss function. We can perform optimisation on a quantum neural network similar to how it is done on a classical neural network. In both cases, we perform a forward pass of the model and calculate a loss function. We can then backpropagate over the network and update our trainable parameters. This is the equivalent of the third pillar of machine learning, mentioned in section 1.

During training we use the mean squared error (MSE) as loss function.[4] This allows us to find a distance between our predictions and truth, represented by the value of the loss function

$$L = \frac{1}{n} \sum_{i=1}^{n} \left(y_i^{\text{truth}} - f(w, b, x_i) \right)^2 . \tag{3.1}$$

We will train our model using vanilla gradient descent and quantum gradient descent [18]. The latter is a quantum optimisation algorithm designed to be performed on a hybrid network we have proposed above. Further, we will exploit the advantage in model size of a variational quantum classifier compared to a classical neural network to improve its backpropagation method.

[4]Often, for classification tasks using classical neural networks, the cross entropy is a preferred measure for the loss function. We find the difference between cross entropy and MSE to be numerically irrelevant for the application discussed in section 5 and therefore follow the choice for the loss function of refs. [10, 45]. On testing the difference between using MSE or cross entropy to define the loss function, we find that either loss function results in very similar model performance, with both methods achieving around 70% accuracy.

3.1 Backpropagation

To perform backpropagation for a network with adjustable parameters $\theta = (w, b)$ we compute the change of it's output when varying θ as the gradient $\frac{\partial}{\partial\theta} f$. For a quantum circuits the gradient of the network output is calculated using the parameter-shift rules [46, 47]. Being able to calculate gradients for quantum circuit outputs opens up the possibility of using gradient descent methods to train our variational quantum circuit. The methodology is identical to how optimisation and training is performed on classical neural networks.

For the parameter-shift rules to be correctly applied to a quantum circuit certain conditions must be met. We can represent a unitary gate in the form

$$U(\theta) = e^{i\theta V}, \tag{3.2}$$

where θ are our network parameters and V is the Hermitian generator of $U(\theta)$. For a circuit f that includes gates that can be represented in the form of eq. (3.2), if V has at least two distinct eigenvalues, the parameter-shift rules provide the relation

$$\frac{\partial}{\partial\theta} f = r\left[f(\theta + s) - f(\theta - s)\right], \tag{3.3}$$

where the shift $s = \pi/4r$. The value of r is an arbitrary normalisation factor which we choose in our implementation to be $r = 1/2$. Following eq. (3.3) we can calculate gradients over quantum gates by shifting the parameters. As the difficulty of calculating $\frac{\partial}{\partial\theta} f$ has been reduced to probing the quantum circuit at different parameter points, it is now possible to evaluate the gradient fast and efficiently on a quantum device.

3.2 From gradient descent to quantum gradient descent

The geometry of the parameter space has a direct impact on the reliability and efficiency of an optimisation algorithm [48]. Thus, 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 l_2 Euclidean geometry is an appropriate choice for variational models [49].

For our problem at hand, we propose to augment the vanilla gradient descent method, often used in classical neural networks, by the quantum gradient descent method [18].

In the vanilla gradient descent method, the network parameter vector θ_t at iteration step t is updated with the goal that θ_{t+1} results in a smaller loss function $L(\theta)$. Thus, one approach is to update θ_t in the direction of the steepest decline, $-\nabla L(\theta)$, weighted by the learning rate η

$$\theta_{t+1} = \theta_t - \eta\nabla L(\theta). \tag{3.4}$$

However, this optimisation is performed on the geometry of an l_2 vector space, which influences the performance and how new parameters are found. While all parameters are updated with the same step size, the rate at which the loss function changes for each model parameter can vary greatly. By using this form of gradient descent it is possible to miss the global minimum in the space of the loss function. An improvement would be to change the coordinate system to ensure the loss function changed consistently with each step for each parameter or to find an optimisation method that was invariant under re-parametrisation.

One way to address this problem is the use of natural gradient descent, which makes use of the Fisher Information Matrix [50, 51] and is a classical extension to vanilla gradient descent method. The parameters of a network (the weights and biases) exist on a parameter space that has a Riemannian geometry. The Fisher Information Matrix is the metric that defines this space. Since this metric includes information on the geometric structure of the Riemannian space of the network parameters, its inclusion into the gradient descent optimisation leads the network to learn more effectively. In addition, it is invariant under re-parametrisation, and thus advantageous in finding an effective parametrisation. Algorithmically natural gradient descent can be written as

$$\theta_{t+1} = \theta_t - \eta F^{-1} \nabla L(\theta)\,, \tag{3.5}$$

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 l_2 geometry. Although the inclusion of F^{-1} in eq. (3.5) in general improves the performance of the optimisation algorithm, in most classical deep neural networks calculating the inverse of a large matrix becomes computationally prohibitively expensive. However, in our hybrid network, which benefits from a small model size, the parameter space is rather 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 does indeed have a geometry that can be described by an invariant 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 [18]

$$\theta_{t+1} = \theta_t - \eta g^{+} \nabla L(\theta)\,, \tag{3.6}$$

where g^+ is the pseudo-inverse of the Fubini-Study metric. We implement this algorithm using the PennyLane package [52], which will allow us to find the steepest descent in the parameter space of the quantum states. The approach of eq. (3.6) 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. Thus, we propose to optimise our weights using quantum gradient descent (3.7) while using vanilla gradient descent for the classical bias term b. Calculating both gradients at each optimisation step,

$$\begin{aligned} \theta^w_{t+1} &= \theta^w_t - \eta g^{+} \nabla^w L(\theta)\,, \\ \theta^b_{t+1} &= \theta^b_t - \eta \nabla^b L(\theta)\,, \end{aligned} \tag{3.7}$$

ensures our entire range of parameters is optimised simultaneously.

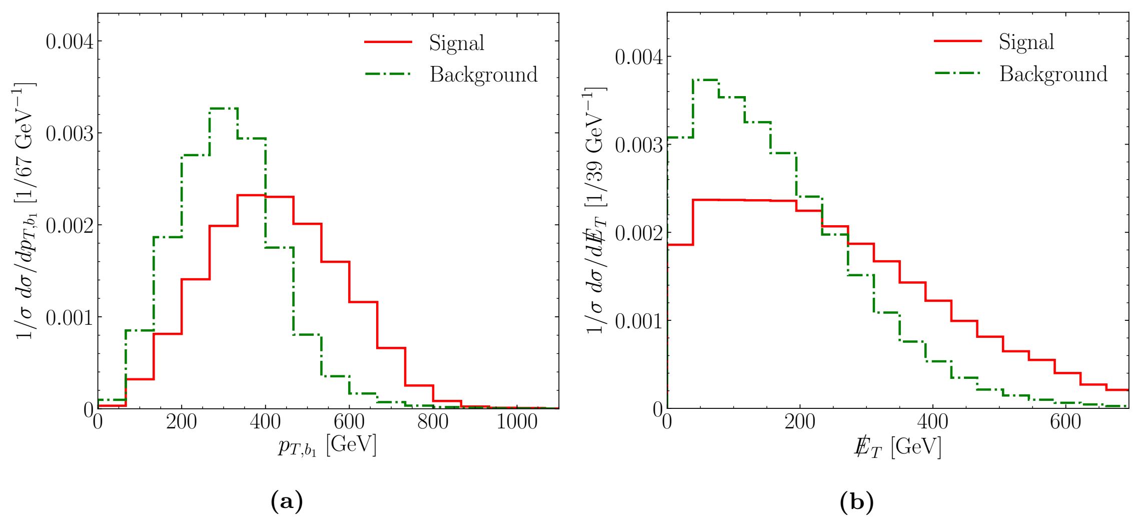


Figure 3. Distribution of signal and background of the (a) p_T of the hardest b-jet and the (b) missing energy.

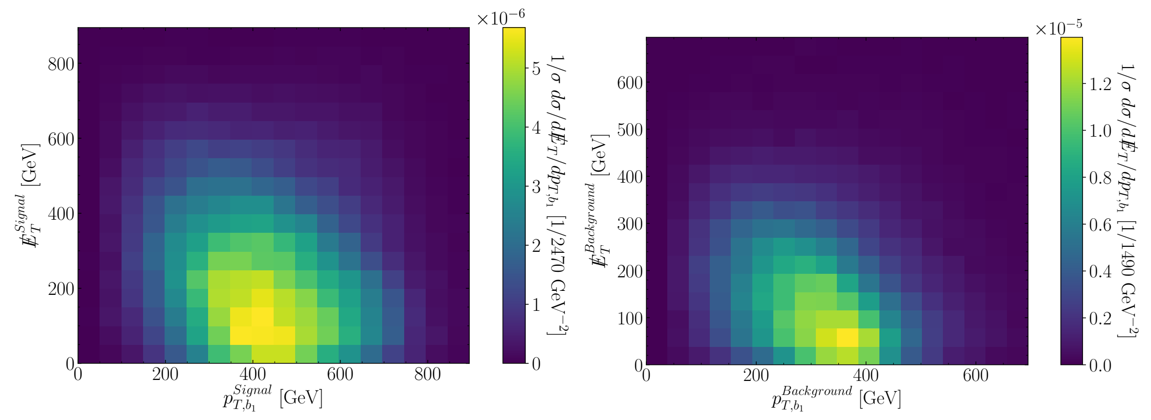
4 Analysis setup

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. When the top quarks are decayed we have forced one quark to have a hadronic decay while the other has a leptonic decay. A heavy new boson, Z' [53], is used as signal, with a mass of 2 TeV and a width chosen to be 89.6 GeV [54]. Similar to the background 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. All events are generated using MadGraph5_aMC@NLO [55] while the parton showering and hadronisation is performed with Pythia 8.2.

Using the Cambridge-Aachen algorithm [56] the hadrons and the non-isolated leptons are clustered into jets with radius $R = 1.0$. This is based on work using fat jets to reconstruct highly boosted top quarks [57–60]. Using FastJet [61] the k_T algorithm is implemented to recluster the hardest two fat jets into jets with radius $R = 0.2$. Based on proximity to a B-meson, jets are b-tagged while 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.

The selection of these events is then based on numerous criteria. For the two fat jets in an event, one must contain at least one b-jet while the other 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.

In the following, the analysis performed is exclusively based on the transverse momentum of one b-jet (p_{T,b_1}) and the event's missing energy ($\not{E}_T$). We show these observable's distributions in figure 3 and heatmaps in figure 4.



(a) **(b)**

Figure 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 $x_{scaled} \in [0, \pi]$. 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.

5 Network performance

We are comparing three models: a classic neural network trained with standard gradient descent (NN-GD), a VQC trained with standard gradient descent (VQC-GD) and a VQC trained with our quantum gradient descent method (VQC-QGD) of section 3.2.

The VQC model consists of two qubits, corresponding to the two features p_{T,b_1} and $\not{E}_T$, and two layers. Each layer has a rotation gate for each qubit followed by two CNOT gates. We implement this model, depicted in figure 2, using PennyLane [52] and train it for 30 epochs with a batch size of 32 events and an initial learning rate of $\eta = 0.01$. During training, for all models, we reduce the learning rate value whenever the loss plateaus. However, learning rate reduction, in this instance, appears to have little effect on the performance of the network during training. The networks poor capacity to discriminate signal from background is reflective of the similarity between the two. Figure 4 shows the probability density for the events to populate areas in the feature space (p_T, $\not{E}_T$). The similarity between signal and background prevents the networks to benefit from a continuous learning rate reduction, for classical NNs and our hybrid method alike.

We anticipate that a significant advantage of the variational quantum classifier lies in its smaller network structure, which allows to employ computationally more expensive optimisation algorithms, as detailed in section 3, giving in turn rise to a faster learning rate. 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 or processes with small production cross section.

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

Table 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).

Thus, to compare the network's ability to learn quickly, we limit ourselves to 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 the size of training samples have on the model performance, we train a second set of models using only 500 events each. While we carry out the training on the PennyLane's inbuilt simulator throughout, we test their performance on the PennyLane simulator, the IBM Q simulator[5] and IBM Q Yorktown.[6] Accessing the IBM hardware was done through PennyLane's Qiskit plugin [62, 63]. For all backends, in training and testing, we use a total of 8192 shots.

To provide a baseline we trained a classical neural network with a vanilla gradient descent optimiser. To provide a fair and instructive comparison the network has been constructed to have a similar number of trainable parameters as the variational classifier model. The network consists of one hidden layer with 3 nodes and a ReLu activation function. The rest of the hyperparameters match what was used to train the variational classifier. To implement the network we used Keras [64] with a TensorFlow backend [65].

We found that training a classical network of this size was unstable, sometimes resulting in the loss plateauing around 1 and being unable to classify the samples. To account for the instability we saw during training of the classifier we ran each model 15 times. The results presented in figure 5 show the average loss from these runs, for each model. We see, from figure 5, optimisation using the quantum gradient leads to a faster convergence than using the traditional gradient descent optimisation and the classical neural network.

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 where used for testing. Figure 6 shows the ROC curve for the chosen VQC-QGC, VQC-GD and NN-GD models. Table 1 shows the performance of the quantum gradient descent method when the test data is applied to it. We see that the model, trained on the simulator, still performs well on the real hardware. In figure 6 we see an example of the variational classifier output before the decision boundary is applied and the ROC curve for each model.

[5]32-qubit backend: IBM Q team, "IBM Q simulator backend specification V0.1.547," (2020). Retrieved from https://quantum-computing.ibm.com.

[6]5-qubit backend: IBM Q team, "IBM Q 5 Yorktown (ibmqx2) backend specification V2.1.0," (2020). Retrieved from https://quantum-computing.ibm.com.

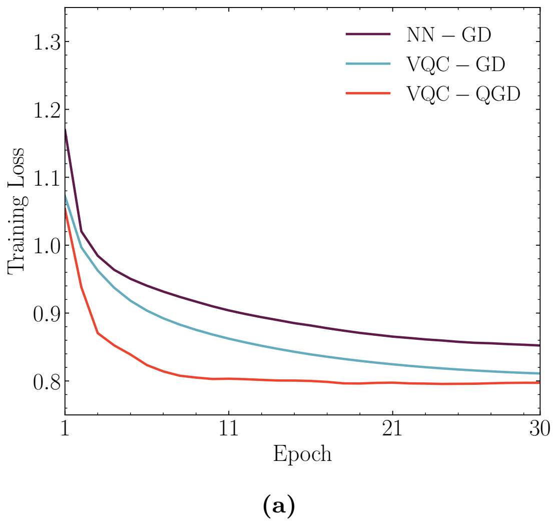
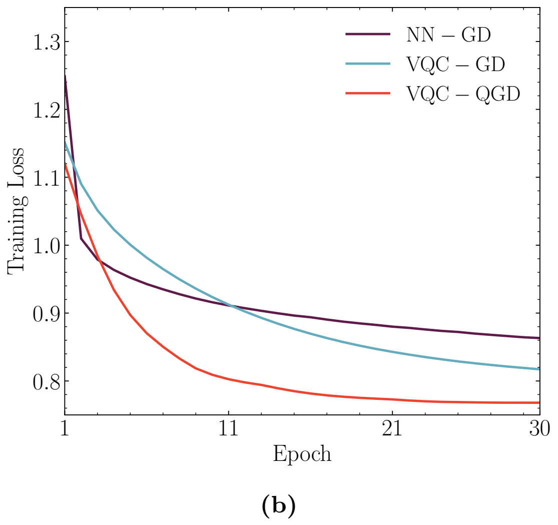
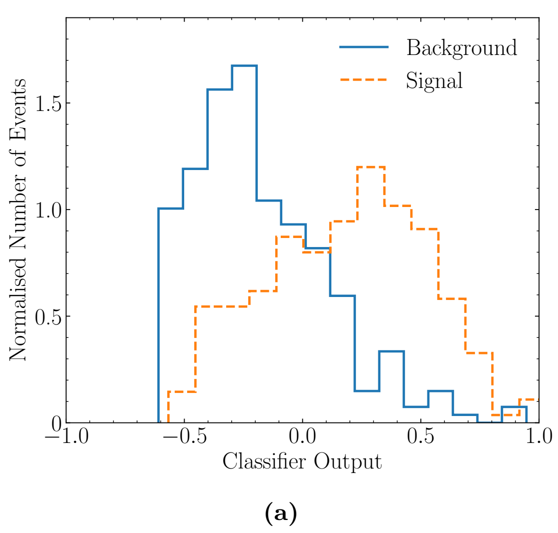


Figure 5. Comparison of the averaged training history for 15 runs of the QVC models trained with quantum gradient descent, QVC models trained using vanilla gradient descent and the classical NN models. Figure (a) show models trained with 1500 samples and figure (b) shows models trained with 500 samples.





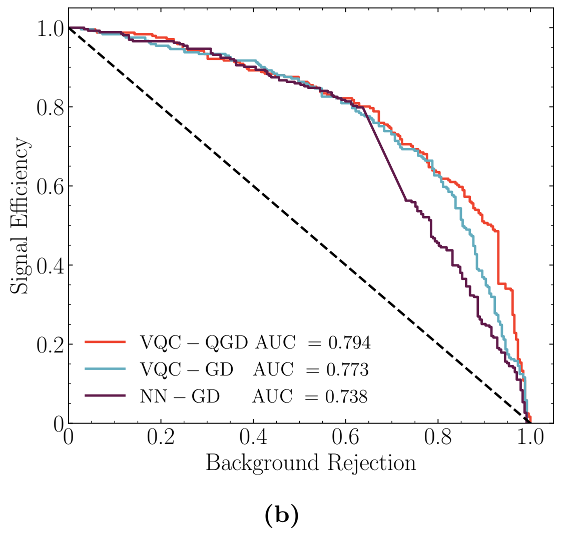


Figure 6. (a) Output of a QVC model trained with quantum gradient descent and (b) ROC curve for a QVC model trained with quantum gradient descent, a QVC model trained with vanilla gradient descent and the classical NN.

6 Conclusions

One of the tasks with paramount importance for searches of new physics at collider experiments is the design of methods to distinguish rare signal events from large Standard Model backgrounds. In recent years increasing effort was dedicated to developing novel machine learning methods to help find correlations in high-dimensional parameter spaces. Harnessing the advantages found in quantum computing and combining them with classical neural networks to form a hybrid approach would provide another way to continue the improvement of these algorithms, possibly already accessible on near-term devices.

Quantum machine learning is an emergent research field that aims to apply these benefits to machine learning. To explore the potential quantum advantage that could come along with quantum machine learning we propose a novel hybrid neural network, based on a variational quantum classifier. Variational quantum classifier models are in many ways analogous classical neural networks. An advantage that a VQC classifier provides over a classical neural network is its small model size. The model proposed uses a quantum algorithm equivalent of natural gradient descent. Typically, due to the need to invert large matrices, natural gradient descent is computationally prohibitive on deep neural networks. However, thanks to the model-size advantage of the VQC we can make use of quantum gradient descent to optimise our network.

Thus, we combine the use of quantum gradient descent to optimise the quantum gate parameters in the model while using classical gradient descent to optimise the classical bias term. This model was used to perform a Z' resonance search. We compared the performance 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.

Acknowledgments

We acknowledge use of the IBM Q for this work. We thank Steve Abel for helpful discussions. We acknowledge funding from the STFC under grant ST/P001246/1 and ST/T001011/1.

A The Fubini-Study metric and 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 [66–68]. The FST can be found by taking the real part of the Quantum Geometric Tensor (QGT).

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 investigating the distance between the two 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

$$ds^2 \equiv 1 - |\langle\psi_0(\theta)|\psi_0(\theta + d\theta)\rangle|^2 \,. \tag{A.1}$$

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

$$a_n = \langle\psi_n(\theta + d\theta)|\psi_0(\theta)\rangle \,, \tag{A.2}$$

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

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

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

$$G_{ij} = \langle\partial_i\psi_0|\partial_j\psi_0\rangle - \langle\partial_i\psi_0|\psi_0\rangle \langle\psi_0|\partial_j\psi_0\rangle \,. \tag{A.4}$$

This tensor therefore signifies the distance between the two quantum states [69]. The Fubini-Study metric is the real part of this tensor, $g_{ij}(\theta) = \mathrm{Re}[G_{ij}(\theta)]$. We can view the Fubini-Study metric as a distance measure between the wave functions, or transition probability between the states [67].

Consequently, G_{ij} can be calculated on quantum hardware [18]. We consider a variational circuit where each layer l is parametrised by θ_l and includes gates $U(\theta_l)$. These gates U and their functional relation to the Hermitian generator matrix V are described in section 2.2 and eq. (3.2), resulting in the relations

$$\begin{aligned} \partial_i U_l(\theta_l) &= -iV_i U_l(\theta_l) \,, \\ \partial_j U_l(\theta_l) &= -iV_j U_l(\theta_l) \,, \end{aligned} \tag{A.5}$$

where V_i and V_j are Hermitian generator matrices. From eq. (A.5) we can find

$$\langle\partial_i\psi_\theta|\partial_j\psi_\theta\rangle = \langle\psi_l|V_iV_j|\psi_l\rangle \,, \tag{A.6}$$

$$i\,\langle\psi_\theta|\partial_j\psi_\theta\rangle = \langle\psi_l|V_j|\psi_l\rangle \,. \tag{A.7}$$

By considering both (A.6) and (A.7) a representation of the Quantum Geometric Tensor can be formed for a block of parameters that exist in layer l

$$G^l_{ij} = \langle\psi_l|V_iV_j|\psi_l\rangle - \langle\psi_l|V_i|\psi_l\rangle \langle\psi_l|V_j|\psi_l\rangle \,. \tag{A.8}$$

The quantum states ψ_l can be determined experimentally from the variational quantum classifier. Importantly, this approximation of the QGT also allows to find the Fubini-Study metric by taking the real part, such that $g^l_{ij} = \mathrm{Re}[G^l_{ij}]$. To calculate the inverse, we use the Moore-Penrose pseudo inverse

$$g^+ = (g^T g)^{-1} g^T \,. \tag{A.9}$$

This method allows to finding an inverse matrix even if the matrix cannot be inverted, as shown in eq. (A.9). In cases where the matrix is invertible the matrix pseudo inverse and matrix inverse are identical.

References

[1] D. Silver et al., *Mastering the game of go without human knowledge*, *Nature* **550** (2017) 354.

[2] D. Silver et al., *A general reinforcement learning algorithm that masters chess, shogi, and go through self-play*, *Science* **362** (2018) 1140.

[3] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, arXiv:1409.1556.

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

[5] I. Sutskever, O. Vinyals and Q. Le, *Sequence to sequence learning with neural networks*, *Adv. Neural Inf. Process. Syst.* **4** (2014).

[6] I.J. Goodfellow et al., *Generative Adversarial Networks*, arXiv:1406.2661 [INSPIRE].

[7] 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 [arXiv:1902.05563] [INSPIRE].

[8] E. Farhi and H. Neven, *Classification with quantum neural networks on near term processors*, arXiv:1802.06002.

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

[10] M. Schuld, A. Bocharov, K.M. Svore and N. Wiebe, *Circuit-centric quantum classifiers*, *Phys. Rev. A* **101** (2020) 032308.

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

[12] A. Mari, T.R. Bromley, J. Izaac, M. Schuld and N. Killoran, *Transfer learning in hybrid classical-quantum neural networks*, arXiv:1912.08278.

[13] A. Peruzzo et al., *A variational eigenvalue solver on a photonic quantum processor*, *Nat. Commun.* **5** (2014) .

[14] E. Farhi, J. Goldstone and S. Gutmann, *A quantum approximate optimization algorithm*, arXiv:1411.4028.

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

[16] E. Farhi, J. Goldstone, S. Gutmann and M. Sipser, *Quantum computation by adiabatic evolution*, quant-ph/0001106.

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

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

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

[20] S. Abel and M. Spannowsky, *Observing the fate of the false vacuum with a quantum laboratory*, arXiv:2006.06003 [INSPIRE].

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

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

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

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

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

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

[27] C.W. Bauer, W.A. de Jong, B. Nachman and D. Provasoli, *Quantum Algorithm for High Energy Physics Simulations*, *Phys. Rev. Lett.* **126** (2021) 062001 [arXiv:1904.03196] [INSPIRE].

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

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

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

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

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

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

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

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

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

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

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

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

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

[41] CMS collaboration, *Search for Anomalous $t\bar{t}$ Production in the Highly-Boosted All-Hadronic Final State*, *JHEP* **09** (2012) 029 [*Erratum ibid.* **03** (2014) 132] [arXiv:1204.2488] [INSPIRE].

[42] ATLAS collaboration, *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. C* **78** (2018) 565 [arXiv:1804.10823] [INSPIRE].

[43] ATLAS collaboration, *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. D* **99** (2019) 092004 [arXiv:1902.10077] [INSPIRE].

[44] R. LaRose and B. Coyle, *Robust data encodings for quantum classifiers*, arXiv:2003.01695.

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

[46] K. Mitarai, M. Negoro, M. Kitagawa and K. Fujii, *Quantum circuit learning*, *Phys. Rev. A* **98** (2018) 032309.

[47] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac and N. Killoran, *Evaluating analytic gradients on quantum hardware*, *Phys. Rev. A* **99** (2019) 032331.

[48] B. Neyshabur, R. Salakhutdinov and N. Srebro, *Path-sgd: Path-normalized optimization in deep neural networks*, arXiv:1506.02617.

[49] A. Harrow and J. Napp, *Low-depth gradient measurements can improve convergence in variational hybrid quantum-classical algorithms*, arXiv:1901.05374.

[50] S.-i. Amari, *Natural gradient works efficiently in learning*, *Neural Comput.* **10** (1998) 251.

[51] S. ichi Amari, R. Karakida and M. Oizumi, *Fisher information and natural gradient learning of random deep networks*, arXiv:1808.07172.

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

[53] G. Altarelli, B. Mele and M. Ruiz-Altaba, *Searching for New Heavy Vector Bosons in $p\bar{p}$ Colliders*, *Z. Phys. C* **45** (1989) 109 [*Erratum ibid.* **47** (1990) 676] [INSPIRE].

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

[55] J. Alwall 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 [arXiv:1405.0301] [INSPIRE].

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

[57] T. Plehn and M. Spannowsky, *Top Tagging*, *J. Phys. G* **39** (2012) 083001 [arXiv:1112.4441] [INSPIRE].

[58] T. Plehn, M. Spannowsky and M. Takeuchi, *How to Improve Top Tagging*, *Phys. Rev. D* **85** (2012) 034029 [arXiv:1111.5034] [INSPIRE].

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

[60] D.E. Soper and M. Spannowsky, *Finding physics signals with event deconstruction*, *Phys. Rev. D* **89** (2014) 094005 [arXiv:1402.1189] [INSPIRE].

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

[62] D.C. McKay et al., *Qiskit backend specifications for openqasm and openpulse experiments*, arXiv:1809.03452.

[63] H. Abraham et al., *Qiskit: An open-source framework for quantum computing*, (2019), DOI.

[64] F. Chollet et al., *Keras*, https://keras.io, (2015).

[65] M. Abadi et al., *Tensorflow: Large-scale machine learning on heterogeneous distributed systems*, arXiv:1603.04467.

[66] T.W.B. Kibble, *Geometrization of Quantum Mechanics*, *Commun. Math. Phys.* **65** (1979) 189 [INSPIRE].

[67] D.C. Brody and L.P. Hughston, *Geometric quantum mechanics*, *J. Geom. Phys.* **38** (2001) 19 [quant-ph/9906086] [INSPIRE].

[68] R. Cheng, *Quantum geometric tensor (fubini-study metric) in simple quantum system: A pedagogical introduction*, arXiv:1012.1337.

[69] M. Kolodrubetz, D. Sels, P. Mehta and A. Polkovnikov, *Geometry and non-adiabatic response in quantum and classical systems*, *Phys. Rept.* **697** (2017) 1.