



August 03, 2022

Quantum Machine Learning for b -jet charge identification

Alessio Gianelle¹, Patrick Koppenburg², Donatella Lucchesi^{1,3}, Davide Nicotra^{3,4},
Eduardo Rodrigues⁵, Lorenzo Sestini¹, Jacco de Vries⁴, Davide Zuliani^{1,3,6}

¹*INFN Sezione di Padova, Padova, Italy*

²*Nikhef National Institute for Subatomic Physics, Amsterdam, Netherlands*

³*Università degli Studi di Padova, Padova, Italy*

⁴*Universiteit Maastricht, Maastricht, Netherlands*

⁵*Oliver Lodge Laboratory, University of Liverpool, Liverpool, United Kingdom*

⁶*European Organization for Nuclear Research (CERN), Geneva, Switzerland*

Abstract

Machine Learning algorithms have played an important role in hadronic jet classification problems. The large variety of models applied to Large Hadron Collider data has demonstrated that there is still room for improvement. In this context Quantum Machine Learning is a new and almost unexplored methodology, where the intrinsic properties of quantum computation could be used to exploit particles correlations for improving the jet classification performance. In this paper, we present a brand new approach to identify if a jet contains a hadron formed by a b or \bar{b} quark at the moment of production, based on a Variational Quantum Classifier applied to simulated data of the LHCb experiment. Quantum models are trained and evaluated using LHCb simulation. The jet identification performance is compared with a Deep Neural Network model to assess which method gives the better performance.

Published in JHEP 08 (2022) 014

© 2022 CERN for the benefit of the LHCb collaboration. CC BY 4.0 licence.

1 Introduction

Machine Learning (ML) methods are widely used in experimental particle physics [1] data manipulation. One of the most successful applications is the classification of hadronic jets at the Large Hadron Collider (LHC) experiments [2]. Jets are streams of particles produced via fragmentation and hadronization of quarks and gluons that emerge from particle collisions, *e.g.* proton-proton collisions at LHC. They are complex objects formed by many detectable particles, and it is possible to identify their properties by exploiting particle content and correlations, commonly referred to as jet substructure. Typical jet classification problems are the identification of the heavy-flavour hadron produced in the jet hadronization (*e.g.* b hadron vs c hadron) or the identification of the charge of the heavy-flavour quark that constitutes this hadron (*e.g.* b vs \bar{b}). State-of-the-art ML methods, such as Deep Neural Networks (DNN) [3], Convolutional Neural Networks (CNNs) [4], Recurrent Neural Networks (RNNs) [5], Tensor Networks [6] and Graph Neural Networks [7, 8], have been applied to jets data collected by the LHC experiments, with a clear improvement of the classification performance with respect to classical non-ML methods [9–11].

Recently, Quantum Computing (QC) has set the scene for a revolution in ML. The new approach consists of using quantum circuits to tackle classification tasks, in the framework of Quantum Machine Learning (QML) [12]: data are embedded into a quantum state, which is then passed to a variational quantum circuit, and by varying the circuit parameters a training procedure is performed by means of minimising a classical loss function. Probability measurements of the final state are then used to perform the classification. Given the intrinsic properties of quantum computation, namely superposition and entanglement, the new approach could lead to new insights from the classification point of view. Jets that originate from gluons, or quarks of a certain charge and flavor, would have a characteristic particle content and correlations between them, which could be exploited to aid the identification of the original particle. It is interesting to study if QML, by exploiting the quantum nature of the algorithm, could enhance the classification performance.

QML techniques have recently been applied to solve High Energy Physics (HEP) problems, such as signal versus background separation [13–16], anomaly detection [17], and particle track reconstruction [18, 19]. A more detailed review of QML applications to HEP can be found in Ref. [20]. This paper presents the first application of QML to the task of jet flavour identification. QML methods are performed on simulators and applied to simulated LHCb samples, to identify the charge of the b quark that forms the b hadron produced in the jet hadronization. In the rest of the paper, this task is simply referred as b -jet charge tagging.

The paper is structured in the following way: Sec. 2 provides a description of the LHCb jet reconstruction and identification together with the used dataset. In Sec. 3 the considered QML algorithms are presented while the analysis flow is described in Sec. 4. The results are discussed in Sec. 5. The conclusions and future developments are presented in Sec. 6.

2 Jet reconstruction and identification at LHCb

LHCb [21] is a single-arm spectrometer designed to study b and c hadrons in the forward region of proton-proton collisions. The reference system used at LHCb is defined by the z -axis (the beam axis) parallel to the proton beam, the x -axis parallel to the gravity acceleration, and the y -axis perpendicular to the other two. The direction of particle momentum is identified by the angles θ and ϕ , where θ is the angle between the momentum and the z -axis, and ϕ is the azimuthal angle between the projection of the momentum in the xy plane and the y -axis. The pseudorapidity η is defined as $\eta = -\log \left[\tan \left(\frac{\theta}{2} \right) \right]$. The LHCb detector covers the region in the pseudorapidity range $2 < \eta < 5$, and consists of a tracking system and a particle identification system [22]. The tracking system is formed by a vertex detector, several tracking stations and a dipole magnet. The vertex detector efficiently reconstructs the decay vertex of b - and c -hadron decays, while the tracking stations measure the trajectories (tracks) of charged particles and their momenta. The particle identification system is formed by two Ring Imaging Cherenkov detectors, two calorimeters (electromagnetic and hadronic) and a muon detector, that allows to precisely determine the type of the particles produced in the collision. Jets are reconstructed using inputs selected by the Particle Flow algorithm [23]. These are charged particles detected by the tracking system, and neutral particles reconstructed in the calorimeter system as energy clusters isolated from tracks [24]. The selected particles represent the input of the anti- k_t clustering algorithm [25] for jet clusterization. The jet clusterization is done using FASTJET [26] with radius $R = 0.5$. The jet momentum is defined as the sum of the momenta of the particles forming the jet. The jet axis is defined as the direction of the jet momentum. Particles inside a jet are approximately contained in a cone structure with a distance from the jet axis $\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2} = 0.5$, where $\Delta\eta$ is the difference in pseudo-rapidity and $\Delta\phi$ is the difference in the azimuthal angle with respect to the jet axis.

The goal is to distinguish between jets that contain a b or \bar{b} hadron just after the hadronization, *i.e.* in the instant of b -hadron production and not at decay, since neutral B -mesons can undergo flavour oscillation. Therefore the analysis is restricted to a sample of jets that belong to these two categories, labelled as b jets and \bar{b} jets.

This preliminary selection is performed in two steps:

- reconstruction of a vertex (secondary vertex) significantly displaced from the primary proton-proton interaction point, using tracks detected by the vertex detector [27], representing the b -hadron decay point;
- identification of the jet that contains the secondary vertex within its cone.

The b -jet charge tagging subsequently becomes a binary classification problem where the jet can belong to one of the two exclusive categories: b jets or a \bar{b} jets. The charge of the b quark at production is correlated to the charge of the b -hadron decay products. This correlation is not perfect, since neutral B -mesons can oscillate, and the charge of the b quark at production may be different from the charge at decay. As an example, in semi-leptonic decays, the b hadron can produce a muon, whose charge is directly related to the charge of the b quark. However, due to possible B -meson oscillations and to the large number of particles, including muons, in a jet the information is diluted and that has to be taken into account.

Two types of b -jet charge tagging algorithms are currently used in LHCb:

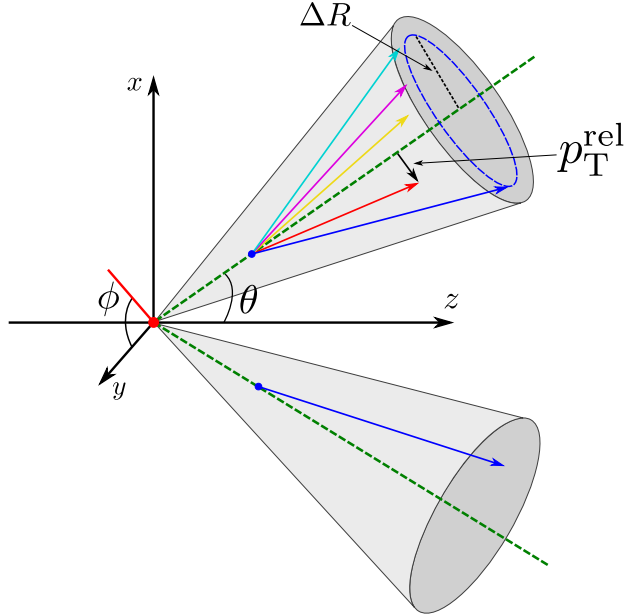


Figure 1: Sketch representing possible jet tagging methods. In the exclusive method the information comes from a particle, e.g. the muon, whose charge is correlated to the b hadron (lower jet); in the inclusive method, the information is extracted from the jet constituents (upper jet). The magnitude of the particle momentum transverse to the jet axis is labelled as p_T^{rel} .

- *exclusive algorithms*, based on information coming from particles inside the jet strictly correlated with the b -hadron decay, such as the muon;
- and *inclusive algorithms*, which aim to exploit the jet sub-structure, *i.e.* information coming from the jet constituents, as shown in Fig. 1.

The QML approach presented in this paper belongs to the category of *inclusive algorithms*. However, for the sake of comparison, the results are compared to the *muon tagging* [28], which is an exclusive algorithm. This tagger selects the muon with the highest momentum with respect to the z axis, p_T , inside the jet. This simple requirement is sufficient to identify the muon coming from a b hadron and therefore to infer the quark charge by measuring the muon charge. The efficiency of this algorithm is limited by the probability that a b hadron decays to a final state with a muon, which is $\sim 10\%$ [29].

The performance of different b -jet charge tagging algorithms are compared using the tagging power ϵ_{tag} [30–32], defined as

$$\epsilon_{\text{tag}} = \epsilon_{\text{eff}}(2a - 1)^2 \quad (1)$$

as the figure of merit, where ϵ_{eff} is the tagging efficiency, *i.e.* the fraction of jets where the classifier takes a decision, and a is the accuracy, *i.e.* the fraction of correctly tagged jets with respect to the tagged jets. The tagging power is the effective fraction of events that contribute to the statistical uncertainty in a measurement where the b -jet charge tagging is applied.

Dataset	Muon			Kaon			Pion			Electron			Proton			Q
	p_T^{rel}	q	ΔR	p_T^{rel}	q	ΔR	p_T^{rel}	q	ΔR	p_T^{rel}	q	ΔR	p_T^{rel}	q	ΔR	
Complete	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Muon	✓	✓	✓													✓

Table 1: Summary of the features contained in the two datasets.

2.1 Data samples description

LHCb simulated samples are used in the studies presented in this paper. The $b\bar{b}$ di-jets samples are produced within the LHCb simulation framework [33], which uses PYTHIA8.1 [34] with a specific LHCb configuration [35], to generate proton-proton interactions and jet fragmentation and hadronization at a center-of-mass energy of 13 TeV, and an internal implementation of EVTGEN [36] to simulate b hadron decays. The GEANT4 software [37], embedded in the LHCb framework, is used to simulate the detector response. Pairs of b and \bar{b} jets are selected by requiring for each jet a p_T greater than 20 GeV/c and a η in the range $2.2 < \eta < 4.2$, to ensure that they are well inside the instrumented part of the detector. After the pre-selection, a fixed number of 16 different features related to the jet substructure are used as input to the classifiers. Among the reconstructed particles inside a jet the muon, kaon, pion, electron and proton with the highest p_T are selected. For each particle three physical variables are considered: the magnitude of the transverse momentum to the jet axis (p_T^{rel}), the charge (q), and the distance, measured in the (η, ϕ) space, between the particle and the jet axis (ΔR). If a particle type is missing, the relative features are set to 0. The last feature is the weighted jet charge Q , defined as the sum of the charges of the particles inside the jet weighted with the particles p_T^{rel} [38–41]

$$Q = \frac{\sum_i (p_T^{\text{rel}})_i q_i}{\sum_i (p_T^{\text{rel}})_i}. \quad (2)$$

The analysis for the b -jet identification is performed with two datasets. The *complete dataset* includes the events selected with the 16 features described above. The *muon dataset* contains jets with at least one muon and only four features: p_T^{rel} , ΔR , q of the muon and the weighted jet charge Q . Table 1 summarises the characteristics of the data samples.

3 Quantum Machine Learning models

A quantum algorithm is implemented by means of a quantum circuit, namely a collection of linked quantum gates acting on a n -qubit quantum state: the measurements on the final state represent the outcome of the quantum algorithm. Parametrized Quantum Circuits (PQCs) [42] are a type of circuit that contains adjustable gates with tunable parameters. The Variational Quantum Classifier (VQC) [43] is a hybrid quantum-classical algorithm to perform classification tasks using a Machine Learning model based on a PQC with the following structure:

Data encoding data x , the features representing the jet substructure in this application, are pre-processed and encoded into a subset of the parameters of a PQC. The stage produces a quantum state $|x\rangle$ representing the input jet.

Variational circuit the state $|x\rangle$ is processed by a PQC, $U(\theta)$, featuring trainable parameters θ to be optimised during the training phase. This stage produces a final state $|\psi\rangle = U(\theta)|x\rangle$.

Prediction expectation values computed on the final state $|\psi\rangle$ are mapped to probabilities for the two labels, P_b and $P_{\bar{b}}$. The training process aims to match the label predictions with the true charge of the b -hadron in the jet in the instant of production, available in the simulations.

Two different PQC models are studied in this work: Amplitude Embedding and Angle Embedding, described below.

3.1 Amplitude Embedding

The Amplitude Embedding model consists in a PQC made by an embedding circuit followed by a variational circuit. The schematic representation of this model is shown in Fig. 2. The embedding circuit consists of an *Amplitude Encoder* that encodes up to 2^n features into the amplitude of a n -qubit quantum state, or equivalently, a vector of N features can be encoded using $\lceil \log_2 N \rceil$ qubits:

$$|x\rangle = \sum_{i=1}^{2^n} x_i |n_i\rangle \quad (3)$$

where x_i is the i^{th} feature and $|n_i\rangle$ is the i^{th} vector of the computational basis. The definition requires the x vector to be normalised,

$$\sum_i |x_i|^2 = 1. \quad (4)$$

If the number of features to encode is not a power of 2, the remaining amplitudes can be padded with constant values. This model embeds the 16 (4) variables of the *complete dataset* (*muon dataset*) into the amplitudes of a 4-qubit (2-qubit) quantum state. The variational stage consists of a variable number L of *strongly entangling layers*. A strongly entangling layer consists of trainable generic rotational gates $R(\alpha_i, \beta_i, \gamma_i)$ applied to each qubit followed by a collection of CNOT gates applied to neighbouring pairs of qubits, considering the last one as a neighbour of the first one. The complexity of this kind of circuit can be tuned by changing the number of strongly entangling layers L : for a generic n -qubit circuit, the number of trainable parameters of the model N_{par} is equal to

$$N_{\text{par}} = 3 \times n \times L. \quad (5)$$

On the final state, the expectation of the Pauli operator of the first qubit $\langle \sigma_z^0 \rangle \in [-1, +1]$ is measured and used to define the probabilities P_b and $P_{\bar{b}}$ of being a b -jet and a \bar{b} -jet, respectively:

$$P_b = \frac{1}{2}(\langle \sigma_z^0 \rangle + 1) \quad (6)$$

$$P_{\bar{b}} = \frac{1}{2}(1 - \langle \sigma_z^0 \rangle) = 1 - P_b. \quad (7)$$

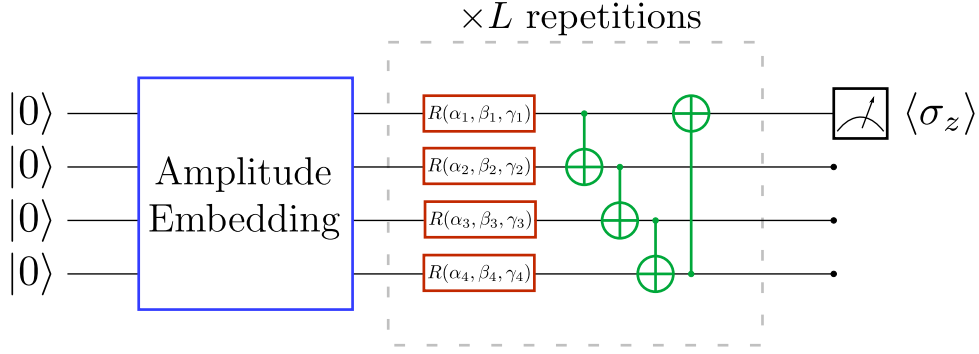


Figure 2: Circuit representation of the Amplitude Embedding model. In blue, variables are embedded into the amplitudes of a quantum state. In red, trainable generic rotational gates to be optimised during the training phase. In green, CNOT gates entangling qubits with a circular topology.

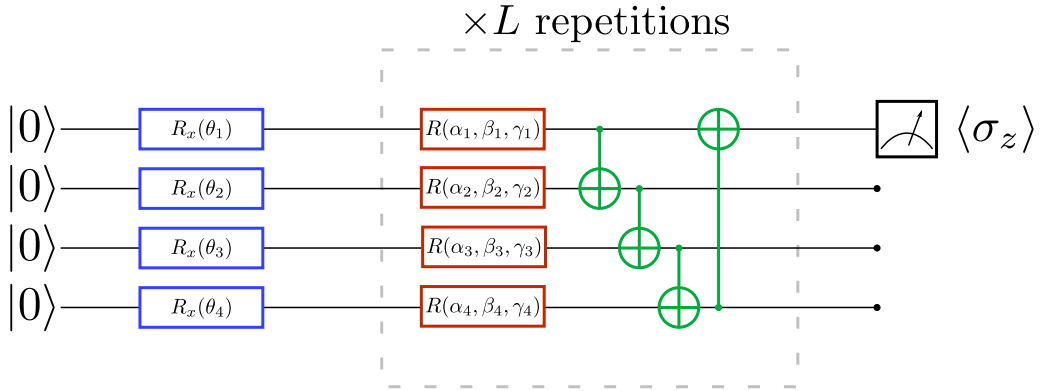


Figure 3: Circuit representation of the Angle Embedding model. In blue, x -axis rotational gates used to embed the variables into the quantum circuit. In red, trainable generic rotational gates to be optimised during the training phase. In green, CNOT gates entangling qubits with a circular topology.

3.2 Angle Embedding

The structure of the Angle Embedding model, represented in Fig. 3, differs from the Amplitude Embedding model in the encoding used to embed the features of the datasets into the quantum state: in this case, the embedding circuit consists in a *Angle Encoder* that embeds 16 (4) features of the *complete dataset* (*muon dataset*) as rotation angles of 16 (4) x -axis rotational gates $R_x(\theta_i)$. Therefore, this circuit structure requires a one-to-one correspondence between qubits and input features: that makes it impractical to adopt with high-dimensionality datasets, due to computational constraints of quantum simulators. The variational stage of the circuit is identical to the Amplitude Embedding model, featuring a variable number L of strongly entangling layers that can be opportunely chosen to tune the number of parameters N_{par} , defined in Eq. 5, and, therefore, the complexity. The measurement of the expectation value of the Pauli σ_z operator is mapped to the tagging probabilities P_b and $P_{\bar{b}}$ as expressed in Eq. 6 and Eq. 7, identically to the Amplitude Embedding model.

4 b-jets identification procedure

Quantum circuits are simulated by means of noiseless simulators (noise impact is studied in Sec. 5.3) using PennyLane [44], a Python framework designed specifically for QML applications. The quantum circuit is embedded into a classical optimisation algorithm, using the Jax [45] Python library. Since the quantum algorithms results are compared to classical DNN ones, the same analysis is performed with a standard feed-forward deep neural network, implemented using the Keras [46] framework with the TensorFlow [47] back-end. Additional details on the structure and the optimisation of the DNN are reported in App. A.

4.1 Training and testing phases

The *muon* and *complete* datasets are both split into training and testing sub-datasets: about 60% of the samples are used in the training process that includes also the validation and the remaining 40% are used to test, evaluate and compare the classifiers. In the *muon dataset* analysis, 60000 jets are used for training and 40000 jets are used for testing. The *complete dataset* training is performed on 400000 jets and remaining 290000 are used for testing and assessing performance. In the analysis of the *muon dataset* (*complete dataset*), the Angle Embedding and Amplitude Embedding classifiers are studied and compared to a DNN with the same 4 (16) input variables. The training process aims to find the values of the model parameters θ that minimise the Mean Squared Error loss function

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N (P_b^i(\theta) - T^i)^2, \quad (8)$$

where N is the number of training jets, P_b^i is the predicted probability, defined in Eq. 6, for the i -th jet, and T^i is the target probability for the i -th jet, *i.e.*, 1 for a b jet and 0 for a \bar{b} jet. Due to the large number of jets in the datasets, the quantum models are trained implementing a mini-batch gradient descent [48] algorithm using the ADAM optimiser [49] to minimise Eq. 8. The training dataset is split in several mini-batches containing a fixed number of training samples. During each training step, the gradient of Eq. 8 is evaluated, averaging over the training samples of a mini-batch, and used to update model parameters. A training epoch is completed when the whole training dataset is processed, namely after a number of steps equal to the number of mini-batches. Unless specified otherwise, the models are trained with learning rate¹ $\xi = 0.01$ for 100 epochs, while the mini-batch size is fixed to the maximum value allowed by memory constraints. The output of the classifier gives the probability that a jet is generated by a b or a \bar{b} quark. The label with the highest probability is assigned to the jet, *i.e.* if $P_b > 0.5$ ($P_b < 0.5$) then it is classified as a b jet (\bar{b} jet). In Fig. 4a the output distributions for the two classes (b and \bar{b} jets) after the training procedure are shown; a separation between the two distributions around 0.5 is visible leading to a good classification. It should be noted that the P_b distribution is shifted toward 1 for b quarks, and toward 0 for \bar{b} quarks, as expected. Fig. 4b shows the output distribution for the Angle Embedding classifier on 16 qubits: in green the jets that

¹In Machine Learning, the learning rate ξ of an optimisation algorithm is usually defined as the scaling factor applied to the gradient of the loss function when updating the parameters. It can be tuned as a parameter of the learning process.

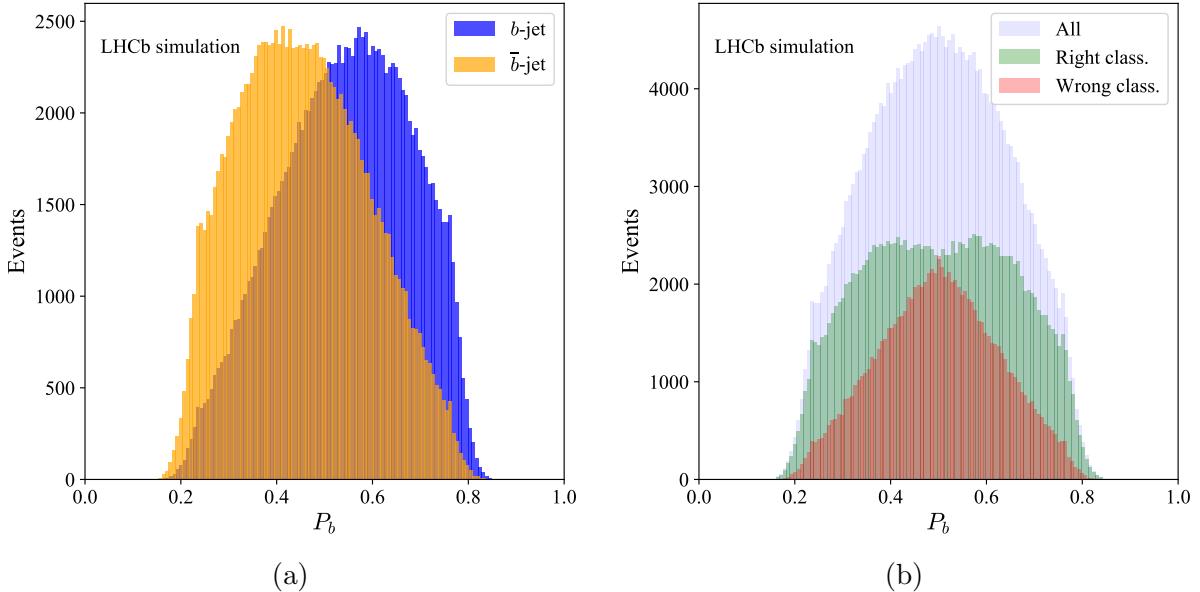


Figure 4: Probability distributions for jet tagged to b (blue) and \bar{b} quarks (yellow), showing separation around 0.5 (a). Probability distribution for the Angle Embedding circuit: jet correctly (wrongly) tagged are plotted in green (red), showing around 0.5 worse classification. The probability distribution for all jets is shown in grey (b).

are correctly classified, in red the jets that are wrongly classified and the sum of all jets in grey. As expected, correctly classified jets tend to stay close to 0 and 1 while the wrongly classified jets are peaked around 0.5, where the prediction power is minimum. Figure 5 shows the Receiver Operating Characteristic curve (ROC) and the Area Under Curve (AUC) for the DNN and the quantum classifiers for the *muon dataset* and the *complete dataset*.

5 Results on b -jet charge tagging

The performance of the classifiers is evaluated by using the jet tagging power, defined in Eq. 1. The tagging power is computed as a function of the jet p_T and η for both the quantum and the classical classifiers. In order to optimise the tagging power, a region symmetric with respect to 0.5 is defined, where no classification is performed. The width Δ_{cut} of the excluded region is defined for each classifier by maximising the tagging power evaluated using all the jets in the dataset. Such an exclusion region reduces the tagging efficiency because less jets are tagged, but enhances the identification probability. The probability distributions and the excluded region are shown in Fig. 6: indeed the region where the prediction power is minimum is excluded. The width Δ_{cut} of the excluded region for each classifier and for *muon* and *complete dataset* are summarised in Tab. 2. For comparison, the unoptimised tagging power, obtained by identifying as b (\bar{b}) the jets with P_b above (below) 0.5, is presented in App. B.

The tagging power ϵ_{tag} as a function of jet p_T and η for the classical and quantum classifiers applied to the *muon dataset* is shown in Fig. 7. All the distributions have similar behaviour demonstrating that no bias is created by any algorithm. The tagging

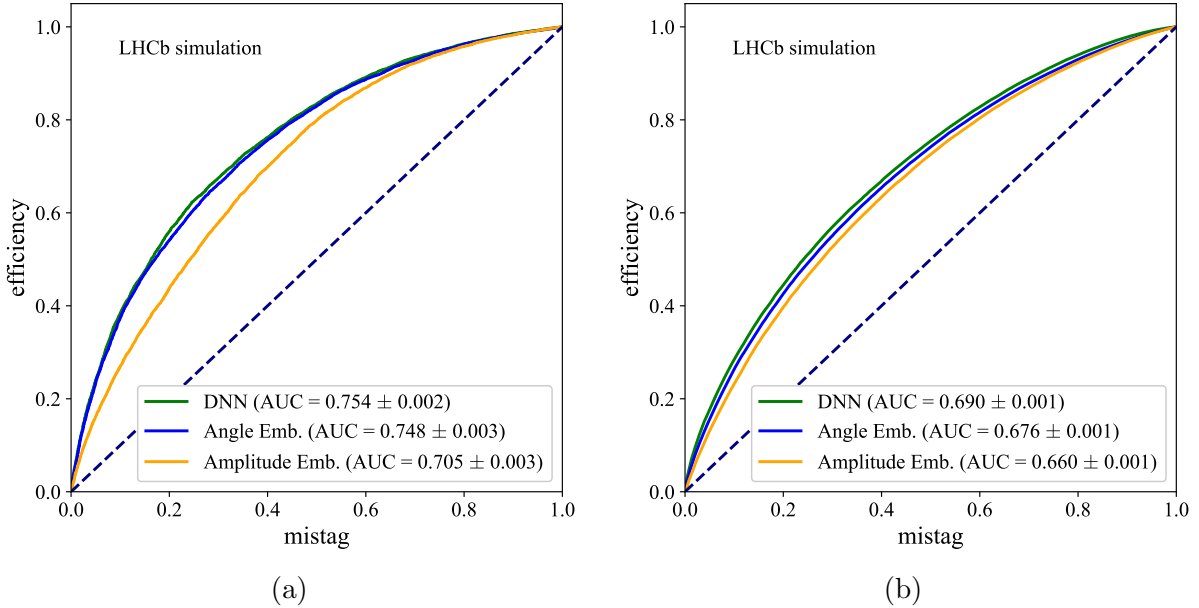


Figure 5: ROC distributions and AUC score for DNN (green), Angle Embedding (blue) and Amplitude Embedding circuits (yellow) for the *muon dataset* (a) and the *complete dataset* (b). The dashed line represents a random classifier.

Table 2: Width Δ_{cut} for different classifiers and dataset.

Dataset	Classifier		
	DNN	Angle Embedding	Amplitude Embedding
Muon	0.30	0.25	0.16
Complete	0.21	0.19	0.12

power dependence on the jet p_T is as expected, since at high p_T the reconstruction and identification of the jet particle content is more difficult, leading to a lower tagging power [50]. The DNN shows slightly better performance compared to the Angle Embedding algorithm, even though the two results are compatible within 2σ in each p_T and η bin. Both algorithms outperform the Amplitude Embedding model and the muon tagging approach. The muon tagging performance is expected, since it only uses the muon charge q for the prediction. The other algorithms use also the muon p_T , the muon ΔR and the weighted jet charge Q . The simple 2-qubit Amplitude Embedding model shows a slightly better performance with respect to the muon tagging, but still worse than the DNN and the Angle Embedding model.

The tagging power ϵ_{tag} for the DNN and the quantum classifiers on the *complete dataset* as function of jet p_T and η , is shown in Fig. 8. Also in this case, the usual dependence on the jet p_T is visible. As expected, the tagging power of the QML and DNN is higher in the *complete dataset* with respect to the *muon dataset*, since a larger number of features is used. As before, for QML and DNN the performance is above the muon tagging approach, given that these classifiers use the information coming from the jet substructure. Differently from the application to the *muon dataset*, in the *complete dataset* the QML algorithms perform slightly worse than the DNN, with slightly better

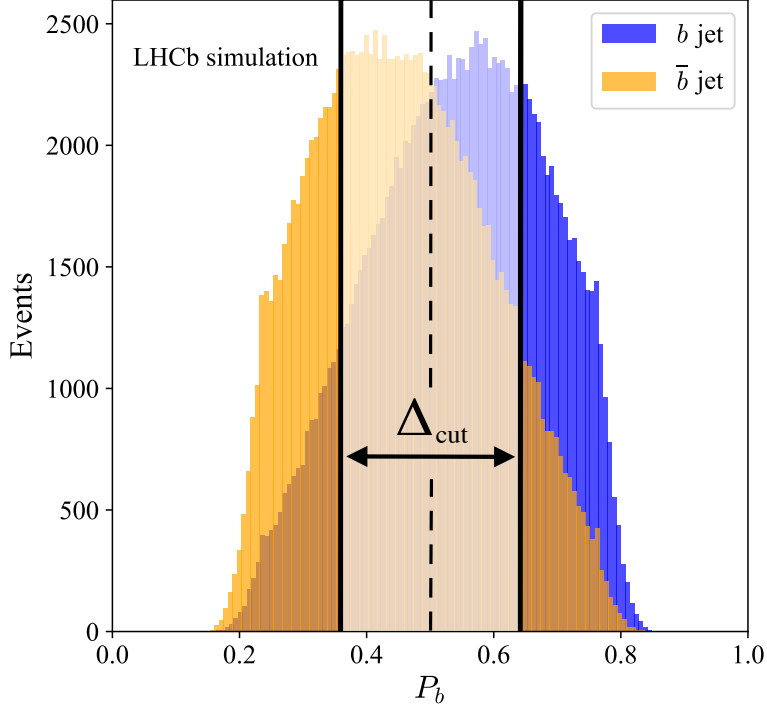


Figure 6: Probability distributions for jet tagged to (blue) b and (yellow) \bar{b} quarks

performance for the Angle Embedding structure than the Amplitude Embedding. It can be deduced that the DNN makes a better use of the features when a larger number of them is used.

5.1 Dependence of the results on number of training events and circuit depth

The dependence of the quantum algorithms performance on the number of training events and the circuit complexity has to be evaluated if near-term applications on quantum hardware are considered. These parameters have an impact on the execution times and therefore on the possibility to use it. The performance dependence on the number of training samples is an interesting parameter to compare QML and DNN methods, in order to assess the differences between the two approaches. Given the high computational efforts of simulating complex circuits with several qubits, only the *muon dataset* is used.

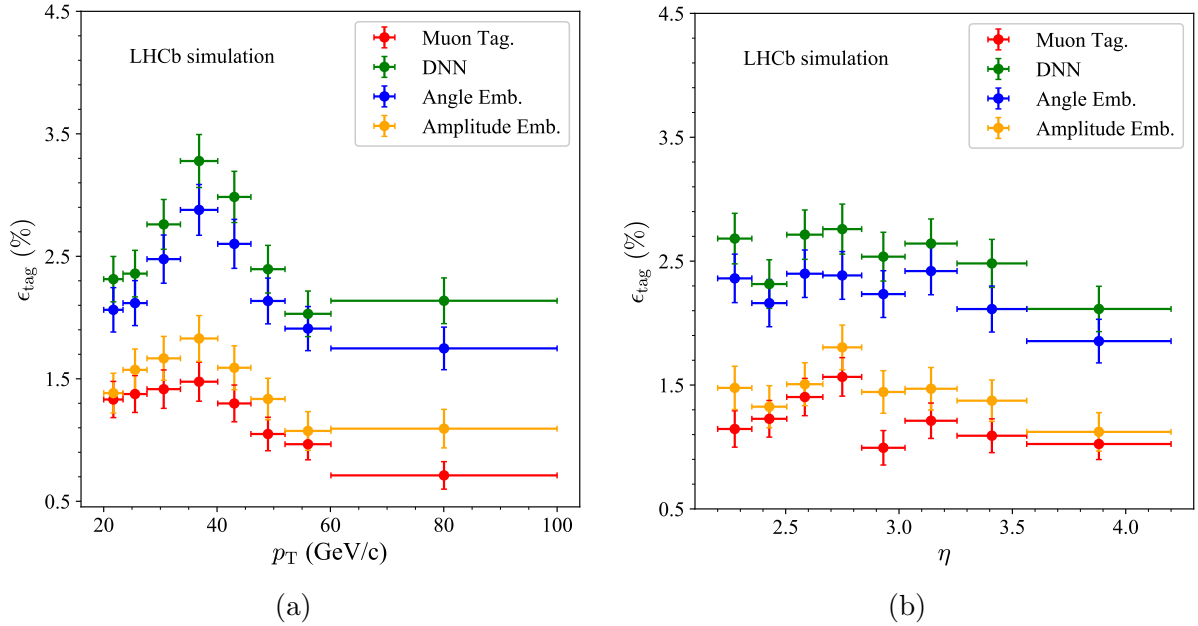


Figure 7: Tagging power ϵ_{tag} with respect to (a) jet p_T and (b) jet η for the *muon dataset*. The Angle Embedding circuit and the DNN show similar performance.

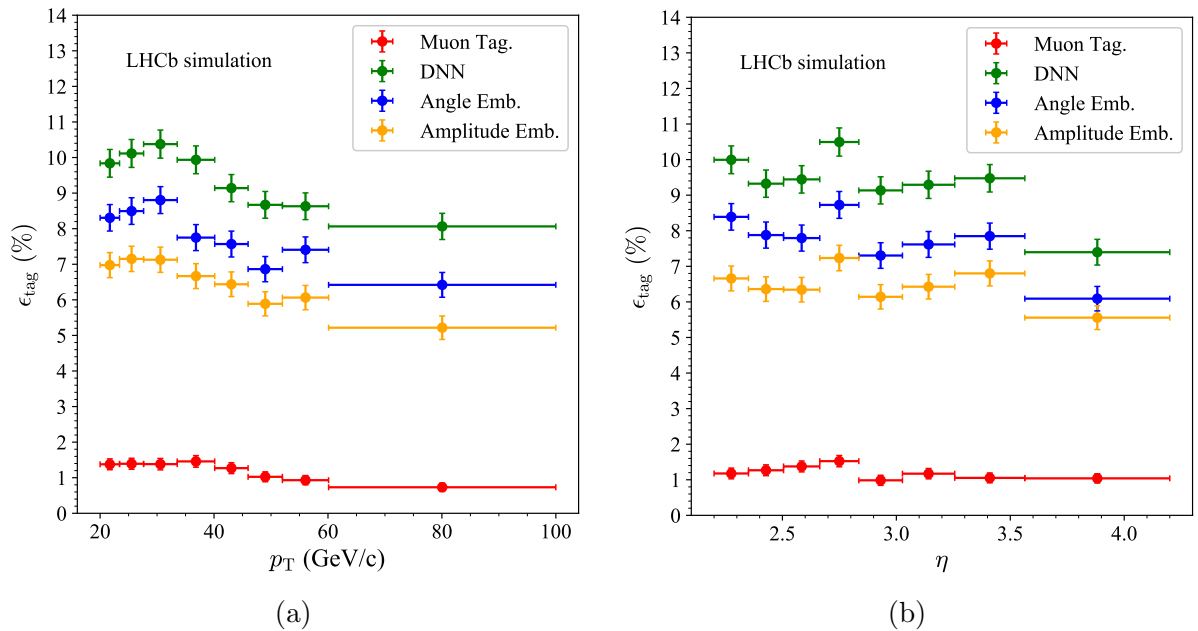


Figure 8: Tagging power ϵ_{tag} with respect to (a) jet p_T and (b) jet η for the *complete dataset*. The quantum algorithms perform slightly worse than the DNN, with the Angle Embedding circuit performing better than the Amplitude Embedding circuit.

For QML, the Angle Embedding structure is considered with different number of strongly entangled layers and different number of training events. The results are compared with the same DNN considered in the previous section; a comparison with more complex networks is described in App. A. The metric used to quantify the goodness of the quantum

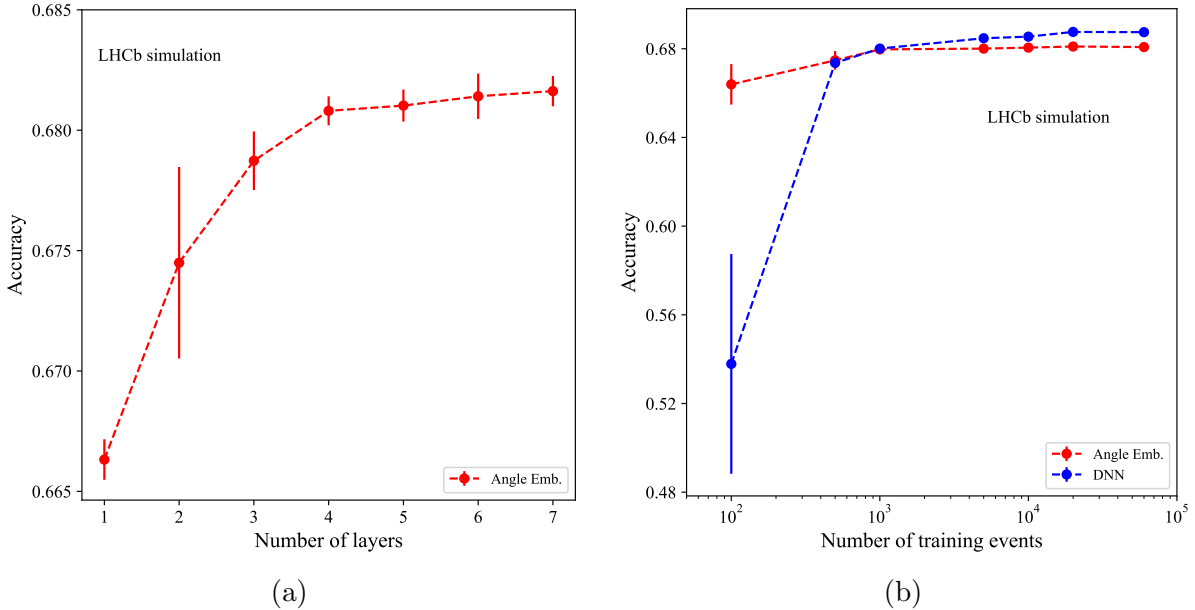


Figure 9: (a) Accuracy of the Angle Embedding structure on the *muon dataset* versus the number of layers. (b) Accuracy of the (red) Angle Embedding structure and (blue) DNN on the *muon dataset* versus the number of training events.

classifier is the accuracy on a test subset of 40000 jets. The performance is calculated averaging over 10 training rounds. In Fig. 9 (a) the accuracy of the Angle Embedding circuit is shown as a function of the number of layers of the circuit. As expected, by increasing the depth of the circuit, and therefore its complexity, the accuracy increases. This behaviour stops at around 5 layers, where the accuracy is saturating and no further improvement is evident. It is clear that, for a given number of features and training data, the Angle Embedding model does not profit of an arbitrarily large number of layers, therefore it is possible to keep a low number of layers, and subsequently a lower complexity of the circuit, to obtain the best performance. This would reduce also the computing time and resources needed for the simulation.

The accuracy as a function of the number of training events for the Angle Embedding circuit and the DNN is shown in Fig. 9 (b). Increasing the number of training events the performance of the quantum algorithm is similar to the DNN, but when the number of training events decreases the quantum algorithm keeps very high performance, while the DNN is not able to perform a good classification. This means that, with respect to the DNN, the QML method reaches optimal performance with a lower number of events. Considering the fact that state-of-the-art ML algorithms require very large data sets to get meaningful performance, this unique feature of QML algorithms needs further investigation, which could lead to a better understanding on how these algorithms are using the input features.

5.2 Time performance

Time performance is a fundamental figure of merit to understand the feasibility of simulating such quantum algorithms. Here, the time performance for the Angle Embedding

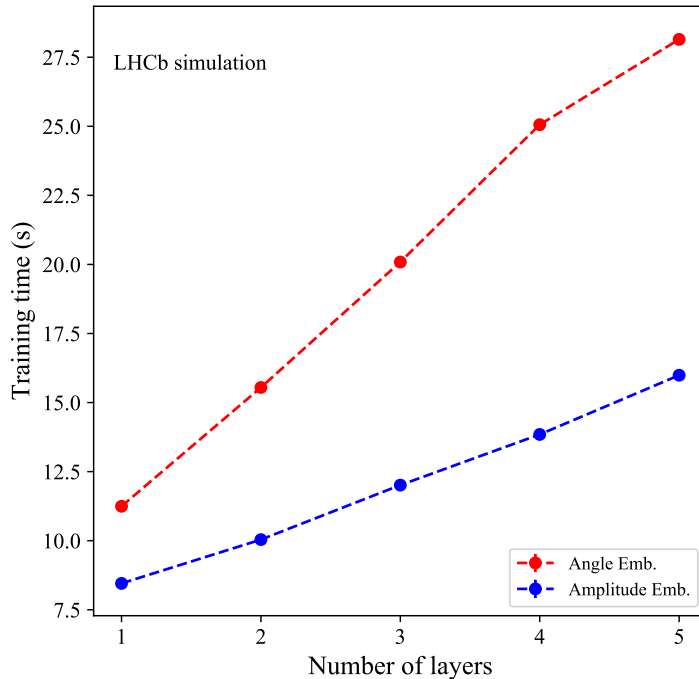


Figure 10: Training time for 100 epochs for the Angle Embedding (red) and Amplitude Embedding (blue) circuits on the *muon dataset* with respect to the number of circuit layers.

and the Amplitude Embedding circuits is evaluated for the *muon dataset*, as a function of the number of strongly entangling layers in the circuit structure. The quantum algorithms are trained using 4 NVIDIA Volta V100 GPUs, and the training time to train 60000 jets for 100 epochs is evaluated. Results are shown in Fig. 10. Results show that it takes less time to train the Amplitude Embedding circuit than the Angle Embedding structure, and the training time increases as the number of layers, although with a greater rate for the Angle Embedding circuit: this may be expected since the complexity of the circuit increases with the number of layers, therefore it takes more time to simulate the quantum circuit.

5.3 Noise models

In order to assess the performance of the algorithms in quantum hardware it is important to understand the impact of noise on quantum circuits. Two kinds of noise can affect quantum algorithms:

- coherent noise: it originates from unitary errors in the application of quantum gates. This leads to the construction of a different quantum state with respect to the desired one. A typical source of this kind of noise is non-ideal calibrations of the quantum hardware;
- incoherent noise: this noise results from the interaction between the quantum hardware and the environment. This noise gives quantum states that are not pure anymore and are described by mixed states, *i.e.* probability distributions over different states.

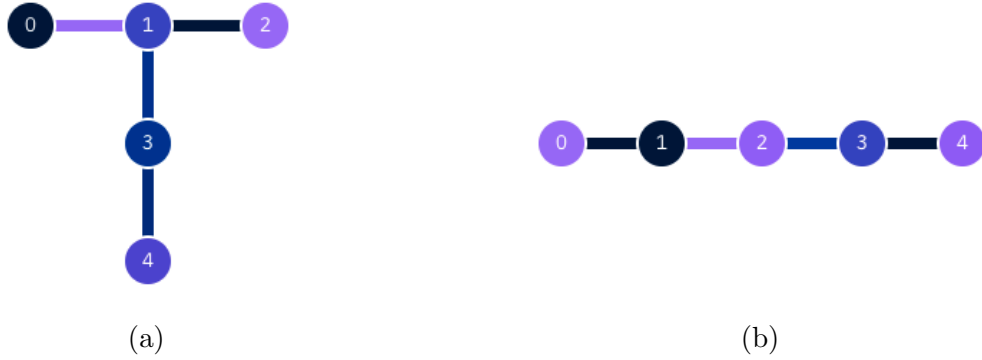


Figure 11: Qubit structure of `ibmq-belem` (a) and `ibmq-santiago` (b).

The simulations of noise contribution taking into account both sources of noise in quantum circuit measurements have been performed using the *pennylane-qiskit* plugin [44, 51]. This plugin allows to simulate noise models coming from different real IBM quantum computers [52], including state preparation and readout errors, and to keep the PennyLane syntax. The result is a simulation of a quantum algorithm on a real device structure. Four IBM quantum computers are considered: `ibmq-belem`, `ibmq-santiago`, `ibmq-jakarta` and `ibmq-toronto`, which have different numbers of qubits (respectively 5, 5, 7 and 27 qubits), different quantum volumes² (respectively 16, 32, 16 and 32) and different qubits structure, as shown in Fig. 11 for the `ibmq-santiago` and `ibmq-belem` which have the same number of qubits.

Studies are performed on the Angle Embedding circuit structure with three strongly entangled layers. A small subset of the *muon dataset* is used because simulating circuits including noise contribution is more time and computationally consuming; on the other hand, with a low number of events the quantum algorithm performance is sufficiently high, as shown in Fig.9. In this way, a subset of 1000 jets of the *muon dataset* is selected for training while validation is performed on a subset of 10000 jets. For each noise model the training is performed for 50 epochs using ADAM with a learning rate $\xi = 0.01$ and batch size of 10 jets. The results are averaged over five rounds of training, using five independent training subsets. The relevant figure of merit to assess noise models performance is the accuracy on the validation test. The results are shown in Fig. 12 and summarised in Tab. 3. Models including noise need more epochs to reach convergence, but in the end the results are consistent with those of noiseless simulations within error. Such a result demonstrates that the proposed circuit model for the *muon dataset* is robust to noise.

6 Conclusions

The first application of QML algorithms to identify the charge of the b hadron produced in the jet hadronization, at the LHCb experiment has been presented. The results using the *muon dataset* show that the Angle Embedding structure almost reaches the performance of the DNN while being better than the muon tagging approach. The amplitude encoding

²The quantum volume is the maximum size of a quantum circuit that can be effectively implemented on a noisy intermediate-scale quantum device. In this paper, the definition from Ref. [53] is adopted.

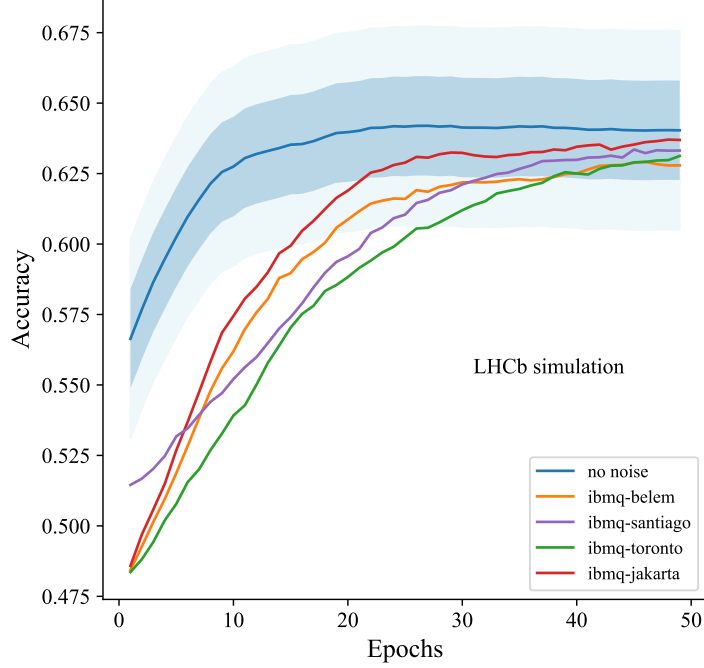


Figure 12: Validation accuracy for noise models as a function of the number of epochs. Blue (light-blue) band represent 1σ (2σ) uncertainty bounds for the noiseless model.

Table 3: Accuracy for noisy circuits obtained with the *muon dataset*.

Noise model	accuracy
no noise	0.640 ± 0.017
ibmq-belem	0.629 ± 0.047
ibmq-santiago	0.633 ± 0.038
ibmq-jakarta	0.637 ± 0.042
ibmq-toronto	0.631 ± 0.044

structure presents no evident improvement with respect to the muon tagging method. When the *complete dataset* is used, the best quantum algorithm performance is given by Angle Embedding structure. The study of the performance dependence on the circuit depth has shown that the number of layers is a parameter to be optimised. More layers, which means more complex structures, do not necessarily result in improved performance, since saturation is reached. The impact of the noise on the results appears to be negligible, suggesting that the proposed circuits could be implemented on a quantum hardware if available. QML algorithms achieve performance consistent with classical methods like the DNN with low-complexity circuits and a smaller number of training events. This could have important implications for LHC experiments where often the training phase is the most expensive in terms of resources. However, when a large number of features is employed, the DNN performs better than QML algorithms. Here huge improvements are expected when the hardware will be available. In fact, the comparison of QML models to classical kernel methods [54] shows that QML models achieve classification tasks separating data in Hilbert spaces whose dimension scales exponentially with the

number of qubits involved. As this number increases, quantum simulations on classical computers become unfeasible.

The full exploitation of QML in high energy physics experiments at colliders, just began. As for any brand new tool, unexpected applications may manifest themselves while using. For example, results obtained involving tensor network methods [6, 55] have shown that quantum algorithms can allow to study correlations among the features. This is done by measuring the entanglement correlations between qubits of an optimised multi-qubit system. Given that a quantum circuit is a unitary matrix, the same approach applied to tensor networks can be applied in QML by easily inverting the unitary matrix representing the circuit. This new and yet not investigated approach could allow to extract information on jet constituents correlations starting from the measurement of the entanglement between the qubits of a trained VQCs and possibly help in improving the classification performance.

Acknowledgements

The authors would like to thank the LHCb Data Processing & Analysis (DPA) project colleagues for supporting this publication and reviewing the work. The authors acknowledge support from INFN (Italy) and from the Department of Physics and Astronomy of the University of Padova. The authors would also like to thank CERN Openlab for useful and fruitful discussions. The authors thank the CINECA Supercomputing Center for awarding access to Marconi100 cluster based at CINECA (Bologna, Italy).

A Deep Neural Network model and further comparisons

The DNN is built following a classical feed-forward structure shown in Fig. 13: it starts with a Batch Normalisation Layer [56] and it applies several Dense layers, each one followed by a Dropout [57] layer. Depending on the number of input features the input vector for the DNN changes (4 variables for the muon dataset and 16 for the complete one). For the hidden layers the ReLu³ activation is used while a sigmoid⁴ function is applied to the output node. The network is trained using the ADAM optimiser [49]. The model is trained for 250 epochs with an early stopping of 25 epochs on the test loss function and a learning rate of 0.0001. The hyperparameters of the DNN (such as depth, number of nodes per layer, dropout rate, normalization moment, and kernel initialization) are optimised using the *hyperopt* package [58], maximising the accuracy in the test dataset by exploring different parameter spaces. Given the classification task, we take cross entropy as loss function. A scheme of the DNN model is shown in Fig. 13. Different DNN structures have also been considered, in order to have a fair comparison between quantum and classical classifiers. Usual state-of-the-art networks, such as CMS DeepJet algorithm [9], make use of LSTM [59] and convolutional layers, resulting in an improvement of performance. RNN structures are also under study within the LHCb experiment as potential flavour tagging algorithms. Therefore the following DNN models are studied:

³ReLu is the rectifier activation function, defined as $\text{ReLu}(x) = \max(0, x)$

⁴The sigmoid activation function is defined as $\text{sig}(x) = [1 + \exp(-x)]^{-1}$

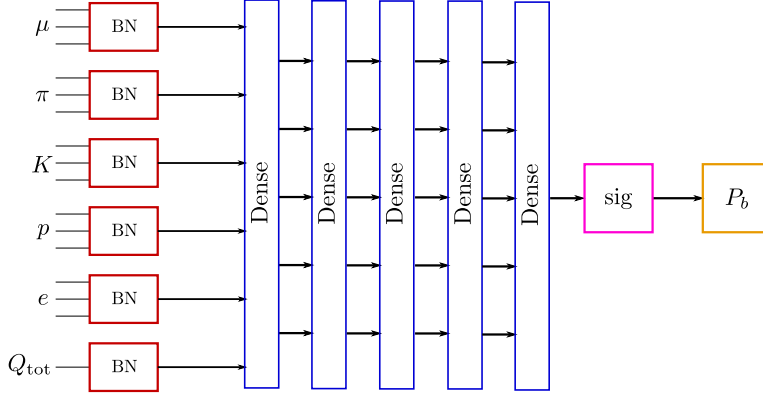


Figure 13: DNN structure.

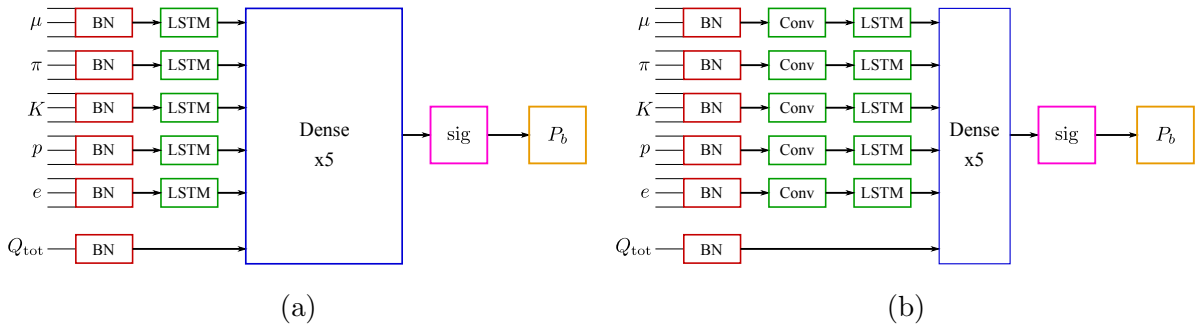


Figure 14: DNN structure for (a) the “LSTM” model and for (b) the “LSTM+CONV” model .

- “LSTM” model: starting from the DNN structure we apply a LSTM layer to the particle features (therefore excluding the ”global” variable of the total jet charge) before the first Dense layer. The Dense structure is identical to the DNN structure.
- “LSTM+Conv” model: from the LSTM model we firstly apply a convolutional layer of the particle features (as before not applied to the total jet charge).

The two models are shown in Fig. 14. The DNN performance in terms of tagging power is compared to the LSTM and LSTM+CONV models. Results for tagging power as function of the jet p_T and η are shown in Fig. 15, where results for different models are comparable within the error and therefore allow us to consider just the DNN model for the comparison with quantum algorithms.

B Unoptimised tagging power distributions

In this section the distributions for the unoptimised tagging power are shown, both for the muon and the complete datasets. This is done in order to check if there are performance biases when cutting on the probability distributions to maximise the tagging power. In Figs. 16 and 17 the unoptimised tagging power distributions as function of jet p_T and η respectively for the muon and the complete datasets are shown. No evident biases are present and the same considerations done for the optimised distributions are valid.

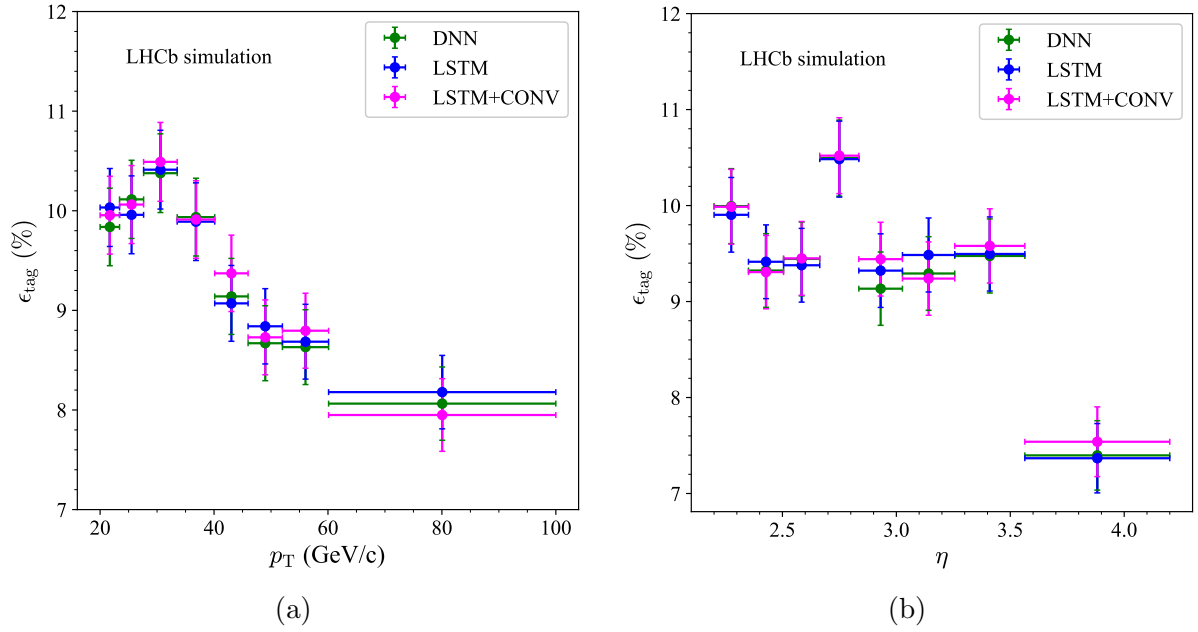


Figure 15: Tagging power ϵ_{tag} with respect to (a) jet p_T and (b) jet η for the complete dataset applied to different DNN models.

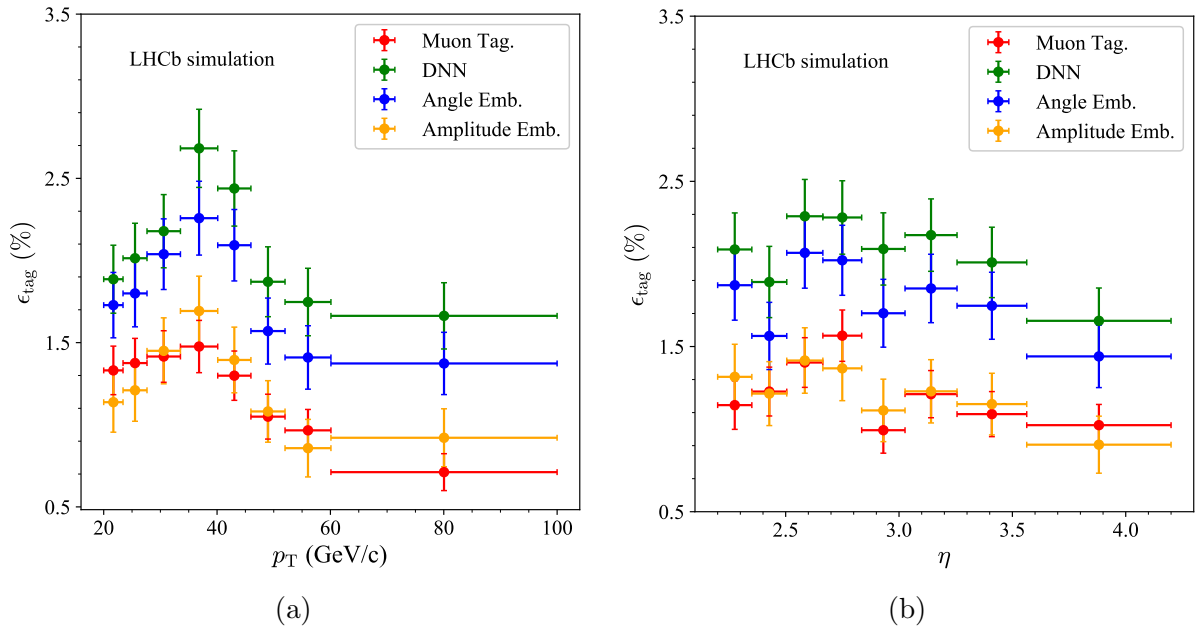


Figure 16: Unoptimised tagging power ϵ_{tag} with respect to (a) jet p_T and (b) jet η for the muon dataset. The angle embedding circuit and the DNN show similar performance.

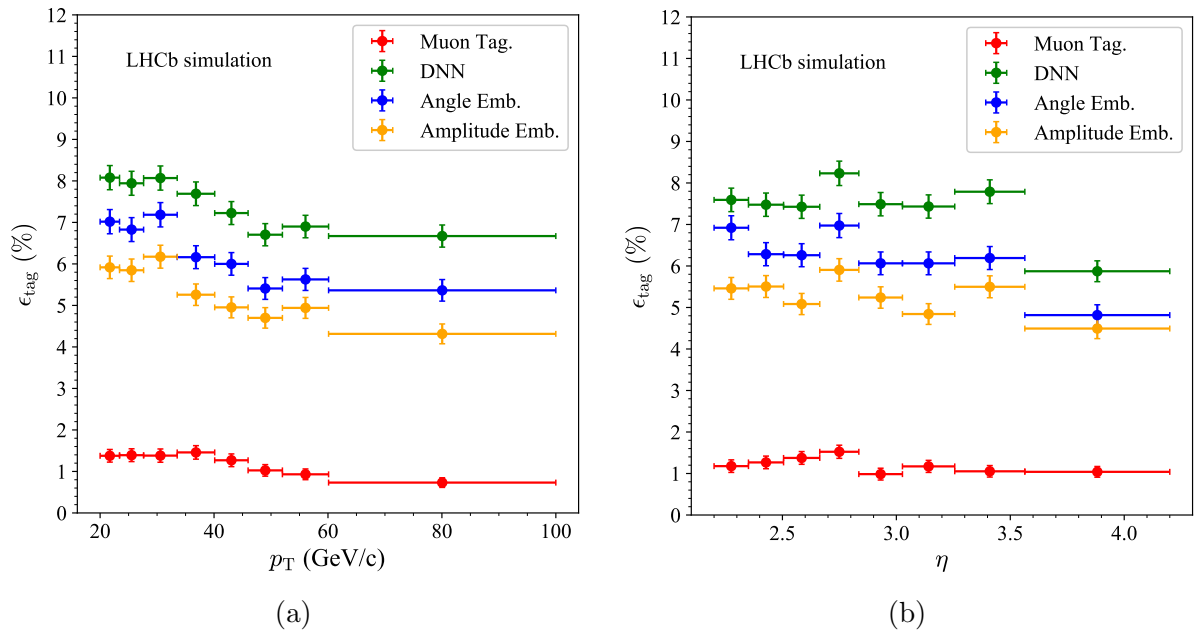


Figure 17: Unoptimised tagging power ϵ_{tag} with respect to (a) jet p_T and (b) jet η for the complete dataset. The angle embedding circuit and the DNN show similar performance.

References

- [1] D. Guest, K. Cranmer, and D. Whiteson, *Deep Learning and its Application to LHC Physics*, Ann. Rev. Nucl. Part. Sci. **68** (2018) 161, [arXiv:1806.11484](#).
- [2] A. J. Larkoski, I. Moult, and B. Nachman, *Jet Substructure at the Large Hadron Collider: A Review of Recent Advances in Theory and Machine Learning*, Phys. Rept. **841** (2020) 1, [arXiv:1709.04464](#).
- [3] P. Baldi *et al.*, *Jet Substructure Classification in High-Energy Physics with Deep Neural Networks*, Phys. Rev. **D93** (2016) 094034, [arXiv:1603.09349](#).
- [4] J. Lin, M. Freytsis, I. Moult, and B. Nachman, *Boosting $H \rightarrow b\bar{b}$ with Machine Learning*, JHEP **10** (2018) 101, [arXiv:1807.10768](#).
- [5] ATLAS Collaboration, *Identification of Jets Containing b -Hadrons with Recurrent Neural Networks at the ATLAS Experiment*, ATL-PHYS-PUB-2017-003, CERN, Geneva, 2017.
- [6] T. Felser *et al.*, *Quantum-inspired machine learning on high-energy physics data*, npj Quantum Information **7** (2021) 111.
- [7] J. Shlomi, P. Battaglia, and J.-R. Vlimant, *Graph neural networks in particle physics*, Machine Learning: Science and Technology **2** (2021) 021001.
- [8] H. Qu and L. Gouskos, *ParticleNet: Jet Tagging via Particle Clouds*, Phys. Rev. **D101** (2020) 056019, [arXiv:1902.08570](#).
- [9] E. Bols *et al.*, *Jet flavour classification using DeepJet*, Journal of Instrumentation **15** (2020) P12012.
- [10] ATLAS collaboration, M. Aaboud *et al.*, *Measurements of b -jet tagging efficiency with the ATLAS detector using $t\bar{t}$ events at $\sqrt{s} = 13$ TeV*, JHEP **08** (2018) 089, [arXiv:1805.01845](#).
- [11] ATLAS collaboration, *A new tagger for the charge identification of b -jets*, ATL-PHYS-PUB-2015-040, 2015.
- [12] J. Biamonte *et al.*, *Quantum machine learning*, Nature **549** (2017) 195–202.
- [13] K. Terashi *et al.*, *Event Classification with Quantum Machine Learning in High-Energy Physics*, Computing and Software for Big Science **5** (2021) .
- [14] S. L. Wu *et al.*, *Application of quantum machine learning using the quantum kernel algorithm on high energy physics analysis at the LHC*, Phys. Rev. Res. **3** (2021) 033221, [arXiv:2104.05059](#).
- [15] A. Blance and M. Spannowsky, *Quantum Machine Learning for Particle Physics using a Variational Quantum Classifier*, doi: 10.1007/JHEP02(2021)212 [arXiv:2010.07335](#).

- [16] J. Y. Araz and M. Spannowsky, *Classical versus Quantum: comparing Tensor Network-based Quantum Circuits on LHC data*, arXiv:2202.10471.
- [17] A. Blance and M. Spannowsky, *Unsupervised event classification with graphs on classical and photonic quantum computers*, JHEP **21** (2020) 170, arXiv:2103.03897.
- [18] C. Tüysüz *et al.*, *Particle track reconstruction with quantum algorithms*, EPJ Web Conf. **245** (2020) 09013.
- [19] L. Funcke *et al.*, *Studying quantum algorithms for particle track reconstruction in the LUXE experiment*, in *20th International Workshop on Advanced Computing and Analysis Techniques in Physics Research: AI Decoded - Towards Sustainable, Diverse, Performant and Effective Scientific Computing*, 2022, arXiv:2202.06874.
- [20] W. Guan *et al.*, *Quantum machine learning in high energy physics*, Machine Learning: Science and Technology **2** (2021) 011003.
- [21] LHCb collaboration, A. A. Alves, Jr. *et al.*, *The LHCb Detector at the LHC*, JINST **3** (2008) S08005.
- [22] LHCb collaboration, R. Aaij *et al.*, *LHCb Detector Performance*, Int. J. Mod. Phys. A **30** (2015) 1530022, arXiv:1412.6352.
- [23] ALEPH collaboration, D. Buskulic *et al.*, *Performance of the ALEPH detector at LEP*, Nucl. Instrum. Meth. **A360** (1995) 481.
- [24] LHCb collaboration, R. Aaij *et al.*, *Study of forward $Z + jet$ production in pp collisions at $\sqrt{s} = 7 TeV$* , JHEP **01** (2014) 033, arXiv:1310.8197.
- [25] M. Cacciari, G. P. Salam, and G. Soyez, *The anti- k_t jet clustering algorithm*, JHEP **04** (2008) 063, arXiv:0802.1189.
- [26] M. Cacciari, G. P. Salam, and G. Soyez, *FastJet User Manual*, Eur. Phys. J. C **72** (2012) 1896, arXiv:1111.6097.
- [27] LHCb collaboration, R. Aaij *et al.*, *Identification of beauty and charm quark jets at LHCb*, JINST **10** (2015) P06013, arXiv:1504.07670.
- [28] LHCb collaboration, R. Aaij *et al.*, *First measurement of the charge asymmetry in beauty-quark pair production*, Phys. Rev. Lett. **113** (2014) 082003, arXiv:1406.4789.
- [29] Particle Data Group, M. Tanabashi *et al.*, *Review of Particle Physics*, Phys. Rev. **D98** (2018) 030001.
- [30] D0 Collaboration, *Measurements of B_d mixing using opposite-side flavor tagging*, Phys. Rev. **D74** (2006) 112002, arXiv:0609034v1.
- [31] CDF Collaboration, *Measurements of B_0 oscillations and calibration of flavor tagging in semileptonic decays*, CDF Note **8235** (2006).
- [32] LHCb collaboration, R. Aaij *et al.*, *Opposite-side flavour tagging of b mesons at the lhcb experiment*, The European Physical Journal C **72** (2012) .

- [33] LHCb collaboration, M. Clemencic *et al.*, *The LHCb simulation application, Gauss: Design, evolution and experience*, J. Phys. Conf. Ser. **331** (2011) 032023.
- [34] T. Sjöstrand, S. Mrenna, and P. Skands, *A brief introduction to PYTHIA 8.1*, Computer Physics Communications **178** (2008) 852.
- [35] I. Belyaev *et al.*, *Handling of the generation of primary events in gauss, the LHCb simulation framework*, Journal of Physics: Conference Series **331** (2011) 032047.
- [36] D. J. Lange, *The EvtGen particle decay simulation package*, Nucl. Instrum. Meth. **A462** (2001) 152.
- [37] S. Agostinelli *et al.*, *Geant4—a simulation toolkit*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **506** (2003) 250.
- [38] D. Krohn, M. D. Schwartz, T. Lin, and W. J. Waalewijn, *Jet charge at the lhc*, Phys. Rev. Lett. **110** (2013) 212001.
- [39] R. D. Field and R. P. Feynman, *A parametrization of the properties of quark jets*, Nuclear Physics B **136** (1978) 1.
- [40] ATLAS collaboration, *Jet Charge Studies with the ATLAS Detector Using $\sqrt{s} = 8$ TeV Proton-Proton Collision Data*, ATLAS-CONF-2013-086, 2013.
- [41] ATLAS collaboration, *Measurement of the Jet Vertex Charge algorithm performance for identified b -jets in $t\bar{t}$ events in pp collisions with the ATLAS detector*, ATLAS-CONF-2018-0220, 2018.
- [42] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, *Parameterized quantum circuits as machine learning models*, Quantum Science and Technology **4** (2019) 043001.
- [43] V. Havlíček *et al.*, *Supervised learning with quantum-enhanced feature spaces*, Nature **567** (2019) 209–212.
- [44] V. Bergholm *et al.*, *Pennylane: Automatic differentiation of hybrid quantum-classical computations*, 2020.
- [45] J. Bradbury *et al.*, *JAX: composable transformations of Python+NumPy programs*, <http://github.com/google/jax>, 2018.
- [46] F. Chollet *et al.*, *Keras*, <https://keras.io>, 2015.
- [47] M. Abadi *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, <https://www.tensorflow.org/>, 2015.
- [48] L. Bottou, F. E. Curtis, and J. Nocedal, *Optimization methods for large-scale machine learning*, SIAM Review **60** (2018) 223, arXiv:<https://doi.org/10.1137/16M1080173>.

- [49] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.
- [50] CDF, D0 collaboration, T. Wright, *B-Tagging at CDF and D0, lessons for LHC*, in *17th Symposium on Hadron Collider Physics 2006 (HCP 2006)*, 2006, arXiv:0707.1712.
- [51] M. S. ANIS *et al.*, *Qiskit: An open-source framework for quantum computing*, 2021. doi: 10.5281/zenodo.2573505.
- [52] IBM Quantum. <https://quantum-computing.ibm.com/>, 2021.
- [53] A. W. Cross *et al.*, *Validating quantum computers using randomized model circuits*, Phys. Rev. A **100** (2019) 032328.
- [54] M. Schuld, *Supervised quantum machine learning models are kernel methods*, arXiv:2101.11020.
- [55] M. L. Wall and G. D’Aguanno, *Tree-tensor-network classifiers for machine learning: From quantum inspired to quantum assisted*, Phys. Rev. A **104** (2021) 042408.
- [56] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), **37**, (Lille, France), 448–456, PMLR, 2015.
- [57] N. Srivastava *et al.*, *Dropout: a simple way to prevent neural networks from overfitting*, The journal of machine learning research **15** (2014) 1929.
- [58] J. Bergstra, D. Yamins, and D. Cox, *Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures*, in *Proceedings of the 30th International Conference on Machine Learning* (S. Dasgupta and D. McAllester, eds.), **28**, (Atlanta, Georgia, USA), 115–123, PMLR, 2013.
- [59] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural Computation **9** (1997) 1735.