Jet tagging via particle clouds

Huilin Qu[*]
Department of Physics, University of California, Santa Barbara, California 93106, USA

Loukas Gouskos[†]
CERN, CH-1211 Geneva 23, Switzerland

(Received 12 August 2019; accepted 28 February 2020; published 26 March 2020)

How to represent a jet is at the core of machine learning on jet physics. Inspired by the notion of point clouds, we propose a new approach that considers a jet as an unordered set of its constituent particles, effectively a "particle cloud." Such a particle cloud representation of jets is efficient in incorporating raw information of jets and also explicitly respects the permutation symmetry. Based on the particle cloud representation, we propose ParticleNet, a customized neural network architecture using Dynamic Graph Convolutional Neural Network for jet tagging problems. The ParticleNet architecture achieves state-of-the-art performance on two representative jet tagging benchmarks and is improved significantly over existing methods.

DOI: 10.1103/PhysRevD.101.056019

I. INTRODUCTION

A jet is one of the most ubiquitous objects in proton-proton collision events at the LHC. In essence, a jet is a collimated spray of particles. It serves as a handle to probe the underlying elementary particle produced in the hard scattering process that initiates the cascade of particles contained in the jet.

One of the most important questions about a jet is which type of elementary particle initiates it. Jets initiated by different particles exhibit different characteristics. For example, jets initiated by gluons tend to have a broader energy spread than jets initiated by quarks. High-momentum heavy particles (e.g., top quarks and W, Z, and Higgs bosons) that decay hadronically can lead to jets with distinct multiprong structures. Therefore, the identity of the source particle can be inferred from properties of the reconstructed jet. Such particle identity information provides powerful insights into the collision events under study and therefore can help greatly in separating events originating from different physics processes and improving the sensitivity of both searches for new particles and measurements of the standard model processes.

The study on jet tagging, i.e., the identification of the elementary particle initiating a jet, has a long history. Methods based on the QCD theory have been proposed and continuously improved for discriminating quark and gluon jets [1–7], tagging jets originating from high-momentum heavy particles [8–18], etc. See Refs. [19–24] for more in-depth reviews. Recently, machine learning (ML) has injected fresh blood in jet tagging. Jets are regarded as images [25–35] or as sequences [36–48], trees [49,50], graphs [51], or sets [52] of particles, and ML techniques, most notably deep neural networks (DNNs), are used to build new jet tagging algorithms automatically from (labeled) simulated samples or even (unlabeled) real data [53–56], leading to new insights and improvements in jet tagging.

In this paper, we propose a new deep-learning approach for jet tagging using a novel way to represent jets. Instead of organizing a jet's constituent particles into an ordered structure (e.g., a sequence or a tree), we treat a jet as an *unordered* set of particles [57]. This is very analogous to the point cloud representation of three-dimensional (3D) shapes used in computer vision, where each shape is represented by a set of points in space, and the points themselves are also unordered. Therefore, a jet can be viewed as a "particle cloud." Based on Dynamic Graph Convolutional Neural Network (DGCNN) [58], we design ParticleNet, a customized neural network architecture that operates directly on particle clouds for jet tagging. The ParticleNet architecture is evaluated on two jet tagging benchmarks and is found to achieve significant improvements over all existing methods.

[*] hqu@ucsb.edu
[†] loukas.gouskos@cern.ch

II. JET REPRESENTATIONS

The efficiency and effectiveness of ML techniques on jet physics relies heavily on how a jet is represented. In this section, we review the mainstream jet representations and introduce the particle cloud representation.

A. Image-based representation

The image representation has its root in the reconstruction of jets with calorimeters. A calorimeter measures the energy deposition of a jet on fine-grained spatial cells. Treating the energy deposition on each cell as the pixel intensity naturally creates an image for a jet. When jets are formed by particles reconstructed with the full detector information (e.g., using a particle-flow algorithm [59,60]), a jet image can be constructed by mapping each particle onto the corresponding calorimeter cell and sum up the energy if more than one particle is mapped to the same cell.

The image-based approach has been extensively studied for various jet tagging tasks, e.g., W boson tagging [25–29,35], top tagging [32–34], and quark-gluon tagging [30,31]. Convolutional neural networks (CNNs) with various architectures were explored in these studies, and they were found to achieve sizable improvement in performance compared to traditional multivariate methods using observables motivated by QCD theory. However, the architectures investigated in these papers are in general much shallower compared to state-of-the-art CNN architectures used in image classification tasks (e.g., ResNet [61] or Inception [62]); therefore, it remains to be seen that if deeper architectures can further improve the performance.

Despite the promising performance, the image-based representation has two main shortcomings. While it can include all information without loss when a jet is measured by only the calorimeter, once the jet constituent particles are reconstructed, how to incorporate additional information of the particles is unclear, as it involves combining nonadditive quantities (e.g., the particle type) of multiple particles entering the same cell. Moreover, treating jets as images also leads to a very sparse representation: a typical jet has $\mathcal{O}(10)$ to $\mathcal{O}(100)$ particles, while a jet image typically needs $\mathcal{O}(1000)$ pixels (e.g., 32×32) in order to fully contain the jet; therefore, more than 90% of the pixels are blank. This makes the CNNs highly computationally inefficient on jet images.

B. Particle-based representation

A more natural way to represent a jet, when particles are reconstructed, is to simply view the jet as a collection of its constituent particles. This approach allows for the inclusion of any kind of features for each particle and therefore is significantly more flexible than the image representation. It is also much more compact compared to the image representation, though at the cost of being variable length, as each jet may contain a different number of particles.

A collection of particles, though, is a rather general concept. Before applying any deep-learning algorithm, a concrete data structure has to be chosen. The prevailing choice is a sequence, in which particles are sorted in a specific way (e.g., with decreasing transverse momentum) and organized into a one-dimensional (1D) list. Using particle sequences as inputs, jet tagging tasks have been tackled with recurrent neural networks (RNNs) [36–39,45], 1D CNNs [40–44] and physics-oriented neural networks [46–48]. Another interesting choice is a binary tree, which is well motivated from the QCD theory perspective. Recursive neural networks (RecNNs) are then a natural fit and have been studied in Refs. [49,50].

One thing to note about the sequence or tree representation is that they both need the particles to be sorted in some way, as the order of the particles is used implicitly in the corresponding RNNs, 1D CNNs, or the RecNNs. However, the constituent particles in a jet have no intrinsic order; thus, the manually imposed order may turn out to be suboptimal and impair the performance.

C. Jet as a particle cloud

An even more natural representation than particle sequences or trees would be an unordered, permutation-invariant *set* of particles. As a special case of the particle-based representations, it shares all the advantages of particle-based representations, especially the flexibility to include arbitrary features for each particle. We refer to such representation of a jet as a *particle cloud*, analogous to the point cloud representation of 3D shapes used in computer vision. They are actually highly similar, as both are essentially unordered sets of entities distributed irregularly in space. In both clouds, the elements are not unrelated individuals but are rather correlated, as they represent higher-level objects (i.e., jets or 3D shapes) that have rich internal structures. Therefore, deep-learning algorithms developed for point clouds are likely to be helpful for particle clouds, i.e., jets, as well.

The idea of regarding jets as unordered sets of particles was also proposed in Ref. [52] and is in parallel to our work. The Deep Sets framework [63] was adapted to construct the infrared and collinear safe Energy Flow Network and the more general Particle Flow Network. However, different from the DGCNN [58] approach adopted in this paper, the Deep Sets approach does not explicitly exploit the local spatial structure of particle clouds, but only processes the particle clouds in a global way. Another closely related approach is to represent a jet as a graph whose vertices are the particles. Message-passing neural networks (MPNNs) with different variants of adjacency matrices were explored on such jet graphs and were found to show better performance than the RecNNs [51]. However, depending on how the adjacency matrix is defined, the MPNNs may not respect the permutation symmetry of the particles.

III. NETWORK ARCHITECTURE

The permutation symmetry of the particle cloud makes it a natural and promising representation of jets. However, to achieve the best possible performance, the architecture of the neural network has to be carefully designed to fully exploit the potential of this representation. In this section, we introduce ParticleNet, a CNN-like deep neural network for jet tagging with particle cloud data.

A. Edge convolution

CNNs have achieved overwhelming success in all kinds of machine-learning tasks on visual images. Two key features of CNNs contribute significantly to their success. First, the convolution operation exploits translational symmetry of images by using shared kernels across the whole image. This not only greatly reduces the number of parameters in the network but also allows the parameters to be learned more effectively, as each set of weights will use all locations of the image for learning. Second, CNNs exploit a hierarchical approach [64] for learning image features. The convolution operations can be effectively stacked to form a deep network. Different layers in the CNNs have different receptive fields and therefore can learn features at different scales, with the shallower layers exploiting local neighborhood information and the deeper layers learning more global structures. Such a hierarchical approach proves an effective way to learn images.

Motivated by the success of CNNs, we would like to adopt a similar approach for learning on point (particle) cloud data. However, regular convolution operation cannot be applied on point clouds, as the points there can be distributed irregularly, rather than following some uniform grids as the pixels in an image. Therefore, the basis for a convolution, i.e., a “local patch” of each point on which the convolution kernel operates, remains to be defined for point clouds. Moreover, a regular convolution operation, typically in the form $\sum_j K_j x_j$ where K is the kernel and x_j denotes the features of each point, is not invariant under permutation of the points. Thus, the form of a convolution also needs to be modified to respect the permutation symmetry of point clouds.

Recently, the edge convolution (“EdgeConv”) operation has been proposed in Ref. [58] as a convolutionlike operation for point clouds. EdgeConv starts by representing a point cloud as a graph, whose vertices are the points themselves, and the edges are constructed as connections between each point to its k nearest neighboring points. In this way, a local patch needed for convolution is defined for each point as the k nearest neighboring points connected to it. The EdgeConv operation for each point x_i then has the form

$$\boldsymbol{x}'_i = \mathop{\square}_{j=1}^{k} \boldsymbol{h}_\Theta(\boldsymbol{x}_i, \boldsymbol{x}_{i_j}), \tag{1}$$

where $\boldsymbol{x}_i \in \mathbb{R}^F$ denotes the feature vector of the point x_i and $\{i_1, \ldots, i_k\}$ are the indices of the k nearest neighboring points of the point x_i. The edge function $\boldsymbol{h}_\Theta : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}^{F'}$ is some function parametrized by a set of learnable parameters $\boldsymbol{\Theta}$, and $\square$ is a channelwise symmetric aggregation operation, e.g., max, sum, or mean. The parameters $\boldsymbol{\Theta}$ of the edge function are shared for all points in the point cloud. This, together with the choice of a symmetric aggregation operation $\square$, makes EdgeConv a permutationally symmetric operation on point clouds [65].

In this paper, we follow the choice in Ref. [58] to use a specialized form of the edge function,

$$\boldsymbol{h}_{\boldsymbol{\Theta}}(\boldsymbol{x}_i, \boldsymbol{x}_{i_j}) = \bar{\boldsymbol{h}}_{\boldsymbol{\Theta}}(\boldsymbol{x}_i, \boldsymbol{x}_{i_j} - \boldsymbol{x}_i), \tag{2}$$

where the feature vectors of the neighbors, $\boldsymbol{x}_{i_j}$, are substituted by their differences from the central point $\boldsymbol{x}_i$ and $\bar{\boldsymbol{h}}_{\boldsymbol{\Theta}}$ can be implemented as a multilayer perceptron (MLP) whose parameters are shared among all edges. For the aggregation operation $\square$, however, we use mean, i.e., $\frac{1}{k}\sum$, throughout this paper, which shows better performance than the max operation used in the original paper.

One important feature of the EdgeConv operation is that it can be easily stacked, just as regular convolutions. This is because EdgeConv can be viewed as a mapping from a point cloud to another point cloud with the same number of points, only possibly changing the dimension of the feature vector for each point. Therefore, another EdgeConv operation can be applied subsequently. This allows us to build a deep network using EdgeConv operations, which can learn features of point clouds hierarchically.

The stackability of EdgeConv operations also brings another interesting possibility. Basically, the feature vectors learned by EdgeConv can be viewed as new coordinates of the original points in a latent space, and then, the distances between points, used in the determination of the k nearest neighbors, can be computed in this latent space. In other words, the proximity of points can be dynamically learned with EdgeConv operations. This results in the DGCNN [58], in which the graph describing the point clouds are dynamically updated to reflect the changes in the edges, i.e., the neighbors of each point. Reference [58] demonstrates that this leads to better performance than keeping the graph static.

B. ParticleNet

The ParticleNet architecture makes extensive use of EdgeConv operations and also adopts the dynamic graph update approach. However, a number of different design choices are made in ParticleNet compared to the original DGCNN to better suit the jet tagging task, including the

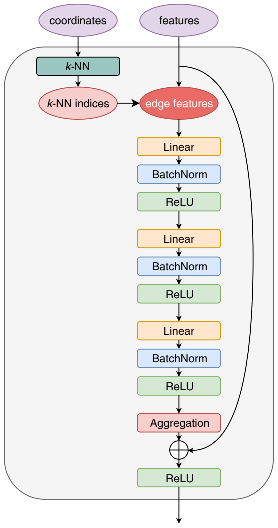
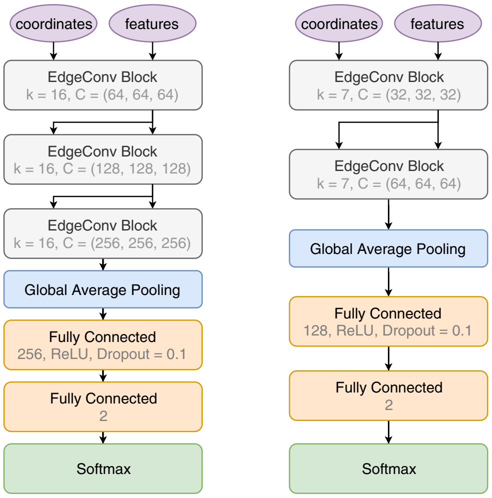


FIG. 1. The structure of the EdgeConv block.



(a) (b)

FIG. 2. The architectures of the (a) ParticleNet and the (b) ParticleNet-Lite networks.

number of neighbors, the configuration of the MLP in EdgeConv, the use of shortcut connection, etc.

Figure 1 illustrates the structure of the EdgeConv block implemented in this paper. The EdgeConv block starts with finding the k nearest neighboring particles for each particle, using the "coordinates" input of the EdgeConv block to compute the distances. Then, inputs to the EdgeConv operation, the "edge features" are constructed from the "features" input using the indices of k nearest neighboring particles. The EdgeConv operation is implemented as a three-layer MLP. Each layer consists of a linear transformation, followed by a batch normalization [66] and then a rectified linear unit (ReLU) [67]. Inspired by ResNet [61], a shortcut connection running parallel to the EdgeConv operation is also included in each block, allowing the input features to pass through directly. An EdgeConv block is characterized by two hyperparameters, the number of neighbors k, and the number of channels $C = (C_1, C_2, C_3)$, corresponding to the number of units in each linear transformation layer.

The ParticleNet architecture used in this paper is shown in Fig. 2(a). It consists of three EdgeConv blocks. The first EdgeConv block uses the spatial coordinates of the particles in the pseudorapidity-azimuth space to compute the distances, while the subsequent blocks use the learned feature vectors as coordinates. The number of nearest neighbors k is 16 for all three blocks, and the number of channels C for each EdgeConv block is (64, 64, 64), (128, 128, 128), and (256, 256, 256), respectively. After the EdgeConv blocks, a channelwise global average pooling operation is applied to aggregate the learned features over all particles in the cloud. This is followed by a fully connected layer with 256 units and the ReLU activation. A dropout layer [68] with a drop probability of 0.1 is included to prevent overfitting. A fully connected layer with two units, followed by a softmax function, is used to generate the output for the binary classification task.

A similar network with reduced complexity is also investigated. Compared to the baseline ParticleNet architecture, only two EdgeConv blocks are used, with the number of nearest neighbors k reduced to 7 and the number of channels C reduced to (32, 32, 32) and (64, 64, 64) for the two blocks, respectively. The number of units in the fully connected layer after pooling is also lowered to 128. This simplified architecture is denoted as "ParticleNet-Lite" and is illustrated in Fig. 2(b). The number of arithmetic operations is reduced by almost an order of magnitude in ParticleNet-Lite, making it more suitable when computational resources are limited.

The networks are implemented with Apache MXNet [69], and the training is performed on a single Nvidia GTX 1080 Ti graphics card (GPU). A batch size of 384 (1024) is used for the ParticleNet (ParticleNet-Lite) architecture due to GPU memory constraint. The AdamW optimizer [70], with a weight decay of 0.0001, is used to minimize the cross entropy loss. The one-cycle learning rate (LR) schedule [71] is adopted in the training, with the LR selected following the LR range test described in Ref. [71], and slightly tuned afterward with a few trial trainings. The training of ParticleNet (ParticleNet-Lite) network uses an initial LR of 3×10^{-4} (5×10^{-4}), rising to the peak LR of 3×10^{-3} (5×10^{-3}) linearly in eight epochs and then decreasing to the initial LR linearly in another eight epochs. This is followed by a cooldown phase of four epochs, which gradually reduces the LR to 5×10^{-7} (1×10^{-6}) for better convergence. A snapshot of the model is saved at the end of each epoch, and the model snapshot showing the best accuracy on the validation dataset is selected for the final evaluation.

IV. RESULTS

The performance of the ParticleNet architecture is evaluated on two representative jet tagging tasks: top tagging and quark-gluon tagging. In this section, we show the benchmark results.

A. Top tagging

Top tagging, i.e., identifying jets originating from hadronically decaying top quarks, is commonly used in searches for new physics at the LHC. We evaluate the performance of the ParticleNet architecture on this task using the top tagging dataset [72], which is an extension of the dataset used in Ref. [46] with some modifications. Jets in this dataset are generated with PYTHIA8 [73] and passed through DELPHES [74] for fast detector simulation. No multiple parton interaction or pileup is included in the simulation. Jets are clustered from the DELPHES E-Flow objects with the anti-k_T algorithm [75] using a distance parameter $R = 0.8$. Only jets with transverse momentum $p_T \in [550, 650]$ and pseudorapidity $|\eta| < 2$ are considered. Each signal jet is required to be matched to a hadronically decaying top quark within $\Delta R = 0.8$, and all three quarks from the top decay also within $\Delta R = 0.8$ of the jet axis. The background jets are obtained from a QCD dijet process. This dataset consists of 2×10^6 jets in total, half signal and half background. The official splitting for training (1.2×10^6 jets), validation (400,000 jets), and testing (400,000 jets) is used in the development of the ParticleNet model for this dataset.

In this dataset, up to 200 jet constituent particles are stored for each jet. Only kinematic information, i.e., the 4-momentum (p_x, p_y, p_z, E), of each particle is available. The ParticleNet model takes up to 100 constituent particles with the highest p_T for each jet, and uses seven variables derived from the 4-momentum for each particle as inputs, which are listed in Table I. The $(\Delta\eta, \Delta\phi)$ variables are used as coordinates to compute the distances between particles in the first EdgeConv block. They are also used together with the other five variables, $\log p_T$, $\log E$, $\log \frac{p_T}{p_T(\text{jet})}$, $\log \frac{E}{E(\text{jet})}$, and ΔR, to form the input feature vector for each particle.

We compare the performance of ParticleNet with three alternative models [76]:

(i) *ResNeXt-50.*—The ResNeXt-50 model is a very deep two-dimensional (2D) CNN using jet images as inputs. The ResNeXt architecture [78] was proposed for generic image classification, and we modify it slightly for the jet tagging task. The model is trained on the top tagging dataset starting from randomly initialized weights. The implementation details can be found in the Appendix A. Note that the ResNeXt-50 architecture is much deeper and therefore has a much larger capacity than most of the CNN architectures [25,27–35] explored for jet tagging so far, so evaluating its performance on jet tagging will shed light on whether architectures for generic image classification are also applicable to jet images.

(ii) *P-CNN.*—The P-CNN is a 14-layer 1D CNN using particle sequences as inputs. The P-CNN architecture was proposed in the CMS particle-based DNN boosted jet tagger [42] and showed significant improvement in performance compared to a traditional tagger using boosted decision trees and jet-level observables. The model is also trained on the top tagging dataset from scratch, with the implementation details in Appendix B.

(iii) *PFN:* The Particle Flow Network (PFN) [52] is a recent architecture for jet tagging which also treats a jet as an unordered set of particles, the same as the particle cloud approach in this paper. However, the network is based on the Deep Sets framework [63], which uses global symmetric functions and does not exploit local neighborhood information explicitly as the EdgeConv operation. Since the performance of PFN on this top tagging dataset has already been

TABLE I. Input variables used in the top tagging task (TOP) and the quark-gluon tagging task (QG) with and without PID information.

Variable	Definition	TOP	QG	QG-PID
$\Delta\eta$	Difference in pseudorapidity between the particle and the jet axis	×	×	×
$\Delta\phi$	Difference in azimuthal angle between the particle and the jet axis	×	×	×
$\log p_T$	Logarithm of the particle's p_T	×	×	×
$\log E$	Logarithm of the particle's energy	×	×	×
$\log \frac{p_T}{p_T(\text{jet})}$	Logarithm of the particle's p_T relative to the jet p_T	×	×	×
$\log \frac{E}{E(\text{jet})}$	Logarithm of the particle's energy relative to the jet energy	×	×	×
ΔR	Angular separation between the particle and the jet axis $\left(\sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}\right)$	×	×	×
q	Electric charge of the particle			×
`isElectron`	If the particle is an electron			×
`isMuon`	If the particle is a muon			×
`isChargedHadron`	If the particle is a charged hadron			×
`isNeutralHadron`	If the particle is a neutral hadron			×
`isPhoton`	If the particle is a photon			×

TABLE II. Performance comparison on the top tagging benchmark dataset. The ParticleNet, ParticleNet-Lite, P-CNN, and ResNeXt-50 models are trained on the top tagging dataset starting from randomly initialized weights. For each model, the training is repeated for nine times using different randomly initialized weights. The table shows the result from the median-accuracy training, and the standard deviation of the nine trainings is quoted as the uncertainty to assess the stability to random weight initialization. Uncertainty on the accuracy and AUC are negligible and therefore omitted. The performance of PFN on this dataset is reported in Ref. [52], and the uncertainty corresponds to the spread in ten trainings.

	Accuracy	AUC	$1/\varepsilon_b$ at $\varepsilon_s = 50\%$	$1/\varepsilon_b$ at $\varepsilon_s = 30\%$
ResNeXt-50	0.936	0.9837	302 ± 5	1147 ± 58
P-CNN	0.930	0.9803	201 ± 4	759 ± 24
PFN	⋯	0.9819	247 ± 3	888 ± 17
ParticleNet-Lite	0.937	0.9844	325 ± 5	1262 ± 49
ParticleNet	**0.940**	**0.9858**	$\mathbf{397 \pm 7}$	$\mathbf{1615 \pm 93}$

reported in Ref. [52], we did not reimplement it but just include the results for comparison.

The results are summarized in Table II and also shown in Fig. 3 in terms of receiver operating characteristic (ROC) curves. A number of metrics are used to evaluate the performance, including the accuracy, the area under the ROC curve (AUC), and the background rejection ($1/\varepsilon_b$, i.e., the reciprocal of the background misidentification rate) at a certain signal efficiency (ε_s) of 50% or 30%. The background rejection metric is particularly relevant to physics analysis at the LHC, as it is directly related to the expected contribution of background, and is commonly used to select the best jet tagging algorithm. The ParticleNet model achieves state-of-the-art performance on the top tagging benchmark dataset and improves over previous methods significantly. Its background rejection power at 30% signal efficiency is roughly 1.8 (2.1) times as good as PFN (P-CNN) and about 40% better than ResNeXt-50. Even the ParticleNet-Lite model, with significantly reduced complexity, outperforms all the previous models, achieving about 10% improvement with respect to ResNeXt-50. The large performance improvement of the ParticleNet architecture over the PFN architecture is likely due to a better exploitation of the local neighborhood information with the EdgeConv operation.

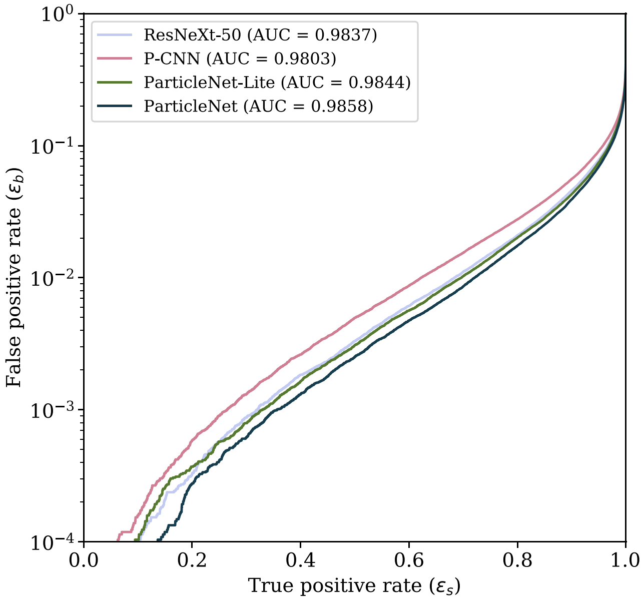


FIG. 3. Performance comparison in terms of ROC curves on the top tagging benchmark dataset.

B. Quark-gluon tagging

Another important jet tagging task is quark-gluon tagging, i.e., discriminating jets initiated by quarks and by gluons. The quark-gluon tagging dataset from Ref. [52] is used to evaluate the performance of the ParticleNet architecture on this task. The signal (quark) and background (gluon) jets are generated with PYTHIA8 using the $Z(\to \nu\nu) + (u, d, s)$ and $Z(\to \nu\nu) + g$ processes, respectively. No detector simulation is performed. The final state non-neutrino particles are clustered into jets using the anti-k_T algorithm [75] with $R = 0.4$. Only jets with transverse momentum $p_T \in [500, 550]$ and rapidity $|y| < 2$ are considered. This dataset consists of 2 million jets in total, half signal and half background. We follow the recommended splitting of $1.6 \times 10^6/200,000/200,000$ for training, validation, and testing in the development of the ParticleNet model on this dataset.

One important difference of the quark-gluon tagging dataset is that it includes not only the four momentum but also the type of each particle (i.e., electron, photon, pion, etc.). Such particle identification (PID) information can be quite helpful for jet tagging. Therefore, we include this information in the ParticleNet model and compare it with the baseline version using only the kinematic information. The PID information is included in an experimentally realistic way by using only five particle types (electron, muon, charged hadron, neutral hadron, and photon), as well as the electric charge, as inputs. These six additional variables, together with the seven kinematic variables, form the input feature vector of each particle for models with PID information, as shown in Table I.

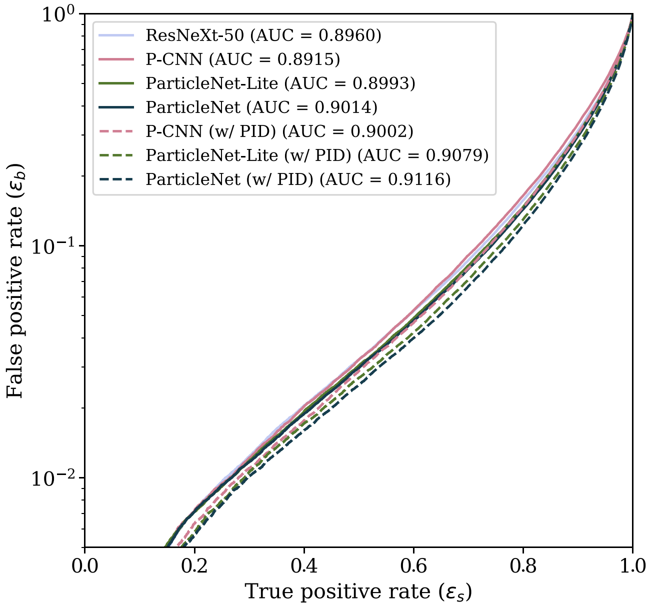


FIG. 4. Performance comparison in terms of ROC curves on the quark-gluon tagging benchmark dataset.

Table III compares the performance of the ParticleNet model with a number of alternative models introduced in Sec. IV A. Model variants with and without PID inputs are also compared. Note that for the ResNeXt-50 model only the version without PID inputs is presented, as it is based on jet images which cannot incorporate PID information straightforwardly. The corresponding ROC curves are shown in Fig. 4. Overall, the addition of PID inputs has a large impact on the performance, increasing the background rejection power by 10%–15% compared to the same model without using PID information. This clearly demonstrates the advantage of particle-based jet representations, including the particle cloud representation, as they can easily integrate any additional information for each particle. The best performance is obtained by the ParticleNet model with PID inputs, achieving almost 15% improvement on the background rejection power compared to the PFN-Ex (PFN using experimentally realistic PID information) and P-CNN models. The ParticleNet-Lite model achieves the second-best performance and shows about 7% improvement with respect to the PFN-Ex and P-CNN models.

TABLE III. Performance comparison on the quark-gluon tagging benchmark dataset. The ParticleNet, ParticleNet-Lite, P-CNN, and ResNeXt-50 models are trained on the quark-gluon tagging dataset starting from randomly initialized weights. The training is repeated nine times for the ParticleNet model using different randomly initialized weights. The table shows the result from the median-accuracy training, and the standard deviation of the nine trainings is quoted as the uncertainty to assess the stability to random weight initialization. Because of limited computational resources, the training of other models is performed only once, but the uncertainty due to random weight initialization is expected to be fairly small. The performance of PFN on this dataset is reported in Ref. [52], and the uncertainty corresponds to the spread in ten trainings. Note that a number of PFN models with different levels of PID information are investigated in Ref. [52], and "PFN-Ex," also using experimentally realistic PID information, is shown here for comparison.

	Accuracy	AUC	$1/\varepsilon_b$ at $\varepsilon_s = 50\%$	$1/\varepsilon_b$ at $\varepsilon_s = 30\%$
ResNeXt-50	0.821	0.8960	30.9	80.8
P-CNN	0.818	0.8915	31.0	82.3
PFN	⋯	0.8911	30.8 ± 0.4	⋯
ParticleNet-Lite	0.826	0.8993	32.8	84.6
ParticleNet	0.828	0.9014	33.7	85.4
P-CNN (w/ PID)	0.827	0.9002	34.7	91.0
PFN-Ex (w/ PID)	⋯	0.9005	34.7 ± 0.4	⋯
ParticleNet-Lite (w/ PID)	0.835	0.9079	37.1	94.5
ParticleNet (w/ PID)	**0.840**	**0.9116**	$\mathbf{39.8 \pm 0.2}$	$\mathbf{98.6 \pm 1.3}$

TABLE IV. Number of parameters, inference time per object, and background rejection of different models. The CPU inference time is measured on an Intel Core i7-6850K CPU with a single thread using a batch size of 1. The GPU inference time is measured on a Nvidia GTX 1080 Ti GPU using a batch size of 100.

	Parameters	Time (CPU) (ms)	Time (GPU) (ms)	$1/\varepsilon_b$ at $\varepsilon_s = 30\%$
ResNeXt-50	1.46×10^6	7.4	0.22	1147 ± 58
P-CNN	348,000	1.6	0.020	759 ± 24
PFN	82,000	**0.8**	**0.018**	888 ± 17
ParticleNet-Lite	**26,000**	2.4	0.084	1262 ± 49
ParticleNet	366,000	23	0.92	$\mathbf{1615 \pm 93}$

V. MODEL COMPLEXITY

Another aspect of machine-learning models is the complexity, e.g., the number of parameters and the computational cost. Table IV compares the number of parameters and the computational cost of all the models used in the top tagging task in Sec. IV A. The computational cost is evaluated using the inference time per object, which is a more relevant metric than the training time for real-life applications of machine-learning models. The inference time of each model is measured on both the CPU and the GPU, using the implementations with Apache MXNet. For the CPU, to mimic the event processing workflow typically used in collider experiments, a batch size of 1 is used, and the inference is performed in single-thread mode. For the GPU, a batch size of 100 is used instead, as the full power of the GPU cannot be revealed with a very small batch size (e.g., 1) due to the overhead in data transfer between the CPU and the GPU. The ParticleNet model achieves the best classification performance at the cost of speed, being more than an order of magnitude slower than the PFN and the P-CNN models, but still it is not prohibitively slow even on the CPU. In addition, the current implementation of the EdgeConv operation used in the ParticleNet model is not as optimized as the regular convolution operation; therefore, further speed-up is expected from an optimized implementation of EdgeConv. On the other hand, the ParticleNet-Lite model provides a good balance between speed and performance, showing more than 40% improvement in performance while being only a few times slower than the PFN and P-CNN models. Notably, it is also the most economical model, outperforming all previous approaches with only 26,000 parameters, thanks to the effective exploitation of the permutation symmetry of the particle clouds. Overall, PFN is the fastest model on both the CPU and the GPU, making it a suitable choice for extremely time-critical tasks.

VI. CONCLUSION

In this paper, we present a new approach for machine learning on jets. The core of this approach is to treat jets as particle clouds, i.e., unordered sets of particles. Based on this particle cloud representation, we introduce ParticleNet, a network architecture tailored to jet tagging tasks. The performance of the ParticleNet architecture is compared with alternative deep-learning architectures, including the jet image–based ResNeXt-50 model, the particle sequence–based P-CNN model, and the particle set–based PFN model. On both the top tagging and the quark-gluon tagging benchmarks, ParticleNet achieves state-of-the-art performance and improves significantly over existing methods. Although the very deep image–based ResNeXt-50 model also shows significant performance improvement over shallower models like P-CNN and PFN on the top-tagging benchmark, indicating that deeper architectures can generally lead to better performance, the gain with the ParticleNet architecture is more substantial. Moreover, the high performance is achieved in a very economical way as the number of trainable parameters is a factor of 4 (56) lower in ParticleNet (ParticleNet-Lite) compared to ResNeXt-50. Such lightweight models are particularly useful for applications in high-energy physics experiments, especially for online event processing in which low latency and memory consumption is critical.

While we only demonstrate the power of the particle cloud representation in jet tagging tasks, we think that it is a natural and generic way of representing jets (and even the whole collision event) and can be applied to a broad range of particle physics problems. Applications of the particle cloud approach to, e.g., pileup identification, jet grooming, jet energy calibration, etc., would be particularly interesting and worth further investigation.

ACKNOWLEDGMENTS

We thank Gregor Kasieczka, Tilman Plehn, and Michael Russel for creating the top tagging dataset and sharing it with us as well as Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler for creating the quark-gluon tagging dataset and making it publicly accessible. This work was supported by the U.S. Department of Energy under Grant No. DE-SC0011702.

APPENDIX A: IMPLEMENTATION DETAILS OF ResNeXt-50

The ResNeXt-50 model uses jet images as inputs. Each image is constructed from the constituent particles by projecting them onto a 2D grid of 64 × 64 pixels in size, corresponding to a granularity of 0.025 rad in the pseudorapidity-azimuth space. The intensity of each pixel is the sum of p_T of all the particles within the pixel rescaled by the inverse of the jet p_T.

The original 50-layer ResNeXt architecture [78] was developed for images of size 224 × 224 and a classification task with 1000 classes. To adapt to the smaller size of the jet images and the significantly fewer number of output classes, the number of channels in all but the first convolutional layers is reduced by a factor of 4, and a dropout layer with a drop probability of 0.5 is added after the global pooling layer.

The network is implemented with Apache MXNet and trained with the Adam optimizer with a minibatch size of 256. The network is trained for 30 epochs, with a starting learning rate of 0.01, and subsequently reduced by a factor of 10 at the 10th and 20th epochs. A snapshot of the model is saved at the end of each epoch, and the model snapshot showing the best accuracy on the validation dataset is selected for the final evaluation.

APPENDIX B: IMPLEMENTATION DETAILS OF P-CNN

The particle-level convolutional neural network (P-CNN) [42] is a deep 1D CNN architecture customized for boosted jet tagging. Each input jet is represented as a sequence of particles with a fixed length of 100. The particles are organized in descending order of p_T. The sequence is padded with zeros if a jet has less than 100 particles and truncated if it has more than 100 particles.

The P-CNN architecture is similar to the ResNet model [61,79] for image classification but uses 1D convolution instead. It features a total of 14 convolutional layers, all with a kernel size of 3. The number of channels for the 1D convolutions is either 32, 64, or 128. The convolutions are followed by a global pooling, then by a fully connected layer of 512 units with ReLU activation and a dropout layer with a drop rate of 0.5, before producing the classification output.

The network is implemented with Apache MXNet and trained with the Adam optimizer with a minibatch size of 1024. The network is trained for 30 epochs, with a starting learning rate of 0.001, and subsequently reduced by a factor of 10 at the 10th and 20th epochs. A snapshot of the model is saved at the end of each epoch, and the model snapshot showing the best accuracy on the validation dataset is selected for the final evaluation.

[1] J. Gallicchio and M. D. Schwartz, Quark and Gluon Tagging at the LHC, Phys. Rev. Lett. **107**, 172001 (2011).

[2] J. Gallicchio and M. D. Schwartz, Quark and gluon jet substructure, J. High Energy Phys. 04 (2013) 090.

[3] A. J. Larkoski, J. Thaler, and W. J. Waalewijn, Gaining (mutual) information about quark/gluon discrimination, J. High Energy Phys. 11 (2014) 129.

[4] B. Bhattacherjee, S. Mukhopadhyay, M. M. Nojiri, Y. Sakaki, and B. R. Webber, Associated jet and subjet rates in light-quark and gluon jet discrimination, J. High Energy Phys. 04 (2015) 131.

[5] D. F. de Lima, P. Petrov, D. Soper, and M. Spannowsky, Quark-Gluon tagging with shower deconstruction: Unearthing dark matter and Higgs couplings, Phys. Rev. D **95**, 034001 (2017).

[6] P. Gras, S. Höche, D. Kar, A. Larkoski, L. Lönnblad, S. Plätzer, A. Siódmok, P. Skands, G. Soyez, and J. Thaler, Systematics of quark/gluon tagging, J. High Energy Phys. 07 (2017) 091.

[7] C. Frye, A. J. Larkoski, J. Thaler, and K. Zhou, Casimir meets Poisson: Improved quark/gluon discrimination with counting observables, J. High Energy Phys. 09 (2017) 083.

[8] D. E. Kaplan, K. Rehermann, M. D. Schwartz, and B. Tweedie, Top Tagging: A Method for Identifying Boosted Hadronically Decaying Top Quarks, Phys. Rev. Lett. **101**, 142001 (2008).

[9] Y. Cui, Z. Han, and M. D. Schwartz, W-jet tagging: Optimizing the identification of boosted hadronically-decaying W bosons, Phys. Rev. D **83**, 074023 (2011).

[10] T. Plehn, M. Spannowsky, and M. Takeuchi, How to improve top tagging, Phys. Rev. D **85**, 034029 (2012).

[11] D. E. Soperand and M. Spannowsky, Finding top quarks with shower deconstruction, Phys. Rev. D **87**, 054012 (2013).

[12] C. Anders, C. Bernaciak, G. Kasieczka, T. Plehn, and T. Schell, Benchmarking an even better top tagger algorithm, Phys. Rev. D **89**, 074047 (2014).

[13] G. Kasieczka, T. Plehn, T. Schell, T. Strebler, and G. P. Salam, Resonance searches with an updated top tagger, J. High Energy Phys. 06 (2015) 203.

[14] J. Thaler and K. Van Tilburg, Identifying boosted objects with N-subjettiness, J. High Energy Phys. 03 (2011) 015.

[15] J. Thaler and K. Van Tilburg, Maximizing boosted top identification by minimizing N-subjettiness, J. High Energy Phys. 02 (2012) 093.

[16] A. J. Larkoski, G. P. Salam, and J. Thaler, Energy correlation functions for jet substructure, J. High Energy Phys. 06 (2013) 108.

[17] I. Moult, L. Necib, and J. Thaler, New angles on energy correlation functions, J. High Energy Phys. 12 (2016) 153.

[18] A. J. Larkoski, S. Marzani, G. Soyez, and J. Thaler, Soft drop, J. High Energy Phys. 05 (2014) 146.

[19] A. Abdesselam *et al.*, Boosted objects: A probe of beyond the Standard Model physics, Eur. Phys. J. C **71**, 1661 (2011).

[20] A. Altheimer *et al.*, Jet substructure at the Tevatron and LHC: New results, new tools, new benchmarks, J. Phys. G **39**, 063001 (2012).

[21] A. Altheimer *et al.*, Boosted objects and jet substructure at the LHC. Report of BOOST2012, held at IFIC Valencia, 23rd–27th of July 2012, Eur. Phys. J. C **74**, 2792 (2014).

[22] D. Adams *et al.*, Towards an understanding of the correlations in jet substructure, Eur. Phys. J. C **75**, 409 (2015).

[23] A. J. Larkoski, I. Moult, and B. Nachman, Jet substructure at the Large Hadron Collider: A review of recent advances in theory and machine learning, Phys. Rep. **841**, 1 (2020).

[24] L. Asquith *et al.*, Jet substructure at the Large Hadron Collider: Experimental review, Rev. Mod. Phys. **91**, 045003 (2019).

[25] J. Cogan, M. Kagan, E. Strauss, and A. Schwarztman, Jet-images: Computer vision inspired techniques for jet tagging, J. High Energy Phys. 02 (2015) 118.

[26] L. G. Almeida, M. Backović, M. Cliche, S. J. Lee, and M. Perelstein, Playing tag with ANN: Boosted top identification with pattern recognition, J. High Energy Phys. 07 (2015) 086.

[27] L. de Oliveira, M. Kagan, L. Mackey, B. Nachman, and A. Schwartzman, Jet-images—Deep learning edition, J. High Energy Phys. 07 (2016) 069.

[28] P. Baldi, K. Bauer, C. Eng, P. Sadowski, and D. Whiteson, Jet substructure classification in high-energy physics with deep neural networks, Phys. Rev. D **93**, 094034 (2016).

[29] J. Barnard, E. N. Dawe, M. J. Dolan, and N. Rajcic, Parton shower uncertainties in jet substructure analyses with deep neural networks, Phys. Rev. D **95**, 014018 (2017).

[30] P. T. Komiske, E. M. Metodiev, and M. D. Schwartz, Deep learning in color: Towards automated quark/gluon jet discrimination, J. High Energy Phys. 01 (2017) 110.

[31] ATLAS Collaboration, Quark versus Gluon Jet Tagging Using Jet Images with the ATLAS Detector, Tech. Rep. No. ATL-PHYS-PUB-2017-017 (CERN, Geneva, 2017).

[32] G. Kasieczka, T. Plehn, M. Russell, and T. Schell, Deep-learning top taggers or the end of QCD?, J. High Energy Phys. 05 (2017) 006.

[33] S. Macaluso and D. Shih, Pulling out all the tops with computer vision and deep learning, J. High Energy Phys. 10 (2018) 121.

[34] S. Choi, S. J. Lee, and M. Perelstein, Infrared safety of a neural-net top tagging algorithm, J. High Energy Phys. 002 (2019) 132.

[35] F. A. Dreyer, G. P. Salam, and G. Soyez, The Lund jet plane, J. High Energy Phys. 12 (2018) 064.

[36] D. Guest, J. Collado, P. Baldi, S.-C. Hsu, G. Urban, and D. Whiteson, Jet flavor classification in high-energy physics with deep neural networks, Phys. Rev. D **94**, 112002 (2016).

[37] J. Pearkes, W. Fedorko, A. Lister, and C. Gay, Jet constituents for deep neural network based top quark tagging, arXiv:1704.02124.

[38] S. Egan, W. Fedorko, A. Lister, J. Pearkes, and C. Gay, Long short-term memory (LSTM) networks with jet constituents for boosted top tagging at the LHC, arXiv:1711.09059.

[39] K. Fraserand and M. D. Schwartz, Jet charge and machine learning, J. High Energy Phys. 10 (2018) 093.

[40] CMS Collaboration, CMS Phase 1 heavy flavour identification performance and developments, Tech. Rep. No. CMS-DP-2017-013, 2017.

[41] CMS Collaboration, Performance of the DeepJet b tagging algorithm using 41.9/fb of data from proton-proton collisions at 13 TeV with Phase 1 CMS detector, Tech. Rep. No. CMS-DP-2018-058, 2018.

[42] CMS Collaboration, Boosted jet identification using particle candidates and deep neural networks, Tech. Rep. No. CMS-DP-2017-049, 2017.

[43] M. Stoye, J. Kieseler, H. Qu, L. Gouskos, M. Verzetti, and A. Stakia, Deepjet: Generic physics object based jet multiclass classification for LHC experiments, in *Deep Learning for Physical Sciences Workshop at the 31st Conference on Neural Information Processing Systems (NIPS)* (2017), https://dl4physicalsciences.github.io.

[44] CMS Collaboration, Machine learning-based identification of highly Lorentz-boosted hadronically decaying particles at the CMS experiment, Tech. Rep. No. CMS-PAS-JME-18-002 (CERN, Geneva, 2019).

[45] ATLAS Collaboration, Identification of jets containing b-hadrons with recurrent neural networks at the ATLAS experiment, Tech. Rep. No. ATL-PHYS-PUB-2017-003 (CERN, Geneva, 2017).

[46] A. Butter, G. Kasieczka, T. Plehn, and M. Russell, Deep-learned top tagging with a Lorentz layer, SciPost Phys. **5**, 028 (2018).

[47] G. Kasieczka, N. Kiefer, T. Plehn, and J. M. Thompson, Quark-gluon tagging: Machine learning meets reality, SciPost Phys. **6**, 069 (2019).

[48] M. Erdmann, E. Geiser, Y. Rath, and M. Rieger, Lorentz boost networks: Autonomous physics-inspired feature engineering, arXiv:1812.09722.

[49] G. Louppe, K. Cho, C. Becot, and K. Cranmer, QCD-aware recursive neural networks for jet physics, J. High Energy Phys. 01 (2019) 057.

[50] T. Cheng, Recursive neural networks in quark/gluon tagging, Comput. Softw. Bi.g. Sci. **2**, 3 (2018).

[51] I. Henrion, J. Brehmer, J. Bruna, K. Cho, K. Cranmer, G. Louppe, and G. Rochette, Neural message passing for jet physics, in *Deep Learning for Physical Sciences Workshop at the 31st Conference on Neural Information Processing Systems (NIPS)* (2017), https://dl4physicalsciences.github.io.

[52] P. T. Komiske, E. M. Metodiev, and J. Thaler, Energy flow networks: Deep sets for particle jets, J. High Energy Phys. 01 (2019) 121.

[53] E. M. Metodiev, B. Nachman, and J. Thaler, Classification without labels: Learning from mixed samples in high energy physics, J. High Energy Phys. 10 (2017) 174.

[54] P. T. Komiske, E. M. Metodiev, B. Nachman, and M. D. Schwartz, Learning to classify from impure samples with high-dimensional data, Phys. Rev. D **98**, 011502(R) (2018).

[55] A. Andreassen, I. Feige, C. Frye, and M. D. Schwartz, JUNIPR: A framework for unsupervised machine learning in particle physics, Eur. Phys. J. C **79**, 102 (2019).

[56] P. T. Komiske, E. M. Metodiev, and J. Thaler, An operational definition of quark and gluon jets, J. High Energy Phys. 11 (2018) 059.

[57] The idea of regarding jets as unordered sets of particles was also proposed in Ref. [52] independently recently. We provide comparison to their approach in later sections.

[58] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, Dynamic graph CNN for learning on point clouds, ACM Trans. Graph. **38**, 146 (2019).

[59] A. M. Sirunyan *et al.* (CMS Collaboration), Particle-flow reconstruction and global event description with the CMS detector, J. Instrum. **12**, P10003 (2017).

[60] M. Aaboud *et al.* (ATLAS Collaboration), Jet reconstruction and performance using particle flow with the ATLAS Detector, Eur. Phys. J. C **77**, 466 (2017).

[61] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, Las Vegas, NV, 2016) pp. 770–778, https://ieeexplore.ieee.org/document/7780459.

[62] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, Rethinking the inception architecture for computer vision, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, Las Vegas, NV, 2016) pp. 2818–2826, https://ieeexplore.ieee.org/document/7780677.

[63] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, Deep sets, *Advances in Neural Information Processing Systems* (Curran Associates, Inc., Long Beach, CA, 2017) pp. 3391–3401, https://papers.nips.cc/paper/6931-deep-sets.

[64] M. D. Zeiler and R. Fergus, Visualizing and understanding convolutional networks, in *Computer Vision—ECCV 2014*, edited by D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars (Springer, Cham, 2014), pp. 818–833.

[65] Unlike other approaches in the literature (e.g, Deep Sets [63]), EdgeConv is not designed as a universal approximator for any permutation-invariant functions. Specifically, the permutation invariance of the EdgeConv layer refers to the fact that the output does not depend on the ordering of the input points. However, to use EdgeConv, one needs to specify a set of features to be used as the "coordinates" for the computation of distances needed by the nearest neighbor finding. This choice of a "coordinate system" then leads to a canonical ordering of the points in that space where the neighbor relationship is fully determined. Since EdgeConv is performed with the k nearest neighbors for each point, any permutation of the points that changes the neighbor relationship will actually lead to a change in the network output.

[66] S. Ioffeand and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in *Proceedings of the 32nd International Conference on International Conference on Machine Learning, Lille, France* (JMLR.org, 2015), pp. 448–456.

[67] X. Glorot, A. Bordes, and Y. Bengio, Deep sparse rectifier neural networks, in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (PMLR, Fort Lauderdale, FL, 2011) pp. 315–323, http://proceedings.mlr.press/v15/glorot11a.html.

[68] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. **15**, 1929 (2014).

[69] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems, in *Workshop on Machine Learning Systems at the 29st Conference on Neural Information Processing Systems (NIPS)* (LearningSys.org, Montreal, Canada, 2015).

[70] I. Loshchilovand and F. Hutter, Fixing weight decay regularization in adam, in *Proceedings of the 7th International Conference on Learning Representations (ICLR)* (ICLR.cc, New Orleans, LA, 2019).

[71] L. N. Smith, A disciplined approach to neural network hyper-parameters: Part 1—Learning rate, batch size, momentum, and weight decay, arXiv:1803.09820 (2018).

[72] G. Kasieczka, T. Plehn, and M. Russel, Top quark tagging reference dataset, https://doi.org/10.5281/zenodo.2603256.

[73] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen, and P. Z. Skands, An introduction to PYTHIA8.2, Comput. Phys. Commun. **191**, 159 (2015).

[74] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi (DELPHES 3 Collaboration), DELPHES 3, A modular framework for fast simulation of a generic collider experiment, J. High Energy Phys. 02 (2014) 057.

[75] M. Cacciari, G. P. Salam, and G. Soyez, The anti-k_t jet clustering algorithm, J. High Energy Phys. 04 (2008) 063.

[76] A comprehensive comparison between a wide range of machine learning approaches on this top tagging dataset is presented in Ref. [77], in which an earlier version of ParticleNet is also included.

[77] G. Kasieczka *et al.*, The machine learning landscape of top taggers, SciPost Phys. **7**, 014 (2019).

[78] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, Aggregated residual transformations for deep neural networks, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, Honolulu, HI, 2017) pp. 1492–1500, https://ieeexplore.ieee.org/document/8100117.

[79] K. He, X. Zhang, S. Ren, and J. Sun, Identity mappings in deep residual networks, in *Computer Vision—ECCV 2016*, edited by B. Leibe, J. Matas, N. Sebe, and M. Welling (Springer, Cham, 2016), pp. 630–645.