

Quantum-inspired Machine Learning on high-energy physics data

Marco Trenti,¹ Lorenzo Sestini,² Alessio Gianelle,² Davide Zuliani,^{1,2}
Timo Felser,^{1,2,3} Donatella Lucchesi,^{1,2} and Simone Montangero^{1,2}

¹*Dipartimento di Fisica e Astronomia “G. Galilei”, Università di Padova, I-35131 Padova, Italy*

²*INFN, Sezione di Padova, I-35131 Padova, Italy.*

³*Theoretische Physik, Universität des Saarlandes, D-66123 Saarbrücken, Germany.*

(Dated: April 30, 2020)

One of the most challenging big data problems in high energy physics is the analysis and classification of the data produced by the Large Hadron Collider at CERN. Recently, machine learning techniques have been employed to tackle such challenges, which, despite being very effective, rely on classification schemes that are hard to interpret. Here, we introduce and apply a quantum-inspired machine learning technique and, exploiting tree tensor networks, we show how to efficiently classify b -jet events in proton-proton collisions at LHCb and to interpret the classification results. In particular, we show how to select important features and adapt the network geometry based on information acquired in the learning process. Moreover, the tree tensor network can be adapted for optimal precision or fast response in time without the need of repeating the learning process. This paves the way to high-frequency real-time applications as needed for current and future LHC event classification to trigger events at the tens of MHz scale.

Artificial Neural Networks (NN) are a well-established tool for applications in Machine Learning and they are of increasing interest in both research and industry [1–6]. Inspired by biological neural networks, they are able to recognize patterns while processing a huge amount of data. In fact, NNs describe a functional mapping containing many variational parameters, which are optimised during the training procedure. Recently, deep connections between Machine Learning and quantum physics have been shown and continue to be uncovered [7]. On one hand, NNs have been applied to describe the behavior of complex quantum many-body systems [8–10], while on the other hand quantum-inspired technologies and algorithms are taken into account to solve Machine Learning tasks [11–13].

One particular numerical method originated from quantum physics which has been increasingly compared to NNs are Tensor Networks (TNs) [14–16]. TNs have been developed to investigate quantum many-body systems on classical computers by efficiently representing the quantum wavefunction $|\psi\rangle$ in a compact form and they have proven to be an essential tool for a broad range of applications [17–26]. The accuracy of the TN approximation can be controlled with the so-called *bond-dimension* χ , an auxiliary dimension for the indices of the connected local tensors. Recently, it has been shown that TN methods can be applied to solve Machine Learning (ML) tasks [13, 16, 27–30].

In this paper, we present a novel TN approach for the supervised learning problem of identifying the charge of b -quarks (i.e. b or \bar{b}) produced in high-energy proton-proton collisions at the Large Hadron Collider (LHC) accelerator at CERN. Due to their original development focusing on quantum systems, TNs allow to easily compute quantities such as quantum correlations or entanglement entropy and thereby to gain insight into the learned

data from a distinct point of view for the application in ML [16, 30]. We demonstrate the approach effectiveness and, more importantly, that it allows introducing novel algorithms to simplify and explain the learning process, unveiling a pathway to a novel explainable Artificial Intelligence.

In particular, hereafter we briefly describe the LHCb experiment and its simulation framework, the main observables related to b -jets physics and the relevant quantities for this analysis. The produced b -quarks cannot exist as free particles, they manifest themselves as bound states (hadrons) or as narrow cones of particles produced by the hadronization (jets). The b -jets are detected by the apparatus that in the case of LHCb experiment [31] is located in the forward region of proton-proton collisions, as illustrated in Fig. 1. We then present the quantum-inspired TTN and the biological-inspired Deep NN (DNN) used for the analysis, together with the LHCb simulated data [32, 33] used as input and the output of the two classifiers. We further compare the performance obtained by the DNN and the TTN in separating jets initiated by b -quark or \bar{b} -quark in the considered simulated data. Finally, we present different quantities obtained by the TTN classifier which are not easily accessible for NNs, such as the correlation and entanglement entropy captured within the classifier for distinguishing the different events. We introduce the Quantum-Information Post-learning feature Selection (QuIPS), a protocol exploiting these quantities to efficiently reduce the complexity of the Machine Learning model based on the information the single features provide for the classification problem. Furthermore, we introduce the Quantum-Information Adaptive Network Optimisation (QIANO), which adapts the TN representation by reducing the number of free parameters based on the captured information within the TN while aiming to maintain the highest accuracy possible. Therewith, we can optimise the trained TN classifier for a targeted prediction speed with-

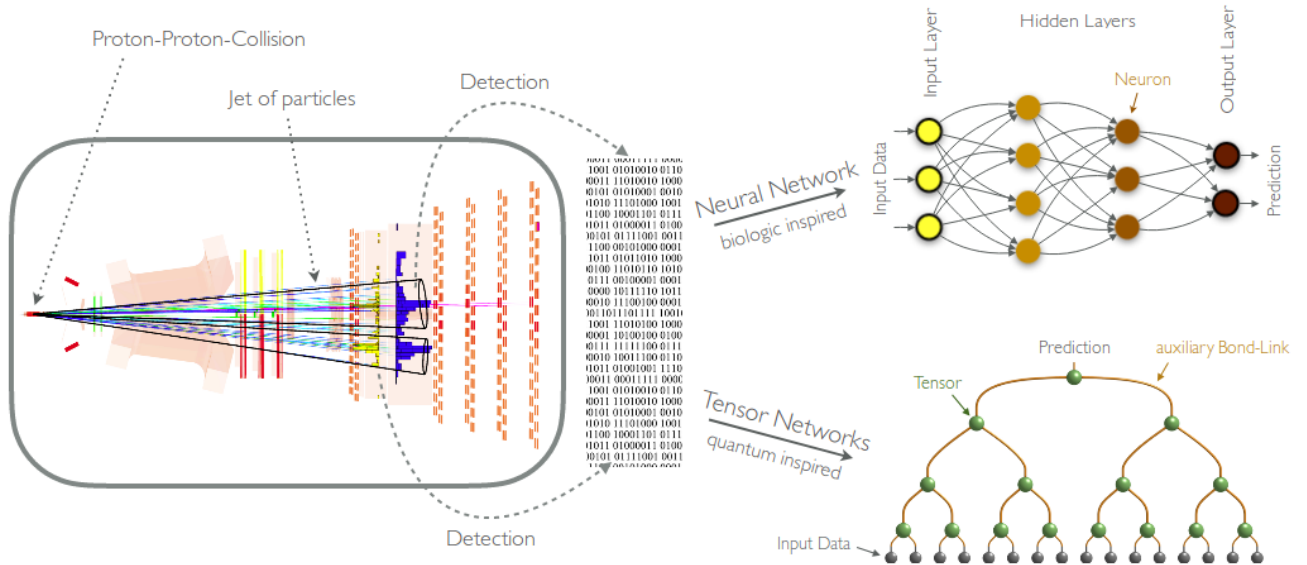


FIG. 1: Data flow of the LHCb experiment at CERN. After Proton-Proton collisions b - and \bar{b} -quarks are created, they fragment into jets to be detected (left). Selected features of the detected particle data are used as input of the Machine Learning analysis by NNs and TNs to determine the charge of the initial quark (right).

out the necessity to relearn a new model from scratch. The presented analytical insights from the TTN lead to a deeper physical understanding of the LHCb data and can be exploited to improve further analysis of high energy problems.

I. LHCb FRAMEWORK AND DATA DESCRIPTION

LHCb is an experiment located in the LHC accelerator at CERN, Geneve, mainly dedicated to the study of the physics of b - and c -quarks produced in proton-proton collisions. The LHCb detector includes a high-precision tracking system, that provides the measurement of the momentum of charged particles, and a particle identification system that distinguishes different types of charged hadrons, photons, electrons and muons [34]. The energy of charged and neutral particles is measured by electromagnetic and hadronic calorimeters.

LHCb is fully instrumented in the phase space region of proton-proton collisions defined by the pseudo-rapidity (η) range [2,5], with η defined as

$$\eta = -\log \left[\tan \left(\frac{\theta}{2} \right) \right],$$

where θ is the angle between the particle momentum and the beam axis. The direction of particles momenta can be fully identified by η and by the azimuthal angle ϕ , defined as the angle in the plane transverse to the beam axis. The projection of the momentum in this plane is called transverse momentum (p_T). In the following we work with physics natural units.

At LHCb jets are reconstructed using a Particle Flow algorithm [35] for charged and neutral particles selection and using the anti- k_t algorithm [36] for clusterization. The jet momentum is defined as the sum of the momenta of the particles that form the jet, while the jet axis is defined as the direction of the jet momentum. The particles that form the jet are contained in a cone of radius $\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2} = 0.5$, where $\Delta\eta$ and $\Delta\phi$ are respectively the pseudo-rapidity difference and the azimuthal angle difference between the particles momenta and the jet axis. For each particle inside the jet cone the momentum relative to the jet axis (p_T^{rel}) is defined as the projection of the particle momentum in the plane transverse to the jet axis.

A topic of great interest for the experiment is the identification of the charge of the quark that generated a b -jet, *i.e.* b or \bar{b} . Such identification can be used in many physics measurements, and it is the core of the determination of the charge asymmetry in b -pairs production, which is sensitive to physics beyond the Standard Model [37].

The separation between b - and \bar{b} -jets is a highly difficult task because the b -quark fragmentation produce dozens of particles via non-perturbative Quantum Chromodynamics processes, resulting in non-trivial correlations between them and the original particle. The algorithms used to identify the charge of the b -quarks are called tagging methods. Two categories of tagging algorithms exist: based on one single particle inside the jet, muon, and/or inclusively exploiting the jet sub-structure, *i.e.* information on all the jet constituents, as shown in Fig. 2.

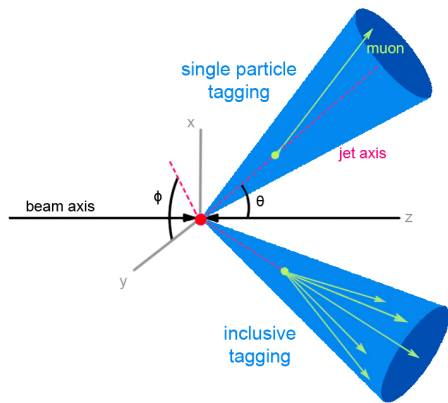


FIG. 2: Simple sketch showing the two possible tagging algorithms used at LHCb: a single particle tagging algorithm, exploiting information coming from one single particle (muon), and the inclusive tagging algorithm which exploits the information on all the jet constituents.

The tagging algorithm performance is typically quantified with the tagging power ϵ_{tag} . This tagging power represents the effective fraction of jets that contribute to the statistical uncertainty in an asymmetry measurement [38, 39]. Thus, the tagging power ϵ_{tag} takes into account the efficiency ϵ_{eff} , *i.e.* the fraction of jets where the classifier takes a decision, and the prediction accuracy a , *i.e.* the fraction of classified jet where the right decision is taken,

$$\epsilon_{tag} = \epsilon_{eff} \cdot (2a - 1)^2.$$

LHCb measured the $b\bar{b}$ forward-central asymmetry using the dataset collected in the LHC Run I [40] using the muon tagging approach: In this method, the muon with the highest momentum in the jet cone is selected, and its electric charge is used to decide on the b -quark charge. In fact, if this muon is produced in the original semi-leptonic decay of the b -hadron, its charge is totally correlated with the b -quark charge. Up to date, the muon tagging method gives the best performance on the b - vs \bar{b} -jet discrimination. Although this method can distinguish between b - and \bar{b} -quark with good accuracy, its efficiency is low as it is only applicable on jets where a muon is found and it is intrinsically limited by the b -hadrons branching ratio in semi-leptonic decays. Additionally, the muon tagging may fail in some scenarios, where the selected muon is produced not by the decay of the b -hadron but in other decay processes. In these cases, the muon may not be completely correlated with the b -quark charge.

The LHCb simulation datasets used for our analysis are produced with a Monte Carlo technique using the framework GAUSS [41], which makes use of PYTHIA 8 [42] to generate proton-proton interactions and jet fragmentation and uses EvtGen [43] to simulate b -hadrons decay. The GEANT4 software [44, 45] is used to simu-

late the detector response, and the signals are digitized and reconstructed using the LHCb analysis framework.

II. MACHINE LEARNING METHODS

The identification of the b -quark charge described in sec. I can be formulated in terms of a supervised learning problem. As described in detail in App. A, we implemented a TTN as a classifier and applied it to the LHCb problem analysing its performance; the same is done for a DNN and both algorithms are compared with the muon tagging approach. Both, the TTN and the DNN, use as input for the supervised learning 16 features of the jet substructure. The 16 features are determined as follows: the muon with the highest p_T among all other detected muons in the jet cone is selected and the same is done for the highest p_T kaon, pion, electron, and proton. In this way, 5 particles of different types are selected. For each particle, three observables are considered: (i) The momentum relative to the jet axis (p_T^{rel}), (ii) the particle charge (q), and (iii) the distance in the (η, ϕ) space between the particle and the jet axis (ΔR), resulting in 5×3 observables. If a particle type is not found in a jet, the related features are set to 0. The 16th feature is the total jet charge Q , defined as the weighted average of the particles charges inside the jet, using the particles p_T^{rel} as weights:

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

The used dataset contains b and \bar{b} -jets produced in proton-proton collisions at a center-of-mass energy of 13 TeV [32, 33]. First, pairs of b -jets and \bar{b} -jets are selected by requiring a jet p_T greater than 20 GeV and η in the range [2.2, 4.2] for both jets. Then, the dataset of about 700k jets (samples) is split into two datasets: 60% of the samples are used in the training process while the remaining 40% are used as test set to evaluate and compare the different methods.

We train the TTN as described in App. A and analyse the data with different bond dimensions χ . The auxiliary dimension χ controls the number of free parameters within the variational TTN ansatz. While the TTN is able to capture more information from the training data with increasing bond dimension χ , choosing χ too large may lead to overfitting and thus can worsen the results in the test set. For the DNN we use an optimized network with three hidden layers of 96 nodes (see App. E for details). Hereafter, we aim to compare the best possible performance of both approaches therefore, we optimised the hyper-parameters of both methods in order to obtain the best possible results from each of them, TTN and DNN.

For each event prediction, both methods give as output the probability \mathcal{P}_b to classify a jet as generated by a b - or a \bar{b} -quark. This probability (*i.e.* the confidence

of the classifier) is normalized in the following way: for values of probability $\mathcal{P}_b > 0.5$ ($\mathcal{P}_b < 0.5$) a jet is classified as generated by a b -quark (\bar{b} -quark), with an increasing confidence going to $\mathcal{P}_b = 1$ ($\mathcal{P}_b = 0$). Therefore a completely confident classifier returns a probability distribution peaked at $\mathcal{P}_b = 1$ and $\mathcal{P}_b = 0$ for jets classified as generated by b - and \bar{b} -quark respectively.

III. JET CLASSIFICATION PERFORMANCE

In the following, we present the jet classification performance for the TTN and the DNN applied to the LHCb dataset, comparing both ML techniques also with the muon tagging approach.

We introduce a threshold Δ symmetrically around the prediction confidence of $\mathcal{P}_b = 0.5$ in which we classify the event as unknown. We optimise the cut on the predictions of the classifiers (*i.e.* their confidences) to maximise the tagging power for each method based on the training samples. In the following analysis we find $\Delta^{\text{TTN}} = 0.40$ ($\Delta^{\text{DNN}} = 0.20$) for the TTN (DNN). Thereby, we predict for the TTN (DNN) a b -quark with confidences $\mathcal{P}_b > C^{\text{TTN}} = 0.70$ ($\mathcal{P}_b > C^{\text{DNN}} = 0.60$), a \bar{b} -quark with confidences $\mathcal{P}_b < 0.30$ ($\mathcal{P}_b < 0.40$) and no prediction for the range in between.

Applying both ML approaches after the training procedure on the test data, we obtain similar performances in terms of the prediction accuracy. Taking the threshold for classifying data as unknown into account, the TTN takes a decision in $\epsilon_{eff}^{\text{TTN}} = 54.5\%$ of the cases with an overall accuracy of $a^{\text{TTN}} = 70.56\%$, while the DNN decides in $\epsilon_{eff}^{\text{DNN}} = 55.3\%$ of the samples with $a^{\text{DNN}} = 70.49\%$ (see App. H for further details). We further checked both approaches for biases in physical quantities to ensure that both methods are able to properly capture the physical process behind the problem and thus that they can be used as valid tagging methods for LHCb events (see App. F).

In Fig. 3a we present the tagging power of the different approaches with respect to the jet transverse momentum p_T . Evidently, both Machine Learning methods perform significantly better than the muon tagging approach for the complete range of jet transverse momentum p_T , while the TTN and DNN both show comparable performances within the statistical uncertainties.

In Figs. 3c and 3d we present the histograms of the confidences for predicting a b -flavored jet for all samples in the test data set for the DNN and the TTN respectively. Interestingly, even though both approaches give similar performances in terms of overall precision and tagging power, the prediction confidences are fundamentally different. For the DNN, we see a Gaussian-like distribution with, in general, not very high confidences for each prediction. Thus, we obtain less correct predictions with high confidences, but at the same time, fewer wrong predictions with high confidences compared to the TTN. On the other hand, the TTN shows a flatter distribution

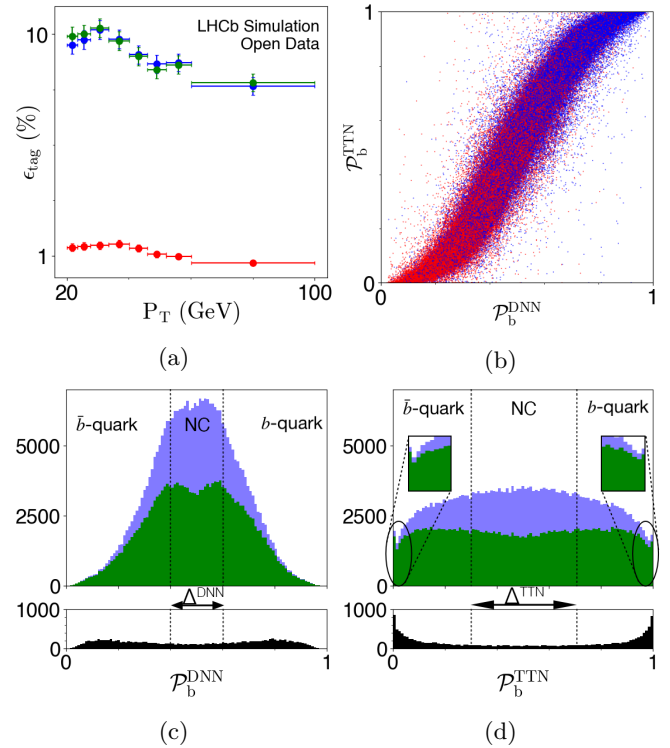


FIG. 3: Comparison of the DNN and TTN analysis: (a) Tagging power for the DNN (green), TTN (blue) and the muon tagging (red), (b) scatter plot of DNN and TTN predictions with $b(\bar{b})$ -quarks in blue(red), (c) probability distribution for the DNN, and (d) for the TTN. In the two distributions (c)+(d), the correctly classified events (green) are shown in the total distribution (light blue). Below, in black all samples where a muon was detected in the jet.

including more predictions - correct and incorrect - with higher confidence. Remarkably though, we can see peaks for extremely confident predictions (around 0 and around 1) for the TTN. These peaks can be traced back to the presence of the muon; noting that the charge of which is a well-defined predictor for a jet generated by a b -quark. The DNN lacks these confident predictions exploiting the muon charge.

Finally, in Fig. 3b the scatter plot of TTN and DNN is presented: for each jet classified as coming from a b -quark (green dots) or \bar{b} -quark (red dots) we relate the outputs of the two classifiers to spot any correlation between them. The graph shows that despite the insights of the different confidence distributions the outputs of the two classifiers are linearly correlated, with a Pearson correlation factor $r = 0.97$.

In conclusion, the two different approaches result in similar outcomes in terms of prediction performances. However, the underlying information used by the two discriminators is inherently different. For instance, the DNN predicts more conservatively, in the sense that the confidences for each prediction tend to be lower compared

with the TTN. Additionally, the DNN does not exploit the presence of the muon as strongly as the TTN, even though the muon is a good predictor for the classification.

IV. EXPLOITING INSIGHTS INTO THE DATA WITH TTN

The TTN analysis allows to efficiently measure the captured correlations and the entanglement within the classifier. These measurements give insight into the learned data and can be exploited to identify the most important features typically used for the classifications.

Therefore, we interpret the TTN classifier Ψ as a set of quantum many-body wavefunctions $|\psi_l\rangle$ - one for each of the class label l (see App. A for further details). To perform the classification, each feature x_i is encoded by the *local feature map*

$$\Phi^{[i]}(x_i) = \left[\cos\left(\frac{\pi x'_i}{2}\right), \sin\left(\frac{\pi x'_i}{2}\right) \right], \quad (1)$$

thus each feature x_i is represented by a quantum spin. Accordingly, each sample x is mapped into a product state $\Phi(x)$. Alongside, when we classify a sample x , we compute the overlap $\langle \Phi(x) | \psi_l \rangle$ for all labels l with the product state $\Phi(x)$ resulting in the weighted probabilities

$$\mathcal{P}_l = \frac{|\langle \Phi(x) | \psi_l \rangle|^2}{\sum_l |\langle \Phi(x) | \psi_l \rangle|^2}$$

for each class. We stress, that we can encode the input data in different non-linear feature maps as well (see App. A 4).

We can now calculate the correlation functions

$$C_{i,j}^l = \frac{\langle \psi_l | \sigma_i^z \sigma_j^z | \psi_l \rangle}{\langle \psi_l | \psi_l \rangle}$$

for each pair of features (located at site i and j), to gain an insight into the different information the features provide. In case of maximum correlation or anti-correlation among them for all classes l , the information of one of the features can be obtained by the other one and thus one can be neglected. In case of no correlation among them, the two features may provide fundamentally different information for the classification. For both labels ($l = b, \bar{b}$) the results are very similar, thus in Fig. 4a we present only $l = b$ (see App. B 1 for further discussion on the correlation measurements).

The correlation analysis presented above allows pinpointing if two features give independent information. However, the correlation itself does not tell if this information is important for the classification. We thus, computed the entanglement entropy S of each feature, as reported in Fig. 4b. The entanglement entropy S reflects the shared information between two TN bipartitions. The entanglement S is measured via the Schmidt

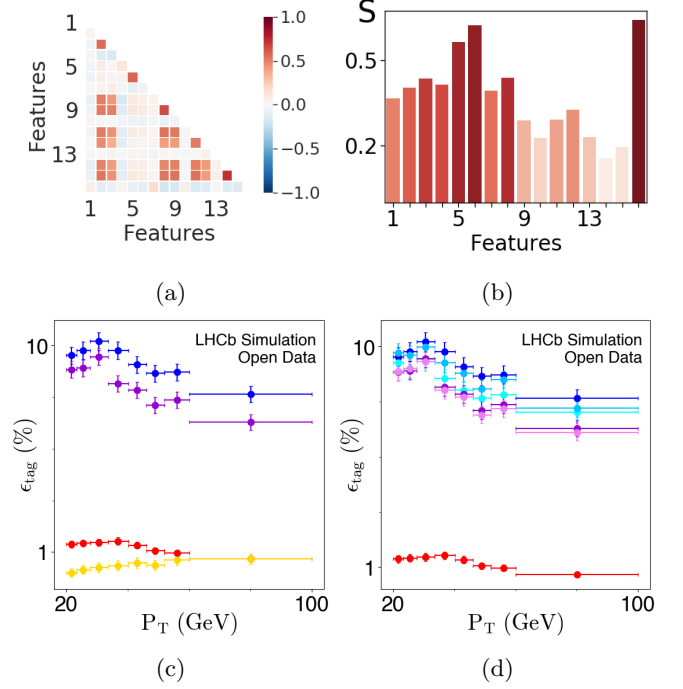


FIG. 4: Exploiting the information provided by the learned TTN classifier: (a) Correlations between the 16 input features (blue for anti-correlated, white for uncorrelated, red for correlated). (b) Entropy of each feature as measure for the information provided for the classification. (c) Tagging power for learning on all features (blue), the best 8 proposed by QuIPS exploiting insights from (a)+(b) (magenta), the worst 8 (yellow) and the muon tagging (red). (d) Tagging power for decreasing bond dimension truncated after training: The complete model (blue shades for $\chi = 100$, $\chi = 50$, $\chi = 5$), for using the QuIPS best 8 features only (violet shades for $\chi = 16$, $\chi = 5$), and the muon tagging (red).

decomposition, that is, decomposing $|\psi\rangle$ into two bipartitions $|\psi_\alpha^A\rangle$ and $|\psi_\alpha^B\rangle$ [46] such that

$$\Psi = \sum_{\alpha}^{\chi} \lambda_{\alpha} |\Psi_{\alpha}^A\rangle \otimes |\Psi_{\alpha}^B\rangle,$$

where λ_{α} are the Schmidt-coefficients (non-zero, normalised singular values of the decomposition). The entanglement entropy is then defined as $S = -\sum_{\alpha} \lambda_{\alpha}^2 \ln \lambda_{\alpha}^2$. Consequently, the minimal entropy $S = 0$ is obtained only if we have one single non-zero singular value $\lambda_1 = 1$. In this case, we can completely separate the two bipartitions as they share no information. On the contrary, higher S mean that information is shared among the bipartitions.

In the Machine Learning context, the entropy can be interpreted as follows: If the features in one bipartition provide no valuable information for the classification task, the entropy will be zero. On the other hand, S increases the more information between the two bi-

partitions are exploited. This analysis can be used to optimize the learning procedure: whenever $S = 0$, the feature can be discarded with no loss of information for the classification. Thereby, a new model with fewer features and fewer tensors can be introduced. The new more efficient model results in the same predictions in less time. On the contrary, whenever a bipartition entropy is high, highlights which features - or combination of features - are important for the correct predictions. In conclusion, if the entropy of a feature bipartition is low, we can discard one of them providing negligible loss of information. Moreover, if the bipartition entropy is significantly large, features can be reordered for a better representation of the classifying wavefunction. Finally, if two features are completely (anti-)correlated we can neglect at least one.

Driven by the previous analysis, we introduce the *Quantum Information Post-learning feature Selection* (QuIPS) algorithm, which combines the insights of both of these measurements - correlations and entropy - to rank the input features according to their importance for the classification (see App. B). Employing QuIPS, we discarded half of the features by selecting the 8 most important ones: i.-iii. charge, momenta, and distance of the muon; iv.-vi. charge, momenta, and distance of the kaon; vii. charge of the pion; viii. total detected charge. To test the QuIPS performance, we compared it with an independent but more time-expensive analysis on the importance of the different particle types (see App. G): the two approaches perfectly matched. Finally, we studied two new models, one composed of the 8 most important features proposed by the QuIPS, and, for comparison, another with the 8 discarded features. In Fig. 4c we show the tagging power for the different analyses with the complete 16-sites (model M_{16}), the best 8 (B_8), the worst 8 (W_8) and the muon tagging. Remarkably, we see that the models M_{16} and B_8 give comparable results, while model W_8 results are even worse than the classical approach. These performances are confirmed by the prediction accuracy of the different models: While we only lose less than 1% of accuracy from M_{16} to B_8 , the accuracy of the model W_8 drastically drops to around 52% - that is, almost random predictions. Finally, in this particular run, the model B_8 has been trained 4.7 times faster with respect to model M_{16} and predicts 5.5 times faster as well (The actual speed-up highly depends on the bond-dimension and other hyperparameters, see App. D for details).

A critical point of interest in high energy physics applications is the prediction time. Indeed, short prediction times are necessary to perform real-time event selection. In the LHCb Run 2 data-taking, the high-level software trigger takes a decision approximately every $1 \mu\text{s}$ [34] and higher rates are expected in future Runs. Consequently, we aim to exploit the QuIPS to efficiently reduce the prediction computational time while maintaining a comparable high prediction power. Another step we can undertake to reduce the prediction time is to reduce the bond dimension χ after the training procedure. Here,

we introduce the *Quantum information Adaptive Network Optimization* (QIANO) performing this truncation in a way ensuring to introduce the least infidelity possible (see App. C). In other words, QIANO can adjust the bond dimension χ to achieve a targeted prediction time while keeping the prediction accuracy reasonably high. We stress that this can be done without relearning a new model, as it would be the case with NN.

Finally, we apply QuIPS and QIANO to reduce the information in the TTN in an optimal way for a targeted balance between prediction time and accuracy. In Fig. 4d we show the tagging power taking the original TTN and truncate it to different bond-dimensions χ . We can see, that even though we compress quite heavily, the overall tagging power does not change significantly. In fact, we only drop about 0.03% in the overall prediction accuracy, while at the same time improving the average prediction time from $345 \mu\text{s}$ to $37 \mu\text{s}$. Applying the same idea to the model B_8 we can reduce the average prediction time efficiently down to $19 \mu\text{s}$ (see App. D for more details), compatible to current real-time classification rate.

V. CONCLUSIONS

We analysed an LHCb dataset for the classification of b - and \bar{b} -jets with two different ML approaches, a DNN and a TTN. We showed that we obtained with both techniques a tagging power about one order of magnitude higher than the classical muon tagging approach, which up to date is the best-published result for this classification problem. We pointed out that, even though both approaches result in similar tagging power, they treat the data very differently. In particular, TTN efficiently recognises the importance of the presence of the muon as a strong predictor for the jet classification.

We further explained the crucial benefits of the TTN approach over the DNNs, namely (i) the ability of efficiently measuring correlations and the entanglement entropy, and (ii) the power of compressing the network while keeping a high amount of information (to some extent even lossless compression). We showed how the former quantum-inspired measurements help to set up a more efficient ML model: in particular, by introducing an information-based heuristic technique, we can establish the importance of single features based on the information captured within the trained TTN classifier only. Using this insight, we introduced the QuIPS, which can significantly reduce the model complexity by discarding the least-important features maintaining high prediction accuracy. This selection of features based on their informational importance for the trained classifier is one major advantage of TNs targeting to efficiently decrease training and prediction time. Regarding the latter benefit of the TTN, we introduced the QIANO, with which once we learned a TTN, we can decrease its prediction time by optimally decreasing its representative power based on information from the quantum entropy, ensuring that

each truncation introduces the least infidelity possible. In contrast to DNNs, with the QIANO we do not need to set up a new model and train it from scratch, but we can optimise the network post-learning adaptively to the used CPU and required prediction time of the final application.

Finally, given the importance of prediction time in the LHCb experiment, we showed that using QuIPS and QIANO we can efficiently compress the trained TTN to target a given prediction time. In particular, we decreased our prediction times from $345\mu s$ to $19\mu s$. Finally, while we used only one CPU for the predictions, by parallelising the tensor contractions on GPUs one can obtain a speed-up from 10 to 100 times [47]. Thus, we are confident that it is possible to reach the MHz prediction rate while still obtaining results significantly better than the classical muon tagging approach.

Further applications of our approach in the LHCb experiment is the discrimination between b -jets, c -jets and light flavour jets, which was already tackled by a Machine Learning approach using Boosted Decision Tree classifiers [48]. A fast and efficient real-time identification of b - and c -jets can be the key point for several studies in high energy physics, ranging from the search for the rare Higgs boson decay in two c -quarks, up to the search for new particles decaying in a pair of heavy-flavour quarks ($b\bar{b}$ or $c\bar{c}$). Given the optimal performance of the presented method, we envisage a multitude of possible future applications in high-energy experiments at CERN and in other fields of science.

VI. DATA AVAILABILITY

This paper is based on data obtained by the LHCb experiment, but is analyzed independently, and has not been reviewed by the LHCb collaboration. The data are available in the official LHCb open data repository [32, 33].

VII. ACKNOWLEDGMENTS

We are very grateful to Konstantin Schmitz for valuable comments and discussions on the Machine Learning comparison. We thank Miles Stoudenmire for fruitful discussions on the implementation of the Tensor Networks Machine Learning code.

This work is partially supported by the Italian PRIN 2017 and Fondazione CARIPARO, the Horizon 2020 research and innovation programme under grant agreement No 817482 (Quantum Flagship - PASQuaS), the QuantERA projects QTFLAG and QuantHEP, and the DFG project TWITTER. We acknowledge computational resources by CINECA, the Cloud Veneto and by the BwU-niCluster.

We acknowledge the LHCb Collaboration for the valuable help and the Istituto Nazionale di Fisica Nucleare

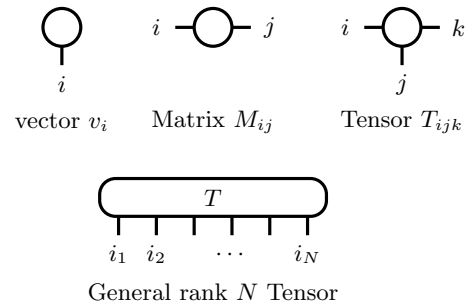


FIG. 5: Graphical representation of tensors from a Tensor Networks. Top (from left to right) a vector, a matrix and a rank-3 tensor. Bottom: A general rank- N tensor at the bottom.

and the Department of Physics and Astronomy of the University of Padova for the support.

Appendix A: Tree Tensor Networks

Tensor Networks (TNs) have been developed for decades to investigate quantum many-body systems on classical computers. They provide an efficient representation of a quantum wavefunction $|\psi\rangle$ in a compact form and thereby, they have proven to be an essential tool for a broad range of applications [17–21, 23, 24]. In a mathematical context, a TN approximates a high-order tensor by a set of low-order tensors that are contracted in a particular underlying geometry and have common roots with other decompositions, such as the Singular Value Decomposition (SVD) or Tucker decomposition [49]. The accuracy of the TN approximation can be controlled with the so-called *bond-dimension* χ , an auxiliary dimension for the indices of the connected local tensors. Among others, some of the most successful TN representations are the Matrix Product State (MPS) - or Tensor Trains [18, 27, 50, 51], the Tree Tensor Network (TTN) - or Hierarchical Tucker decomposition [30, 52, 53], and the Projected Entangled Pair States (PEPS) [54, 55].

In the following, we briefly describe the main principle of Tensor Networks and the concepts we refer to within the paper or later on in the appendix. For a more detailed insight into Tensor Networks, we refer to more comprehensive reviews and text books [18, 23, 25, 28, 55].

1. Graphical representation of Tensor Networks

Within the original Tensor Network development and applications in physics, a graphical representation of the underlying mathematical tensor notation has been established for the sake of compactness. In a nutshell, we represent TNs with circles - for the tensors - and lines

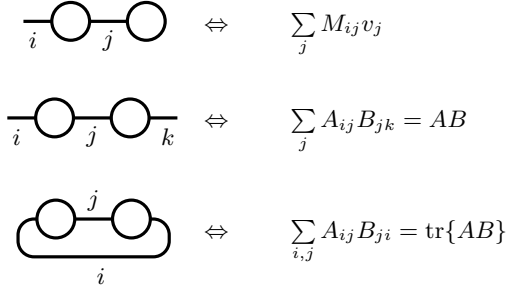


FIG. 6: Graphical representation of Tensor contractions: A vector-matrix multiplication, a matrix-matrix-multiplication and the trace of the result of a matrix-matrix-multiplication (from top to bottom).

connecting the different tensors. Each line, as well referred to as link, indicates a contraction of the two connected tensors over a coinciding index. Fig. 5 shows the graphical representation of Tensors with different ranks: A vector, a matrix, and a rank-3 tensor from left to right at the top, and a general rank-N tensor at the bottom.

Within TN algorithms, the tensors are constantly manipulated. The most important operations thereby are the contraction of two tensors, reshaping a tensor and performing a factorisation. The contraction of two tensors generalises the linear algebra well-known matrix-matrix-multiplication to tensors with arbitrary rank. As matrices are a special form of tensors, the matrix multiplication can be identified in terms of TN as a contraction of two rank-two tensors over one coinciding link j (see Fig. 6). Generalising this statement for arbitrary tensors A and B , the tensor contraction can be performed over several coinciding links m_1, \dots, m_μ . The resulting tensor $C = AB$ given by the summation over all indices m for coinciding links as follows.

$$C_{l,n} = \sum_m A_{l,m} B_{m,n} ,$$

with $l = (l_1, \dots, l_\lambda), m = (m_1, \dots, m_\mu)$
and $n = (n_1, \dots, n_\nu)$

Therefore, the links $l_k \in \{l_1, \dots, l_\lambda\}$ and $n_k \in \{n_1, \dots, n_\nu\}$ are the remaining indices after the contraction.

For a general TN, a link connecting two tensors always indicates a contraction of both tensors. The algorithmic complexity of such a tensor contraction scales with the dimension of all involved links to $\mathcal{O}(d_{l_1} \dots d_{l_\lambda} d_{m_1} \dots d_{m_\mu} d_{n_1} \dots d_{n_\nu})$ (although the scaling can be reduced, when carried out as optimised matrix-matrix multiplication). Due to this complexity, the contractions play a crucial role in the efficiency of algorithms for Tensor Networks.

2. Tree Tensor Network representation

In its original idea, the TTN represents an arbitrary pure quantum state $|\psi\rangle$ as a decomposition of the complete exponentially large tensor Ψ . The corresponding separable Hilbert space of the system $\mathcal{H} = \mathcal{H}_1 \otimes \dots \otimes \mathcal{H}_L$ consists of L local subspaces \mathcal{H}_i , where each local state space \mathcal{H}_k shall be d -dimensional. The most general pure state in such a system can be written

$$|\psi\rangle = \sum_{i_1, \dots, i_L=1}^d c_{i_1, \dots, i_L} |i_1\rangle \otimes |i_2\rangle \otimes \dots \otimes |i_L\rangle , \quad (\text{A1})$$

where i_k describes the local state space of the site k . This complete representation seeks for d^L coefficients c_{i_1, \dots, i_L} describing each possible combination of the local states. These coefficients can be recast in a tensor of rank L , where each leg i_k of this tensor corresponds to a local Hilbert space \mathcal{H}_k . The normalisation $\sum_{i_1, \dots, i_L} |c_{i_1, \dots, i_L}|^2 = 1$ of the state $|\psi\rangle$ thereby coincides with the Frobenius norm $\|\Psi\|_F$ of the rank- L tensor. The TTN further decomposes this rank- L tensor c_{i_1, \dots, i_L} into a set of hierarchically connected rank-3 tensors.

3. Machine Learning with Tree Tensor Network

Even though DNNs have been highly developed in recent decades by industry and research, the first approaches of Machine Learning with TN yield already to comparable results when applied to standardised datasets [13, 27, 56]. In particular, we implemented a TTN as a classifier for supervised learning problems. In this section, we give insights into the TTN Machine Learning algorithm.

As for a general supervised learning problem, the data samples are given as input vectors \mathbf{x} . Each sample \mathbf{x} is encoded into a higher dimensional feature space by a *feature map* $\Phi(\mathbf{x})$, and subsequently classified with the *decision function* $f(\mathbf{x})$.

$$f(\mathbf{x}) = W \cdot \Phi(\mathbf{x}) . \quad (\text{A2})$$

In general, the complete *weight tensor* W can be used as a classifier, however, this tensor W becomes exponentially large with increasing numbers of features given in the dataset. Therefore, we represent W as a quantum-inspired Tensor Network, in particular a Tree Tensor Network, building on the idea proposed for an MPS in [27].

A TN with N sites addresses a global space $\mathcal{H} = \otimes_{j=1}^N \mathcal{H}_j$ spanned by a tensor product of N local subspaces \mathcal{H}_j . Each subspace \mathcal{H}_j can in general have a different dimension $\dim \mathcal{H}_j = d_j$. For the application in Machine Learning, a natural feature map suited for a TN is a product of N *local feature maps* Φ^{s_j} where

$j \in \{1, 2, \dots, N\}$ and $s_j \in \{1, 2, \dots, d_j\}$. All N local feature maps together determine the global feature map

$$\Phi(x) = \Phi^{s_1}(x_1)\Phi^{s_2}(x_2)\cdots\Phi^{s_N}(x_N). \quad (\text{A3})$$

To point out the connection with quantum mechanics in this Machine Learning ansatz, we can describe the TTN classifier as a set of quantum many-body wavefunctions $W^l = |\psi_l\rangle$ - one for each of the class label l . Consequently, when we predict a sample x we calculate its overlap $\langle\Phi(x)|\psi_l\rangle$ for all labels l with the product state $\Phi(x)$ given by a global feature map. The final prediction output for each class is then given by the weighted probabilities

$$\mathcal{P}_l = \frac{|\langle\Phi(x)|\psi_l\rangle|^2}{\sum_l |\langle\Phi(x)|\psi_l\rangle|^2}. \quad (\text{A4})$$

For the identification of jets, the features x_i are the detected physical observables from the LHCb simulation described in Section I. We rescale each of the features x_i to $x'_i = x_i/x_{i,\max} \in [0, 1]$ with respect to the corresponding maximum value $x_{i,\max}$ of all samples within the complete training set. We encode the rescaled features x'_i - following the inspiration of quantum spins - by choosing the local feature map

$$\Phi^{s_i}(x_i) = \left[\cos\left(\frac{\pi x'_i}{2}\right), \sin\left(\frac{\pi x'_i}{2}\right) \right]. \quad (\text{A5})$$

In this way, we can think about each single feature x_i being represented by a quantum spin (where $x_i = 0$ is mapped to a spin down and $x_i = x_{i,\max}$ to a spin up). Accordingly, each sample x is mapped to the product state $\Phi(x)$. After the transformation, the i -th feature is addressed by the i -th site of the TTN. In general, we can exploit different, more expressive feature maps then the chosen one of Eq. A5 (see App. A 4).

For the learning procedure of the TN, we aim to minimise the quadratic cost function

$$C = \frac{1}{2} \sum_{n=1}^{N_T} \sum_l (f_l(x_n) - \delta_{l,L_n})^2,$$

where the index n runs over all N_T training samples and δ_{l,L_n} is a Kronecker delta with L_n being the correct label for the n -th sample. Thus $\delta_{l,L_n} = 1$, if the label l equals the known label L_n for the supervised learning. We optimise the complete network by subsequently performing a gradient descent on local tensors until the cost function C converges. We sweep through the network from the bottom to the top, so that after one sweep every tensor has been optimised once, concluding one learning iteration. In contrast to Ref. [27], we keep the label l fixed at the top tensor and optimise each tensor separately rather than optimising in the space of two tensors at once.

Furthermore, we initialise the TTN by performing the unsupervised learning proposed in Ref. [13] up to the topmost layer in the tree and adding a random tensor on top connecting the remaining two bipartitions with the label l for the classification. We start with optimising the random top-tensor via conjugate gradient descent and afterward start iteratively sweeping through the network from the bottom to top.

4. Higher-Dimensional Local Feature Maps

In Eq. (A5), we presented the local feature map as a 2-dimensional vector inspired by the quantum spin representation. In general, we are not restricted to this feature map, as the different samples x can be mapped by using more expressive feature maps, e.g., taking polynomial orders (e.g. $\Phi_i(x) = [1, x_i, x_i^2]$) or higher order spherical feature maps defined as

$$\Phi_d^{s_j}(x_j) = \sqrt{\frac{d-1}{s_j-1}} (\cos(\frac{\pi}{2}x_j))^{d-s_j} (\sin(\frac{\pi}{2}x_j))^{s_j-1}.$$

We analysed the data with different orders of the spherical feature map and presented in Sec. III the results obtained by the 5-th order map $\Phi_{d=5}^{s_j}$, as this order lead to the best prediction accuracy. Anyhow, the different feature maps all result in similar prediction accuracies in the end and the fundamental insights we obtained did not change. As an example, for the 2-dimensional feature map $\Phi_{d=2}^{s_j}$ we obtained an accuracy of $a = 70.34\%$ in contrast to $a = 70.56\%$ for $\Phi_{d=5}^{s_j}$ (both after applying the cuts).

In Sec. IV we used the 2-dimensional feature map $\Phi_{d=2}^{s_j}$ for the insights into the TTN by measuring the correlations and entropy. We stress that the operators used to measure correlations have to be adapted to the local Hilbert space as well. For spherical feature maps, we can exploit the Pauli-Matrices for $d = 2$, the Gell-matrices for $d = 3$, or higher representations of the $SU(d)$ Lie group in order to investigate correlations in the classification.

5. Isometrisation

Here, we restrict ourselves to rank-3 tensors for the sake of compactness and since the Tree TN is composed out of rank-3 tensors only. The generalisation to rank- N tensors is straightforward.

A tensor $T^{[k_1, k_2, k_3]}$ of the TTN is *isometrised* with respect to the links k_1, k_2 if it is a unitary matrix when combining the two indices k_1 and k_2 , that is it obeys the isometry condition

$$\sum_{k_1, k_2} (T)_{k_1, k_2}^{k_3} (T^\dagger)_{k'_3}^{k_1, k_2} = \delta_{k_3, k'_3}. \quad (\text{A6})$$

Hence, one isometrised tensor $T^{[k_1, k_2, k_3]}$ performs a unitary transformation on two subspaces (k_1, k_2) . We can isometrise an arbitrary TTN by iteratively performing QR-decompositions [23] on each tensor of the tree from the bottom to the top. In particular, when going from the bottom to the top, we set the unitary Q -tensor as the original tensor T and contract the R -tensor upwards with the connected tensor over the link k_3 . This procedure results in all tensors being isometrised, except for the upmost one. The TTN is then *isometrised* towards the up-most tensor. In the same manner, we can isometrise the TTN as well to different tensors within the network.

After we train a TTN in the Machine Learning application, we isometrise the complete TTN towards the upmost tensor for the predictions. Consequently, the prediction is a *real space renormalisation* of two neighboring sites for each layer within the tree [25, 57]. Each tensor simply performs a unitary transformation together with a truncation of two sites originating from the input sample. Consequently, when assessing the tensor entries we know exactly how the data will be processed for the general prediction.

6. Schmidt-decomposition

In a loop-free TN state $|\psi\rangle$ every link ν between two tensors in the network bipartites the underlying system \mathcal{L} into the subsystems \mathcal{A}^ν and \mathcal{B}^ν . This allows rewriting the TN at every link ν in terms of the Schmidt-decomposition

$$|\psi\rangle = \sum_{\alpha=1}^{\chi_\nu} \lambda_\alpha^{[A, B]} |\psi^{[A]}\rangle \otimes |\psi^{[B]}\rangle \quad (\text{A7})$$

with χ_ν being the bond dimension of the link ν and $\lambda_\alpha^{[A, B]}$ the Schmidt-coefficients (or non-zero, normalised singular values of the decomposition). Thus, at each bipartition, the bond dimension χ_ν of the link ν provides an upper bound for the Schmidt rank and consequently for the bipartite entanglement the TN is able to capture (compare App. B 2). Each of the two Schmidt vector sets forms an orthonormal basis for the associated subspaces [46]. In practice, we exploit the isometry condition for the tensors within the TTN by isometrising to one of the tensors attached to the link ν of the desired bipartition. The Schmidt-values then correspond to all non-zero singular values of an SVD decomposition with respect to attached to the link ν on the tensor the TTN is isometrised towards.

As we will see in the next section, this Schmidt-decomposition will allow us to calculate the information encoded in the TNs and based on this information to efficiently reduce the complexity of the Machine Learning model.

Appendix B: Quantum Information Post-learning feature Selection (QuIPS)

In this section, we introduce the Quantum Information Post-learning feature Selection (QuIPS), a protocol that exploits the information encoded in the TTN to reduce the input features in the model to the most valuable ones for the classification process. In particular, the quantum-inspired measurements of correlations and entropy are used to determine the importance of the different input features after the learning procedure based on the information they provide for the classification. Finally, the QuIPS allows us to rank all the input features according to their importance and to use this ranking to efficiently reduce the model by discarding the least important features.

In the following we first describe the two exploited quantum-inspired measurements, the correlations and the *von Neumann entropy*, and finally give an algorithmic protocol for the QuIPS.

1. Correlation measurements

We can measure the correlations captured within the TTN classifier by exploiting the quantum-correlation measurements. As we chose the local map to represent the input features in quantum spins, we will measure the correlations $C_{i,j}$ in the σ_z basis for each pair of features (located at site i and j), defined as follows:

$$C_{i,j}^l = \frac{\langle \psi_l | \sigma_i^z \sigma_j^z | \psi_l \rangle}{\langle \psi_l | \psi_l \rangle}$$

The correlation results in $C_{i,j}^l = 1$ if the TTN recognizes that the two local features i and j are completely correlated - such that the rescaled input of $x_i \in [0, 1]$ always equals the rescaled input $x_j = x_i$. We obtain $C_{i,j}^l = -1$ for the two local features being completely anti-correlated - such that $x_i = 1 - x_j$. Finally, we obtain $C_{i,j}^l = 0$ in case of the two local features being completely uncorrelated. Thus in case of no correlation, we know that the two features may provide fundamentally different information for the classification. In any way, we can not tell if this information given by the two features is actually important for the classification itself. On the contrary, in the case of complete (anti-)correlation, the two features provide the same information and we can drop one of them in further analysis. As an example, we can take a look at learning from pictures, where the first two pixels are always white in the complete data. Thus, we will measure that both pixels are totally correlated and we know we can discard at least one of them as we can always reconstruct the information from the other one. But, in this case, both pixels would give us no valuable information for the actual classification problem, and we could even go further and discard both. Measuring the correlation only, we have no insight into this information.

Anyway, when we take the entropy measurement in the subsequent section into account, we can measure the information provided for the classification and efficiently discard both pixels in this scenario.

This idea for the correlation measurement can be extended to the use of different local feature maps (see App. A 4) by using different operators as correlators. We mention as well, that this correlation measurement is purely based on the information within the TTN captured after the learning procedure.

In principle, we can further measure correlations or local expectation values in the σ_x -basis. This can help to find further correlations within the data and can give insights on choosing the local feature map. If we find higher correlations in this basis for certain features, it might be interesting to actually change the input basis of the local feature map from a spin in σ_z to the σ_x -basis.

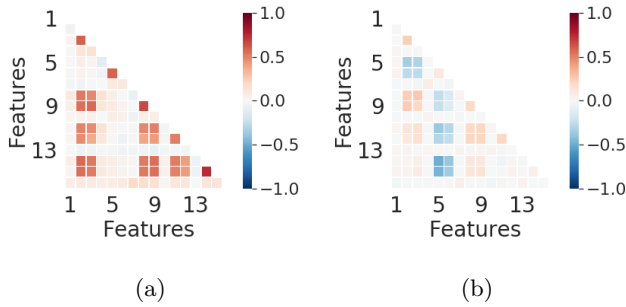


FIG. 7: Correlations for the \bar{b} correlations (a) and correlations measured for the cross-classification $l = b, l' = \bar{b}$ (b)

In the paper, we presented the correlations for the b -quark classification. In Fig. 7 (a) we show the correlations as well for the \bar{b} -quark for sake of completeness. As mentioned before, both cases are very similar and only differ slightly in the magnitude of the single correlations.

We can further generalise the correlation measurement and compute the cross-correlations of the two feature spaces i and j between two different classes l and l'

$$C_{i,j}^{l,l'} = \frac{\langle \psi_l | \sigma_i^z \sigma_j^z | \psi_{l'} \rangle}{\sqrt{\langle \psi_l | \psi_l \rangle \langle \psi_{l'} | \psi_{l'} \rangle}}.$$

2. Entropy measurements

Within the TTN, we have also access to the entanglement entropy. This expresses the correlations within two general bipartitions of the whole system. To compute it, we bring the state $|\psi\rangle$ represented by the TTN into an orthogonal form with the Schmidt decomposition (see App. A 6). The von Neumann entropy is then defined by the Schmidt-coefficients as $S = -\sum_{\alpha} \lambda_{\alpha}^2 \ln \lambda_{\alpha}^2$. Consequently, the minimal entropy $S = 0$ is obtained only if we have one single non-zero singular value of $\lambda_1 = 1$.

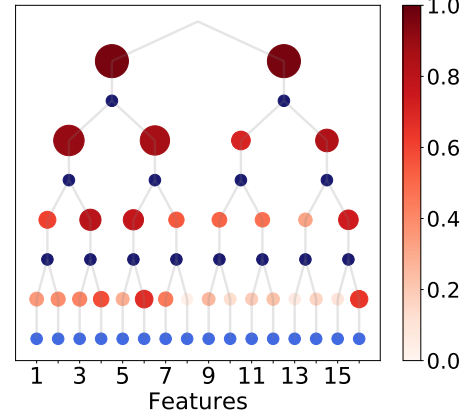


FIG. 8: Normalized entropy measurements at all links ν within the tree for the LHCb classification problem (magnitude of S indicated by darkness and size of the red circles). The lowest layer corresponds to the information of the single input features, while higher layers correspond to the mutual information of the input features connected from below.

In this case, we can completely separate the two bipartitions as they share no information. This idea can be interpreted the Machine Learning context: If the features in one bipartition provide no valuable information for distinguishing between the different classes, the entropy is zero. On the other hand, the entropy increases the more information between the two bipartitions is used for the classification. This criterion can be used to optimize the learning procedure: In the first scenario, we can efficiently discard the features with no - or negligible less - information and introduce a new model with fewer features and less tensor respectively. With this more compact model, we are able to obtain the same predictions while requiring fewer contractions and thereby less time. The second scenario, where the entropy is high, helps us to understand which features - and more general which combination of features - are critically important for the correct predictions.

In fact, we can further exploit the mutual information

$$I_{ik} = S_i + S_k - S_{ik}$$

of two different features i and k if they are attached to the same tensor by measuring the entanglement entropy on a higher layer within the tree as illustrated in Fig. 8 as an example of the $b\bar{b}$ -classification. Here, S_i (S_k) is the corresponding entropy for the two features i (k) and S_{ik} the entropy of the combined features. If we see, that e.g. two features provide the same entropy $S_1 = S_2$, and additionally the coarse-grained space consisting of both features provides the exact same entropy $S_{12} = S_1 = S_2$, we know that the information we obtain from the two features is equivalent. Thus, in this case, we can neglect one of the two (in agreement with the correlation

measurement). The same idea can be extended to the mutual information of different clusters of input features and opens a very promising direction for a deeper understanding of information captured within the TTN.

3. QuIPS Protocol

Both above-mentioned measurements, the correlations together with the entanglement entropy, leave us with the following insights and receipt for increasing the model efficiency using QuIPS:

- If two learned features are completely correlated we can neglect at least one of them without loss of classification accuracy.
- If the entropy for a bipartition (set of features) is low, we can discard all the features.
- If the entropy is significantly large, we may reorder the features for a better representation of the classifying wavefunction.

Therefore, the QuIPS can be described by the following algorithmic idea for setting up the new model with a targeted number of total features:

- i) Add the feature i with highest entropy S_i to the new model
- ii) **while** (`numberOfFeatures < targetNumber`)
 - take next feature j with highest entropy S_j
 - if** ($\forall |C_{i,j}| < 1 - \epsilon \mid \forall i \in \text{new model}$)
 - add feature j
 - end if**
- end while**

Resuming, the QuIPS offers valuable insights into the learned data, in particular, it provides an information-based heuristic for the importance of the single features based on the information within the trained TTN. Thereby, we can significantly reduce the model complexity by discarding the least-important features while still maintaining high prediction accuracy. As an outlook on further investigation in this direction, we can include the mutual information into the heuristic and put more value as well on the ordering of the features. As the TTN classifier is breaking the symmetry of the position of input samples, this ordering of features can lead to critical gains in the performance as well.

Furthermore, an interesting approach is to exploit different metrics for measuring and describing the captured information. For instance, we might use the Kullback-Leibler Divergence [58], which is a more prominent measurement in Machine Learning.

Appendix C: Quantum Information Adaptive Network Optimization (QIANO)

We introduce the Quantum Information Adaptive Network Optimization (QIANO) which performs a reduction of free parameters in the network in a way that ensures to keep the highest amount of information possible. In other words with QIANO we can adjust the bond dimension χ of the TTN classifier targeting a certain prediction time while controlling the prediction accuracy to stay reasonably high. Moreover, this adjustment can be done without the need of relearning a new model, as it would be the case with neural networks. The underlying procedure for this truncation of a TTN is the Singular Value Decomposition (SVD) which will be applied to the tensors within the network.

1. Singular Value Decomposition

Every complex matrix $M \in \mathbb{C}^{m \times n}$ can be decomposed into a matrix $U \in \mathbb{C}^{m \times \min\{m,n\}}$ featuring orthonormal columns, a diagonal matrix $S \in \mathbb{R}^{\min\{m,n\} \times \min\{m,n\}}$ and a matrix $V^\dagger \in \mathbb{C}^{\min\{m,n\} \times n}$ with orthonormal rows such that

$$M = USV^\dagger.$$

The orthonormal columns of U - also referred to as left singular vectors of M - obey $U^\dagger U = \mathbb{1}$. Analogously, the orthonormal rows of V^\dagger - or right singular vectors of M - entail $V^\dagger V = \mathbb{1}$. The diagonal matrix S contains the *singular values* of M (real, non-negative). The number of non-zero singular values equals the Schmidt rank of the matrix M . With a descending order of the singular values, the matrix S is unique for given M , which is in general not the case for U and V .

The SVD provides the best possible approximation of a matrix M by a matrix \tilde{M} with lower rank $\tilde{r} < r$ with respect to its Frobenius norm $\|M\|_F^2 := \sum_{ij} |M_{ij}|^2 = \text{tr}\{MM^\dagger\}$. Indeed, performing the SVD on M leads to

$$\|M\|_F^2 = \text{tr}\{USV^\dagger VS^\dagger U^\dagger\} = \text{tr}\{SS^\dagger\} = \sum_{i=1}^r \sigma_i^2,$$

with r denoting the Schmidt rank of the matrix M . Thus the squared Frobenius Norm equals the sum of the squared non-zero singular values σ_i . Consequently, the error in $\|M\|_F$ made by the approximation \tilde{M} is minimal for taking the highest \tilde{r} singular values into account, together with their corresponding singular vectors in U and V^\dagger . If the singular values σ_i are arranged in descending order, the error $\epsilon_{\tilde{r}}$ in the Frobenius norm can be calculated by the discarded values σ_k [18].

$$\|\tilde{M}\|_F^2 = \sum_{i=1}^{\tilde{r}} \sigma_i^2 = \|M\|_F^2 - \underbrace{\sum_{k=\tilde{r}+1}^r \sigma_k^2}_{=\epsilon_{\tilde{r}}}$$

The SVD is generalised straightforward to the general tensor algebra by splitting the tensor with respect to two different sets of its links.

2. TTN truncation

When we truncate one link in the TTN to a lower bond dimension χ , we can isometrise the network forming the Schmidt-decomposition by an SVD on a certain tensor. We truncate it by throwing away the smallest singular values. The error made within this truncation can be estimated accurately since the induced state fidelity explicitly depends on the sum of the squared discharged singular values. We can proceed by truncating all the links in the network in the same manner reducing the sizes of the tensors and the total space of the TTN representation.

In this process of truncation, we stress again that the local truncation of the links ensures the mathematically best possible approximation within the lower subspace regarding the total fidelity of the TTN state. Anyhow, for the global truncation we point out that as we iterate threw the complete network with local truncations, the ordering of the local truncation may play a role in the final approximation. Even though, this approach has proven to be extremely efficient (see application on the LHCb data in App. D), the global truncation might be performed in an even further optimised way.

3. QIANO Protocol

The QIANO implements a truncation of the TTN as mentioned above after the training procedure which thereby reduces the free parameters in the network while introducing the least possible infidelity for each truncation step. In other words, one adjusts the bond dimension χ targeting a certain prediction time $T_{\text{pred}}(\chi)$ while keeping the critical amount of information captured by the TTN - and thereby a high prediction accuracy - within the smaller subspace we represent our quantum wave function classifier in. A prediction with the truncated TTN equals to perform a set of contractions - or vector-matrix-multiplications - on lower dimensional tensors and thus can be executed in lower computational time. Performing a QIANO, we represent the TTN classifier more compactly based on the quantum-information captured within the TN ansatz, resulting in a more efficient classifier concerning the prediction time versus its prediction accuracy. We can perform this reduction of free parameters without the need of relearning a new model, as it

would be the case with neural networks when we target a certain prediction time. We train the TTN once with a maximum bond dimension χ_{max} and then truncate it depending on the CPU architecture to the dimension $\chi' < \chi_{\text{max}}$ in order to obtain the targeted T_{pred} .

This way of reducing the information in the TTN for the sake prediction time maintaining an optimal balance between prediction time and accuracy is of extreme value in a broad range of Machine Learning applications: The TTN offers the flexibility to adjust the prediction time depending on the requirement and the architecture of the computational system. For the analysis on the CERN data we provide in App. D a deeper insight into the actual speed-up, including the measured prediction times for different bond dimension χ (see, Tab. I).

4. Equivalence with DNNs

It has been shown that artificial NNs can be mapped into TNs [15, 56, 59, 60]. Following the idea of the graph-based mapping, the post-learning reduction of free parameters by truncating the TTN is equivalent to reducing the number of free parameters in the DNN in the way to optimally preserve the amount of represented information within the TTN. Thus, this would not only include the dropout of different weights and neurons in the DNN, but furthermore to some extend restructuring the neuronal connections of the DNN resulting in a different model. With the TTN this post-learning reduction of free parameters can be done without the need of relearning a new model, as it would in general be necessary for DNNs.

Appendix D: Efficient Prediction time speed-up with QuIPS and QIANO

In the following, we present the application of the QuIPS and QIANO protocol on the LHCb problem in more detail.

1. Speed-up for CERN data

Here, we consider both approaches to analyse the speed-up: Discharting features with QuIPS and optimizing the representation power of the TTN with QIANO. In Tab. I we show the different results for the complete 16 features and the best 8 features, both varying the bond dimension χ . Interestingly, when we truncate the TTN classifier from high bond-dimension we lose no prediction accuracy. In fact, we actually increase the accuracy in the test set, which might be due to the tendency to overfitting with increasing χ . We notice a bigger drop in the prediction accuracy only when we truncate down to $\chi = 5$, thus gaining from $\chi = 200$ to $\chi = 10$ almost an order of magnitude in speed-up without decreasing

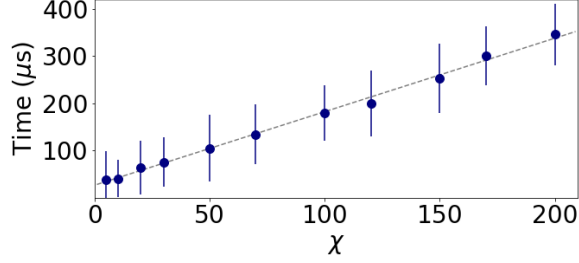


FIG. 9: CPU time for a prediction with respect to the bond dimension χ of the TTN classifier.

the precision. Interestingly, when we train from the beginning with bond dimension $\chi = 10$, we are not able to achieve the same accuracy as we get after truncating from the learned $\chi = 200$ model to $\chi = 10$.

The presented calculations were done at the CINECA Marconi Knights Landing cluster on one single processor core (Intel Xeon 8160, SkyLake at 2.10 GHz).

χ	16 feat. (μ s)	Accuracy (%)	8 feat. (μ s)	Accuracy (%)
200	345	70.27 (63.45)	-	-
100	178	70.34 (63.47)	-	-
50	105	70.26 (63.47)	-	-
20	62	70.31 (63.46)	-	-
16	-	-	19	69.10 (62.78)
10	40	70.36 (63.44)	19	69.01 (62.78)
5	37	69.84 (62.01)	19	69.05 (62.76)

TABLE I: Prediction time of the TTN for different bond dimension χ and prediction accuracy with (without) applied cuts when we reduce the TTN model with QIANO, both for the complete 16 (left) and the QuIPS reduced 8 features (right).

Furthermore, the performance of tensor contractions can be improved by a factor ranging from 10 to 100 bond-dimension depending when executed in parallel on GPUs rather than CPUs [47]. Thus combining the TTN truncation and the post-analysis feature selection, together with a CPU, GPU or FPGA architecture optimally designed for TNs we are confident that we can reach the order of prediction times required for real-time applications in the LHCb experiment.

2. Minimum CPU time

During the prediction, we can perform the contractions in each layer in parallel. So in the following calculation of a lower boundary for the CPU-time we assume a perfect parallelisation and thus only take the contractions of one tensor within each layer into account. This contraction consists of two matrix-vector multiplications; the first

with a $\chi_l \times (\chi_{l+1}\chi_l)$ -matrix and a χ_l -dimensional vector; the second with a $\chi_l \times \chi_{l+1}$ -matrix and χ_l -vector, where $\chi_l = \min\{\chi, d^l\}$ is the actual dimension of the downwards directed links in the l -th layer (the input layer is $l = 1$; the link upwards of the last layer χ_{L+1} has dimension of the classification problem).

Therefore, the number of floating-point operations NF which cannot be parallelised on a higher level, but only within the execution of the single matrix-vector multiplications, is

$$NF = \sum_{l=1}^{L+1} \chi_l^2 \chi_{l+1} + \chi_L \chi_{L+1}.$$

Thus, taking, for instance, a TTN with 8 input features and bond dimension $\chi = 5$, we can calculate the number of floating-point operations required for predicting a sample. In this case, the 4 tensors in the lowest level each address two different $\chi_1 = 2$ -dimensional local input feature spaces and merge them to a $\chi_2 = 4$ -dimensional space. Here we require $16 + 8 = 24$ FLOPS. In the next layer, we work with dimensions $\chi_2 = 4$ and $\chi_3 = 5$, resulting in $80 + 20 = 100$ FLOPS. The last contraction projects the samples onto the $\chi_4 = 1$ -dimensional output space with $25 + 5 = 30$ FLOPS. Totalling 154 FLOPS which can be computed by an ordinary CPU with a lower performance of 2 GFLOPS/s (higher performance with 6 GFLOPS/s) within 77ns (26ns).

This lower bound for the prediction time is very encouraging in high energy physics for LHCb experiments in the real-time event selection. In particular, the high-level software trigger in the Run 2 data-taking has to take a decision approximately every 1μ s [34] and higher rates in the range of 200ns are expected in future Runs. Thus with the bond-dimension $\chi = 5$ of this calculation example, we can reach these time scales and we showed in the paper that using QuIPS and QIANO, we still can obtain results about 6 – 8 times more efficient than the muon tagging for such a TTN.

Finally, let us mention that we did not consider the parallelisation of the actual vector-matrix multiplication in this lower bound for the CPU time. Furthermore, this is a completely theoretical bound neglecting, for instance, the time to copy data into the cache and any overhead from the implementation.

Appendix E: Detailed description of the DNN

The DNN networks studied here were implemented using the Keras [61] framework with the Tensorflow [62] back-end. The network was built alternating a Dropout layer after each Dense layer starting with a Batch Normalization layer. As mentioned before, we use 60% of the total data (totalling about 700k samples) for the learning procedure and 40% of it for the final testing and estimation of the performance. We further divide the learning data into training (60%) and validation set (40%).

The hyper-parameters of the network (Depth, number of nodes per layer, dropout rate, normalization moment, and kernel initialization) were tuned using the hyperopt [63] package exploring different parameter spaces in order to maximise the accuracy in the validation dataset. The ReLu activation was used for the hidden layer while a sigmoid is used for the output node. The network was trained with both ADAM and SGD optimiser optimizing their learning rate.

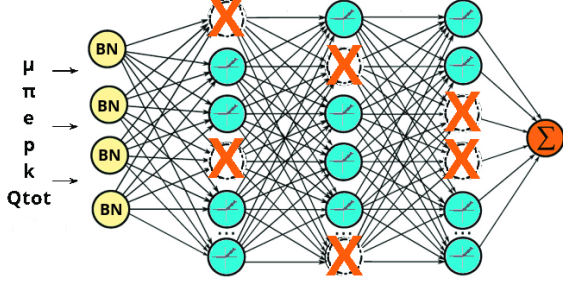


FIG. 10: Pictorial representation of the DNN: after a Batch Normalization layer there are three Dense layers of 96 nodes, followed by a Dropout layer here represented by the X nodes.

The final chosen network architecture (see Fig. 10) consists of three couples of dense plus a dropout layers with 96 nodes per layer and 0.1 dropout rate. The optimization was done with ADAM and learning rate 0.0001. The model was trained for a maximum of 250 epochs, early stopping with a patience parameter of 25 epochs on the loss in the validation set was used. The model used for evaluating the performance on the test set is the model with the best performance on the validation set.

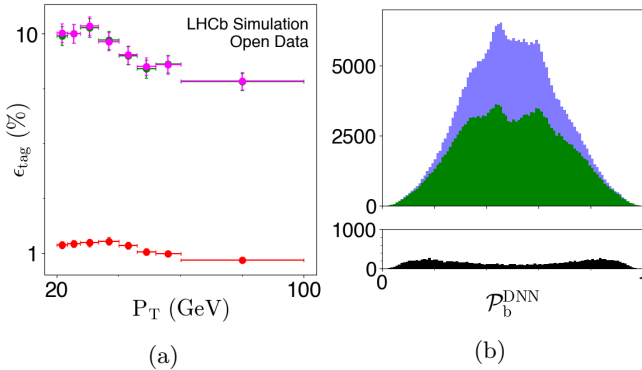


FIG. 11: DNN analysis with quadratic cost function: (a) Tagging power for the DNN trained with cross-entropy loss (green) and with MSE (magenta) and the muon tagging (red), (b) probability distribution for the DNN trained with cross-entropy loss.

Further, we investigated the use of different cost functions for the network optimisation. We performed the

DNN optimisation for the cross-entropy loss function and with the Mean Squared Error (MSE), with which the TTN is trained as well. In the end, both cost functions lead to similar results in the prediction accuracy, the tagging power and the probability distribution (see Fig. 11). Introducing the cuts in the evaluation of the tagging power, we obtain $\epsilon_{eff}^{DNN} = 58.3\%$ with $a^{DNN} = 69.92\%$ for the cross-entropy loss and $\epsilon_{eff}^{DNN} = 55.3\%$ with $a^{DNN} = 70.49\%$ for the MSE.

Appendix F: Check for biases in physical quantities

Once the performances of the TTN and DNN algorithms have been established, it is necessary to check for biases in describing the main physical quantities related to jets physics: in this way, we can probe the feasibility of these new tagging methods to perform physical analysis. Typical quantities describing jets are the transverse momentum p_T (which is the momentum perpendicular to the beam axis direction) and the pseudorapidity η (defined as $\eta = -\ln(\tan(\theta/2))$ where θ is the polar angle); some cuts are applied to these quantities: $p_T > 20$ GeV and $2.2 < \eta < 4.2$, to ensure that both jets are contained in the LHCb acceptance. Every simulated event contains two jets generated by a b - and a \bar{b} -quark, therefore labelled as b and \bar{b} . To perform this check it is sufficient to require the TTN and the DNN to classify the flavour of just one jet: once one jet's flavour has been established we are sure what the other jet's flavour is.

In Figs. 12a and 12b p_T distributions are shown while in Figs. 12c and 12d η distributions are shown for jets generated by b - and \bar{b} -quark respectively, all normalized to one. Results are shown for the TTN and DNN methods, compared to the so-called *Monte Carlo truth* (MC), which is the set of true known features of the jets resulting from the simulation process. From the plots it is clear that no evident biases are present, therefore we can conclude that not only the TTN and the DNN methods perform better than the usual muon tagging method (as seen by plotting the tagging power), but they also describe properly the physics behind the studied processes. This quick check allows us to possibly use a TTN to perform physics analyses, such as measuring the charge asymmetry in $b\bar{b}$ events.

Appendix G: Analysis with single particles

In the following, we present our study on the importance of the different particle types (which we recall to be electron, pion, proton, kaon and muon and their antiparticles respectively) for the final $b\bar{b}$ classification. In particular, we are looking at the contribution from every single particle, including all of the corresponding three features (relative momenta p_T^{rel} , charge q and relative distance ΔR), to the tagging power. This study has been performed for our TTN algorithm and provides (i) a

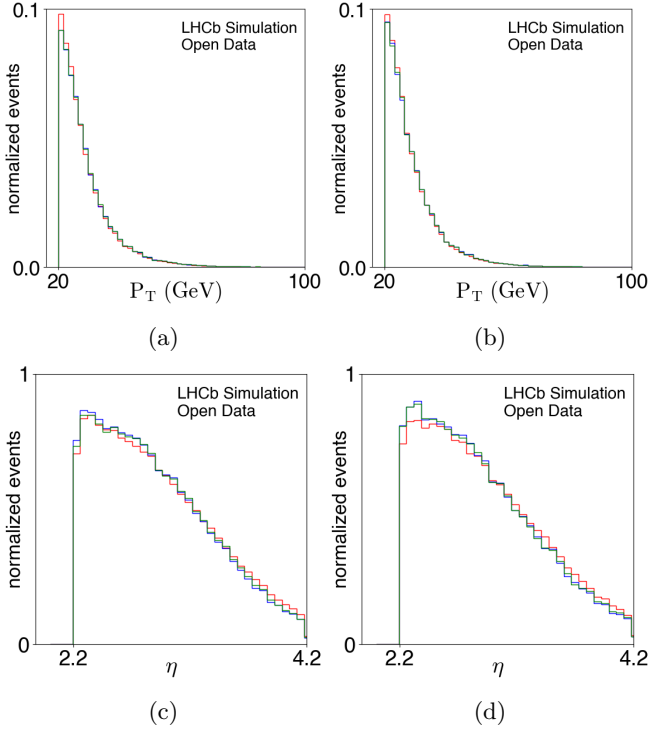


FIG. 12: Jet p_T distributions for TTN (blue) and DNN (green) compared with Monte Carlo (red) truth for b -quark (a) and \bar{b} -quark (b) and η distributions for TTN, DNN and MC for b -quark (c) and \bar{b} -quark (d).

deeper insight on the interesting features from the physical point of view, (ii) further information to validate our feature reduction using the QuIPS protocol which, combining correlation and entropy arguments, reduces the number of features considered without sensibly decreasing the final accuracy and tagging power.

In the first part of this study, we discard one particle type only in the analysis. Thus we learn using only the remaining 4 particles and the total charge, resulting in 5 different simulations corresponding to discarding each particle correspondingly. In order to deselect just the i -th particle we set all its input features $(p_T^{el})_i = 0$, $(q)_i = 0$, and $(\Delta R)_i = 0$ explicitly to zero; both in the train and in the test dataset.

In Fig. 13a the tagging power of the TTN analysis including all particles, and discarding the features of one particle type are shown. It is evident that by removing the kaon for the classification, we have a clear loss in the tagging power for the complete range of transverse momentum p_T of the complete jet. This loss is even more significant for low jet transverse momentum p_T . For other particles, a clear loss exceeding the statistical uncertainty can only be found for certain p_T , such as the pion for high p_T , or the muon for the middle range of p_T . In order to properly understand the importance of the different particles, we can further compute the tagging power by considering only the features of one particle at

a time.

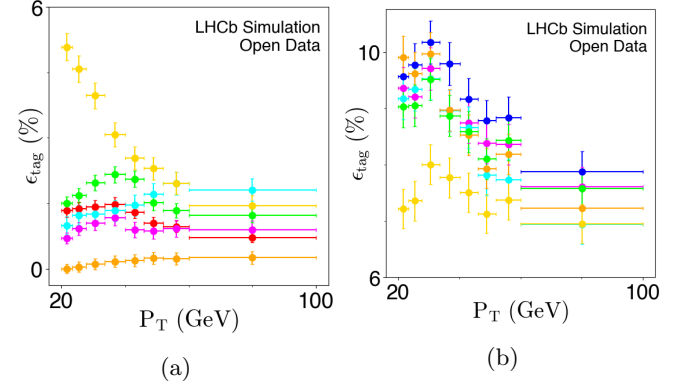


FIG. 13: Tagging power as a function of jet p_T . (a) muon tagging method (red) and TTN with just one particle's features: electron (magenta), pion (cyan), proton (orange), kaon (gold) and muon (lime) (b) TTN (blue) compared with the same network but with one particle's features removed: w/o electron (magenta), w/o pion (cyan), w/o proton (orange), w/o kaon (yellow) and w/o muon (lime).

In Fig. 13b we show the tagging power for using just one of the particle types for the classification: the contribution of the kaon at low jet p_T is evident, meaning that in order to get high values of the tagging power we need to exploit the features of the kaon. Here, we can see again, even more clearly, the contribution of the pion for high transverse momentum p_T of the jet, and of the muon for the middle range of p_T , while the features of the proton do not play an important role in the tagging power. Interestingly, the TTN analysis using only the muon (lime curve) is still more efficient than the usual muon tagging: this is due to the fact that while the muon tagging algorithm considers only the charge of the muon, the TTN also considers other features such as p_T^{el} and ΔR .

Concluding this study, we figured out that the kaon is the most crucial particle for the $b\bar{b}$ -classification followed by the muon or by the pion for very high transverse momentum p_T of the detected jet. These insights perfectly align with the quantum-information based insights provided by our QuIPS protocol: Next to the total charge of the jet Q , the QuIPS suggests for this problem to use all available features of the kaon and muon, together with one feature (the charge) of the pion when reducing the total number of features from 16 to 8.

Appendix H: Other comparisons between TTN and DNN

When investigating the performances of the TTN and DNN methods, the two algorithms gave the same performances (as their accuracy and tagging power is the same

within the statistical error) but we showed by the different shapes of the TTN and DNN confidence distributions that they exploit different kinds of information. Further, we found some correlation between the outputs of the two algorithms. Here, we further investigate this aspect of correlation in the predictions of the different methods by computing the so-called *confusion matrix*, a graphic comparison between the outputs of the two classifiers.

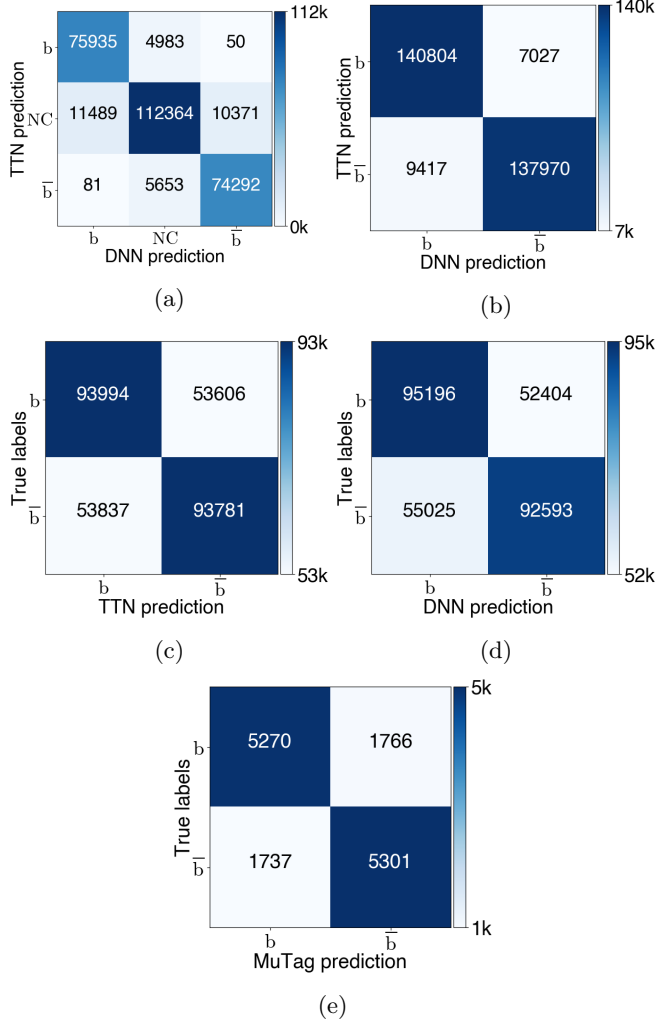


FIG. 14: Confusion matrix with (a) or without (b) cuts, for the TTN and the DNN: possible outputs (b for b -quark, NC for *not classified* and \bar{b} for \bar{b} -quark) for the DNN are shown in the horizontal axis while possible outputs for the TTN are shown in the vertical axis and the confusion matrix of DNN (c), TTN (d) and muon tagging (e) versus the TRUE label (in the vertical axis).

In Fig. 14a the confusion matrix for the TTN and the DNN methods is shown: the output of every jet analysed by the DNN (whose output could be b (\bar{b}) for a jet generated by a b -quark (\bar{b} -quark) or NC for a non-classified jet) is compared to the output of the same jet classified by the TTN. The fact that a jet is classified as

NC comes from applying cuts to the confidence distributions to maximize the tagging power. Despite the output (i.e. the confidence distributions) of the TTN and the DNN being different, the two algorithms tend to classify jets in the same way: whenever the DNN classify a jet as generated by a b -quark it is very unlikely that the TTN wrongly classifies it as generated by a \bar{b} -quark (it happens $\sim 0.1\%$ of the times) and vice-versa; moreover in 7% – 12% of the cases a jet is classified by just one classifier, while the other one does not classify it. As a last remark, when one classifier does not classify a jet (e.g. for the TTN this corresponds to the central row) the other one does not classify the same jet in 90% of the cases, meaning that the TTN and the DNN classify (and do not classify) jets for the majority of the data in the same way. This aspect is also confirmed by considering Fig. 14b, where no cuts on the confidence distributions are considered: when the TTN classify a b -quark (\bar{b} -quark) so does the DNN 95% (94%) of the times.

We further checked the results of the two classifiers for the true labels coming from the MC simulation (i.e. the accuracy of the algorithm). In Figs. 14c and 14d confusion matrices between true labels and DNN and TTN respectively (without cuts on confidence distributions) are shown: in 64% of the times the jet's flavour is correctly classified, both for the TTN and the DNN. In Fig. 14e the same comparison is shown for the muon tagging approach: the jet's flavour is correctly classified 75% of the times but it is evident that the number of classified jet has been reduced.

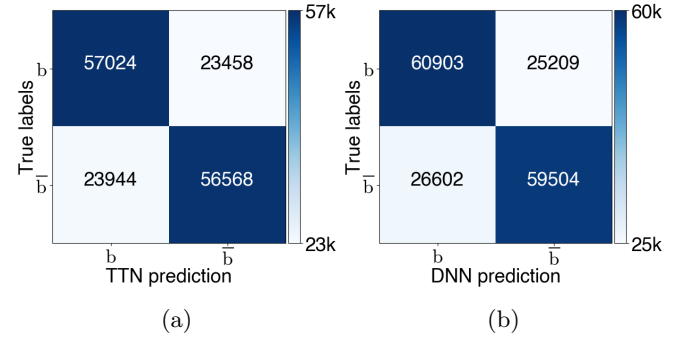


FIG. 15: Confusion matrix with cuts for the TTN (a) and the DNN (b): possible outputs (b for b -quark and \bar{b} for \bar{b} -quark) for the TTN and DNN are shown in the horizontal axis versus the TRUE labels in the vertical axis.

In Figs. 15b and 15a confusion matrices between the two classifiers and the true labels are shown, with cuts maximizing the tagging power applied to the confidence distributions: when applying the cuts we correctly classify between 70-71% of the times, both for the TTN and the DNN. Therefore we can conclude that both the TTN and the DNN have almost the same accuracy compared to the muon tagging algorithm, but they are able to process a bigger amount of jets, resulting in greater

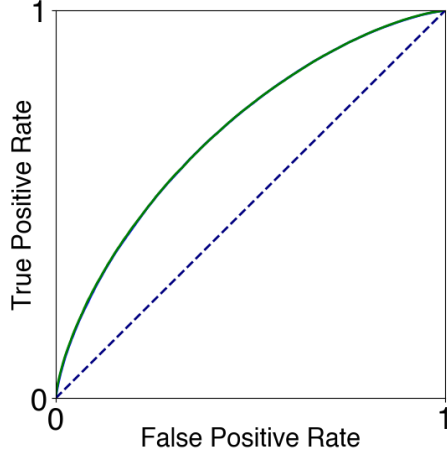


FIG. 16: ROC curves for the DNN (green) and the TTN (blue, but completely covered by DNN), compared with the *line of no-discrimination* (dotted navy-blue line).

values for efficiency and therefore greater values of tagging power.

As a last comparison between the two classifiers we consider the Receiving Operator Characteristic (ROC) curve, in order to check the ability of the TTN and the DNN to classify b - and \bar{b} -jets as the discrimination threshold is varied. Therefore, we plot the rate of correctly tagged jets (defined as *True Positive Rate*, TPR) against the rate of wrongly tagged jets (defined as *False Positive Rate*, FPR). In Fig. 16 the two ROC curves are plotted and compared with the so called *line of no-discrimination*, which represents a randomly guessing classifier: the two ROC curves for tTN and DNN are perfectly coincident, and the Area Under the Curve (AUC) for the two classifiers is the almost same ($AUC^{TTN} = 0.689$ and $AUC^{DNN} = 0.690$).

This last check further confirms the similarity between the TTN and DNN in tagging b - and \bar{b} -jets despite relying on totally different confidence distributions.

-
- [1] C. M. Bishop, *Neural Networks for Pattern Recognition* (Oxford University Press, 1996).
 - [2] S. S. Haykin *et al.*, *Neural networks and learning machines*, Vol. 3 (Pearson, 2009).
 - [3] M. A. Nielsen, *Neural networks and deep learning* (Determination press, 2015).
 - [4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT press, 2016).
 - [5] Y. LeCun, Y. Bengio, and G. Hinton, *Nature* **521**, 436 (2015).
 - [6] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, *Nature* **529**, 484 (2016).
 - [7] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborov, *Reviews of Modern Physics* **91** (2019), 10.1103/revmodphys.91.045002.
 - [8] D.-L. Deng, X. Li, and S. Das Sarma, *Physical Review B* **96** (2017), 10.1103/physrevb.96.195145.
 - [9] Y. Nomura, A. S. Darmawan, Y. Yamaji, and M. Imada, *Physical Review B* **96** (2017), 10.1103/physrevb.96.205152.
 - [10] G. Carleo and M. Troyer, *Science* **355**, 602606 (2017).
 - [11] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers* (Springer, Cham, 2018).
 - [12] S. Das Sarma, D.-L. Deng, and L.-M. Duan, *Physics Today* **72**, 4854 (2019).
 - [13] E. M. Stoudenmire, *Quantum Science and Technology* **3**, 034003 (2018).
 - [14] M. Collura, L. Dell'Anna, T. Felser, and S. Montangero, "On the descriptive power of neural-networks as constrained tensor networks with exponentially large bond dimension," (2019), [arXiv:1905.11351 \[quant-ph\]](#).
 - [15] J. Chen, S. Cheng, H. Xie, L. Wang, and T. Xiang, *Physical Review B* **97** (2018), 10.1103/physrevb.97.085104.
 - [16] Y. Levine, D. Yakira, N. Cohen, and A. Shashua, "Deep learning and quantum entanglement: Fundamental connections with implications to network design," (2017), [arXiv:1704.01552 \[cs.LG\]](#).
 - [17] I. P. McCulloch, *Journal of Statistical Mechanics: Theory and Experiment* **2007**, P10014 (2007).
 - [18] U. Schollwck, *Annals of Physics* **326**, 96192 (2011).
 - [19] S. Singh and G. Vidal, *Phys. Rev. B* **88**, 115147 (2013).
 - [20] M. Dalmonte and S. Montangero, *Contemporary Physics* **57**, 388 (2016).
 - [21] M. Gerster, M. Rizzi, P. Silvi, M. Dalmonte, and S. Montangero, *Phys. Rev. B* **96**, 195123 (2017).
 - [22] M. C. Bauls, R. Blatt, J. Catani, A. Celi, J. I. Cirac, M. Dalmonte, L. Fallani, K. Jansen, M. Lewenstein, S. Montangero, C. A. Muschik, B. Reznik, E. Rico, L. Tagliacozzo, K. V. Acoleyen, F. Verstraete, U. J. Wiese, M. Wingate, J. Zakrzewski, and P. Zoller, "Simulating lattice gauge theories within quantum technologies," (2019), [arXiv:1911.00003 \[quant-ph\]](#).
 - [23] P. Silvi, F. Tschirsich, M. Gerster, J. Jünemann, D. Jaschke, M. Rizzi, and S. Montangero, *SciPost Phys. Lect. Notes*, 8 (2019).
 - [24] T. Felser, P. Silvi, M. Collura, and S. Montangero, "Two-dimensional quantum-link lattice quantum electrodynamics at finite density," (2019), [arXiv:1911.09693 \[quant-ph\]](#).
 - [25] S. Montangero, *Introduction to Tensor Network Methods* (Springer International Publishing, 2018).
 - [26] M. C. Bañuls and K. Cichy, *Reports on Progress in Physics* **83**, 024401 (2020).
 - [27] E. Stoudenmire and D. J. Schwab, in *Advances in Neural Information Processing Systems 29*, edited by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Curran Associates, Inc., 2016) pp. 4799–4807.
 - [28] A. Novikov, M. Trofimov, and I. Oseledets, "Exponential machines," (2016), [arXiv:1605.03795 \[stat.ML\]](#).
 - [29] V. Khrulkov, A. Novikov, and I. Oseledets, "Expressive power of recurrent neural networks," (2017),

- arXiv:1711.00811 [cs.LG].
- [30] D. Liu, S.-J. Ran, P. Wittek, C. Peng, R. B. García, G. Su, and M. Lewenstein, *New Journal of Physics* **21**, 073059 (2019).
 - [31] A. A. Alves Jr. *et al.* (LHCb collaboration), *JINST* **3**, S08005 (2008).
 - [32] R. Aaij *et al.* (LHCb collaboration), “LHCb open data website,” <http://opendata.cern.ch/docs/about-lhcb> (2020).
 - [33] R. Aaij *et al.* (LHCb collaboration), “Jet tagging dataset samples for LHCb quark flavour identification studies,” 10.7483/OPENDATA.LHCB.N75T.TJPE (2020).
 - [34] R. Aaij *et al.* (LHCb collaboration), *Int. J. Mod. Phys. A* **30**, 1530022 (2015), arXiv:1412.6352 [hep-ex].
 - [35] N. I. M. A (ALEPH collaboration), *Nucl. Instrum. Meth. A* **360**, 481 (1994).
 - [36] M. Cacciari, G. P. Salam, and G. Soyez, *JHEP* **04**, 063 (2008), arXiv:0802.1189 [hep-ph].
 - [37] C. W. Murphy, *Phys. Rev. D* **92**, 054003 (2015), arXiv:1504.02493 [hep-ph].
 - [38] V. A. D0 collaboration *et al.*, *Phys. Rev. D* **74**, 112002 (2006), arXiv:0609034v1 [hep-ex].
 - [39] T. A. CDF collaboration *et al.*, *CDF Note* **8235** (2006).
 - [40] R. Aaij *et al.* (LHCb collaboration), *Phys. Rev. Lett.* **113**, 082003 (2014), arXiv:1406.4789 [hep-ex].
 - [41] M. Clemencic *et al.*, *J. Phys. Conf. Ser.* **331**, 032023 (2011).
 - [42] T. Sjöstrand, S. Mrenna, and P. Skands, *Comput. Phys. Commun.* **178**, 852 (2008), arXiv:0710.3820 [hep-ph].
 - [43] D. J. Lange, *Nucl. Instrum. Meth. A* **462**, 152 (2001).
 - [44] S. Agostinelli *et al.* (Geant4 collaboration), *Nucl. Instrum. Meth. A* **506**, 250 (2003).
 - [45] J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Dubois, *et al.* (Geant4 collaboration), *IEEE Trans. Nucl. Sci.* **53**, 270 (2006).
 - [46] M. Nielsen and I. Chuang, eds., *Quantum Computation and Quantum Information* (Cambridge University Press, 2000).
 - [47] A. Milsted, M. Ganahl, S. Leichenauer, J. Hidary, and G. Vidal, “Tensornetwork on tensorflow: A spin chain application using tree tensor networks,” (2019), arXiv:1905.01331 [cond-mat.str-el].
 - [48] R. Aaij *et al.* (LHCb), *JINST* **10**, P06013 (2015), arXiv:1504.07670 [hep-ex].
 - [49] L. R. Tucker, *Psychometrika* **31**, 279 (1966).
 - [50] S. Stlund and S. Rommer, *Physical Review Letters* **75**, 35373540 (1995).
 - [51] I. V. Oseledets, *SIAM Journal on Scientific Computing* **33**, 22952317 (2011).
 - [52] M. Gerster, P. Silvi, M. Rizzi, R. Fazio, T. Calarco, and S. Montangero, *Phys. Rev. B* **90**, 125154 (2014).
 - [53] W. Hackbusch and S. Khn, *Journal of Fourier Analysis and Applications* **15**, 706 (2009).
 - [54] F. Verstraete and J. I. Cirac, “Renormalization algorithms for quantum-many body systems in two and higher dimensions,” (2004), arXiv:cond-mat/0407066 [cond-mat.str-el].
 - [55] R. Ors, *Annals of Physics* **349**, 117158 (2014).
 - [56] I. Glasser, N. Pancotti, and J. I. Cirac, “From probabilistic graphical models to generalized tensor networks for supervised learning,” (2018), arXiv:1806.05964 [quant-ph].
 - [57] L. Tagliacozzo, G. Evenbly, and G. Vidal, *Physical Review B* **80** (2009), 10.1103/physrevb.80.235127.
 - [58] S. Kullback and R. A. Leibler, *Ann. Math. Statist.* **22**, 79 (1951).
 - [59] E. Robeva and A. Seigal, “Duality of graphical models and tensor networks,” (2017), arXiv:1710.01437 [math.ST].
 - [60] I. Glasser, N. Pancotti, M. August, I. D. Rodriguez, and J. I. Cirac, *Physical Review X* **8** (2018), 10.1103/physrevx.8.011006.
 - [61] F. Chollet *et al.*, “Keras,” <https://keras.io> (2015).
 - [62] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” (2015), software available from tensorflow.org.
 - [63] J. Bergstra, D. Yamins, and D. D. Cox, in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML13 (JMLR.org, 2013) p. I115I123.