Reconstructing charged particle track segments with a quantum-enhanced support vector machine

Philippa Duckett, Gabriel Facini, Marcin Jastrzebski, Sarah Malik, and Tim Scanlon

Department of Physics and Astronomy, University College London, Gower St, London WC1E 6BT, United Kingdom and Center for Data Intensive Science and Industry, University College London, Gower St, London WC1E 6BT, United Kingdom

Sébastien Rettie

Department of Physics and Astronomy, University College London, Gower St, London WC1E 6BT, United Kingdom and European Organization for Nuclear Research (CERN), Geneva 1211, Switzerland

(Received 30 January 2023; accepted 26 January 2024; published 8 March 2024)

Reconstructing the trajectories of charged particles from the collection of hits they leave in the detectors of collider experiments like those at the Large Hadron Collider (LHC) is a challenging combinatorics problem and computationally intensive. The tenfold increase in the delivered luminosity at the upgraded High Luminosity LHC will result in a very densely populated detector environment. The time taken by conventional techniques for reconstructing particle tracks scales worse than quadratically with track density. Accurately and efficiently assigning the collection of hits left in the tracking detector to the correct particle will be a computational bottleneck and has motivated studying possible alternative approaches. This paper presents a quantum-enhanced machine learning algorithm that uses a support vector machine (SVM) with a quantum-estimated kernel to classify a set of three hits (triplets) as either belonging to or not belonging to the same particle track. The performance of the algorithm is then compared to a fully classical SVM. The quantum algorithm shows an improvement in accuracy versus the classical algorithm. Model complexity metrics are used to hint at an explanation for favorable performance of the quantum kernel.

DOI: 10.1103/PhysRevD.109.052002

I. INTRODUCTION

The Large Hadron Collider (LHC) is currently the highest energy particle collider in the world. It accelerates beams of protons to almost the speed of light and then collides them at a centre-of-mass energy of 13.6 TeV at the center of large, multipurpose particle detectors that are designed to reconstruct the outcome of those collisions. Among the key physics objectives of the LHC are precise measurements of the properties of the Higgs boson, shedding light on the elusive particle(s) that may constitute dark matter, and searching for a wide breadth of new physics phenomena beyond the Standard Model (SM) via exotic decay signatures like long-lived particles.

To attain these physics goals, the LHC is preparing for an upgrade that will deliver an order of magnitude more data to the experiments by increasing the intensity of the proton beams, resulting in a higher instantaneous luminosity and thus many more collisions taking place every time the proton bunches cross [1]. At this upgraded High Luminosity LHC (HL-LHC) the number of concurrent, overlapping proton-proton interactions (pileup) is expected to reach up to 200, a significant increase from the average predicted pileup of 60 for the current Run 3. Such a step change in the running conditions of the collider will significantly increase our capabilities to fulfil the goals of the LHC program. However, it also presents challenges. The significant increase in detector occupancy will impact the performance of the entire pipeline, including data acquisition, processing, and analysis, as well as simulating the collisions in the detector. This presents significant overhead on the computational resources, with some elements, such as reconstructing charged particle trajectories, becoming a major bottleneck.

To address these high demands on the computational resources, numerous approaches are being pursued, ranging from the development of more efficient algorithms and the application of state-of-the-art machine learning techniques to the use of graphics processing units (GPUs) [2,3] to execute code that is parallelizable. One of the intriguing new avenues being pursued to tackle these challenges is quantum computing. This new paradigm offers a fundamentally new form of computing by leveraging the phenomena of quantum mechanics and opens the prospect of significantly speeding up our current algorithms and performing calculations that could only be done to some approximation with classical computers.

Particle physics has seen a surge of interest in ascertaining how quantum computers may impact the future of the field and establishing the scenarios in which they may be most advantageous. The current noisy intermediate scale quantum (NISQ) devices [4], while a stepping stone on the way to universal, fault-tolerant quantum computers, have enabled many of these proof-of-principle studies to be performed. This exploratory phase of applying current NISQ era quantum computers to challenging problems in particle physics will pave the way for the emergence of new ideas and techniques needed to fully exploit quantum computation and identify the specific problems for which they are most suitable.

Quantum computing algorithms have been studied for a range of different scenarios in high energy physics. The calculation of simple scattering processes via the helicity spinor formalism and the simulation of a parton shower was demonstrated in [5]. A quantum walk framework was proposed in [6], demonstrating that the parton shower is more naturally and efficiently simulated using a quantum walk in two dimensions. Quantum computing has also been applied to jet clustering [7–9], classification of collisions of interest from those that are not [10–12], and anomaly detection in searches for new physics [13].

The challenging task of connecting the hits left by charged particles in the tracking detector and associating them with the same particle has been studied from several different perspectives including: quantum associative memory to store all the different track patterns and subsequently employ Grover's search algorithm to search through the database and recall the right track pattern [14]; quantum graph neural networks [15,16]; and quantum annealing devices to minimise an objective function [17–19].

This paper approaches the problem of track reconstruction by proposing a hybrid quantum-classical algorithm that uses a support vector machine with a quantum-estimated kernel. The problem is decomposed into that of classifying short segments of tracks. Often such segments can form the "seeds" for extrapolating to the full trajectory of the track. This "seeding" step is expected to be a large consumer of CPU time at the HL-LHC [2]. Simplifications are implemented to fit the limitations of the presently available quantum simulators.

II. DATA PREPROCESSING

This study utilises the TrackML dataset [20,21] which has been widely used for proof-of-principle studies of classical machine learning algorithms and quantum-based approaches. The dataset provides a simplified simulation of the detector geometry and conditions expected at the HL-LHC. It features a silicon tracking detector with 10 cylindrical layers in the central region and disk geometry in the forward regions, which is typically representative of the ATLAS [22] and CMS detectors [23]. The detector is segmented into three subdetectors differing in their spatial resolution, with the inner pixel detector comprising of 4 layers, followed by a short strip detector of 4 layers and then a 2-layer long strip detector. These tracking detectors are immersed in a strong magnetic field aligned with the direction of the proton-proton beam, so charged particles moving through these detectors will typically follow an approximately helical trajectory and show curved trajectories in the transverse *x*–*y* plane which is perpendicular to the beam line. Figure 1 shows the layout of this virtual detector used to produce the TrackML dataset and the coverage of each subdetector in the *r*–*z* plane, where *r* is the radial dimension and measures the distance from the beam line and *z* is the distance along the beam line. For the analysis in this paper, only the hits in the barrel region of the detector are used to reduce the total number of hits to a level that can be processed within the current computational constraints.

The TrackML dataset contains 10,000 simulated events. The process of interest is top-antitop production and overlaid on this "signal" are 200 additional proton-proton collisions to simulate the conditions expected at the HL-LHC. This results in an average of 100,000 hits per event in the tracking detector which must be associated with approximately 10,000 tracks.

The 3-dimensional spatial information for every hit in the detector is provided and this information is used to build

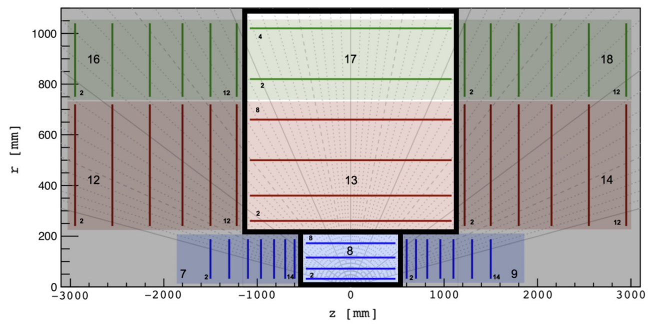
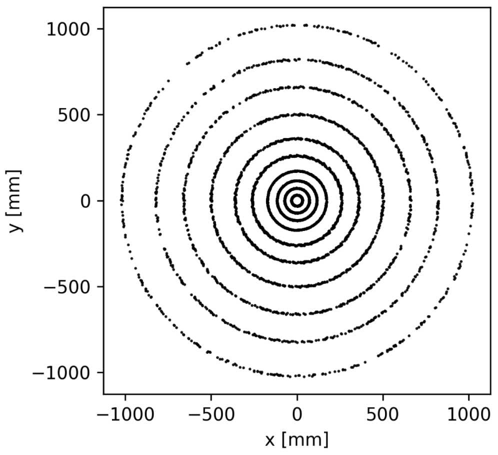


FIG. 1. A schematic of the virtual general-purpose detector simulated in the TrackML challenge and the coverage of each subdetector in the r-z plane. Highlighted is the barrel region used in the analysis. The numbers indicate the various detector components and layers respectively. Original image is taken from [20].



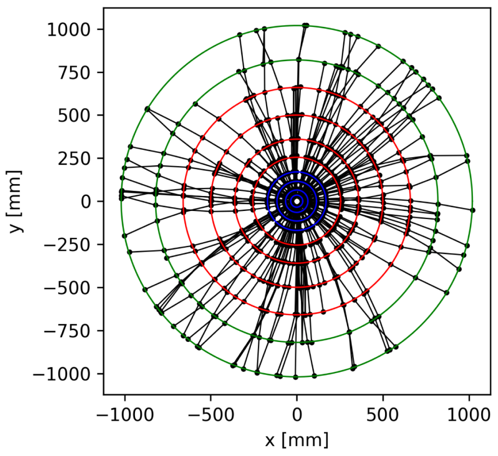


FIG. 2. The 6518 hits in an example event in the x–y detector plane (top) and some of the true tracks reconstructed from those hits (bottom). The hits come from 879 particles which produced triplets in the barrel region. The 10 layers of the detector for the barrel region are also shown. The blue, red, and green layers correspond to the pixel, short strip and long strip detectors, respectively.

the track candidates. The total number of possible combinations of those hits that can lead to a track is very large. Identifying the correct combination of hits that reconstruct the true trajectory of a particle is thus a challenging combinatorics problem. Figure 2 illustrates this by showing all hits for an event in the x–y plane of the detector and a fraction of true reconstructed tracks formed from a combination of those hits. To avoid unphysical hit sequences which would dominate the resulting dataset, selection criteria are applied to reduce the number of possible connections between hits in each event such that they can be processed without overburdening the computational resources. In addition, the problem is formulated as a classification task, with track segments consisting of a set of three hits in adjacent layers of the detector being classified as belonging to a single particle track or not.

TABLE I. The selection criteria applied to select doublets, using the z_0 intercept from the extrapolation of the doublet to the z axis and the ratio of the difference in ϕ and r between each hit forming the doublet.

Variable	Selection
$\frac{\Delta\phi}{\Delta r}$	$\leq 6 \times 10^4$ $[\frac{\text{rad}}{\text{mm}}]$
$\lvert z_0 \rvert$	≤ 100 [mm]

Such objects often form seeds from which the rest of the track reconstruction starts.

The hits are described by three coordinates; r, ϕ, and z, where ϕ is defined as the angle around the z axis. A total of 310 events have been processed for classification. The first step in constructing the triplets is to make a dataset of doublets, which are defined as two consecutive hits in the detector. Selection criteria are applied to reduce the size of the doublet dataset and improve its quality. The following observables are used in the selection; the intercept from the extrapolation of the doublet to the z axis, z_0, and the ratio $\frac{\Delta\phi}{\Delta r}$, as calculated from the difference in ϕ and r between each hit forming the doublet. This selection is summarized in Table I, and was originally implemented in [24].

Triplets are constructed by connecting one doublet to the end of another, with the two doublets sharing a hit such that the outer hit of the first (inner) doublet is the same as the inner hit of the second.

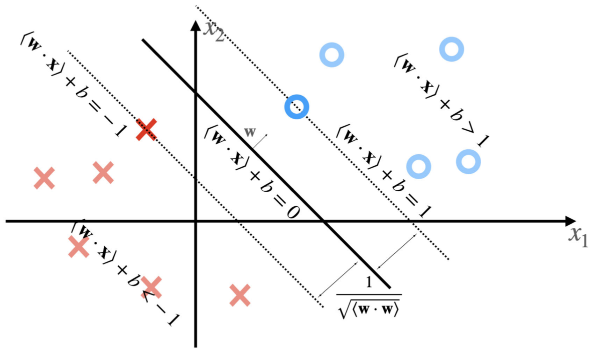
The selection of triplets is based on the estimation of the transverse momentum (p_T) as determined from the three hits, the θ-breaking angle and the ϕ-breaking angle. The angle θ is defined in the r-z plane and a breaking angle is that between the straight lines (connecting the two hits in a doublet) of two doublets that form a triplet. The triplet selection is summarized in Table II.

III. SUPPORT VECTOR MACHINE

The proposed algorithm utilises a support vector machine (SVM) [25], where a kernel function is calculated either on a (simulated) quantum or a classical computer. A support vector machine is a supervised machine learning algorithm that classifies data by drawing linear decision boundaries (hyperplanes) between different groups of data. This paper focuses on discriminating between two classes

TABLE II. The selection criteria applied to select triplets based on the estimated p_T and the θ and ϕ angles between two doublets that form a triplet. A range of values is given when the selection depends upon detector components traversed.

Variable	Selection
θ-breaking angle	$\leq$0.05–0.07 [rad]
ϕ-breaking angle	$\leq$0.05–0.12 [rad]
p_T	$\geq$0.75 [GeV]



x_2
x_1
$\langle \mathbf{w}\cdot\mathbf{x}\rangle + b = -1$
$\langle \mathbf{w}\cdot\mathbf{x}\rangle + b = 0$
$\langle \mathbf{w}\cdot\mathbf{x}\rangle + b = 1$
$\langle \mathbf{w}\cdot\mathbf{x}\rangle + b > 1$
$\langle \mathbf{w}\cdot\mathbf{x}\rangle + b < -1$
$\mathbf{w}$
$\frac{1}{\sqrt{\langle \mathbf{w}\cdot\mathbf{w}\rangle}}$

FIG. 3. A visual representation of two classes of data in a 2-dimensional space, separated by a hyperplane $\langle \mathbf{w}\cdot\mathbf{x}\rangle + b$ (solid line). The highlighted points lying closest to the separation plane are called support vectors and the dotted lines passing through them define the margins.

of data. It takes a training dataset of size N of the form $(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^N, y^N)$, where $\mathbf{x}^i$ is an M-dimensional vector and $y^i = \pm 1$ for data that belongs to one of two classes. The hyperplane is defined by $\langle \mathbf{w}\cdot\mathbf{x}\rangle + b = 0$, where $\mathbf{w}$ is the normal vector to the hyperplane and b is an offset. These parameters are determined during the learning process. For the simple case of linearly separable data, the training points $\mathbf{x}^i$ of the two classes are placed on either side of the decision boundary, satisfying $f(\mathbf{x}^i) = \mathrm{sign}(\langle \mathbf{w}\cdot\mathbf{x}^i\rangle + b) = y_i$, where $f(\mathbf{x})$ is called the decision function. The points closest to the hyperplane are called support vectors and the distance between them is called the margin. The goal is to optimise the parameters of the hyperplane such that the margin is maximized. Figure 3 shows a visual representation of this. Once the hyperplane has been found, a previously unseen data point $\mathbf{z}$ can be classified using the decision function.

The decision boundary is usually defined not in the original data space but in a higher-dimensional *feature space* obtained with a feature map $\phi(\mathbf{x})$. This can introduce nonlinearity while keeping the decision boundary linear. The goal of this operation is to achieve better separation of the two classes. Figure 4 shows a simple example of a feature map's functionality. SVMs are an example of a kernel method, where the kernel $k(\mathbf{x},\mathbf{y}) = \langle \phi(\mathbf{x})\cdot\phi(\mathbf{y})\rangle$ is a function with arguments in the original space of the data, defining a distance measure between two points in the feature space. The remarkable property of this function is that it returns the inner product in the feature space, sidestepping the explicit application of the feature map, which can become computationally expensive for sophisticated feature spaces. In support vector machines, this property can be utilized to find the separation hyperplane. This is possible because linear learning machines can be expressed in a dual representation, following the Karush-Kuhn-Tucker theory [26]. During the optimization of the dual problem, one needs to find a kernel matrix $K_{\mathbf{x},\mathbf{y}} = k(\mathbf{x},\mathbf{y})$ (an $N\times N$ symmetric matrix) from all pairs of training data points. Expressed in its dual form, the decision function becomes:

$$f(\mathbf{x}) = \mathrm{sign}\left(\sum_{i=1}^{N} y^i \alpha^i k(\mathbf{x}^{\mathbf{i}},\mathbf{x}) + b\right), \tag{1}$$

where α^i are the coefficients which need to be optimized. Just like quantum computing, kernel methods perform implicit computations in a possibly intractably-large Hilbert space through the efficient manipulation of data inputs.

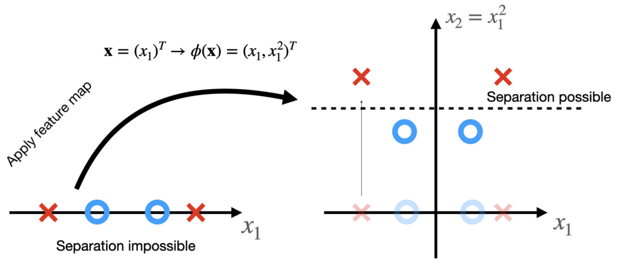


FIG. 4. A visual representation of a simple feature map that takes an inseparable dataset in one-dimension to a two-dimensional feature space. Separation with a linear hyperplane is possible in the new feature space.

IV. QUANTUM KERNEL ESTIMATION

Quantum computers can be utilized in kernel methods if one considers a quantum circuit $\mathcal{U}(\mathbf{x})$ whose gates are parametrized by the original features of some classical data. The result of such a circuit before measurement is a quantum state which exists in a higher-dimensional Hilbert space. This is equivalent to a feature map. The quantum state is defined as [27]:

$$\mathbf{x} \rightarrow \rho(\mathbf{x}) = |\psi(\mathbf{x})\rangle\langle\psi(\mathbf{x})|, \tag{2}$$

where $|\cdot\rangle$ denotes the usual Dirac vector and $\rho(\mathbf{x})$ is obtained via

$$\rho(\mathbf{x}) = \mathcal{U}^\dagger(\mathbf{x})\rho_0\mathcal{U}(\mathbf{x}), \tag{3}$$

with an initial state ρ_0. An all-zero initial state is used with $|\psi_0\rangle = |0^{\otimes M}\rangle$. The kernel associated with such a feature map is obtained from [28]:

$$K(\mathbf{x},\mathbf{z}) = \mathrm{tr}\{\rho(\mathbf{x})\rho(\mathbf{z})\} = |\langle\psi(\mathbf{x})|\psi(\mathbf{z})\rangle|^2. \tag{4}$$

This inner product can be calculated from the transition amplitude of two states;

$$|\langle\psi(\mathbf{x})|\psi(\mathbf{z})\rangle|^2 = |\langle 0^{\otimes M}|\mathcal{U}^\dagger(\mathbf{x})\mathcal{U}(\mathbf{z})|0^{\otimes M}\rangle|^2. \tag{5}$$

The circuit $\mathcal{U}^\dagger(\mathbf{x})\mathcal{U}(\mathbf{z})|0^{\otimes M}\rangle$ is run repeatedly over R identical runs (shots). The fraction of measurements

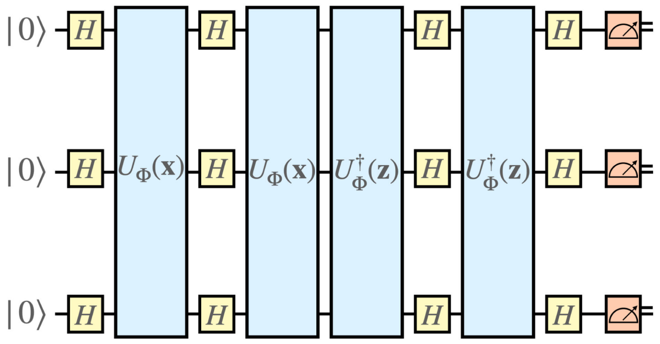


FIG. 5. Quantum circuit diagram used to estimate the kernel and determine the inner product between two quantum states shown for data with three features.

yielding an all-zero output gives an estimation of the kernel function for the two points $\mathbf{x}$ and $\mathbf{z}$, which forms an entry in the kernel matrix. Repeated evaluations for all combinations of the input dataset gives the full kernel matrix. Similar states have large kernel matrix entries while orthogonal points give $k(\mathbf{z},\mathbf{x}) = 0$.

Feature maps of particular interest are those that are difficult to calculate using classical means while providing good classification of data. Ideally, a kernel matrix resulting from Eq. (4) would produce results better than any classical classifier and be calculated significantly faster on a quantum device. The kernel function proposed in [28] is based on the 3-fold forrelation ("Fourier correlation") problem [29]. The function is conjectured to have an exponential separation in complexity between its quantum and classical estimation. Further discussion of the potential for speedup is presented later.

The feature map circuit is of the form $\mathcal{U}(\mathbf{x}) = U_{\phi(\mathbf{x})} H^{\otimes M} U_{\phi(\mathbf{x})} H^{\otimes M}$ where H is the Hadamard gate and

$$U_{\phi(\mathbf{x})} = \exp\left(i \sum_{S \subseteq [M]} \phi_S(\mathbf{x}) \prod_{i \in S} Z_i\right). \tag{6}$$

Z_i is a gate rotating the ith qubit around the Z axis on the Bloch sphere by an amount defined by $\phi_S(\mathbf{x})$. S denotes a subset of qubits. Only subsets with $|S| \leq 2$ are considered. The circuit for kernel estimation with $\mathcal{U}(\mathbf{x})$ in the case of 3-dimensional data is shown in Fig. 5. Ideas for generalizing the circuit have been proposed in [30,31]. Following the latter, we implement unitaries of the form:

$$U_{\phi(\mathbf{x})} = \exp\left(i\alpha \sum_{S \subseteq [M]} \phi_S(\mathbf{x}) \prod_{i \in S} \sigma_i\right), \tag{7}$$

where $\sigma \in X, Y, Z$ and α is a constant factor to regulate the degree of rotation of the qubits. An example of $U_{\phi(\mathbf{x})}$ for 3-dimensional data can be found in Fig. 6.

This quantum-estimated kernel is then used as input to a classical support vector machine which performs the training and classification. The full circuit thus takes data points of dimension M and projects them into an 2^M-dimensional quantum space where the hyperplane separating the two classes of data is calculated.

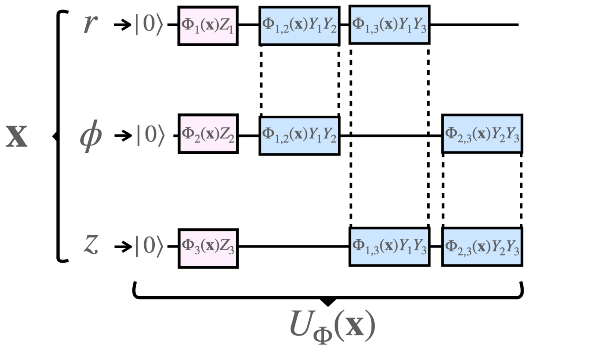


FIG. 6. Circuit diagram used to calculate $U_{\phi(\mathbf{x})}$, part of a kernel estimation circuit, shown for a datapoint with three features, which correspond to the spatial coordinates of a single hit in the tracking detector. The single-qubit gates are shown in pink and two-qubit gates in blue.

V. RESULTS

Results from the quantum-enhanced and fully classical SVM algorithm presented in this section were obtained with optimal models found via a grid search [32] with cross validation and a parameter scan optimizing for validation score and training time. The rotation mappings were chosen as $\phi_k(\mathbf{x}) = x_k$ (one-qubit), and $\phi_{l,m}(\mathbf{x}) = (\pi - x_l)(\pi - x_m)$ (two-qubit). A regularization term C can be included into the optimization loss function, controlling the trade-off between finding a large margin and correctly labeling as much of the training data as possible. Large values of C favor smaller margins. The value of $C = 10^6$ determined from this optimization procedure proved optimal in both the classical and quantum kernels. $\alpha = 0.1$, $\sigma_k = Z$, for single qubit rotations and $\sigma_{l,m} = Y_l Y_m$ for two-qubit rotations were chosen. These were compared to an RBF kernel [33], defined as $K_{\mathrm{RBF}}(\mathbf{x},\mathbf{z}) = \exp -\gamma(\|\mathbf{x}-\mathbf{z}\|^2)$ with $\gamma = 1$. Both data sets were studied with the same quantum and classical models.

The metrics used to quantify the performance of the classifiers are defined below, through the confusion matrix shown in Table III.

TABLE III. Confusion matrix used in defining the performance metrics for the classifiers used in triplet recognition.

	Predicted positive	Predicted negative
Actual positive	True positive (TP)	False negative (FN)
Actual negative	False positive (FP)	True negative (TN)

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}, \tag{8}$$

$$\text{Efficiency} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{9}$$

$$\text{Purity} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{10}$$

A good model is expected to score high in all three metrics. In brief, accuracy gives an overall percentage of correct guesses, efficiency is the fraction of actual true objects correctly recognized and purity measures how often the model mistakes a fake object for a true one.

A. Full detector triplets

The dataset used in this classification consists of triplets within the whole barrel region that passed the preprocessing step described earlier, which ensures a sample purity (the percentage of data points labeled "+1" in the remaining data post-selection) of 52% and 80% for doublets and triplets, respectively and a 99% sample efficiency (the percentage of data points labeled "+1" passing the selection) in both datasets. An average of 4,600 triplets remain per event. The spatial coordinates of each hit in the triplet are used as the input data to the hybrid algorithm. This results in a 9-qubit circuit. To accommodate the dataset into our current computational constraints, it is further divided into 16 equal sections in the ϕ plane, with each section subtending $\frac{2\pi}{16}$ radians in ϕ. A support vector machine is then defined for each of these regions and the relevant quantum kernel estimated. The data is divided into 70 events for training and 15 events for testing, equivalent to a total of around 320,000 and 70,000 triplets, respectively. The performance of both the classical algorithm and the quantum algorithm are evaluated using the three metrics defined above; accuracy, efficiency and purity. Furthermore, since the preprocessing step selects triplets that are more likely to form track candidates, a benchmark scenario is introduced to illustrate the performance of the classical and quantum algorithms on top of the preprocessed data. Triplets are randomly assigned labels based on the sample purity and the classical and quantum classifiers are compared against this benchmark to demonstrate the improvements in classification accuracy.

Figure 7 shows the dependence of the efficiency and purity scores on relevant geometric and kinematic properties of the triplets; the angle ϕ of the first (innermost) hit of the triplet, the pseudorapidity $|\eta|$ of the triplet, the true p_T of the triplet, the number of true particles in the event (particle multiplicity), and the number of hits corresponding to a true track (track length). For fake triplets, there is some ambiguity in determining the true p_T and track length, as the three hits can come from three separate

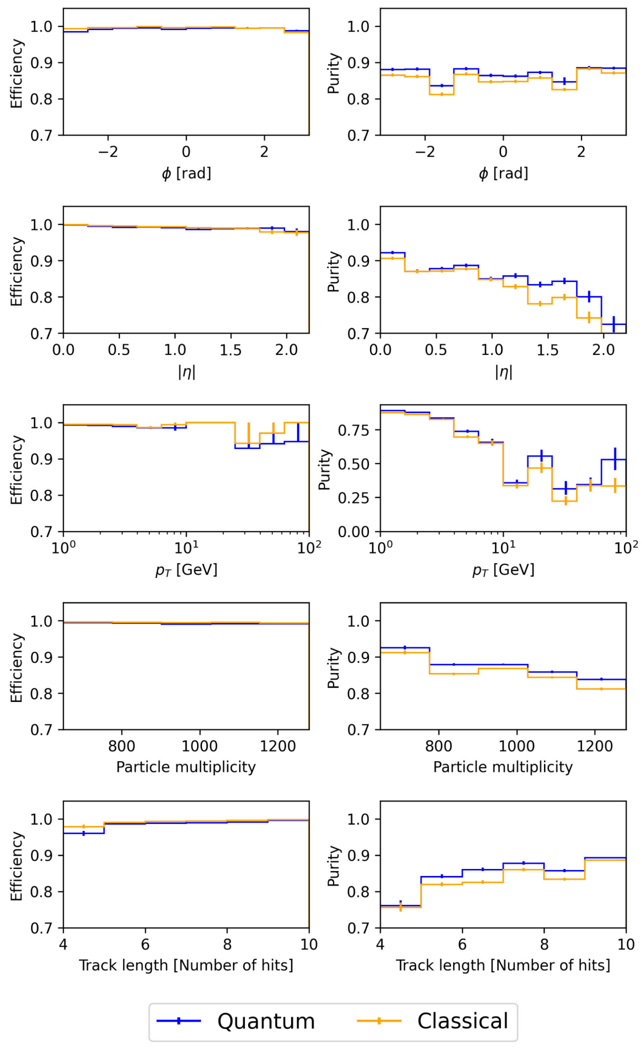


FIG. 7. Track reconstruction efficiency and purity for triplets in the barrel detector as a function of ϕ, $|\eta|$, p_T, particle multiplicity, and the number of hits associated with the track (track length). These are compared for the quantum-estimated kernel and the classical kernel.

particles. In such cases, the choice was made to define these variables using the particle associated with the first hit. The two classifiers show mostly comparable performance and a similar dependence on the observables defined above. Efficiencies close to 1.0 are achieved for most bins. Reduced values of the purity are observed in regions with reduced number of triplets for training, such as high-η and high-p_T. The accuracy scores of the classical and quantum algorithms as a function of the size of the training data are shown in Fig. 8. The training size grows up to the computational limit imposed by the quantum simulator. While the overall performance of the two algorithms follows a similar trend, the classical algorithm performs slightly better at low training size and the quantum algorithm shows an advantage for training size above 6,000 triplets. Both algorithms

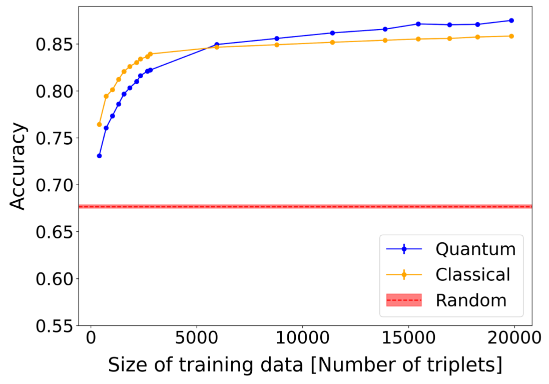


FIG. 8. Accuracy to identify triplets in the barrel detector as a function of the size of the training dataset for the quantum-estimated kernel, classical kernel and selecting random triplets from the preprocessed dataset.

significantly outperform the benchmark scenario of randomly selecting triplets.

It is also instructive to study the performance of these algorithms for different layers of the detector, as the detector occupancy progressively decreases from the inner to the outer layers. Figure 9 shows the comparison between the quantum and classical algorithms for the efficiency and purity in different layers of the detector. While the efficiency and purity are similar between the two algorithms for the full detector, the largest difference in purity occurs for triplets identified in the first layer of the detector (the first hit is in the first layer). Since triplet formation in the innermost region of the detector is part of the seeding step used in many track reconstruction algorithms, we study the performance of our models for triplets identified in the inner detector, with the first hit being in the first layer of the detector.

B. Innermost triplets

This section presents results when restricting the classification of triplets to those in the innermost layers of the tracking detector, with the three hits of a triplet in the first three layers. The reduction in the total number of triplets per event allows more events to be processed before reaching the computational limit. The dataset is split into 310 events for training and 60 events for testing, equivalent to about 300,000 and 60,000 triplets, respectively. The efficiency and purity as a function of triplet parameters are shown in Fig. 10 and the accuracy is shown as a function of the size of training data in Fig. 11. The accuracy indicates a clearer separation between the quantum and classical performances. We see a continued trend of better purity for the quantum-enhanced classifier and a comparable performance in terms of efficiency.

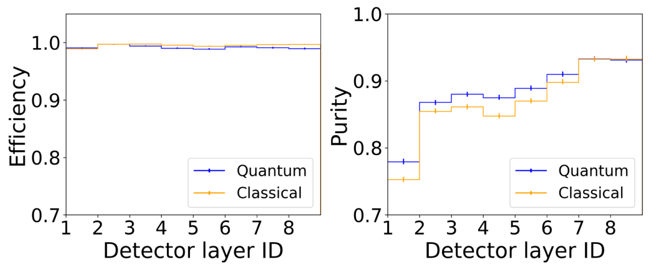


FIG. 9. Efficiency and purity as a function of the detector layer, starting from the innermost layer nearest the center of the detector. Layers 1–4 correspond to the pixel detector, layers 5–8 belong to the short strip detector and layers 9 and 10 are the long strip detector. Triplets are binned by their innermost hit.

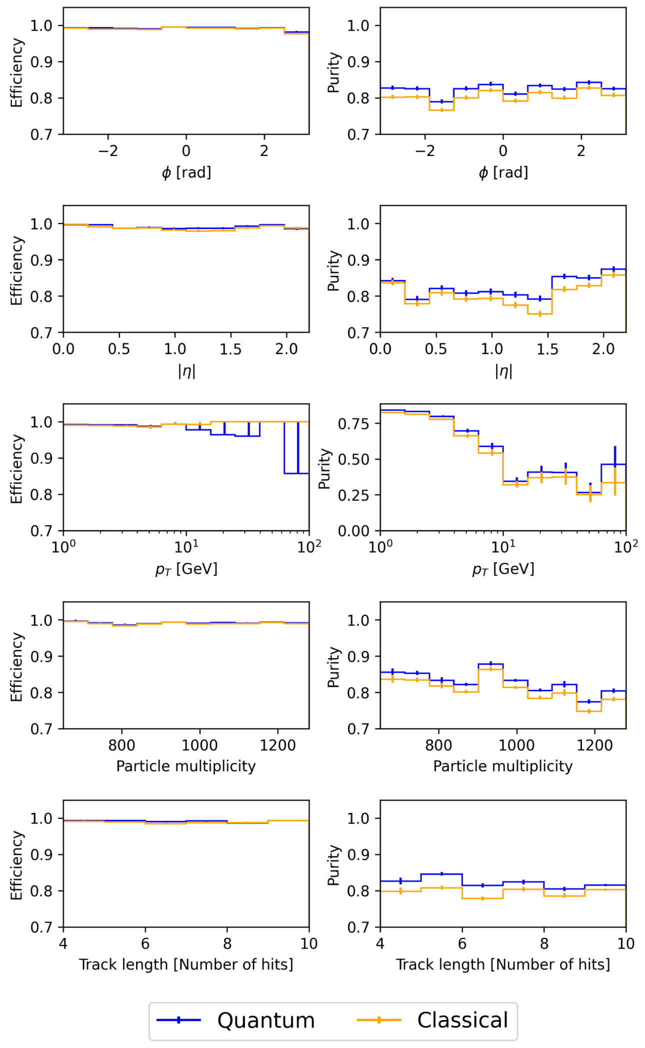


FIG. 10. Track reconstruction efficiency and purity for triplets in the inner detector barrel region as a function of ϕ, $|\eta|$, p_T, particle multiplicity, and the number of hits associated with the track (track length). These are compared for the quantum-estimated kernel and a classical kernel.

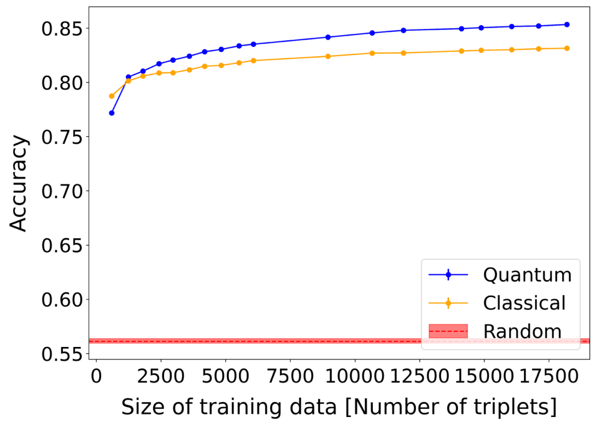


FIG. 11. Accuracy to identify triplets in the inner detector barrel region as a function of the size of the training data for the quantum-estimated kernel, classical kernel and selecting random triplets from the preprocessed data.

C. Model complexity metrics

The authors of [34] introduced a set of metrics which can be used to quantify the differences between the quantum and classical models (kernels) K_Q and K_C, respectively.

The central result of the paper is the prediction error bound:

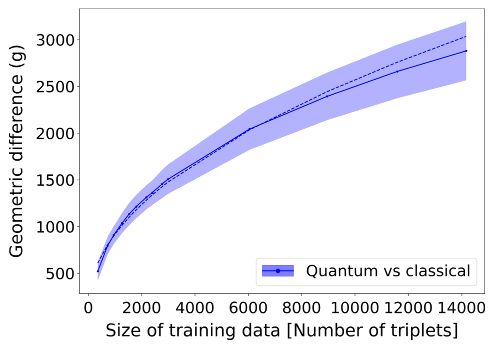
$$\mathbb{E}_{x\sim\mathcal{D}}|h_K(x) - h_T(x)| \leq c\sqrt{\frac{s_K(N)}{N}}. \tag{11}$$

This error bound tells us how much the output of a trained model $[h_K(x)]$ is allowed to differ, on average, from a theoretical target model $[h_T(x)]$ describing the underlying distribution we wish to learn. The quantity s_K plays a crucial role in determining the accuracy of the trained model and describes its complexity. It has a straightforward interpretation $s_K = \|\mathbf{w}\|$ which is the inverse of the length of the margin of the support vector machine introduced earlier. To find this quantity, one can use the trained kernel:

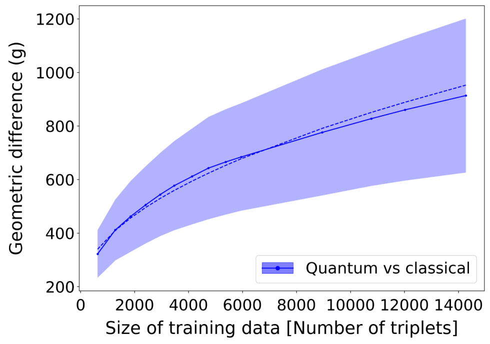
$$s_K^\lambda(N) = y^\dagger(\sqrt{K}(K+\lambda I)^{-2}\sqrt{K})y, \tag{12}$$

where $\lambda = \frac{1}{2C}$.

If $s_K(N)$ is linear or sublinear, a small upper bound on the prediction error in Eq. (11) can be achieved. Note, however, that this theoretical error does not readily translate to the classification error in our task. It describes the difference between the output of the model itself $(\langle\mathbf{w}\cdot\mathbf{x}\rangle + b)$ and some ideal theoretical model, not the labels assigned to points based on those models. This means that while a point mapped by the trained model may be near the ideal point, there is still a possibility that it falls on the opposite side of the decision boundary. However, generally speaking, the closer these points are, on average, to the target model, the lower the likelihood of them falling on the wrong side of the decision boundary.



(a) Full detector triplets



(b) Innermost triplets

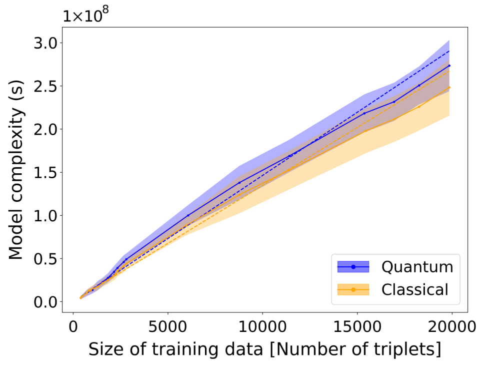
FIG. 12. Geometric difference (solid line) between the quantum and classical models for the (a) full detector triplets and (b) inner detector triplets along with a square-root fit (dashed line).

One can compare two models using their geometric difference g_{CQ}, which describes how different the feature mappings made by the two models are

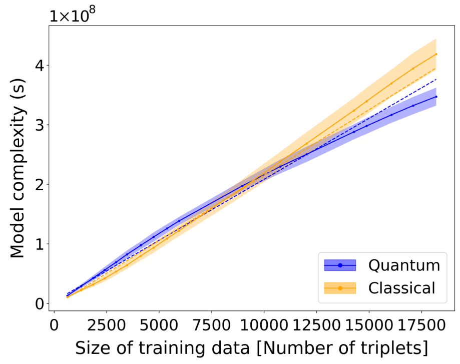
$$g_{CQ} = \sqrt{\left\|\sqrt{K^Q}\sqrt{K^C}(K^C+\lambda I)^{-2}\sqrt{K^C}\sqrt{K^Q}\right\|_\infty}. \tag{13}$$

This quantity is independent of the training labels. The maximal model complexity for one model (e.g. the classical kernel) compared to another (the quantum kernel) is bounded with $s_C \leq g_{CQ}^2 s_Q$. A small value of g indicates that the classical model will always perform similarly or better. A large g_{CQ} means that there exists a dataset (a specific configuration of labels in the training set) for which the quantum model should generalize much better than the classical one. In particular, a scaling of $g \propto \sqrt{N}$ ensures a possible large separation between the two models' complexities.

Figure 12 shows the geometric difference between the two kernels used in the classifications of the different datasets in this study at increasing training sizes. Also included is a best fit to a square-root function of the training size. The two triplet datasets show clear upward trends that



(a) Full detector triplets



(b) Innermost triplets

FIG. 13. Model complexity (solid line) of the quantum and classical models for the (a) full detector triplets and (b) inner detector triplets along with a linear fit (dashed line).

can visually be fit well by a square-root dependence, both reaching high values of g.

In Fig. 13 we compare the growth in training data size of the model complexity of the two kernels for the datasets. The quantum complexity follows a linear or sublinear trend in both of them. Interestingly, in the case of the inner triplets, the classical model complexity becomes larger than the quantum one, tending above a linear fit.

We argue that these results provide a potential explanation of the overall good performance of the quantum and classical models in the two datasets. They can also suggest why the separation between the accuracy results is larger in the inner triplet dataset.

VI. POTENTIALS OF QUANTUM SPEEDUP

In general, we can assume that the evaluation of the matrix elements dominates the complexity of the SVM [35]. Thus, the complexity of the algorithm is $\mathcal{C} = \mathcal{O}(\beta N^2)$, where the β factor depends on the kernel type and method used. For a single value of the kernel, the quantum complexity is $\beta_Q = \mathcal{O}(\epsilon^{-2})$ for some additive error ϵ [28,36]. The current best classical algorithm proposed in [37] has $\beta_C = \mathcal{O}(\epsilon^{-\frac{2}{3}} 2^{\frac{2M}{3}})$. Demanding that $\beta_Q < \beta_C$ for $\epsilon = 10^{-3}$ results in a possible advantage for $M \approx 20$.

In [36] it has been shown that in order for the full kernel matrix to approximate the ideal kernel, the propagation of the desired accuracy into the classification with an SVM causes β_Q to pick up a non-negligible scaling with training size N; $\beta_Q \to \beta'_Q = \mathcal{O}(N^{\frac{8}{3}} \epsilon^{-2})$. It is possible that a similar analysis could introduce an N-scaling to β_C. Regardless, it appears that at the current stage, quantum kernel estimation could provide possible advantage for small datasets where the data has many features. In our study, while quantum advantage in accuracy has been observed, nine features were used. Possible speedup may be obtained if the length of the track segments is extended or additional hit information (beyond the three spatial points) is added to the data features.

Another point to consider for quantum kernel estimation is the noise on current quantum devices. In [38], the authors show that the presence of noise can cause kernel entries produced with Eq. (6) evaluated over different input data to concentrate around some fixed value. The difference between any kernel entry and that value becomes exponentially small with the number of qubits. This results in an exponential number of shots necessary to resolve kernel entries for successful training. This dependence would have to be added into β_Q in order for the required precision to be obtained.

Some proposals for different quantum kernel estimation methods can be found in [36], where a probabilistic algorithm calculates only a subset of the kernel entries and [39] where the quantum computation timescales linearly with N, and building of the kernel is outsourced back to a classical computer. Further studies could include empirical tests of classical and quantum time complexities, study of noise effects in simulations and real quantum devices as well as implementation of other proposed quantum kernel estimation techniques in the context of high energy physics.

VII. SUMMARY

Reconstructing the trajectories of charged particles at particle colliders like the Large Hadron Collider is a challenging, computationally intensive problem. This is expected to become increasingly complex with the upgraded Large Hadron Collider (HL-LHC) where $\mathcal{O}(10^5)$ hits in the tracking detector must be quickly and accurately connected to form around 10,000 tracks.

This paper presents a hybrid quantum-classical algorithm that utilises a support vector machine (SVM) with a quantum-estimated kernel for the classification of track segments in this challenging track reconstruction problem. Using a publicly available dataset that simulates a generic particle detector for the HL-LHC, we apply selection criteria to select doublets (sets of two consecutive hits) and subsequently triplets (sets of three consecutive hits). The proposed algorithm classifies these triplets as either belonging to a particle track or not. A comparison is made between the performance of a quantum-estimated kernel, a classical kernel, and randomly selected triplets from the dataset. The quantum algorithm demonstrates a slightly higher level of performance. Notably, when focusing on triplets from the inner part of the tracking detector, the distinction in performance becomes more pronounced. This is promising as the innermost layers are expected to be the most important for the initial seeding step of track reconstruction. Additionally, complexity metrics are employed to provide insights into the obtained results. This is the first implementation of a quantum-kernel SVM approach in the context of track reconstruction.

ACKNOWLEDGMENTS

Sarah Malik and Tim Scanlon are funded by grants from the Royal Society. Sébastien Rettie acknowledges support from the Banting Postdoctoral Fellowship program, and the Natural Sciences and Engineering Research Council of Canada. We also acknowledge funding from the STFC. We thank Simeon Hatzopoulos for helpful research assistance in visualizing the TrackML dataset and Mohammad Hassanshahi for valuable discussions.

[1] ATLAS and CMS Collaborations, Report on the physics at the HL-LHC and perspectives for the HE-LHC, 10.23731/CYRM-2019-007.Addendum (2019).

[2] ATLAS Collaboration, Technical design report for the phase-II upgrade of the ATLAS trigger and data acquisition system—event filter tracking amendment, 10.17181/CERN.ZK85.5TDL (2022).

[3] D. Contardo, M. Klute, J. Mans, L. Silvestris, and J. Butler, Technical proposal for the phase-II upgrade of the CMS detector, 10.17181/CERN.VU8I.D59J (2015).

[4] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik, Noisy intermediate-scale quantum algorithms, Rev. Mod. Phys. **94**, 015004 (2022).

[5] K. Bepari, S. Malik, M. Spannowsky, and S. Williams, Towards a quantum computing algorithm for helicity amplitudes and parton showers, Phys. Rev. D **103**, 076020 (2021).

[6] K. Bepari, S. Malik, M. Spannowsky, and S. Williams, Quantum walk approach to simulating parton showers, Phys. Rev. D **106**, 056002 (2022).

[7] A. Y. Wei, P. Naik, A. W. Harrow, and J. Thaler, Quantum algorithms for jet clustering, Phys. Rev. D **101**, 094015 (2020).

[8] D. Pires, P. Bargassa, J. a. Seixas, and Y. Omar, A digital quantum algorithm for jet clustering in high-energy physics, arXiv:2101.05618.

[9] J. J. Martínez de Lejarza, L. Cieri, and G. Rodrigo, Quantum clustering and jet reconstruction at the LHC, Phys. Rev. D **106**, 036021 (2022).

[10] K. Terashi, M. Kaneda, T. Kishimoto, M. Saito, R. Sawada, and J. Tanaka, Event Classification with quantum machine learning in high-energy physics, Comput. Softw. Big Sci. **5**, 2 (2021).

[11] A. Blance and M. Spannowsky, Unsupervised event classification with graphs on classical and photonic quantum computers, J. High Energy Phys. 08 (2021) 170.

[12] 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**, 033221 (2021).

[13] V. S. Ngairangbam, M. Spannowsky, and M. Takeuchi, Anomaly detection in high-energy physics using a quantum autoencoder, Phys. Rev. D **105**, 095004 (2022).

[14] I. Shapoval and P. Calafiura, Quantum associative memory in HEP track pattern recognition, EPJ Web Conf. **214**, 01012 (2019).

[15] C. Tüysüz, F. Carminati, B. Demirköz, D. Dobos, F. Fracas, K. Novotny, K. Potamianos, S. Vallecorsa, and J.-R. Vlimant, Particle track reconstruction with quantum algorithms, EPJ Web Conf. **245**, 09013 (2020).

[16] C. Tüysüz, F. Carminati, B. Demirköz, D. Dobos, F. Fracas, K. Novotny, K. Potamianos, S. Vallecorsa, and J.-R. Vlimant, A quantum graph neural network approach to particle track reconstruction, in *Connecting The Dots* (2020), 10.5281/zenodo.4088474.

[17] A. Zlokapa, A. Anand, J.-R. Vlimant, J. M. Duarte, J. Job, D. Lidar, and M. Spiropulu, Charged particle tracking with quantum annealing-inspired optimization, Quantum Mach. Intell. **3**, 27 (2021).

[18] M. Saito, P. Calafiura, H. Gray, W. Lavrijsen, L. Linder, Y. Okumura, R. Sawada, A. Smith, J. Tanaka, and K. Terashi, Quantum annealing algorithms for track pattern recognition, EPJ Web Conf. **245**, 10006 (2020).

[19] F. Bapst, W. Bhimji, P. Calafiura, H. Gray, W. Lavrijsen, L. Linder, and A. Smith, A pattern recognition algorithm for quantum annealers, Comput. Softw. Big Sci. **4**, 1 (2020).

[20] S. Amrouche *et al.*, The tracking machine learning challenge: Accuracy phase, in *The NeurIPS '18 Competition: From Machine Learning to Intelligent Conversations*, edited by S. Escalera and R. Herbrich, The Springer Series on Challenges in Machine Learning (Springer, Cham, 2019).

[21] S. Amrouche *et al.*, The tracking machine learning challenge: Throughput phase, Comput. Softw. Big Sci. **7**, 1 (2023).

[22] G. Aad *et al.* (ATLAS Collaboration), The ATLAS experiment at the CERN Large Hadron Collider, J. Instrum. **3**, S08003 (2008).

[23] S. Chatrchyan *et al.* (CMS Collaboration), The CMS experiment at the CERN LHC, J. Instrum. **3**, S08004 (2008).

[24] S. Farrell *et al.*, Novel deep learning methods for track reconstruction, in *Proceedings of the 4th International Workshop Connecting The Dots 2018* (2018), arXiv:1810.06111.

[25] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods* (Cambridge University Press, Cambridge, England, 2000).

[26] R. K. Sundaram, *A First Course in Optimization Theory* (Cambridge University Press, Cambridge, England, 1996).

[27] M. Schuld and F. Petruccione, Quantum models as kernel methods, in *Machine Learning with Quantum Computers* (Springer International Publishing, Cham, 2021), pp. 217–245.

[28] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Supervised learning with quantum-enhanced feature spaces, Nature (London) **567**, 209 (2019).

[29] S. Aaronson and A. Ambainis, Forrelation: A problem that optimally separates quantum from classical computing, SIAM J. Comput. **47**, 982 (2018).

[30] R. Shaydulin and S. M. Wild, Importance of kernel bandwidth in quantum machine learning, Phys. Rev. A **106**, 042407 (2022).

[31] J.-E. Park, B. Quanz, S. Wood, H. Higgins, and R. Harishankar, Practical application improvement to quantum SVM: Theory to practice, arXiv:2012.07725.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. **12**, 2825 (2011).

[33] M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, and D. M. Hummels, On the training of radial basis function classifiers, Neural Netw. **5**, 595 (1992).

[34] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean, Power of data in quantum machine learning, Nat. Commun. **12**, 2631 (2021).

[35] L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, *Large-Scale Kernel Machines (Neural Information Processing)* (The MIT Press, Cambridge, Massachusetts, USA, 2007).

[36] G. Gentinetta, A. Thomsen, D. Sutter, and S. Woerner, The complexity of quantum support vector machines, Quantum **8**, 1225 (2024).

[37] S. Bravyi, D. Gosset, D. Grier, and L. Schaeffer, Classical algorithms for forrelation, arXiv:2102.06963.

[38] S. Thanasilp, S. Wang, M. Cerezo, and Z. Holmes, Exponential concentration and untrainability in quantum kernel methods, arXiv:2208.11060.

[39] T. Haug, C. N. Self, and M. S. Kim, Quantum machine learning of large datasets using randomized measurements, Mach. Learn. **4**, 015005 (2023).