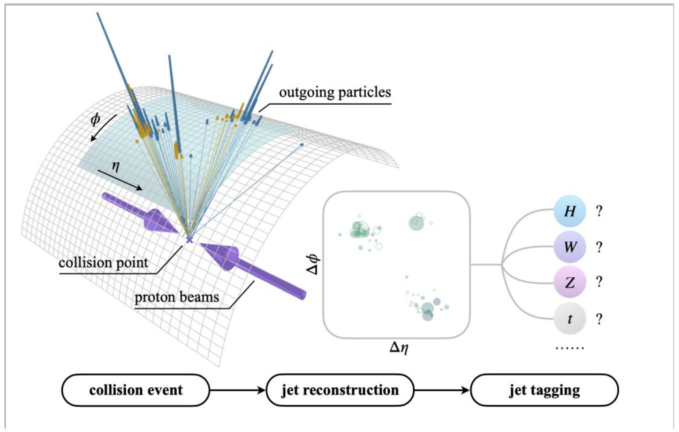
A Review of Machine Learning Applications on Jet Tagging

Konkuk University

Daohan Wang

2023.2.13

Brief motivation of ML in Jet Tagging



Distinguishing boosted heavy particle jets from QCD initiated quark/gluon jet (W/H/Z/top jets, photon-jets…….)

Jet Tagging

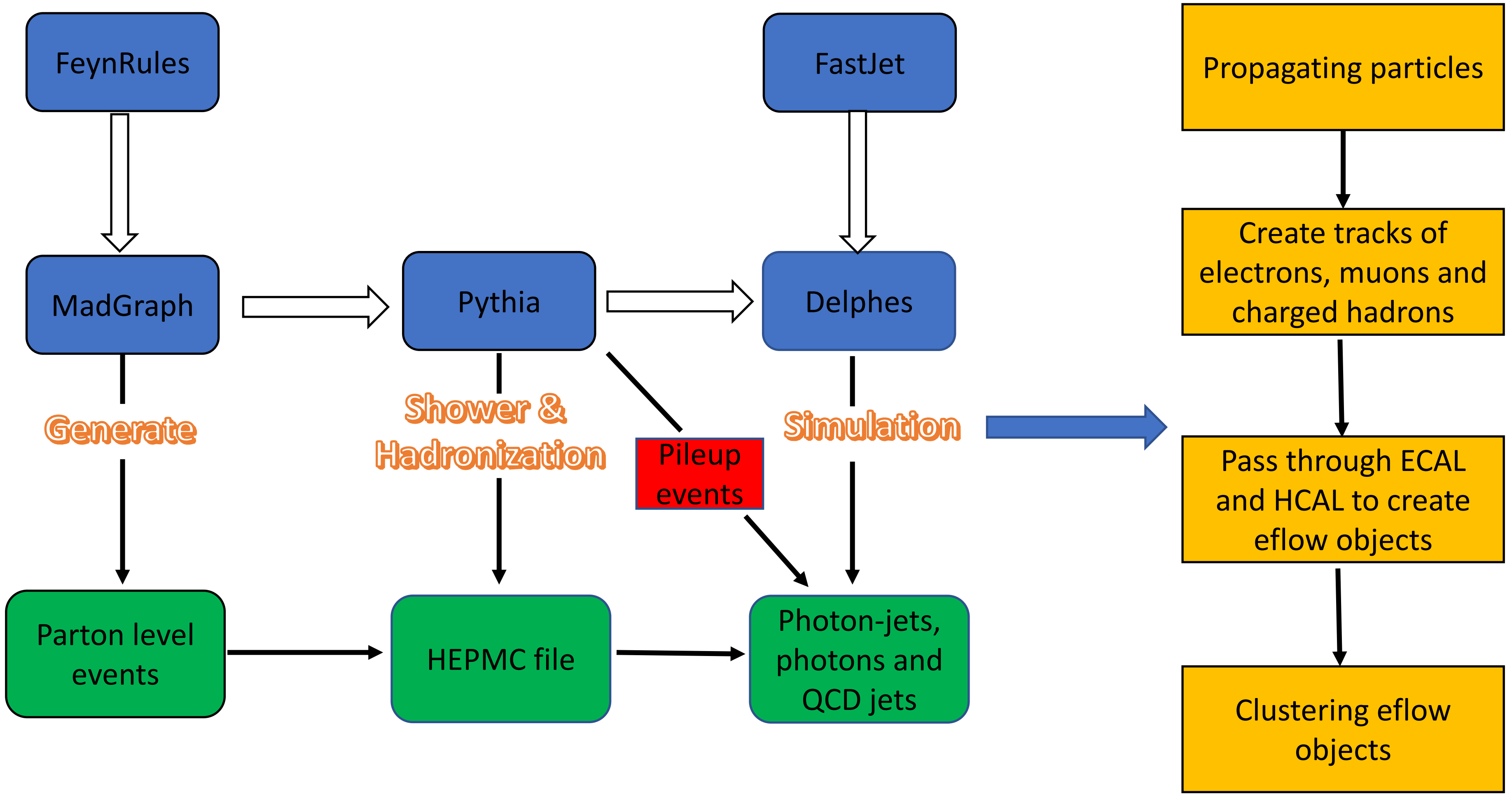
Jet initiated by different particles exhibit different characteristics.

How to represent a jet
How to analyze the representation

- Image-based representation → Convolutional Neural Network
- Particle-based representation →
  - Simple Dense Network
  - Recurrent Neural Network
  - 1D Convolutional Neural Network
- Tree-based representation → Recursive Neural Network
- **Point cloud representation** →
  - Deep Set framework
  - ParticleNet
  - ABCNet
  - LorentzNet
  - Transformer
    - Point Cloud Transformer
    - Particle Transformer

A jet is a spray of particles, produced by the hadronization of a quark/gluon or originate from the decay of high-momenta heavy particles.

Simulation Details



We can use Deep learning to analyze low-level LHC data without constructing high-level observables !

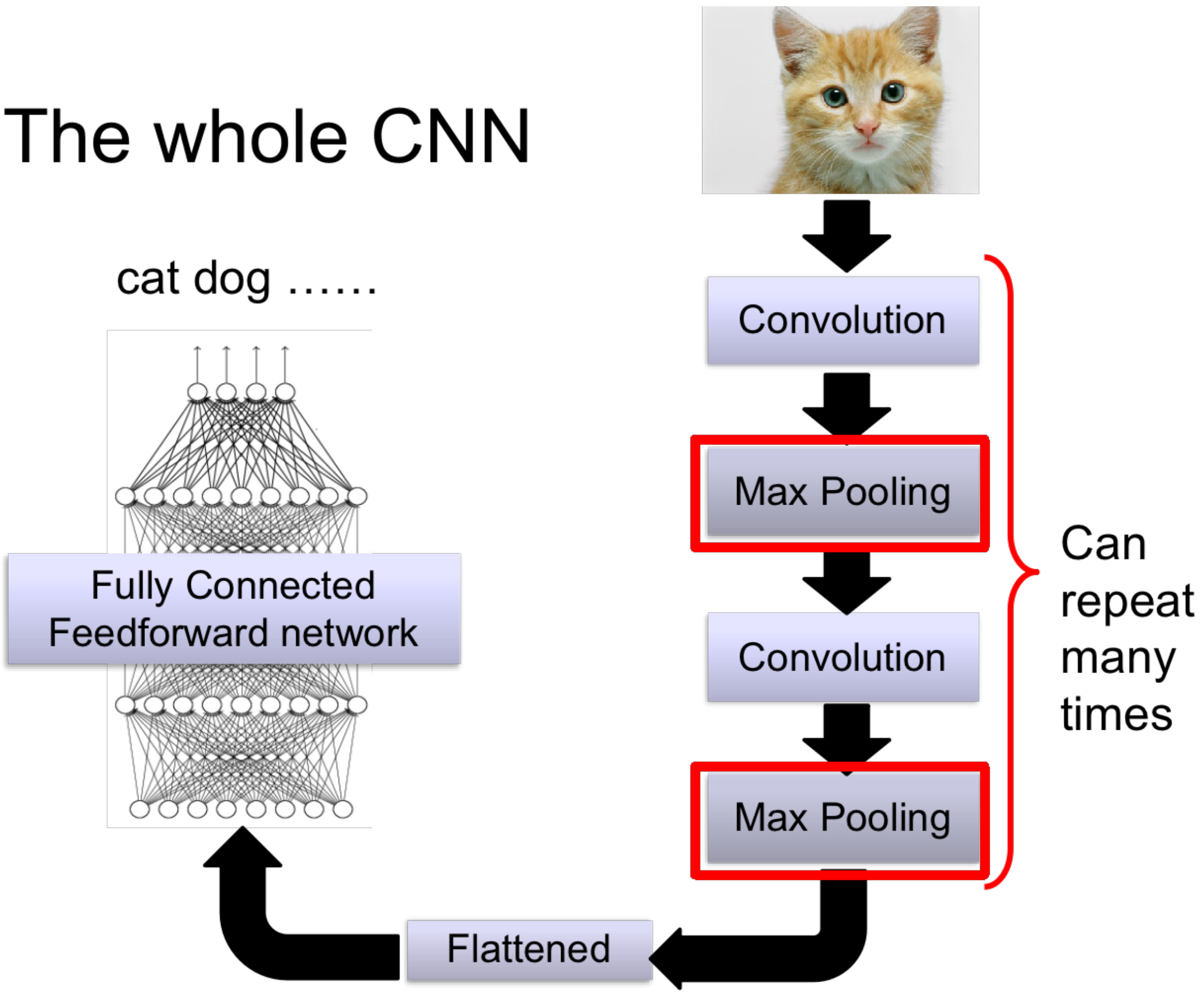
Introduction to Jet-Image

The image-based representation is based on 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
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.

We can construct different channels to characterize more features.
For example, based on energy flow algorithm in Delphes, we can construct 3 channels for
EflowPhoton, EflowNeutralHadron and EflowTracks, respectively.

Introduction to CNN



Advantages:

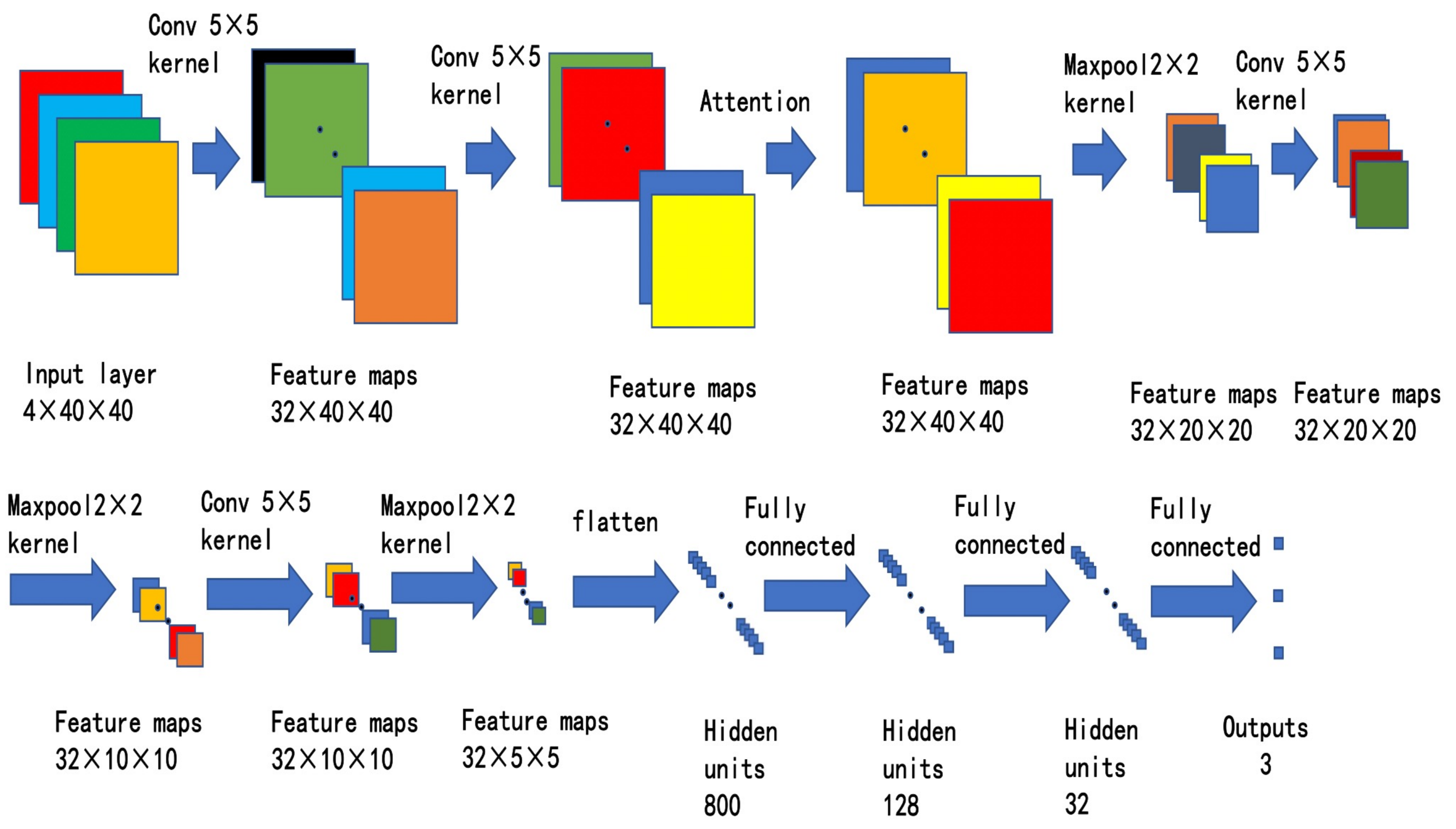
Fewer parameters

Shift, scale and distortion invariance

Local weight sharing

hierarchical channels

CNN Architecture



2106.07018

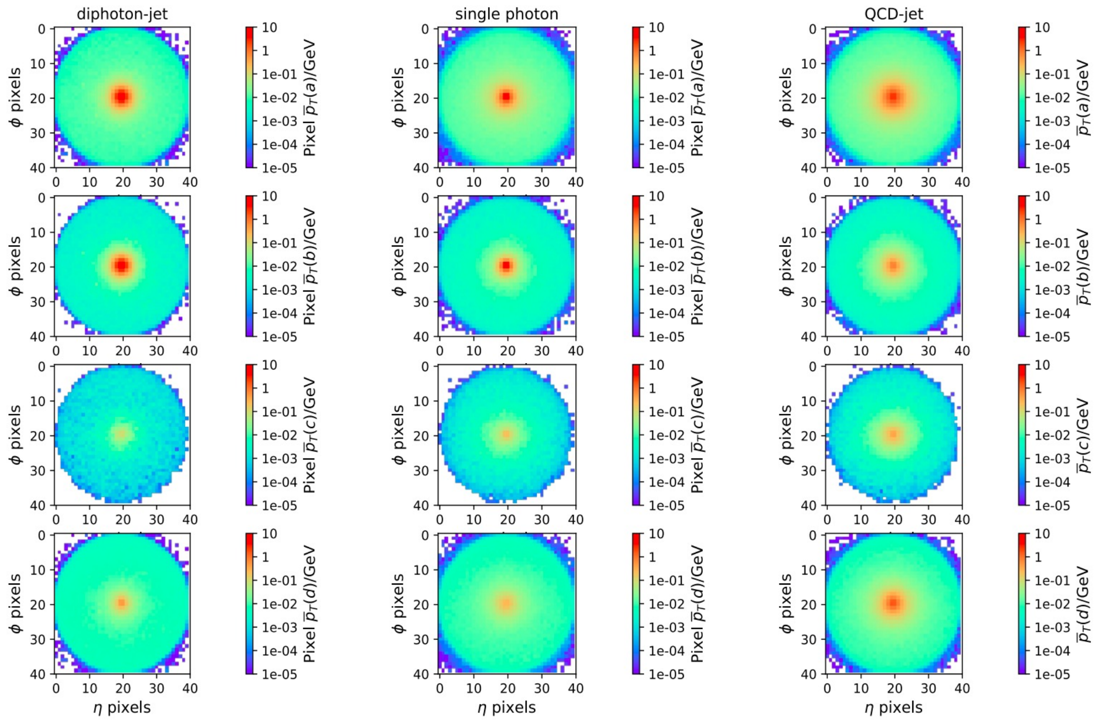
Adam optimizer with learning rate 0.001

Photon-jet label: 0
Single photon label: 1
QCD-jet label:2

Preprocessing
- Shift
- Rotation
- Normalization

The CNN based on jet images achieve sizable improvement in performance compared to traditional multivariate methods

Average Jet Images



(a) diphoton-jet (b) single photon (c) QCD-jet

Two shortcomings of image-based representation for jet tagging:

1, Treating jets as images leads to a very sparse representation.

Without considering the pileup effects, a typical jet has O(10) to O(100) particles, while a jet image typically needs O(1000) pixels (eg.32×32) in order to fully contain the jet, more then 90% of the pixels are blank.

2, How to incorporate additional information of the particles is unclear.

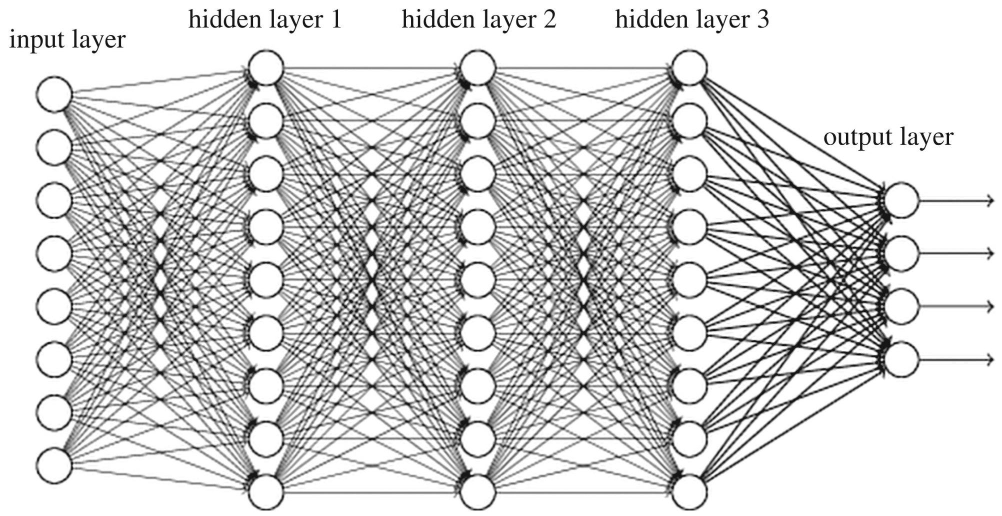
As it involves combining nonadditive quantities (e.g., the particle type) of multiple particles entering the same cell.

Introduction to Particle-based Representation

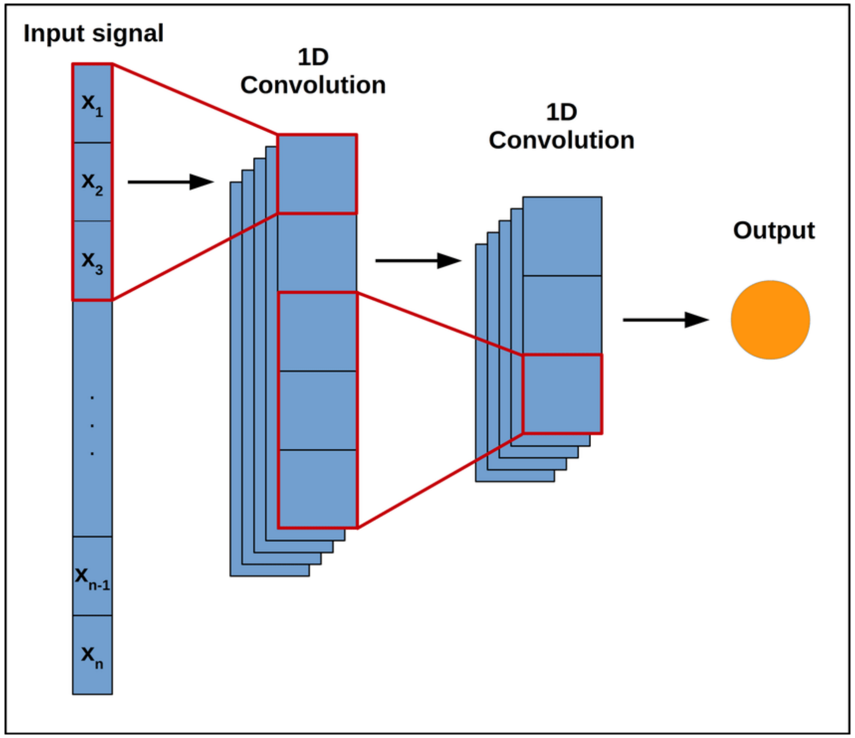
We organize a jet's constituent particles into an ordered structure (sequence or tree) based on the p_T.

Dense Network

We start with N p_T-sorted particles per jet, the arguably simplest deep network architecture is a dense network taking all (p_T, η, ϕ) information as a fixed set. We again improve the training through physics-motivated pre-processing.



Introduction to Particle-based Representation



P-CNN

The particle-level convolutional neural network (P-CNN) is a customized 1-dimensional CNN for jet tagging. Each input jet is represented as a sequence of constituents with a fixed length of N, organized in descending order of p_T. For each constituent, several input features are computed from the 4-momenta of the constituent and used as inputs to the network. The P-CNN is similar to CNN, but only uses a 1-dimensional convolution instead of 2-dimensional convolutions.

Introduction to Particle-based Representation

Two shortcomings of particle-based representation for jet tagging:

1, The jet length are variable.

As each jet may contain a different number of particles.

2, The particles are needed to be sorted in some way.

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.

Introduction to Particle Cloud

We consider a jet as an unordered set of its constituent particles.

↓ **more natural !**

Permutation Invariant

Typical Architectures
- Deep Sets Framework
  - IRC-safe Energy Flow network
  - IRC-not-safe Particle Flow network
- ParticleNet
- ABCNet
- LorentzNet
- Transformer

Introduction to Deep Set framework

The Deep Sets framework was adapted and specialized to particle physics in **1810.05165**

The Deep Sets framework for point clouds demonstrates how permutation-invariant functions of variable-length inputs can be parametrized in a fully general way and it enables a natural visualization of the learned latent space, providing insights as to what exactly the NN is learning.

Observable Decomposition. *An observable $\mathcal{O}$ can be approximated arbitrarily well as:*

$$\mathcal{O}(\{p_1,\ldots,p_M\}) = F\left(\sum_{i=1}^{M}\Phi(p_i)\right), \tag{1.1}$$

where $\Phi : \mathbb{R}^d \to \mathbb{R}^\ell$ is a per-particle mapping and $F : \mathbb{R}^\ell \to \mathbb{R}$ is a continuous function.

IRC safety corresponds to robustness of the observable under collinear splittings of a particle or additions of soft particles.

IRC-Safe Observable Decomposition. *An IRC-safe observable $\mathcal{O}$ can be approximated arbitrarily well as:*

$$z_i = p_{T,i}/\sum_j p_{T,j} \qquad \mathcal{O}(\{p_1,\ldots,p_M\}) = F\left(\sum_{i=1}^{M} z_i\Phi(\hat{p}_i)\right), \tag{1.2}$$

where z_i is the energy (or p_T) and $\hat{p}_i$ the angular information of particle i.

$$\text{EFN:}\quad F\left(\sum_{i=1}^{M} z_i\Phi(\hat{p}_i)\right), \qquad\qquad \text{PFN:}\quad F\left(\sum_{i=1}^{M}\Phi(p_i)\right)$$

Permutation-invariant & Variable lengths

Introduction to Deep Set framework

Many common observables are naturally constructed by simple choices of Φ and F. Furthermore, Φ and F can be parametrized by NN layers, capable of learning essentially any function, in order to explore more complicated observables.

Latent space

$$\mathcal{O}(\{p_1,\ldots,p_M\}) = F\left(\sum_{i=1}^{M}\Phi(p_i)\right),$$

where $\Phi : \mathbb{R}^d \to \mathbb{R}^\ell$ is a per-particle mapping and $F : \mathbb{R}^\ell \to Y$ is a continuous function.

Particle features

Each component of the particle mapping is a filter

Summing $\Phi(p_i)$ over particles induces a latent description of entire jet, which is mapped by F to the value of observable.

Network Implementation

$$\mathcal{O}\left(\{p_i\}_{i=1}^{M}\right) = F\left(\sum_{i=1}^{M} z_i\,\Phi(\hat{p}_i)\right)$$

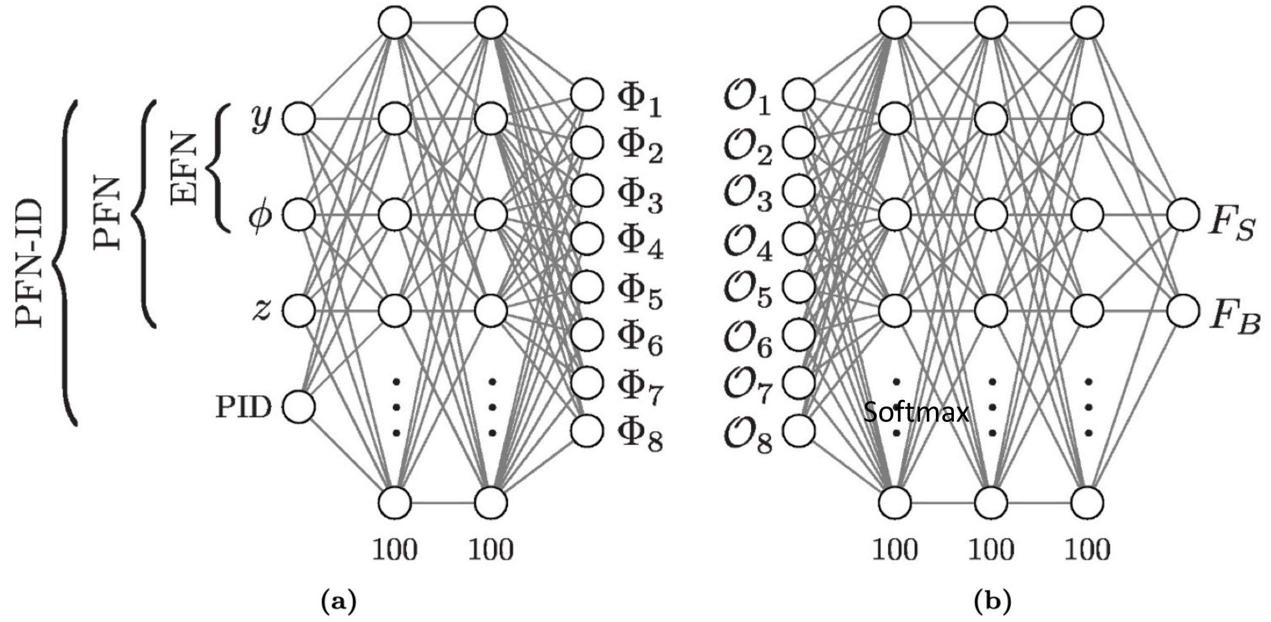


Figure 4. The particular dense networks used here to parametrize (a) the per-particle mapping Φ and (b) the function F, shown for the case of a latent space of dimension $\ell = 8$. For the EFN, the latent observable is $\mathcal{O}_a = \sum_i z_i\,\Phi_a(y_i, \phi_i)$. For the PFN family, the latent observable is $\mathcal{O}_a = \sum_i \Phi_a(y_i, \phi_i, z_i, \text{PID}_i)$, with different levels of particle-ID (PID) information. The output of F is a softmaxed signal (S) versus background (B) discriminant.

Preprocessing $p_{T,i} = \frac{p_{T,i}}{\sum_j p_{T,j}}, y_i = y_i - y_j, \phi_i = \phi_i - \phi_j$

Introduction to ParticleNet (DGCNN)

1902.08570

- Irregular Distribution
- Object of convolution
- Permutation Invariant

→ EdgeConv

- Point cloud ↔ Graph
- Points ↔ Vertices
- Edges ↔ Connections between each point and its k nearest neighboring points

↓ Local Patches

Similar to regular convolution operates on square patches of images.

The EdgeConv parameters are shared for all points !

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

$$R^F \times R^F \rightarrow R^{F'}$$

Advantages:

Easily stacked. A deep network can be built with many EdgeConv operations to learn features of point cloud hierarchically.

The graph describing the point clouds are **dynamically updated** to reflect the changes in the edges, i.e., the neighbors of each point.

Introduction to ParticleNet (DGCNN)

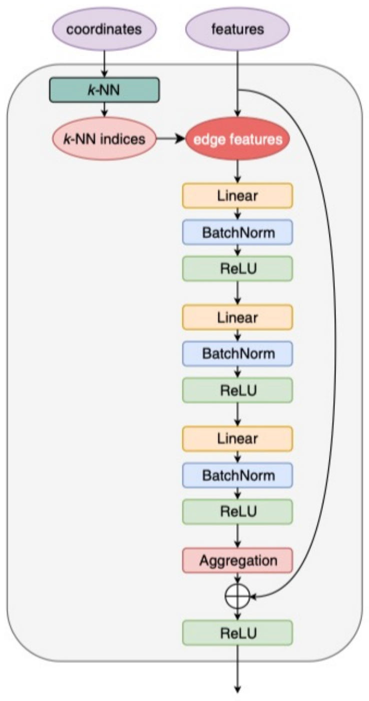


FIG. 1. The structure of the EdgeConv block.

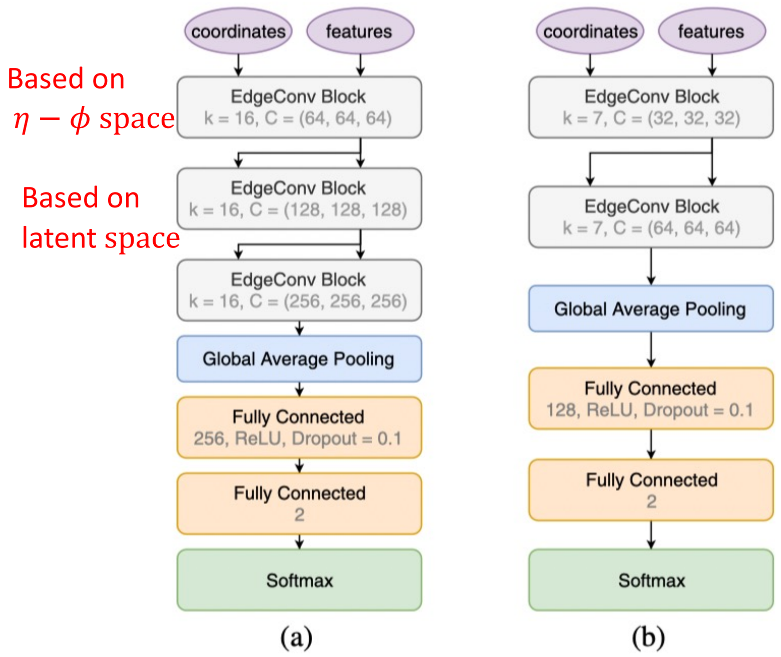


FIG. 2. The architectures of the (a) ParticleNet and the (b) ParticleNet-Lite networks.

The "edge features" are constructed from the "features" input using the indices of k nearest neighboring particles. The EdgeConv operation is implemented as a 3-layer MLP.

After the EdgeConv blocks, a channel wise global average pooling operation is applied to aggregate the learned features over all particles in the cloud.

Introduction to ABCNet

2001.05311

The attention-based cloud net (ABCNet) treats each collider event as an unordered set of points that defines a point cloud. To enhance the extraction of local information, an attention mechanism is used. The main difference between ABCNet and ParticleNet is that ABCNet takes advantage of attention mechanisms to enhance the local feature extraction, allowing for a more compact and efficient architecture. To capture the global information, direct connections for global input features can be directly added.

Key part: GAPLayer (Graph Attention Pooling Layer)

The point cloud is first represented as a graph with vertices represented by the points themselves. The edges are constructed by connecting the points to their k-nearest neighbors, while the edge features, $y_{ij} = (x_i - x_{ij})$, are taken as the difference between features of each point x_i and its k-neighbors x_{ij}.

A GAPLayer is constructed by first encoding each point and edge to a higher-level feature space of dimension F using a single-layer neural network (NN), with learnable parameters θ, in the following form:

Point Feature $x'_i = h(x_i, \theta_i, F)$

Edge Feature $y'_{ij} = h(y_{ij}, \theta_{ij}, F)$ $\quad y_{ij} = (x_i - x_{ij})$

Attention coefficient $c_{ij} = \mathrm{LeakyRelu}(h(x'_i, \theta'_i, 1) + h(y'_{ij}, \theta'_{ij}, 1))$

A single attention feature for each point is $\hat{x}_i = \mathrm{Relu}\left(\sum_j c_{ij} y'_{ij}\right)$

The outputs of each GAPLayer consist of attention features ($\hat{x}_i$) and graph features (y'_{ij}). The graph features are further aggregated in the form:

$$y_{ij}^{max} = max(y'_{ij})$$

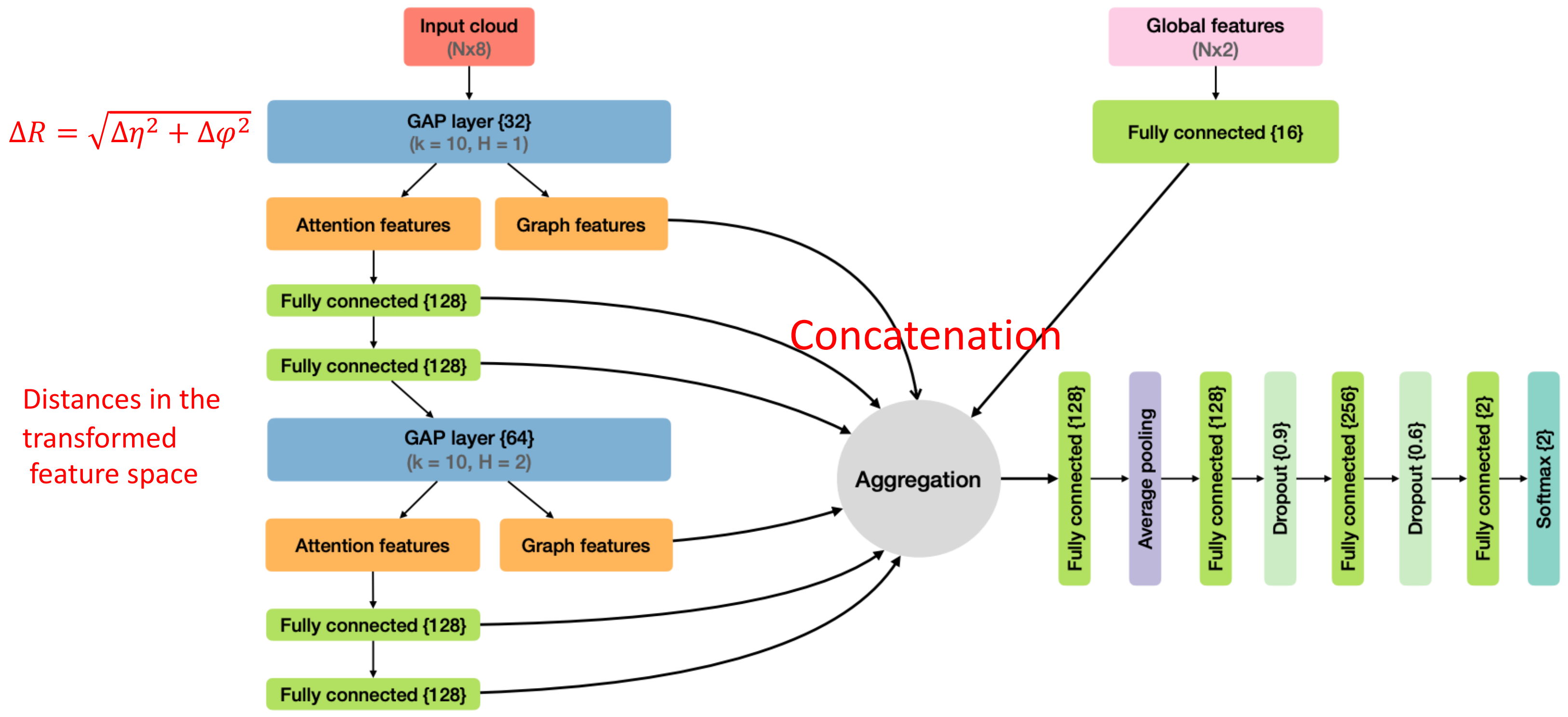


Fig. 1. ABCNet architecture used for quark-gluon tagging. Fully connected layer and encoding node sizes are denoted inside "{}". For each GAPLayer, the number of k-nearest neighbours (k) and heads (H) are given.

Input features: $\Delta\eta$, $\Delta\phi$, log p_T, log E, log $\frac{p_T}{p_{Tj}}$, log $\frac{E}{E_j}$, ΔR, PID

Global features: m_j, p_{Tj}

Introduction to LorentzNet

2201.08187

LorentzNet, a new symmetry-preserving deep learning model for jet tagging. The message passing of LorentzNet relies on an efficient Minkowski dot product attention.

Preprocessing cannot achieve full invariance to arbitrary Lorentz transformations.
LorentzNet directly scalarizes the input 4-vectors to realize Lorentz symmetry.
Specifically, they design Minkowski dot product attention, which aggregates the 4-vectors with weights learned from Lorentz-invariant geometric quantities under the Minkowski metric.

They regard the constituent particles as a point cloud, which is an unordered, permutation invariant set of particles $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N) \in \boldsymbol{R}^{N\times 4}$

Input
- Vector $v_i = (E^i, p_x^i, p_y^i, p_z^i)$
- Scalar $s_i = (s_1^i, s_2^i, \cdots, s_\alpha^i)$
- Graph $G = (V, E)$ where V is the set of nodes and E is the set of edges.

Let Q be the Lorentz transformation, the Lorentz equivariance of $\phi(\cdot)$ means:

$$Q\phi(v) = \phi(Qv), \quad \text{for } \phi(v) \in \mathbb{R}^4; \tag{2.2}$$

$$\phi(v) = \phi(Qv), \quad \text{for } \phi(v) \in \mathbb{R}. \tag{2.3}$$

The construction of the LorentzNet is based on the following universal approximation theorem for the Lorentz group equivariant continuous function.

Proposition 3.1. *[64] A continuous function $\phi : (\mathbb{R}^{N\times 4}) \to \mathbb{R}^4$ is Lorentz-equivariant if and only if*

$$\phi(v_1, v_2, \cdots, v_N) = \sum_{i=1}^{N} g_i(\langle v_i, v_j\rangle_{i,j=1}^{N}) v_i, \tag{3.1}$$

where g_i are continuous Lorentz-invariant scalar functions, and $\langle \cdot, \cdot \rangle$ is the Minkowski inner product.

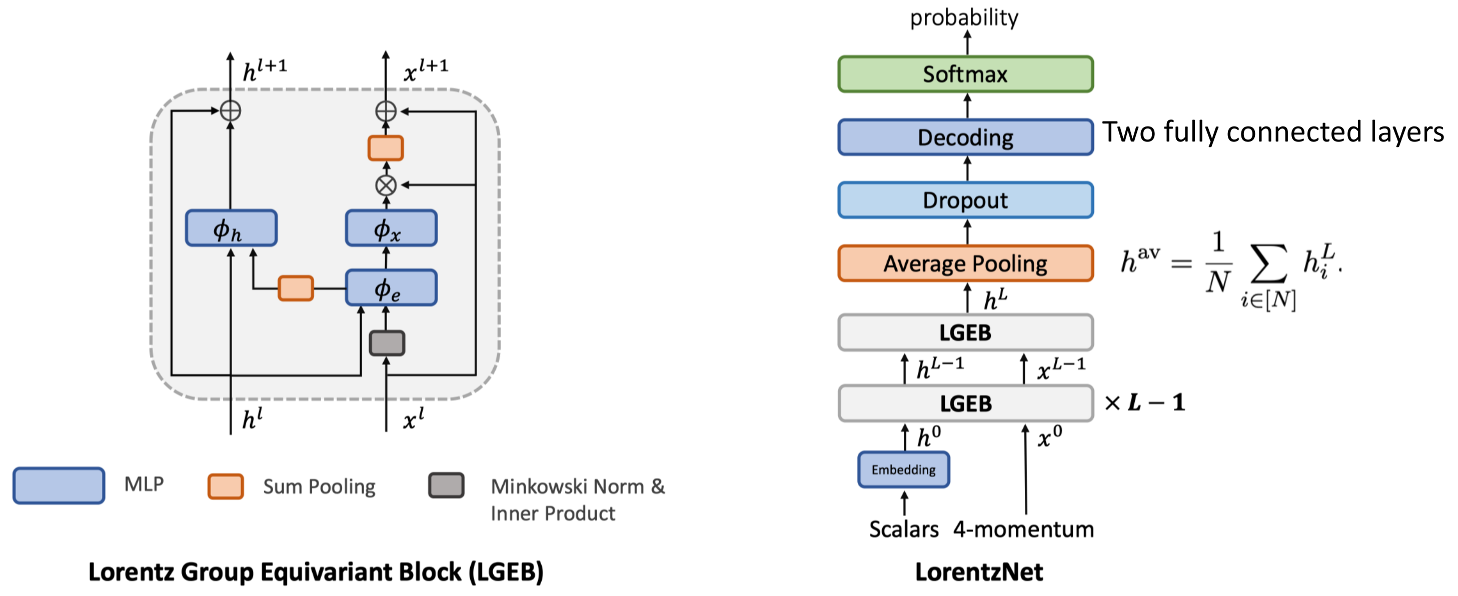


Figure 1. (*left*): the structure of the Lorentz Group Equivariant Block (LGEB). (*right*): the network architecture of the LorentzNet.

The inputs include the PID, mass and the 4-momenta of each particles.

They use $h^l = (h_1^l, h_2^l, \dots, h_N^l)$ to denote the node embedding scalars and $x^l = (x_1^l, x_2^l, \dots, x_N^l)$ to denote the coordinate embedding vectors in the l-th LGEB layer.

Edge message: $m_{ij}^l = \phi_e\left(h_i^l, h_j^l, \psi(\|x_i^l - x_j^l\|^2), \psi(\langle x_i^l, x_j^l\rangle)\right)$ ψ(·) = sgn(·) log(|·|+1)

Minkowski dot product attention: $x_i^{l+1} = x_i^l + c\sum_{j\in[N]} \phi_x(m_{ij}^l)\cdot x_j^l$ $h_i^{l+1} = h_i^l + \phi_h\left(h_i^l, \sum_{j\in[N]} w_{ij} m_{ij}^l\right)$ $w_{ij} = \phi_m(m_{ij}^l) \in [0,1]$

$\phi_x, \phi_h, \phi_e, \phi_m$ are all scalar functions modeled by NN (linear layers).

The l-th message passing step on the graph can be described as

$$m_i^{l+1} = \sum_{j\in\mathcal{N}(i)} M_l(h_i^l, h_j^l, e_{ij})$$
$$h_i^{l+1} = U_l(h_i^l, m_i^{l+1});$$

Introduction to Transformer

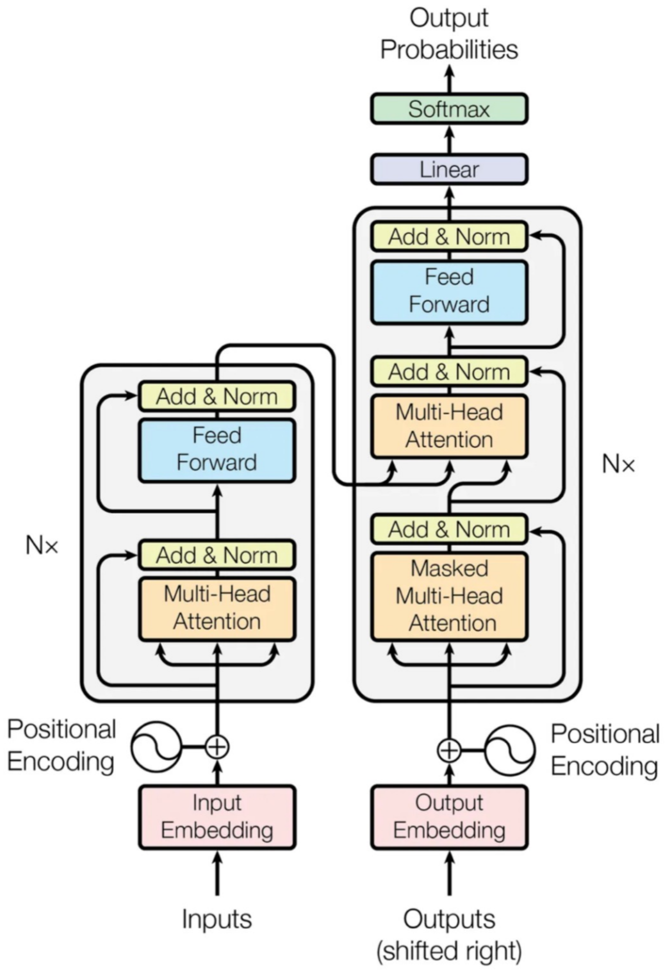


Figure 1: The Transformer - model architecture.

Encoder Decoder

The original Transformer as well as its variants have refreshed the performance records in various tasks, from NLP to CV.

Inputs encoding + Positional encoding

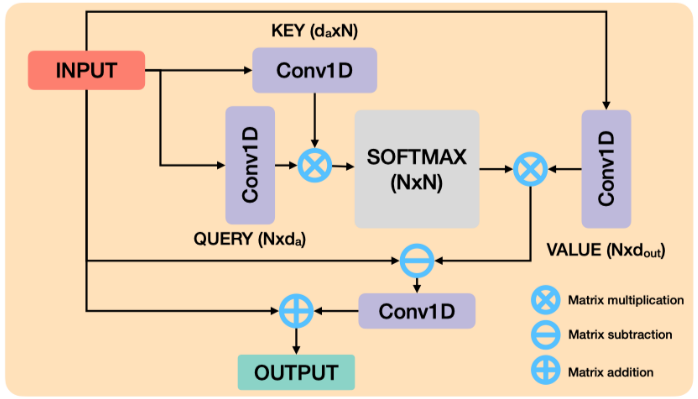
Self-Attention Layers

Query, Key and Value vectors

Matching Q and K **Weighted sum of V and attention weights**

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

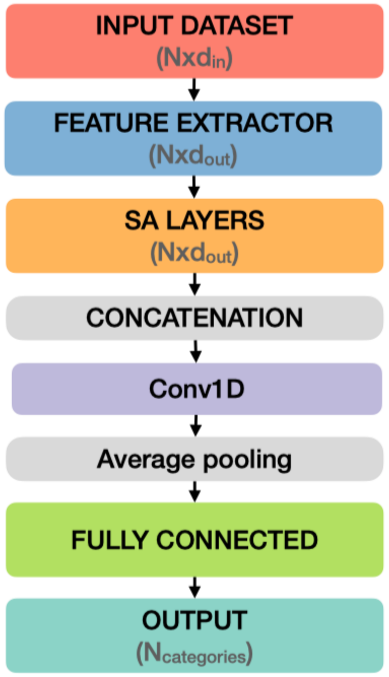
SA LAYER



Introduction to Point Cloud Transformer

2102.05073

Point Cloud Transformer incorporates the advantages of the Transformer architecture to an unordered set of particles resulting from collision events, which means that the Transformer structure has to be modified to define a self-attention operation that is invariant to permutations of the inputs.



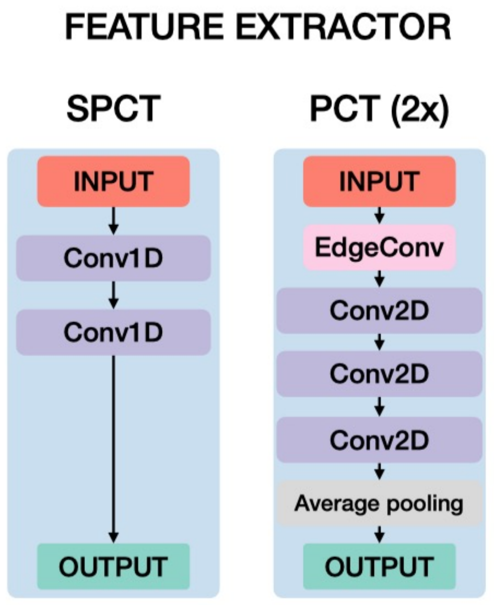
$$F_{in} \in \mathbb{R}^{N\times \mathrm{d_{in}}} \longrightarrow \mathrm{F}_e \in \mathbb{R}^{N\times \mathrm{d_{out}}}$$

Provide the relationship between all particles in the set through attention weights.

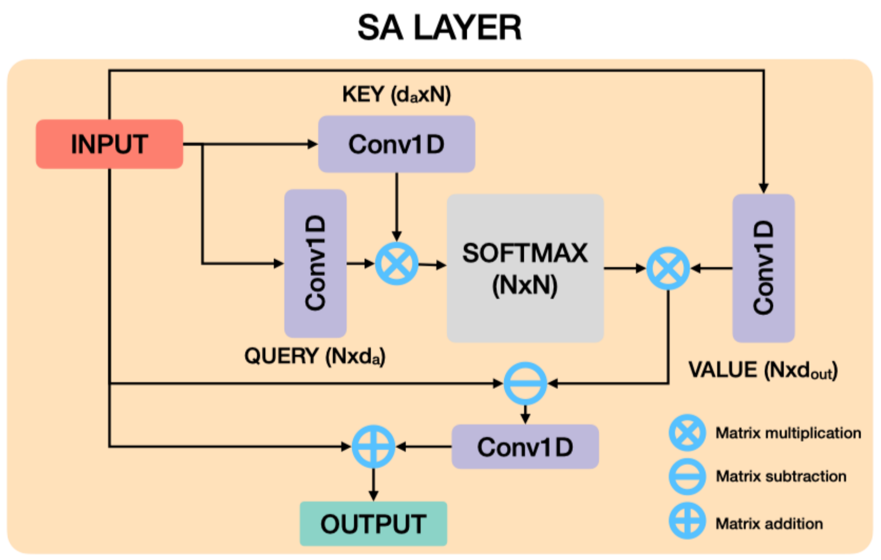
Multiple SA layers can be stacked to achieve different levels of abstraction

To complete the general architecture, the SA layers are combined through a simple concatenation over the feature dimension, followed by a mean aggregation, resulting in the overall means of each feature across all the particles.

Introduction to Point Cloud Transformer



Capture Local Information



EdgeConv uses a k-nearest neighbors approach to define a vicinity for each point in the point cloud.

1st EdgeConv: ΔR

Off-set Attention: $F_{sa} - F_e$ results in a superior classification performance

The output of the feature extractor F_e is used as the input of the first SA layer.

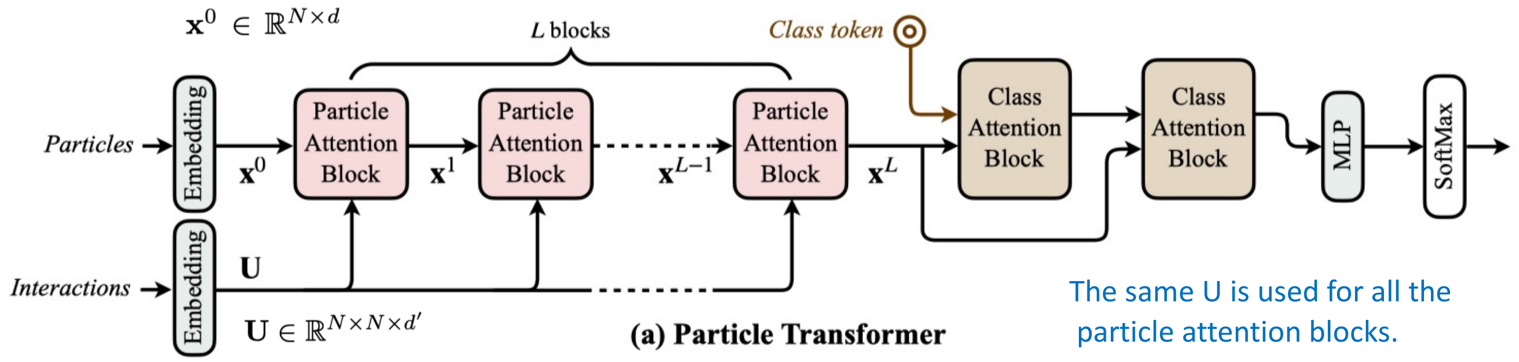
Attention weights are created by matching Q and K through matrix multiplication. These attention weights, after normalization, represent the weighted importance between each pair of particles. The self-attention is then the result of the weighted elements of V, defined as the result of the matrix multiplication between the attention weights and the value matrix.

$$\mathrm{Q, K, V} = \mathrm{F}_e.(\mathrm{W_q, W_k, W_v})$$
$$\mathrm{Q, K} \in \mathbb{R}^{N\times \mathrm{d_a}}, \mathrm{V} \in \mathbb{R}^{N\times \mathrm{d_{out}}}$$
$$\mathrm{A} = \mathrm{Softmax}(\mathrm{Q.K^T})/N, \mathrm{A} \in \mathbb{R}^{N\times N}$$
$$\mathrm{F}_{sa} = \mathrm{A} \cdot \mathrm{V}, \mathrm{F}_{sa} \in \mathbb{R}^{N\times \mathrm{d_{out}}}$$

Introduction to Particle Transformer

2202.03772

Particle Transformer: Incorporating pairwise interactions in the attention mechanism



(a) Particle Transformer

The same U is used for all the particle attention blocks.

Particles form an array of (N, C), each particle has C features.
Interactions form an array of (N, N, C'), each pair of particles has C' features.

Permutation Invariant $\longrightarrow$ **No ad-hoc positional encodings**

Interactions $\left\{ \begin{array}{l} ln\Delta \\ lnk_T \\ lnz \\ lnm^2 \end{array} \right.$

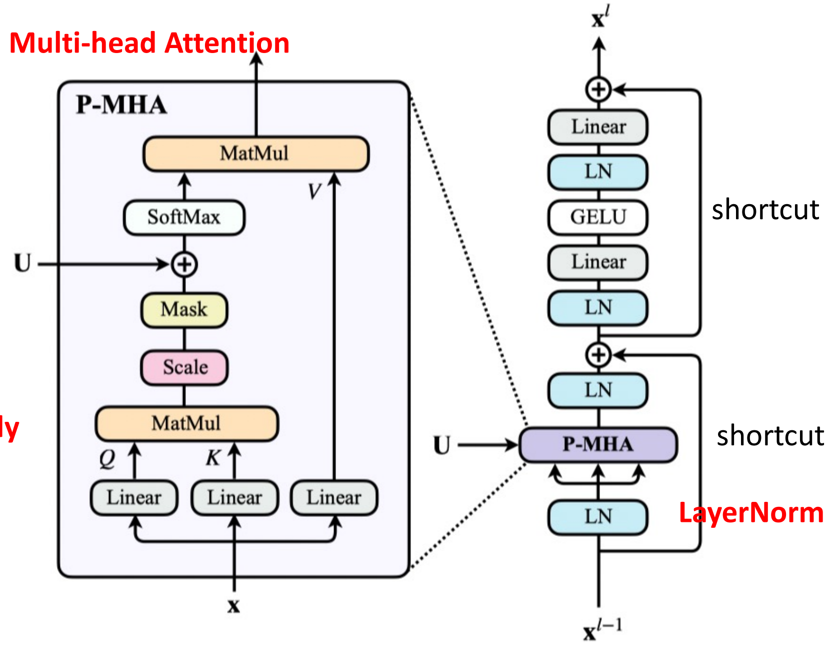
$$\Delta = \sqrt{(y_a - y_b)^2 + (\phi_a - \phi_b)^2},$$
$$k_T = \min(p_{T,a}, p_{T,b})\Delta,$$
$$z = \min(p_{T,a}, p_{T,b})/(p_{T,a} + p_{T,b}),$$
$$m^2 = (E_a + E_b)^2 - \|\mathbf{p}_a + \mathbf{p}_b\|^2,$$

Introduction to Particle Transformer

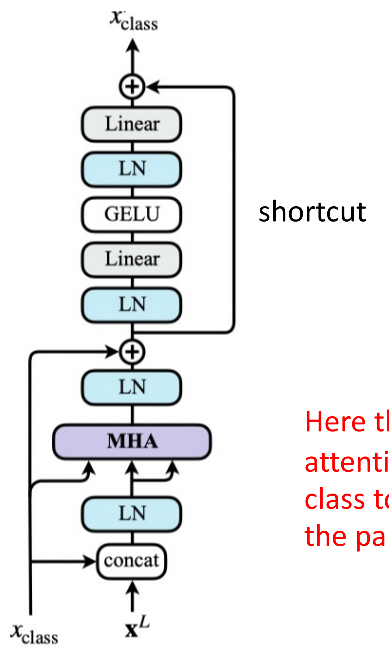
Key part of particle transformer

GELU: Gaussian Error Linear Unit

$$\mathrm{GELU}(x) \approx 0.5x(1+\tanh(\sqrt{2/\pi}(x+0.044715x^3))) \approx x \cdot \mathrm{sigmoid}(1.702x)$$



(b) Particle Attention Block



(c) Class Attention Block

Here they also compute the attention between a global class token x_{class} and all the particles.

Figure 3. The architecture of (a) Particle Transformer (b) Particle Attention Block (c) Class Attention Block.

$$\text{P-MHA}(Q, K, V) = \mathrm{SoftMax}(QK^T/\sqrt{d_k} + \mathbf{U})V,$$

They take U as the attention mask matrix

$$Q = W_q x_{\text{class}} + b_q,$$
$$K = W_k \mathbf{z} + b_k, \qquad (5)$$
$$V = W_v \mathbf{z} + b_v,$$

Q$\in R^{N\times d_1}$

K$\in R^{N\times d_1}$

V$\in R^{N\times d_2}$

U$\in R^{N\times N}$

Scale: $d_k = d_1$ embedding dimension

$$Q_i = QW_i^Q, K_i = KW_i^K, V_i = VW_i^V, \quad i = 1, \cdots, 8$$
$$\mathrm{head}_i = \mathrm{Attention}(Q_i, K_i, V_i), \quad i = 1, \cdots, 8$$
$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, \cdots, \mathrm{head}_8)W^O$$

$We\ set\ d_1 = d_2 = d_o/h$

where $\mathbf{z} = [x_{\text{class}}, \mathbf{x}^L]$ is the concatenation of the class token and the particle embedding after the last particle attention block, $\mathbf{x}^L$.

Typical evaluation metrics for performance

Accuracy

AUC (Area under ROC curve)

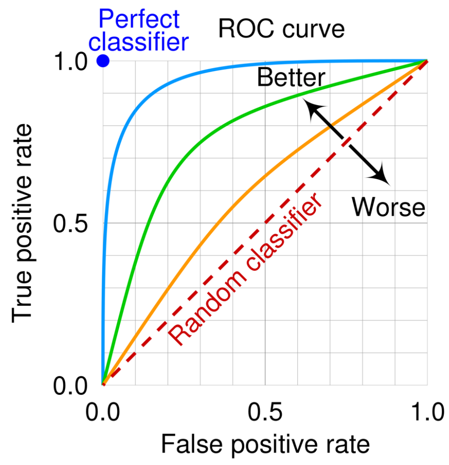
Background rejection ($\frac{1}{\epsilon_b}$) at a certain signal efficiency ϵ_s

Top Tagging

	Acc	AUC	$1/\epsilon_B$ ($\epsilon_S = 0.5$)	$1/\epsilon_B$ ($\epsilon_S = 0.3$)
ResNeXt-50 [17]	0.936	0.9837	302±5	1147±58
P-CNN [17]	0.930	0.9803	201±4	759±24
PFN [33]	-	0.9819	247±3	888±17
ParticleNet-Lite [17]	0.937	0.9844	325±5	1262±49
ParticleNet [17]	**0.940**	**0.9858**	**397±7**	**1615±93**
JEDI-net [21]	0.9263	0.9786	-	590.4
JEDI-net with $\sum O$ [21]	0.9300	0.9807	-	774.6
SPCT	0.928	0.9799	201±9	725±54
PCT	**0.940**	0.9855	392±7	1533±101
LorentzNet	**0.942**	**0.9868**	**498±18**	**2195±173**

Gluon/Quark Discrimination

	Acc	AUC	$1/\epsilon_B$ ($\epsilon_S = 0.5$)	$1/\epsilon_B$ ($\epsilon_S = 0.3$)
ResNeXt-50 [17]	0.821	0.9060	30.9	80.8
P-CNN [17]	0.827	0.9002	34.7	91.0
PFN [33]	-	0.9005	34.7±0.4	-
ParticleNet-Lite [17]	0.835	0.9079	37.1	94.5
ParticleNet [17]	0.840	0.9116	39.8±0.2	98.6±1.3
ABCNet [18]	0.840	0.9126	42.6±0.4	**118.4±1.5**
SPCT	0.815	0.8910	31.6±0.3	93.0±1.2
PCT	**0.841**	**0.9140**	**43.2±0.7**	118.0±2.2
LorentzNet	**0.844**	**0.9156**	**42.4±0.4**	**110.2±1.3**
ABCNet	**0.840**	**0.9126**	**42.6±0.4**	**118.4±1.5**



$$ACC=\frac{TP+TN}{P+N}$$

$$\epsilon_S=\frac{TP}{P}$$

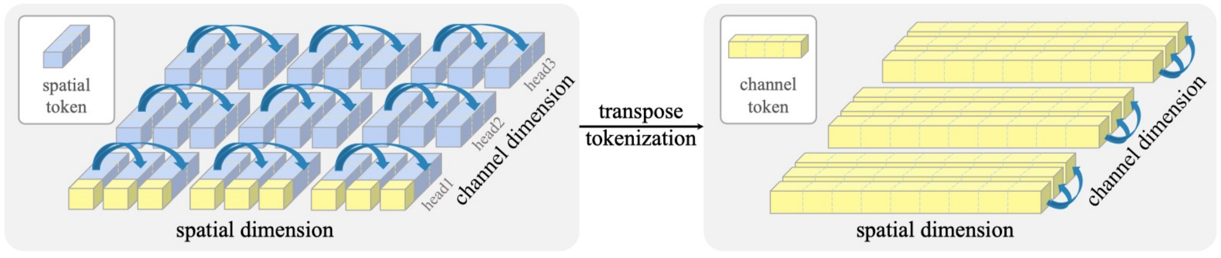
$$\epsilon_B=\frac{FN}{P}$$

Taken from 2102.05073

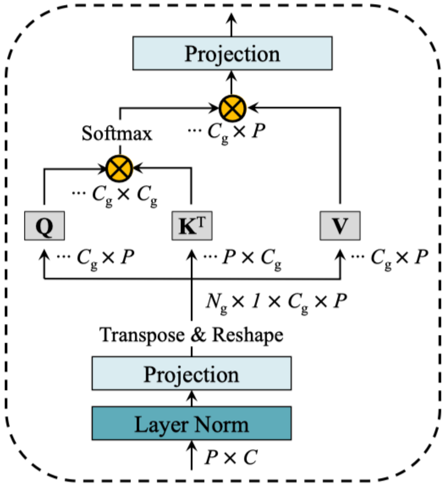
Possible Improvement

Dual Attention Vision Transformer 2204.03645

Self-Attention
- Spatial tokens: Particle numbers (Different particles) Local
- Channel tokens: Feature numbers (Different features) Global



(a) Spatial Window Multihead Self-attention (b) Channel Group Self-attention



With spatial tokens, the spatial dimension defines the token scope, and the channel dimension defines the token feature dimension.

With channel tokens, the channel dimension defines the token scope, and the spatial dimension defines the token feature dimension.

Future Outlook

1, Point Cloud based representation and Transformer show excellent performance.
We can explore more novel methods for particle embedding and self attentions that improve the classification performance and alleviate the computational cost.

2, Point Cloud based representation can be used for other interesting physics problems such as pileup subtraction, jet grooming and jet energy calibration.

3, Based on the Point Cloud representation and EFN/PFN, we can explore different variant architectures and try to obtain a natural visualization of the learned latentspace, providing insights as to what exactly the NN is learning.

4, We can try to incorporate more priors or constraints from physics principles in the architecture designs.

5, How to evaluate the statistical uncertainties of deep learning in jet tagging?

Thanks for your attention !