# Particle Convolution for High Energy Physics

Chase Shimmin
Department of Physics
Yale University
New Haven, CT 06511
chase.shimmin@yale.edu

July 6, 2021

## Abstract

We introduce the Particle Convolution Network (PCN), a new type of equivariant neural network layer suitable for many tasks in jet physics. The particle convolution layer can be viewed as an extension of Deep Sets and Energy Flow network architectures, in which the permutation-invariant operator is promoted to a group convolution. While the PCN can be implemented for various kinds of symmetries, we consider the specific case of rotation about the jet axis the $\eta - \phi$ plane. In two standard benchmark tasks, q/g tagging and top tagging, we show that the rotational PCN (rPCN) achieves performance comparable to graph networks such as ParticleNet. Moreover, we show that it is possible to implement an IRC-safe rPCN, which significantly outperforms existing IRC-safe tagging methods on both tasks. We speculate that by generalizing the PCN to include additional convolutional symmetries relevant to jet physics, it may outperform the current state-of-the-art set by graph networks, while offering a new degree of control over physically-motivated inductive biases.

## 1 Introduction

A common problem in high energy physics is the desire to use state-of-the art machine learning methods to solve real-world problems encountered in physics experiments. Often, the leading edge of machine learning is driven by a standard suite of problems which are motivated by the practical interests of tech industries. For example, convolutional neural networks (CNNs) for image processing and Recurrent Neural Networks (RNNs) for natural language processing are some of the most well-studied and developed areas in deep learning. In an effort to reap the benefits of these advances, physicists have been systematically studying the efficacy of these existing methods when applied to typical problems in our field such triggering, event reconstruction, and object tagging.[1, 2, 3, 4, 5, 6, 7, 8]

However, the structure and underlying processes of the data for which these architectures were developed are often fundamentally different from those present in physics experiments. In order to bridge this gap, physicists have often resorted (with a few exceptions[9, 10, 11, 12]) to remedial measures by re-structuring or re-formatting experimental data from its natural representation to conform with the format required by a specific existing model architecture.

We consider the problem of jet-tagging[13, 14, 15], in which the goal is to discriminate between various types of hadronic decays based on the detailed structure of energy and tracking measurements from the detector. Hadronic jets are the most abundant products of proton collisions at the LHC and are formed when energetic quarks or gluons fragment recursively into lower-energy quarks and gluons until stable particles form, a process known as hadronization. These particles are then observed by the particle detector. Experimentally, jets are defined by a specific clustering algorithm[16] which groups together energy deposits localized within a certain angular scale set by a chosen radius parameter. Jets can be initiated directly from hard QCD processes as well as rare particle decays[17, 18, 19, 20]. The exceedingly large QCD cross sections often lead to a scenario where the rate of background jets overwhelms the signal of interest.

In this work, we will consider two common tasks: quark/gluon (q/g) identification[21, 14, 22, 23, 24], and top-quark tagging[25, 15, 13]. In q/g identification, the goal is to determine whether a jet originated from a final-state quark or gluon, which subsequently fragmented into the pattern of radiation observed by the detector. Top quarks decay before hadronization can occur, and frequently result in three collinear jets, which may overlap significantly in the detector when produced at high momentum. The goal of top-

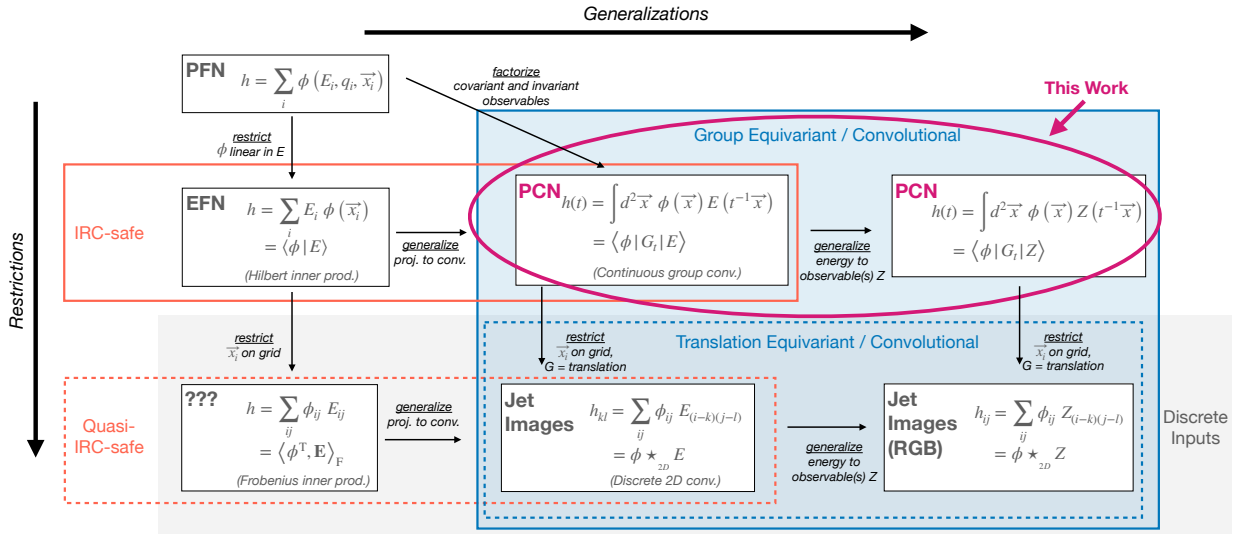


Figure 1: Conceptual map indicating the relationship between ParticleFlow, EnergyFlow, and Jet Image architectures. Particle Convolution (this work) can be seen as a generalization of jet images to continuous space. It can also be seen as a generalization of the EnergyFlow network, where Hilbert space projection has been promoted to (arbitrary) group convolution. Mathematical definitions are provided in section **??**.

tagging is to determine whether or not a large-radius jet pattern originated from a top quark.

In data, a jet is most naturally represented by a variable-sized collection of four-vectors representing momentum as measured by calorimeter deposits and/or charged particle trajectories. These vectors may also be annotated with additional information, such as charge or particle type. The precision with which these constituents are measured depends on the detector properties. For example, charged particle trajectories yield very precise directional measurements but do not include neutral particles, while calorimeter deposits have coarser directional precision but better energy precision.

While the field of HEP has used certain machine learning methods, such as Multilevel Perceptrons[26, 27, 28] and Boosted Decision Trees[29], for decades, these models are not directly amenable to the data in question. Instead, physicists have constructed a large number of mathematical and heuristic *jet substructure observables* which quantify various properties on a per-jet level.[30, 24, 31, 32, 33, 34, 35] These observables may then be used in standard multivariate analysis techniques.

Developments in the field of deep learning have led to a reexamination of the problem of jet tagging, as new architectures have emerged which can operate on constituent-level data. Alternative proposals include casting the particle data into a 2D pixel image in order to apply to a CNNs[4, 5], passing the particle's features one at a time as a sequence into an RNN[2, 1, 9], or embedding groups of nearby constituents into a graph for use in a Graph Neural Networks[36]. While RNNs in particular have seen successful application in experiments, more recently the field has been moving to permutation-invariant set based networks[37, 12] which are easier to train and achieve comparable performance.

Until now, the question of equivariance properties has not received much attention from the field. The CNN-based jet image approach, which respects an approximate 2D translational symmetry, was for a long time the only example of an equivariant architecture that had been studied. While translation equivariance has proved highly useful in image processing tasks, it is unclear whether it is physically meaningful in the context of jet substructure. Very recently, it has been proposed to consider a more physically-relevant class of equivariance, specifically, the Lorentz Group Network[10]. The LGN architecture operates at the constituent level and is fully equivariant with respect to arbitrary lorentz transformations. However, the LGN has not yet proved to work as well as existing methods, possibly due to the exceedingly large memory structures required to implement sufficiently complex networks.

In this paper, we turn our attention to the curious gap between the PFN and CNNs. PFNs, which are the current state-of-the-art in most experimental applications, generally perform as well as any other

methods (with the possible exception of the newly-proposed GNNs), but they possess no particular type of equivariance. On the other hand, the CNN approach which does exhibit equivariance, is generally outperformed by the other methods mentioned, possibly due to the sparsity/discretization or due to the equivariance being of the wrong type.

It turns out there is a specific connection between the PFN and the CNN architectures. They can be viewed from a common mathematical perspective, illustrated schematically in Fig. 1. This mathematical connection, which is elaborated in Sec. 3 and 4, is essentially that with a subtle modification, PFNs can be viewed as the geometrical operation of projection. This projection operation then can be easily promoted to a convolution, which we call Particle Convolution. The Particle Convolution allows us to build networks which feature equivariance with respect to a much larger class of symmetry groups, while operating directly on the constituent-level data. We show that the jet image method is a special case of Particle Convolution in the case of a discrete shift operator and binned coordinates, while the EFN is a special case in which the operator is nullary.

In Sec. 2 we will discuss the concept of equivariance, and motivate the particular case of rotational equivariance for the problem of jet tagging. We then consider in Sec. 3 a formal connection between the permutation-invariant set-based models and the notion of Hilbert space projection. In Sec. 4, we demonstrate how to promote these projective operations to convolutions possessing equivariance properties by construction, using the particular example of rotation. In Sec. 5, we consider some of the technical challenges in implementing Particle Convolution, and detail a more efficient solution based on the notion of steerable functions. In Sec. 6 we provide details for experiments conducted on the two benchmark tasks, q/g tagging and top-tagging, and in Sec. 7 we present results and conclusions.

## 2 Equivariance

In this section, we begin with a formal definition of the mathematical concept of equivariance. Then, before proceeding to the technical details of how this definition can be applied, we present an intuitive argument for how equivariance can benefit machine learning models in the specific application of jet tagging.

A map $f : X \to Y$ is *equivariant* with respect to a group $G$ acting on $X$ if for every $g \in G$, there exists some $\Pi_g : Y \to Y$ such that:

$$f(g \cdot x) = \Pi_g f(x)\,, \ \forall x \in X\,. \tag{1}$$

In other words, given an equivariant function $f$, it is possible to determine the result of $f(g \cdot x)$ by applying either $g$ to the input, or $\Pi_g$ to the output of the function. Note that invariance is a special case of equivariance: the function $f$ is said to be invariant when $\Pi_g$ is equal to the identity for every $g$.

The most commonly known equivariant neural architecture is the Convolutional Neural Network (CNN). These networks are equivariant with respect to discrete translational shifts in one or more dimensions. In particular, two dimensional CNNs excel at computer vision tasks. The intuitive reason behind this is that CNNs can learn generic features, such as textures or edges, and is able to match any of its learned features at any location on an image. If a particular feature is shifted to another location in the image, the CNN's response to that feature will be shifted a corresponding amount. In contrast, a simple fully-connected neural network would need to re-learn a new instance of each feature

Many recent developments in the field of machine learning have focused on the analysis and design of equivariance properties of neural networks. In many cases, equivariance can be generalized to aribrary homogeneous spaces via group convolution, and these architectures often lead to improved performance when a relevant symmetry of the data can be exploited[38, 39, 40].

In the context of jet tagging, we propose to consider the specific case of rotation. The *rotational* Particle Convolution Network (rPCN), detailed in Sec. 4, evaluates a different kind of convolution with respect to rotation about the jet axis in the $\eta - \phi$ plane. This case is physically relevant and also happens to be mathematically simple. The rationale for this particular equivariance stems from the physical processes governing jet formation, which are approximately invariant under rotation about the jet axis. Therefore, common features may emerge in radiation patterns which differ only in arbitrary rotation about the jet axis.

To address this, some works [4] have proposed removing the rotational degree of freedom via a preprocessing step that imposes a standardized reference orientation for all jets. This technique can in some cases improve performance of non-equivariant models, although in other cases it can have a detrimental effect[**?**]. In any case, it effectively manages to render the entire model invariant under *global* rotations.

In many applications we do in fact desire an invariant network response; for example, when determining whether a jet originated from a boosted $Z$ boson decay, we expect the same answer regardless of the random orientation of the parent particle's decay axis. How-

ever, rather than constructing an invariant input to eliminate the rotational variation in the data, we can instead attempt to preserve the structure of the underlying symmetry within the network itself by enforcing equivariance at each layer. This allows a deep network to build rich representations to examine and compare the angular structure of various features.

In the following sections, we will describe one way in which a rotationally-equivariant network can be constructed. This architecture, which we call a Particle Convolution Network, can be generalized to effect equivariance with respect to additional types symmetry as well.

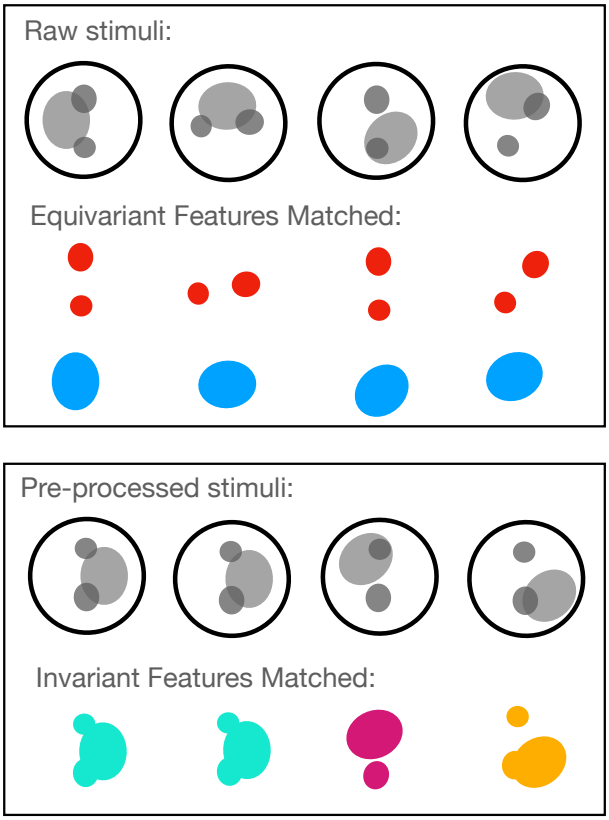


Figure 2: Illustration of invariant vs. equivariant feature matching. Colors correspond different learned filter channels. In this example, the input stimuli are a superposition of two radiation patterns, each of which has an arbitrary angular rotation. The invariant approach is able to match inputs differing by a global rotation; however, it must learn different features for each relative orientation. A network with equivariant filters can match each pattern regardless of relative orientation.

Figure 2 illustrates this point. In this example, we consider two independent features that might appear in a jet: a diffuse blob of low-energy particles, and a pair of high-$p_{\mathrm{T}}$ subjets. If the diffuse particles and the hard subjets are produced at different stages during parton showering, their relative angular orientations about the jet axis could be largely random and uncorrelated. Rotational pre-processing does effectively limit the amount the network needs to learn in the case where two jets differ only by a global rotation. However, such a network still needs to learn a completely different set of features to recognize various relative orientations between the hard and soft particles. On the other hand, an equivariant network might learn a single feature representing "two subjets" and one feature for "diffuse lobe", and by construction understands that these two features independently have a rotational degree of freedom. Having learned such features, the equivariant model would easily detect either element of the substructure at arbitrary angles, and can pass this information to deeper layers within the network. Subsequent layers could then proceed to execute computations which reason about either the absolute or relative angular position of the features.

This is precisely the intuition behind the rotational convolution: features in the jet may be matched by evaluating the projection of the jet onto a series of filters representing learned features. These projections are sampled along a range of rotational orientations, resulting in a periodic "waveform" representing each individual filter's response as a function of angle. At this point, the network has formed a representation consisting of a discrete 1D waveform with multiple channels corresponding to the different filters.

## 3 Particle Projection

In order to define the Particle Convolution, we begin by examining the structure of the "Deep Sets"-based Energy Flow and Particle Flow networks (EFN and PFN). These networks operate on a jet $S$ represented by a set $S = \{s_1, \ldots, s_{|S|}\}$ of observations $s_i$. We shall assume the observations $s_i = (\vec{x}_i, q_i)$ are composed of a 2D coordinate $\vec{x}_i$ representing the direction of particles relative to the jet axis in the $\eta - \phi$ plane[1], and some non-coordinate quantities $q_i$. The non-coordinate observable $q_i = (e_i, \ldots)$ is composed of at least an energy $e_i$ and may be supplemented by additional observables such as charge, mass, etc.

The EFN depends only on the particle coordinate

[1]To simplify notation, we will always assume jets have been centered in the $\eta - \phi$ plane, so that the coordinate $\vec{x} = (\Delta\eta, \Delta\phi) = (\eta, \phi)$.

$\vec{x}_i$ and energy $e_i$, and is defined as:

$$\mathrm{EFN}(S) = F\left(\sum_{k=1}^{|S|} e_k \Phi(\vec{x}_k)\right), \tag{2}$$

where $F$ and $\Phi$ are arbitrary continuous functions, generally represented by neural networks.

Since the learnable function $\Phi$ is defined over (approximately) Euclidean spatial coordinates, we might consider it as an element of the Hilbert space $L^2(\mathbb{R}^2)$. If we likewise consider the values of the non-coordinate observables of the jet $S$ to be represented as a function on $\mathbb{R}^2$, we can "expand" the jet in the continuous basis $|\vec{x}\rangle$. That is, for each collection $S$, we define an associated representation $|Z\rangle_S$ such that

$$\langle \vec{x}|Z\rangle_S = \mathcal{Z}_S(\vec{x}) := \sum_{k=1}^{|S|} Z(q_k)\delta(\vec{x} - \vec{x}_k), \tag{3}$$

where $Z$ denotes some arbitrary function of the non-coordinate observable(s) $q$. That is, in the coordinate basis, $|Z\rangle_S$ corresponds to a function $\mathcal{Z}_S : \mathbb{R}^2 \to \mathbb{R}$ which represents the spatial distribution of the quantity $Z(q)$ within the jet $S$. This function, $\mathcal{Z}_S$, is of course not continuous, and also highly sparse. Strictly speaking, it is not an element of $L^2(\mathbb{R}^2)$, but rather a generalized function. Nonetheless, we shall show that this analytic expression of the jet and its observables provides a useful mathematical tool.

For instance, if we consider an arbitrary continuous function $\Phi : \mathbb{R}^2 \to \mathbb{R}$, we may exploit the definition of the delta function in order to evaluate the Hilbert space inner product

$$\begin{aligned}\langle \Phi, Z\rangle_S &= \int d^2\vec{x}\ \Phi(\vec{x})\mathcal{Z}_S(\vec{x}) && (4)\\ &= \int d^2\vec{x}\ \Phi(\vec{x}) \sum_k Z(q_k)\delta(\vec{x} - \vec{x}_k) && (5)\\ &= \sum_k Z(q_k)\Phi(\vec{x}_k). && (6)\end{aligned}$$

In the special case $Z = \mathrm{E}$ where $\mathrm{E}(q_k) = e_k$ simply returns the energy of the particle $k$, we have:

$$\langle \Phi, \mathrm{E}\rangle_S = \sum_k e_k \Phi(\vec{x}_k). \tag{7}$$

This last equality can immediately be recognized as the inner term of Eq. 2. In other words, we may consider the EFN to be an arbitrary function $F$ acting on the *projection* of a jet's empirical energy distribution $|\mathrm{E}\rangle_S$ onto a learned filter $|\Phi\rangle$:

$$\mathrm{EFN}(S) = F\big(\langle \Phi, \mathrm{E}\rangle_S\big). \tag{8}$$

In the following section, we will exploit this interpretation in order to define general equivariant convolution operators. But first, we will consider the PFN from this perspective. The PFN as defined in **??** is given by

$$\mathrm{PFN}(S) = F\left(\sum_k \Phi(\vec{x}_k, q_k)\right). \tag{9}$$

Comparing with Eq. 2, it is clear that the EFN is a special case of this PFN, where $\Phi$ is linear in the particle energy $e_k$. It is this linearity which guarantees IRC-safety in the EFN case. However, it is the fact that $\Phi$ depends only on $\vec{x}$ that allows the projection Eq. 4 to be related to the EFN. This is because the inner product is defined in terms of an integral with respect to a meaningful topological measure – in this case, two dimensional Euclidean space. As we shall see, this is important for defining convolution with respect to locally compact groups, such as rotation and translation.

Therefore, in order to extend the concept of projection to the more general case, we define a modified version of the PFN directly in terms of the projection operator of Eq. 4:

$$\begin{aligned}\mathrm{PFN}'(S) &= F\left(\langle \Phi, Z\rangle_S\right) && (10)\\ &= F\left(\sum_k Z(q_k)\Phi(\vec{x}_k)\right). && (11)\end{aligned}$$

Here, $F$, $Z$, and $\Phi$ are all arbitrary continuous functions that could be implemented via neural networks. This network is equivalent to a PFN where the per-particle function is required to be separable into terms depending on the coordinate and non-coordinate observables. In principle, this represents a strictly less general model than the original PFN. However, in this form the PFN$'$ readily admits generalization via convolution, which we shall find can result in much more effective models.

## 3.1 Equivariant Projections

Before proceeding to define the Particle Convolution, we first make some observations about the potential for equivariance in the EFN and PFN$'$ architectures, which are based on projection. In particular, it is straightforward to show that whenever $\Phi$ possesses a specific form of equivariance, so does its projection. Suppose $\Phi$ is equivariant with respect to $G$ so that $\Phi(g \cdot x) = \Pi_g \Phi(x)$ for all $x \in X$ and $g \in G$. Then, if

$|Z\rangle_S \to g\,|Z\rangle_S$, we have:

$$\langle\Phi|Z\rangle_S \to \langle\Phi|g|Z\rangle_S \tag{12}$$
$$= \int dx\ \Phi(g\cdot x)\mathcal{Z}_S(x) \tag{13}$$
$$= \int dx\ \Pi_g\Phi(x)\mathcal{Z}_S(x) \tag{14}$$
$$= \Pi_g\,\langle\Phi|Z\rangle_S\ . \tag{15}$$

So we can see that the projection is also equivariant to transformations $g$ applied to the particles of $|Z\rangle_S$.

In practice this is often of limited utility. For example, if $\Phi$ is constant on the $\eta-\phi$ plane, it is invariant *w.r.t.* translations, and clearly so is the projection. If $\Phi$ has some specific periodic behavior under rotations, so that in polar coordinates $\Phi(r,\theta+\delta)=\Phi(r,\theta)$ for some $\delta$, then the projection will also have this periodicity. However, such filters are able to express only limited pattern-matching ability, and may not lead to sufficiently complex representations for the task at hand.

In the following section, we will see that by working with convolutions, arbitrary filters can be learned while retaining equivariance.

# 4 Particle Convolution

Having re-cast the core operation of Particle Flow Networks in terms of the geometric concept of projection, we can immediately generalize the network by promoting projection to convolution. In this section, we demonstrate this concretely with the example of rotational convolution.

The rotational Particle Convolution between a jet $|Z\rangle_S$ and filter $\langle\Phi|$ is defined as the function:

$$h(\Delta;\ \Phi,Z,S) := [\Phi\star Z]_S(\Delta) \tag{16}$$
$$= \langle\Phi|R_\Delta|Z\rangle_S \tag{17}$$
$$= \int d^2\vec{x}\ \Phi(R_\Delta\vec{x})\mathcal{Z}_S(\vec{x}) \tag{18}$$
$$= \sum_k Z(q_k)\Phi(R_\Delta\vec{x}_k)\,, \tag{19}$$

where $\Delta$ is the angle about the jet axis, and $R_\Delta\in SO(2)$ is the corresponding rotation operator. We give the convolution the handle $h(\Delta)$, omitting the independent arguments $(\Phi,Z,S)$ when convenient, to emphasize that the result of the convolution is a *function* of angle.

Let us examine the equivariance of this operation with respect to a rotation of the input particles $|Z\rangle_S$:

$$|Z\rangle_S \to R_\delta|Z\rangle_S \implies \tag{20}$$
$$h(\Delta) \to \langle\Phi|R_\Delta\big[\ R_\delta|Z\rangle_S\big] \tag{21}$$
$$= \langle\Phi|R_{\Delta+\delta}|Z\rangle_S \tag{22}$$
$$= h(\Delta+\delta) \tag{23}$$
$$= T_{-\delta}\ h(\Delta)\,, \tag{24}$$

where $T$ is the coordinate shift operator acting on the function $h$, defined by $T_y f(x)=f(x-y)$.

Since this is true for every $R_\delta\in SO(2)$ and every $\Delta$, we have established the equivariance of the convolution $h(\Delta)$ with respect to the group $SO(2)$ acting on the jet $|Z\rangle_S$. Moreover, we see that rotation of the input particles corresponds to a shift of the output convolution.

This convolution operation comprises the first layer of the rPCN. The network parameters of this layer are encoded by the functions $\Phi$ and $Z$, which could for instance be themselves neural networks. Generally, we would like to pass the result of this first layer to additional layers in a deep neural network. However, it is not clear what to do with a continuous function.

In practice, the convolution can be *sampled* at $n$ discrete points $\Delta_i = 2\pi i/n$. In this case, we can represent the convolution $h$ as a tensor with index $i$ representing the sampled points of the convolutional "waveform":

$$h_i = \langle\Phi|R_{\Delta_i}|Z\rangle_S = \sum_k Z_S(q_k)\Phi(R_{\Delta_i}\vec{x}_i)\,. \tag{25}$$

The sampled convolution is now, strictly speaking, equivariant with respect to the subgroup of discrete rotations by $2\pi i/n$:

$$|Z\rangle_S \to R_{\frac{2\pi j}{n}}\,|Z\rangle_S \implies \tag{26}$$
$$h_i \to \langle\Phi|\,R_{\frac{2\pi(i+j)}{n}}\,|Z\rangle_S = h_{i+j}\,. \tag{27}$$

Hence, a discrete rotation of the particles in $S$ corresponds to a cyclic permutation of the indices $h_{i+j}$.

By sampling a larger number of points $n$, the network can approximate continuous equivariance. Conversely, by limiting the number of samples in accordance with the Shannon-Nyquist theorem, the network can be designed so as to ignore high frequency information which might be considered noise.

Having obtained a shift-equivariant tensor $h_i$, it is straightforward to build deeper equivariant representations by apply the standard 1D CNN layer. The discrete 1D convolution layer is shift equivariant; however, care must be taken to also enforce the cyclic boundary conditions. This can be done by appropriate padding of the inputs: for a convolution with kernel

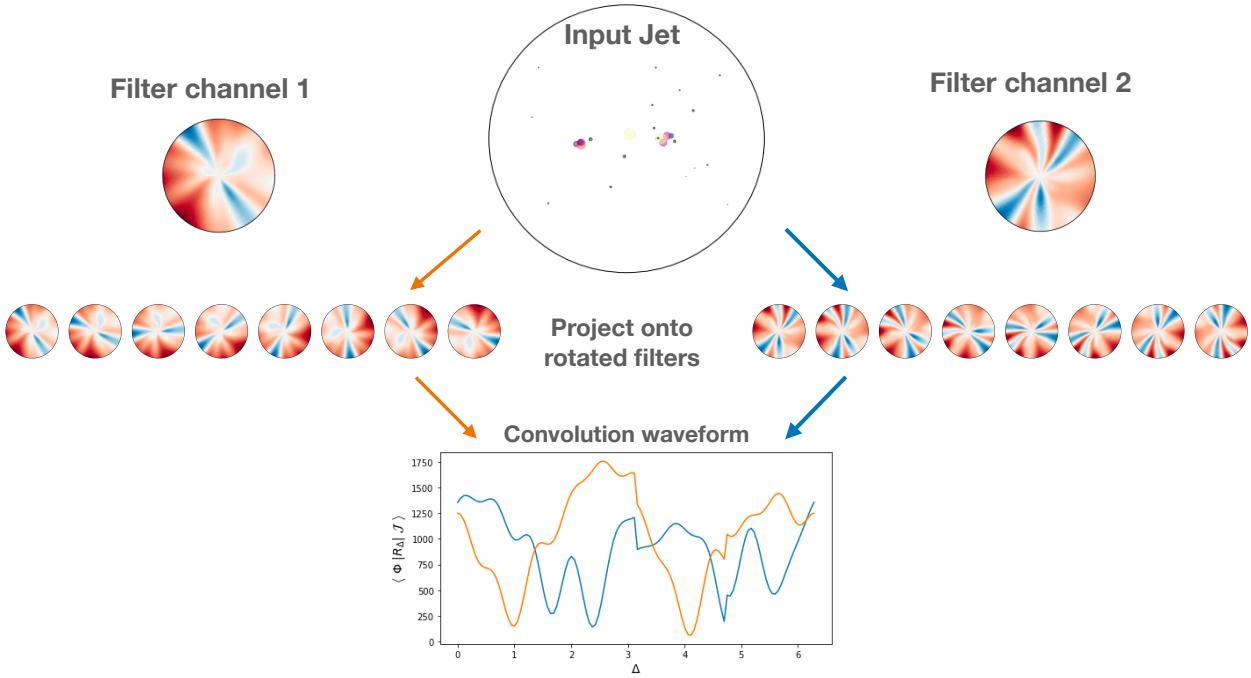


Figure 3: Illustration of the particle convolution of an example jet, represented as a point cloud, and two different filters.

size $k$, the tensor $h = (h_1, \ldots, h_n)$ should be extended as follows:

$$h^+ = (h_{(n-k+1)/2}, \ldots, h_n \Big| h_1, \ldots, h_n \Big| h_1, \ldots, h_{(k-1)/2}) \,. \tag{28}$$

Note that in contrast to conventional CNNs, the number of samples in the convolution output is not reduced, but rather stays the same due to the periodic boundary condition. However, it is possible to reduce the tensor along its sample axis via downsampling.

After processing by any number of additional convolutional layers, an invariant representation may be formed by a global pooling operation. For example, it is clear that taking the maximum or average value of $h_i$ along the sample axis yields an invariant quantity. After the pooling operation, each filter channel yields a single quantity describing an invariant feature of the input jet. Once the invariant is formed, any additional functions applied will also result in invariants. For example, the collection of invariant filter responses may be passed to densely connected layers, and finally to whatever output is suitable for the task objective.

## 5 Steerable Convolutions

In Sec. 4, we defined the rotational Particle Convolution operation. This convolution can in principle be sampled by applying a series of rotations to the particle coordinates $R_{\Delta_i}\vec{x}_k$, and re-evaluating $\Phi$ in the projection of Eq. 25. In practice, this can be problematic as a sizeable neural network may be required to model $\Phi$. For example, in some experiments described in Sec. 6, we consider architectures in which $\Phi$ must be applied to up to 150 particles, and re-sampled at up to 21 orientations. Therefore, even at a moderate batch size of 64 jets, a single network layer of 128 units results in a 32-bit tensor occupying nearly one GiB in memory. This means that models based on direct sampling of convolutions cannot scale well due to memory limitations of current GPUs, and are also very time-intensive to train and optimize.

In this section, we show how the convolution may be implemented more efficiently using *steerable functions*[41, 39]. A function is said to be steerable if it can be expressed as a linear combination of equivariant functions[41]. By structuring a PCN such that the learnable filters $\Phi$ are expressed in an appropriate equivariant basis, it is possible to efficiently sample convolutions at arbitrary points by evaluating $\Phi$ only once.

This works by imposing some particular structure on our learnable filters. Again, we will demonstrate with the specific case of rotation. Consider for example, a filter $\psi$ of the form:

$$\phi_m(\vec{x}) = \rho_m(r)e^{im\theta} \,, \tag{29}$$

where $m$ is an arbitrary integer, $\rho_m$ is a arbitrary function, and $r$ and $\theta$ are polar coordinates about the jet axis in the $\eta - \phi$ plane. For any filter of this form, we can see that the rotation operator acts as:

$$\phi_m(R_\Delta^{-1}\vec{x}) = \rho_m(r)e^{im(\theta-\Delta)} = e^{-im\Delta}\phi_m(\vec{x})\,. \quad (30)$$

We can easily sample the convolution with this function at any point $\Delta$ in terms of the un-rotated projection:

$$\begin{aligned} h(\Delta;\phi_m,Z,S) &= \langle\phi_m|R_\Delta|Z\rangle_S && (31)\\ &= \sum_k \phi_m(R_\Delta\vec{x}_k)Z(q_k) && (32)\\ &= \sum_k e^{im\Delta}\phi_m(\vec{x}_k)Z(q_k) && (33)\\ &= e^{im\Delta}\langle\phi_m|Z\rangle_S\,. && (34)\end{aligned}$$

In other words, we need only compute $h(0)$ once and then use $h(\Delta) = e^{im\Delta}h(0)$. This is, of course, an example of an equivariant projection as described in Sec. 3.1.

Unfortunately, the filter $\phi_m$ has a definite angular frequency $m$. As mentioned in Sec. 3.1, the pattern-matching ability of such a filter is quite limited. However, this suggests we could exploit a similar type of equivariance by imposing a specific structure related to Eq. 29 to construct a more general $\Phi$.

In particular, let us re-define $\Phi$ in terms of a series of functions $\phi_m$:

$$\Phi(\vec{x}) = \Phi(r,\theta) = \sum_{m=-M}^{M}\phi_m(r,\theta) = \sum_m \rho_m(r)e^{im\theta}\,. \quad (35)$$

Here, the hyperparameter $M$ represents a cutoff for angular frequencies captured by the filter. The $2M+1$ radial functions $\rho_m$ are now the arbitrary learnable components of $\Phi$, which could be represented by ordinary neural networks, radial basis functions, *etc.* The relationship between $\Phi$ and this hyperparameter is depicted in Fig 4.

Note that in Eq. 35, $\Phi$ is generally complex-valued, and so are the functions $\rho_m$. For our purposes, we shall constrain it to be real-valued by imposing the condition $\rho_{-m} = \rho_m^\star$. Alternatively, we could define $\Phi$ in terms of sines and cosines; however, the present form allows for simpler mathematical manipulation.

Each of the component functions $\phi_m$ are rotationally equivariant via Eq. 30. However the filter $\Phi$, which combines arbitrarily many frequency modes, does not. The trick is to instead associate $\Phi$ with a column vector

$$\tilde{\Phi} = (\phi_m, ..., \phi_{-m})^{\mathrm{T}}\,, \quad (36)$$

related via:

$$\Phi(\vec{x}) = \mathbf{1}^{\mathrm{T}}\tilde{\Phi}(\vec{x}) = \sum_m \phi_m(\vec{x})\,. \quad (37)$$

Now under a rotation $R_\Delta$, we have:

$$\begin{aligned}\tilde{\Phi}(R_\Delta^{-1}\vec{x}) &= \big(\phi_M(R_\Delta^{-1}\vec{x}),\ldots,\phi_{-M}(R_\Delta^{-1}\vec{x})\big)^{\mathrm{T}} && (38)\\ &= \big(e^{-iM\Delta}\phi_M(\vec{x}),\ldots,e^{iM\Delta}\phi_{-M}(\vec{x})\big)^{\mathrm{T}} && (39)\\ &= \mathbf{A}(\Delta)\tilde{\Phi}(\vec{x})\,, && (40)\end{aligned}$$

where the square matrix

$$\mathbf{A}(\Delta) = \mathrm{diag}\{e^{-iM\Delta},\ldots,e^{iM\Delta}\}\,. \quad (41)$$

In other words, while the function $\Phi$ is not equivariant, the vector of functions $\tilde{\Phi}$ *is*, transforming as $\tilde{\Phi} \to \mathbf{A}(\Delta)\tilde{\Phi}$ under a rotation of coordinates by $\Delta$.

Therefore, in order to perform a convolution with $\Phi$, the network should compute the projection:

$$\langle\tilde{\Phi}|Z\rangle_S := \big(\langle\phi_M|Z\rangle_S,\ldots,\langle\phi_{-M}|Z\rangle_S\big)^{\mathrm{T}}, \quad (42)$$

once for the un-rotated case, and sample additional rotations at points $\Delta_i$ via

$$\tilde{h}_i = \langle\tilde{\Phi}|R(\Delta_i)|Z\rangle_S = \mathbf{A}(\Delta_i)\langle\tilde{\Phi}|Z\rangle_S\,. \quad (43)$$

Finally, the sampled vectors $\tilde{h}_i$ can be "collapsed" into a single numerical value by summing over the $m$-components:

$$h_i = \mathbf{1}^{\mathrm{T}}\tilde{h}_i \quad (44)$$

Note that given a fixed set of sample points $\Delta_i$, the operators $\mathbf{A}(\Delta_i)$ are simply constant, diagonal matrices, and the matrix multiplication of Eq. 43 is a trivial operation for GPUs.

The network based on steerable convolutions is depicted schematically in Figure 5.

Having re-formulated our filter $\Phi$ in terms of the equivariant functions $\phi_m$, we can revisit the question of efficiency and scalability for practical networks. Note that in either case, for a filter $\Phi$ to capture information about given angular frequency $M$, we require $2M+1$ samples for $h_i$. In Eq. 25, we do this by re-evaluating the projection with $\Phi$ $2M+1$ times. In Eq. 43, we evaluate the projection only once, but there are $2M+1$ functions $\phi_m$ which must be evaluated. Therefore, it may seem that nothing has been gained.

However, in practice, the functions $\rho_m$ can be collectively implemented by a single neural network which shares most of its weights. As long as the number of samples required, $2M+1$, is generally less than the size of hidden layers for typical dense networks,

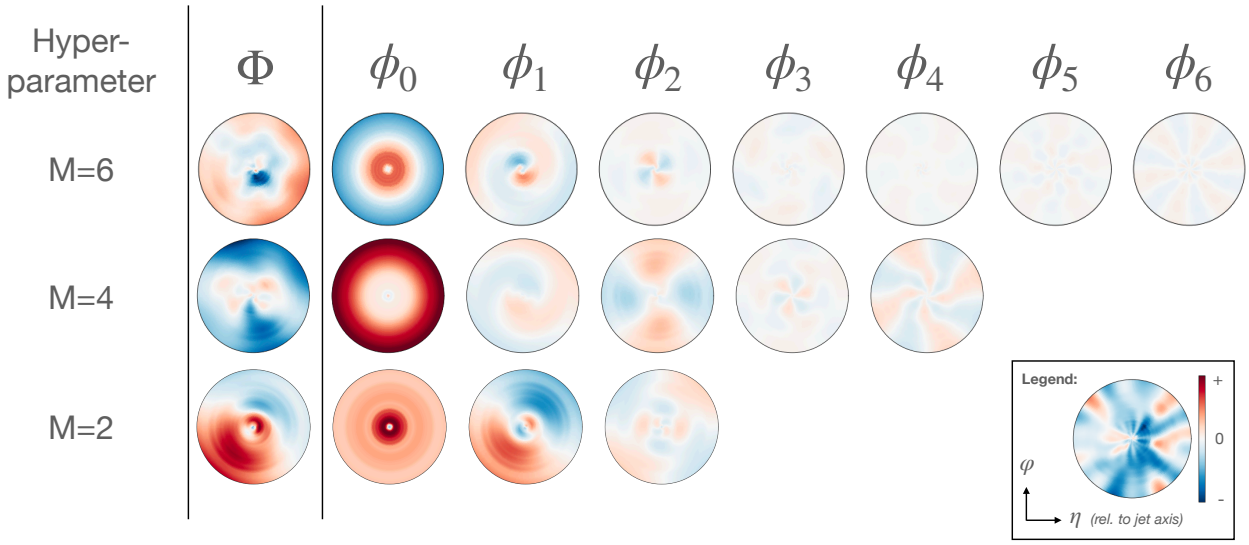


Figure 4: Visualization of the series representation of learned filter instances $\Phi(\vec{x}) = \sum_m \phi_m(r, \theta)$ for various values of the cutoff hyperparameter $M$.

then the network should have a substantially smaller memory footprint. This is because the intermediate tensors for the hidden layers need only be computed for the un-rotated case, as opposed to being calculated at every orientation as in Eq. 25. This allows for smaller networks which are faster to train.

Moreover, the functions $\rho_m$ only need to learn a radial profile, rather than a response on the full 2-dimensional $\eta - \phi$ plane. Therefore, it is reasonable to expect that simpler, smaller networks could be able to achieve equivalent results. Lastly, since $\Phi$ in Eq. 35 is expressed in terms of functions with a definite cutoff frequency $M$, this acts as low-pass filter for angular structure. Without this cutoff, higher frequency information may be aliased to lower modes during sampling, which could lead to a network sensitive to undesirable artifacts.

We find empirically that in either the direct sampling or steerable case, rPCNs of comparable angular resolution and depth achieve similar performance. Therefore, when optimizing hyperparameters for the experiments in section 6, the more efficient steerable convolution architecture is used; however, it is possible that a more exhaustive optimization would find better results via the direct sampling approach.

# 6 Experiments

We conduct experiments in training the rotational Particle Convolution Network for two benchmark problems in jet physics: quark/gluon identification, and top tagging. The details of these datasets are described in Sec. 6.1. For both tasks, we train a general PCN as well as the IRC-safe variant. The coordinates $\vec{x}$ are centered about the jet axis in the $\eta - \phi$ plane, as described in [12], and the scalar feature for each particle is the particle's transverse momentum, *i.e.* $q_k = p_{\mathrm{T},k}$, which is invariant w.r.t. the coordinate-centering operation. In the case of the q/g tagging task, we also test the performance of models when supplied with per-particle identification information as well, $q_k = (p_{\mathrm{T},k}, \mathrm{PID}_k)$.

## 6.1 Datasets

For reference, we consider two typical benchmark problems: q/g identification and top-quark tagging, using the same datasets as in Refs. [12, 36]. The q/g dataset, described in Ref.[12] is comprised of $Z(\nu\nu)+q$ events (signal) and $Z(\nu\nu) + g$ events (background). The events are simulated using Pythia8[42], which performs parton showering and hadronization. No detector simulation is performed for this sample. Particles (excluding neutrinos) are clustered using the anti-$k_{\mathrm{T}}$ algorithm[16] with radius parameter $R = 0.4$. The 4-momenta and particle ID of particles within the leading jet are saved for jets with $p_{\mathrm{T}} \in [500, 550]$ GeV and $|\eta| < 2$. The dataset is split into 1.2 million training events and 400k each of validation and test events. In our experiments, we truncate events with greater than 68 particles by discarding those with the lowest $p_{\mathrm{T}}$.

For the top-tagging benchmark, we use the dataset

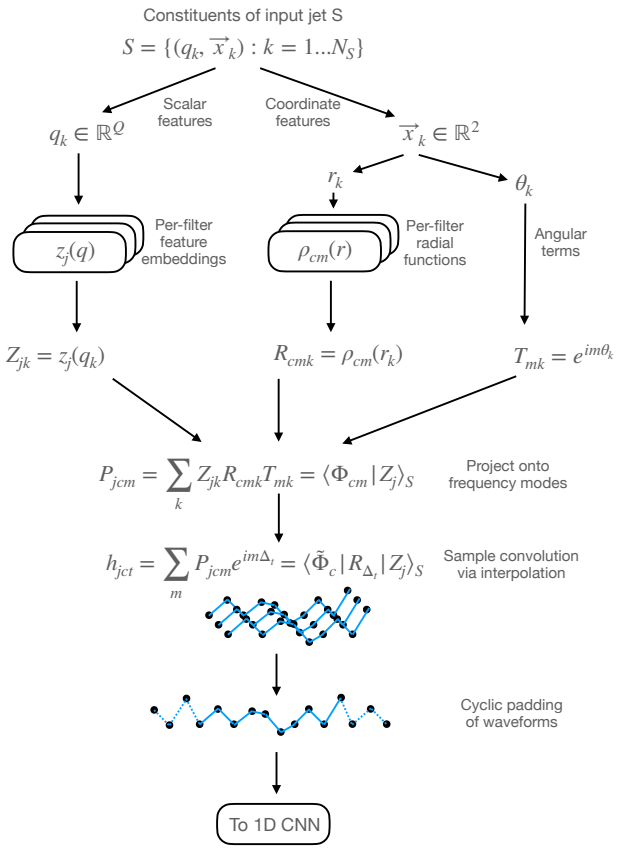


Figure 5: Schematic representation of the rotational Particle Convolution layer as described in Sec. **??**. The functions $z_c$ and $\rho_{cm}$ can be specifically chosen (e.g. when setting $z_c = p_{\mathrm{T}}$ for IRC-safety), or can be implemented as neural networks.

available at Ref.[43]. The events are generated with Pythia8 and passed into Delphes[44] fast detector simulation, without pileup. The signal process is $t\bar{t}$ and the background is inclusive QCD. Jets are reconstructed using Delphes EFlow module, with anti-$k_{\mathrm{T}}$ radius parameter $R = 0.8$, and are required to have $p_{\mathrm{T}} \in [550, 650]$ GeV and $|\eta| < 2$. For each jet, we record the 4-momenta for the leading 140 particle-flow constituents.

## 6.2 Architectures

In previous sections, the discussion has focused on projection and convolution of a jet with a single filter $\Phi$. In practice, we allow a neural network to learn a moderate number of independent filter channels $\Phi_c(r, \theta) = \sum_m \rho_{cm}(r) e^{im\theta}$ and feature embeddings $Z_q$. By convolving each feature embedding with each filter, the network produces a convolution waveform $\langle \Phi_c | R(\Delta) | Z_j \rangle$ with $C \times J$ feature channels, which can be passed on as multi-feature input to a standard 1D CNN architecture. Hence, to define the architecture of a particular rPCN, we must to specify the following:

- The number of filter channels, $C$;
- The number of feature embeddings, $J$;
- The maximum angular frequency mode, $M$;
- The feature embedding function(s), $Z_j(q)$;
- The radial activation function(s), $\rho_{cm}(r)$;
- The remaining 1D convolutional network structure.

Except in the IRC-safe case, we implement both $Z_j$ and $\rho_{cm}$ as densely-connected neural networks. Therefore, $Z_j$ and $\rho_{cm}$ are specified by the number of hidden layers and their units, as well as their nonlinearies. We apply a ReLU nonlinearity at the output of $Z_c$, so that it can more naturally act as an "attention" mechanism which can learn to ignore particles based on their scalar features $q_k$ in the region where the ReLU response is zero. The complex-valued network $\rho_{cm}$ is implemented by a single network which learns real and imaginary parts separately, $\rho_{cm} = \alpha_{cm} + i\beta_{cm}$. We impose the constraint $\rho_{-m} = \rho_m^*$ so that $\Phi_c$ are real by construction; therefore it is sufficient to learn $\alpha_{cm}$ for $m \geq 0$ and $\beta_{cm}$ for $m \geq 1$.

When particle ID is included as part of the scalar input, we follow the "experimentally plausible" labeling scheme of [**?**], where particles are categorized into one of eight types: photons, neutral hadrons, and positively- and negatively-charged muons, electrons, and hadrons. These labels are input to the network via a trainable 3-dimensional embedding layer, which are concatenated with the particle's $p_{\mathrm{T}}$ and passed as a triplet into the $Z_c$ network.

In the IRC-safe case, the function $Z_c$ is simply replaced with the transverse momentum of each particle. Since the majority of trainable parameters in the Particle Convolution network tend to be from the $Z_c$ and $\rho_{cm}$ sub-networks, the IRC-safe networks will usually have substantially fewer parameters than their non-IRC-safe counterparts.

In our experiments, the remainder of the network can be specified by:

- The nonlinearity following the Particle Convolution layer;
- The number of 1D convolution layers, their kernel size, and stride;
- The global pooling operation (maximum or average);

- Any remaining hidden layers, units, and nonlinearity;
- A final 2-unit softmax output layer representing the categorical prediction.

We found that a resnet-like[45] convolutional architecture worked best. We pass the particle convolution output without nonlinearity as a "skip connection" across residual blocks consisting of ReLU activation and batch normalization preceding two 1D CNN layers.

### 6.3 Training

In all experiments, the data are split into train, validation, and test samples in a 6:1:1 ratio. Networks are implemented using Tensorflow v2 [46] and Keras [47]. The loss function used for training is the categorical crossentropy, optimized via Adam[48] with learning rate $10^{-4}$ and batch size 128. The validation loss is used to determine training convergence; training is stopped when the validation loss has not improved for 16 consecutive epochs. The model epoch with the lowest validation loss is evaluated on the test set to report unbiased AUC and rejection metrics.

After training networks by randomly sampling a wide range of hyperparameter configurations, we selected the best-performing architecture for each task based on the validation loss. These configurations were then frozen, and the networks were retrained ten times. The values reported in tables 1 and 2 are the test scores for the specific network whose validation score was nearest to the median value on each task. The errors quoted are the standard deviation of the test scores over the series of ten retrainings.

### 6.4 Inspection

It is possible to visualize the learned functions $\Phi_c(\vec{x})$ in the $\eta - \phi$ plane. Moreover, we can visualize the learned behavior of the nonlinear feature embedding, $Z_j(p_{\mathrm{T}})$. Each convolution represents a learned spatial pattern as well as a gated response based on momentum. Interestingly, it seems that the network often spontaneously learns a sort of binning in $p_{\mathrm{T}}$, as shown in Figure 6. It also tends to ignore many of the lowest-$p_{\mathrm{T}}$ particles, suggesting those particles could potentially be pruned from the input to further speed up the network.

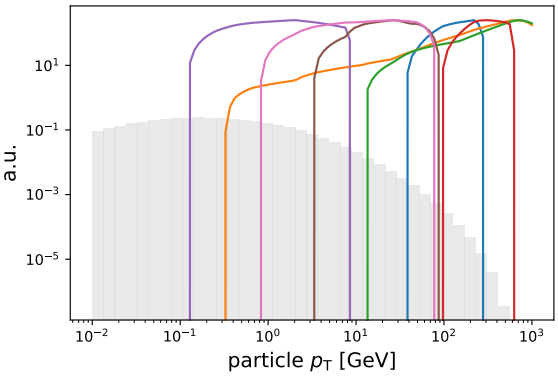


Figure 6: Colored lines representing functions $Z(p_{\mathrm{T}})$ learned by a network on the q/g tagging task. The shaded histogram indicates the distribution of particle $p_{\mathrm{T}}$ over the training dataset.

## 7 Results & Conclusions

The rPCN performance on the two benchmark tasks are given in Tables 1 and 2. These results show that by promoting the permutation-invariant architectures EnergyFlow and ParticleFlow to include rotational convolution significantly improves their performance. In particular, the rPCN significantly improves the state-of-the-art for IRC-safe taggers. When IRC safety can be dispensed with, the rPCN can achieve performance comparable to the current state-of-the-art set by graph-based models such as ParticleNet.

From a practical perspective, the rPCN is much more similar to deep-sets based architectures already in production use by LHC experiments, as compared to the relatively novel graph architectures. The rPCN model also requires minimal preprocessing, is compatible with arbitrary-length collections of particles, and is permutation-equivariant. PCNs in general are conceptually analogous to the jet image approach in the limit of infinitely small pixels and continuous filter kenels, although the rPCN targets rotational rather than translational equivariance as a physically-motivated inductive bias.

Moreover, by careful design of the filter representation $\Phi$, the PCN approach can admit further generalization by constructing convolutional operations that can endow additional equivariance properties to jet tagging models. For example, in forthcoming work we study whether a PCN which is equivariant *w.r.t.* to scaling in the $\eta - \phi$ plane could potentially be useful to capture similar substructure properties across a wide range of jet $p_{\mathrm{T}}$.

## 8 Acknowledgements

The author would like to especially thank Paul Tipton for many iterations of feedback throughout this

project. He thanks Aishik Ghosh, Daniel Whiteson, Dan Guest, and Ema Smith for helpful conversations and comments on this manuscript. Training experiments were made possible by the Grace GPU cluster operated by Yale Center for Research Computing. This work was supported by grant DE-SC0017660 funded by the U.S. Department of Energy, Office of Science.

# References

[1] Taoli Cheng. Recursive Neural Networks in Quark/Gluon Tagging. *Comput. Softw. Big Sci.*, 2(1):3, 2018.

[2] Shannon Egan, Wojciech Fedorko, Alison Lister, Jannicke Pearkes, and Colin Gay. Long Short-Term Memory (LSTM) networks with jet constituents for boosted top tagging at the LHC. 11 2017.

[3] Hui Luo, Ming-xing Luo, Kai Wang, Tao Xu, and Guohuai Zhu. Quark jet versus gluon jet: fully-connected neural networks with high-level features. *Sci. China Phys. Mech. Astron.*, 62(9):991011, 2019.

[4] Luke de Oliveira, Michael Kagan, Lester Mackey, Benjamin Nachman, and Ariel Schwartzman. Jet-images — deep learning edition. *JHEP*, 07:069, 2016.

[5] Josh Cogan, Michael Kagan, Emanuel Strauss, and Ariel Schwarztman. Jet-Images: Computer Vision Inspired Techniques for Jet Tagging. *JHEP*, 02:118, 2015.

[6] Dan Guest, Kyle Cranmer, and Daniel Whiteson. Deep Learning and its Application to LHC Physics. *Ann. Rev. Nucl. Part. Sci.*, 68:161–181, 2018.

[7] Boosted jet identification using particle candidates and deep neural networks. Nov 2017.

[8] Jonathan Shlomi, P. Battaglia, and Jean-Roch Vlimant. Graph neural networks in particle physics. *arXiv: High Energy Physics - Experiment*, 2020.

[9] Gilles Louppe, Kyunghyun Cho, Cyril Becot, and Kyle Cranmer. QCD-Aware Recursive Neural Networks for Jet Physics. *JHEP*, 01:057, 2019.

[10] Alexander Bogatskiy, Brandon Anderson, Jan Offermann, Marwah Roussi, David Miller, and Risi Kondor. Lorentz group equivariant neural network for particle physics. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119. PMLR, 2020.

[11] Eric A. Moreno, Olmo Cerri, Javier M. Duarte, Harvey B. Newman, Thong Q. Nguyen, Avikar Periwal, Maurizio Pierini, Aidana Serikova, Maria Spiropulu, and Jean-Roch Vlimant. JEDI-net: a jet identification algorithm based on interaction networks. *Eur. Phys. J. C*, 80(1):58, 2020.

| | Model | AUC | $1/\epsilon_b\|_{\epsilon_s=50\%}$ |
|---|---|---|---|
| *IRC-safe* | | | |
| | EFN | 0.8824 | $28.6 \pm 0.3$ |
| | rPCN **(Ours)** | **0.8944** | $\mathbf{32.5 \pm 0.4}$ |
| *w/o PID* | | | |
| | PFN | 0.8911 | 30.8 $\pm$0.4 |
| | ResNeXt-50 | 0.8960 | 30.9 |
| | P-CNN | 0.8915 | 31.0 |
| | ParticleNet-Lite | 0.8993 | 32.8 |
| | ParticleNet | **0.9014** | 33.7 |
| | rPCN **(Ours)** | 0.8997 | $\mathbf{34.2 \pm 0.4}$ |
| *w/ PID* | | | |
| | P-CNN | 0.9002 | 34.7 |
| | PFN-Ex | 0.9005 | 34.7 |
| | ParticleNet-Lite | 0.9079 | 37.1 |
| | rPCN +Ex **(Ours)** | 0.9081 | $38.6 \pm 0.5$ |
| | ParticleNet | **0.9116** | $\mathbf{39.8 \pm 0.2}$ |

Table 1: Comparison of known network architectures on the quark-gluon tagging dataset of Ref. [12]. ResNeXt is a deep 2D CNN adapted for jet images in [36], while P-CNN [7] is a 1D convolution operating on ordered lists of particles, also implemented in [36]. Our IRC-safe rPCN model is substantially more sensitive than the EFN. When including a nonlinear $p_{\mathrm{T}}$ function, the rPCN model achieves comparable performance to ParticleNet.


[12] Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler. Energy Flow Networks: Deep Sets for Particle Jets. *JHEP*, 01:121, 2019.

[13] Roman Kogler et al. Jet Substructure at the Large Hadron Collider: Experimental Review. *Rev. Mod. Phys.*, 91(4):045003, 2019.

[14] Jason Gallicchio and Matthew D. Schwartz. Quark and Gluon Tagging at the LHC. *Phys. Rev. Lett.*, 107:172001, 2011.

[15] Andrew J. Larkoski, Ian Moult, and Benjamin Nachman. Jet Substructure at the Large Hadron Collider: A Review of Recent Advances in Theory and Machine Learning. *Phys. Rept.*, 841:1–63, 2020.

[16] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. The anti-$k_t$ jet clustering algorithm. *JHEP*, 04:063, 2008.

[17] Tilman Plehn, Gavin P. Salam, and Michael Spannowsky. Fat Jets for a Light Higgs. *Phys. Rev. Lett.*, 104:111801, 2010.

[18] A. Abdesselam et al. Boosted Objects: A Probe of Beyond the Standard Model Physics. *Eur. Phys. J. C*, 71:1661, 2011.

[19] Jonathan M. Butterworth, Adam R. Davison, Mathieu Rubin, and Gavin P. Salam. Jet substructure as a new Higgs search channel at the LHC. *Phys. Rev. Lett.*, 100:242001, 2008.

[20] Chase Shimmin and Daniel Whiteson. Boosting low-mass hadronic resonances. *Phys. Rev. D*, 94(5):055001, 2016.

[21] Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler. An operational definition of quark and gluon jets. *JHEP*, 11:059, 2018.

[22] Mrinal Dasgupta, Lorenzo Magnea, and Gavin P. Salam. Non-perturbative QCD effects in jets at hadron colliders. *JHEP*, 02:055, 2008.

[23] Jason Gallicchio and Matthew D. Schwartz. Quark and Gluon Jet Substructure. *JHEP*, 04:090, 2013.

[24] Andrew J. Larkoski, Jesse Thaler, and Wouter J. Waalewijn. Gaining (Mutual) Information about Quark/Gluon Discrimination. *JHEP*, 11:129, 2014.

[25] Andrea Banfi, Gavin P. Salam, and Giulia Zanderighi. Infrared safe definition of jet flavor. *Eur. Phys. J. C*, 47:113–124, 2006.

[26] Hermann Kolanoski. Application of artificial neural networks in particle physics. *Nucl. Instrum. Meth. A*, 367:14–20, 1995.

| | Model | AUC | $1/\epsilon_b\|_{\epsilon_s=50\%}$ | $1/\epsilon_b\|_{\epsilon_s=30\%}$ |
|---|---|---|---|---|
| *IRC-safe* | | | | |
| | EFPs | 0.9803 | 184 | 384 |
| | EFN | 0.9789 | 181 | 619 |
| | rPCN **(Ours)** | **0.9821** | $\mathbf{257 \pm 6}$ | $\mathbf{1038 \pm 41}$ |
| *w/o PID* | | | | |
| | P-CNN | 0.9803 | 201 | 759 |
| | PFN | 0.9819 | 247 | 888 |
| | ResNeXt-50 | 0.9837 | 302 | 1147 |
| | ParticleNet-Lite | 0.9844 | 325 $\pm$5 | 1262 $\pm$49 |
| | rPCN **(Ours)** | 0.9845 | 364 $\pm$9 | $\mathbf{1642 \pm 93}$ |
| | ParticleNet | **0.9858** | $\mathbf{397 \pm 7}$ | $1615 \pm 93$ |

Table 2: Comparison of similar network architectures on the top tagging dataset of Ref. [12]. ResNeXt is a deep 2D CNN adapted for jet images in [36], while P-CNN [7] is a 1D convolution operating on ordered lists of particles, also implemented in [36]. As with the q/g task, the rPCN sets a new state-of-the-art for IRC-safe tagging, while achieving comparable performance to the graph-based ParticleNet model.


[27] S. A. Bass, A. Bischoff, J. A. Maruhn, Horst Stoecker, and W. Greiner. Neural networks for impact parameter determination. *Phys. Rev. C*, 53:2358–2363, 1996.

[28] M. Milek and M. P. Patel. Neural network tagging in a toy model. *Nucl. Instrum. Meth. A*, 425:577–588, 1999.

[29] Byron P. Roe, Hai-Jun Yang, Ji Zhu, Yong Liu, Ion Stancu, and Gordon McGregor. Boosted decision trees, an alternative to artificial neural networks. *Nucl. Instrum. Meth. A*, 543(2-3):577–584, 2005.

[30] Jesse Thaler and Ken Van Tilburg. Identifying Boosted Objects with N-subjettiness. *JHEP*, 03:015, 2011.

[31] Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler. Energy flow polynomials: A complete linear basis for jet substructure. *JHEP*, 04:013, 2018.

[32] Ian Moult, Lina Necib, and Jesse Thaler. New Angles on Energy Correlation Functions. *JHEP*, 12:153, 2016.

[33] Andrew J. Larkoski, Ian Moult, and Duff Neill. Power Counting to Better Jet Observables. *JHEP*, 12:009, 2014.

[34] Mrinal Dasgupta, Alessandro Fregoso, Simone Marzani, and Gavin P. Salam. Towards an understanding of jet substructure. *JHEP*, 09:029, 2013.

[35] Andrew J. Larkoski, Simone Marzani, Gregory Soyez, and Jesse Thaler. Soft Drop. *JHEP*, 05:146, 2014.

[36] Huilin Qu and Loukas Gouskos. ParticleNet: Jet Tagging via Particle Clouds. *Phys. Rev. D*, 101(5):056019, 2020.

[37] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep Sets. *arXiv e-prints*, page arXiv:1703.06114, Mar 2017.

[38] Taco S Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant cnns on homogeneous spaces. *Advances in neural information processing systems*, 32:9145–9156, 2019.

[39] M. Weiler, F. Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant cnns. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 849–858, 2018.

[40] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds. *arXiv e-prints*, page arXiv:1802.08219, February 2018.

[41] Patrick Cheng-San Teo. *THEORY AND APPLICATIONS OF STEERABLE FUNCTIONS*. PhD thesis, stanford university, 1998.

[42] Torbjörn Sjöstrand, Stefan Ask, Jesper R. Christiansen, Richard Corke, Nishita Desai, Philip Ilten, Stephen Mrenna, Stefan Prestel, Christine O.

Rasmussen, and Peter Z. Skands. An introduction to PYTHIA 8.2. *Comput. Phys. Commun.*, 191:159–177, 2015.

[43] Gregor Kasieczka, Tilman Plehn, Jennifer Thompson, and Michael Russel. Top quark tagging reference dataset, March 2019.

[44] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi. DELPHES 3, A modular framework for fast simulation of a generic collider experiment. *JHEP*, 02:057, 2014.

[45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv e-prints*, page arXiv:1512.03385, December 2015.

[46] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[47] François Chollet et al. Keras. `https://keras.io`, 2015.

[48] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980, December 2014.