International Institute of Information Technology, Hyderabad

ParticleNet

Jet Tagging via Particle Clouds

Replication Study

Author: Jai Bardhan, Animesh Sinha, Kalp Shah

June 11, 2020

Contents

Chapter 1

Particle Net Simplifications

1.1 Simpler Architectures

1.1.1 The full network

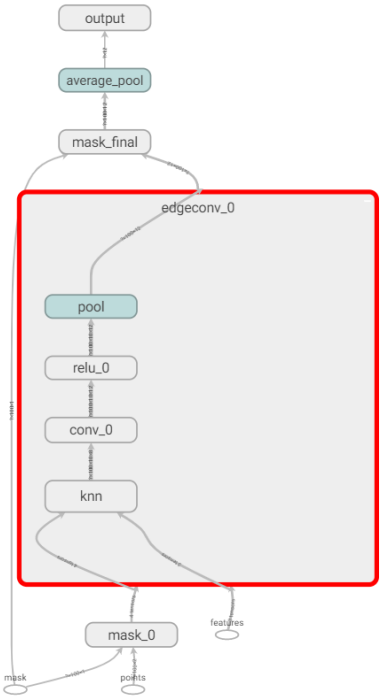


Figure 1.1: Neural Network Graph Visualization

1.2 Distribution of Parameters

1.2.1 Weight Histograms

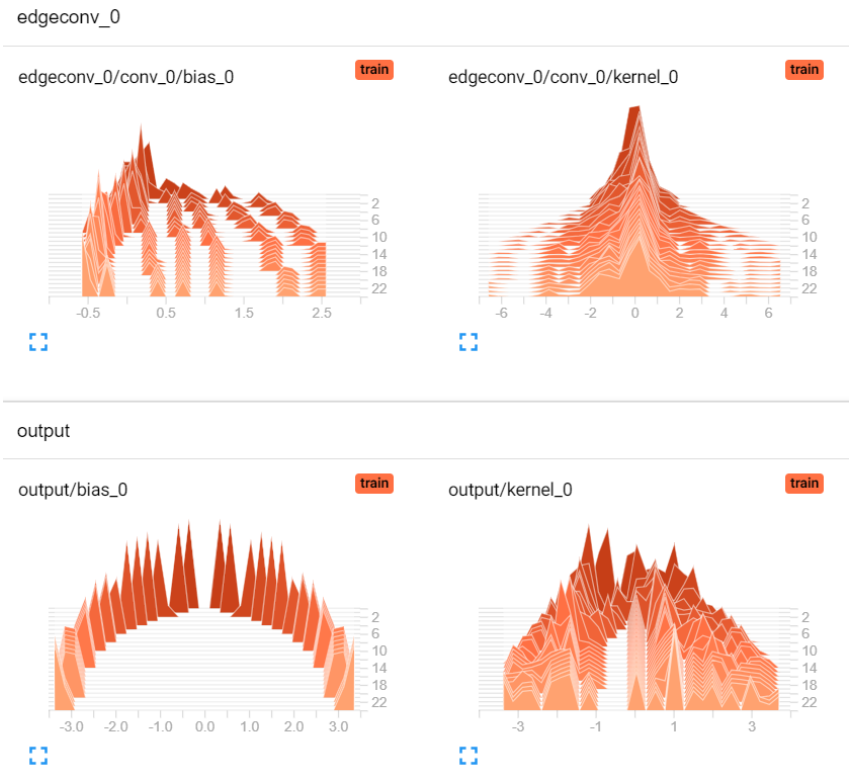


Figure 1.2: TensorBoard Histograms

We see that our biases are very important, and the the network is learning quite something. All layers are important.

1.2.2 Weight Distribution

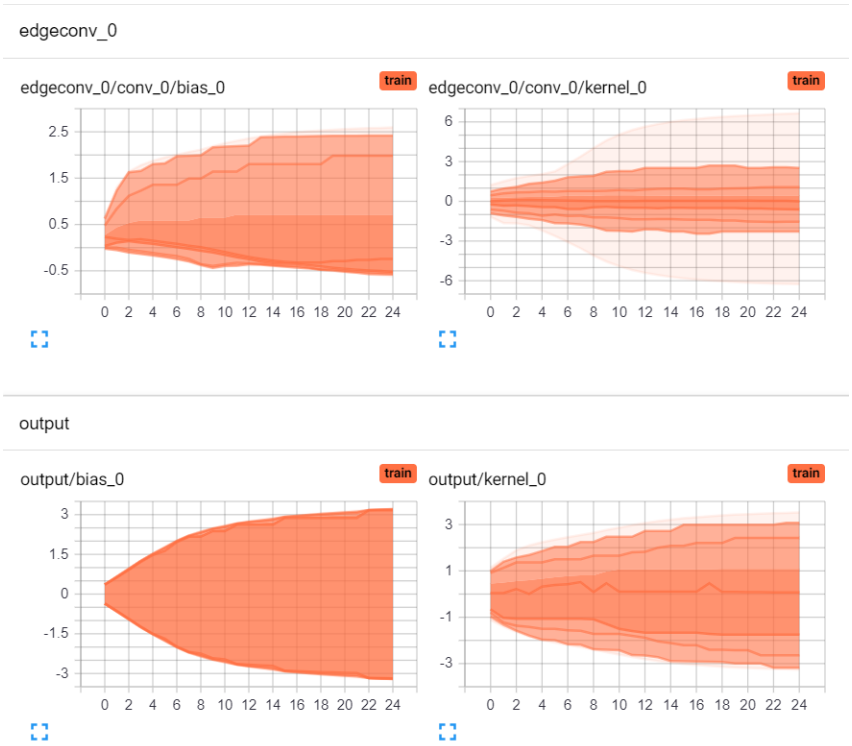


Figure 1.3: TensorBoard Histograms

1.2.3 Feature Polynomials

.	**Hub-η**	**Hub-ϕ**	**Hub-p_t**	**Hub-E**	**Rel-η**	**Rel-ϕ**	**Rel-p_t**	**Rel-E**
00	1.39279	2.95816	0.47035	0.15665	2.13112	2.54264	-1.58122	-1.02798
01	-1.56491	-2.16750	0.32590	0.38175	0.46990	-1.77234	-1.51412	-1.93057
02	1.10430	2.35467	0.66460	0.03594	2.26508	1.87872	-1.13675	-1.95035
03	-3.58598	6.64981	-0.04281	0.11523	0.28928	0.05610	0.43508	-0.46872
04	5.51387	-3.83200	-0.12499	0.62469	4.76349	-3.50846	-0.96857	-0.62145
05	-6.26354	-4.09571	0.32543	-0.21911	-0.16259	-0.11651	-0.15742	0.11673
06	-1.61074	0.18924	-0.50111	0.71576	1.02103	-0.94762	0.00520	-0.64432
07	-1.36603	-0.10369	0.63901	0.74640	2.91987	1.39712	-0.19567	1.04621
08	-1.03466	0.14385	-0.61606	-0.28012	1.07867	-0.29486	0.20878	0.05523
09	-0.87198	-2.81173	0.55644	0.16556	0.96585	-2.10658	-1.37593	-2.11623
10	-0.65812	0.37103	0.27153	0.89163	-1.76185	2.72242	-0.35294	-0.27357
11	-1.14922	0.11895	-1.00611	0.25360	1.05348	-0.16677	0.58487	-0.37585

We train a dataset on these 12 transformed features. The relative weights of the feature follow here, the following are our notes:

- Decision Tree with max-depth 8 gives accuarcy of 81.67% when just trained on faetures 2 and 11.

- Decision Tree with max-depth 8 gives accuarcy of 85.00% when trained on features 2, 11, 9, 3, 1.

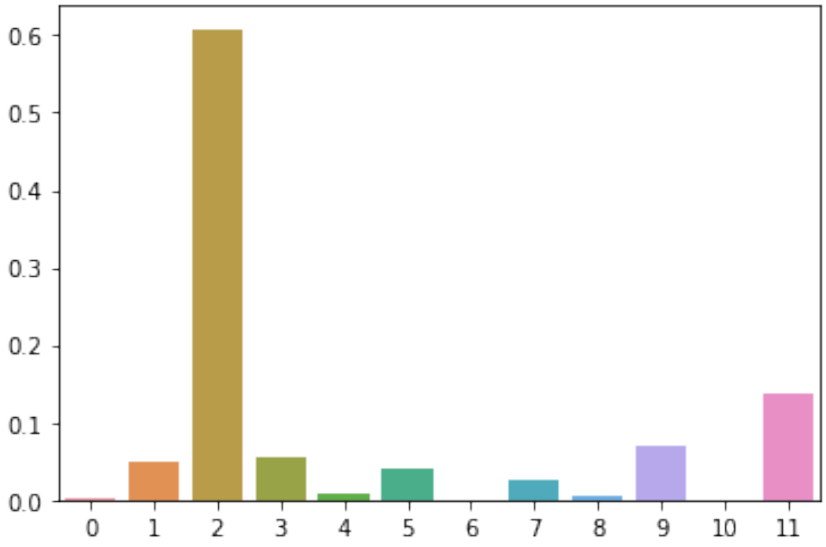


Figure 1.4: Relative Feature importances

Chapter 2

Jet Tagging via Particle Clouds

Representing Jets is a core component of ML on Jet Physics. In this paper, they are represented by particle clouds, which considers Jet as an unordered set of its constituent particles.

2.1 Jets

Jet is a collimated spray of particles. It is one of the most ubiqutous objects in proton-proton collision events at the LHC.

Definition 1.0: Particle Jets

A Jet is defined as a narrow cone of hadrons and other particles produced by the hadronization of a quark or gluon in a particle physics or heavy ion experiment. (-Wikipedia)

Jet exhibits different characteristics when initiated by different particles. Jets initiated by gluons tend to have broader energy spectrum than jets initiated by quarks.

2.1.1 Characteristics

TODO : Find different characteristics due to different initiating particles.

2.1.2 Generating Data

The Large Hadron Collider (LHC) is the place, where all the particle-particle collisions occur and live data can be generated. As this data is too huge, CERN has developed toolchain known as Root, which is a command-line interface and with tools known as Pythia and MadGragh is used to generate the data.

This data is not the actual data which is observed in the LHC, but rather the result of the simulations which are generated locally and is based on the phsics incorporated by them in CERN. They are close enough to the actual data, hence all algorithms are trained on it.

2.2 Jet Tagging

Jet Tagging is one of the most important topics of interest and study in the field of Jet Physics.

Definition 2.0: Jet Tagging

Jet tagging is finding the source particle that initiated the Jet.

The study of Jet Tagging has been done for a long time. The first and initial methods consisted of QCD theory to distinguish between quark and gluon Jets.

The newer approaches involve Machine Learning where Jets are viewed as images, sequences, trees, graphs or set of particles. The tagging isdone using deep NNs.

2.2.1 QCD Theory

Definition 2.0: Quantum Chromodynamics

In theoretical physics, quantum chromodynamics (QCD) is the theory of the strong interaction between quarks and gluons, the fundamental particles that make up composite hadrons such as the proton, neutron and pion.

2.3 Jet Representation

The effectiveness and efficiency of ML techniques on jet physics relies heavily on how a jet is represented. Some of the mainsteam representations and a new representation used by this paper is reviewed and explained here.

2.3.1 Image-Based Representation

One of earliest (and successful) representation of jets is image representation. In this method, the energy deposition on each cell is used pixel intensity which creates an image of the Jet.

The enery deposition of a Jet is measured b a calorimeter. It is done on a fine-grained spatial cells, thus giving a matrix value table, which can be used as a pixel value, thus constructing an image representation of the jet.

This approach is used and studied extensively for various jet tagging tasks. Examples include W boson tagging, top tagging, and quark-gluon tagging. CNNs (and modifications) are used with this representation as they provide a natural extension of Computer Vision tasks.

They have an extensive advantage over traditional methods motivated by QCD theory. The QCD theory motivates multivaraite methods using observables.

There are two major downsides to this reprsentation though, which makes it less efficient and thus not universally used. The first is its inability to take additional information in a meaningful manner. This is due to the fact that the data given by the calorimeter is non additive to the data that can be retrieved later.

Another downside is the sparseness of the generated image. This is due to the fact that jet has particles in the order of $O(10) - O(100)$. This means that the image requires around $O(100)$ pixels. But a jet image requires $O(1000)$ pixels to contain a jet. Hence, most of the pixels are blank. This makes CNNs computatationaly innefective on jet images.

2.3.2 Particle-Based Representation

Another popular way of representing the jet is as a collection of its constituent particles.

2.3.3 Particle Cloud

In the previous section, different representation of Jets were given, which were helpful in jet tagging using ML. In this paper, a different and novel way of representation of jets is proposed known as particle cloud. It is analogous to point cloud reprsentation (3D shapes) which is used in Computer Vision(CV).

In this paper, they designed ParticleNet, which is based on Dynamic Graph Convolution Neural Network (DGCNN). This operates directly on particle clouds for jet tagging.

The idea is that the most natural representation of jets is an unordered, permutation-invariant set of particles. It is a special ase of Particle based representation, thus has all of its benifits (Flexibility, Smaller Size, Efficiency, etc).

As particle cloud is so similar in structure to cloud representation, all the algorithms applicable to it are helpful to make algorithms for particle clouds.

Note 3.1: Message Passing Neural Nets

There are other representations as well which are useful. One of them is represnting jets as graphs, with the vertices as particles. Message Passing NNs (MPNNs) with variations are used on these representations, and depending on how the adjacency matrices are defined, it can have better performance than the RecNNs. There is however a cache that the definition of adjacency matrix might not respect permutation symmetry of the particles.

2.4 Machine Learning Algorithm

This paper talks in detail about ParticleNet, a CNN-like deep NN for Jet Tagging. The permutation symmetry makes particle cloud representation one of the most natural representation, but the best performace can be acheived only if a good algorithm is used.

2.4.1 Convolutional Neural Networks

CNNs are the staple algorithm to use when working with images. Hence, almost every algorithm developed for images is based on CNNs. There are two major reasons why they are so successful.

Definition 4.1

onvolution is a mathematical operation $*$, defined as :

$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

This operation is translation independant. This is defined for a continuous function.

Convolution operations exploit translational symmetry of images by using shared kernels across the complete image. This reduces number of parameters and allows them to be learnt more effectively.

The second reason is that CNNs use heirarchical approach of learning features. This means that CNN layers can be effectively statcked to form a deep network.

Note 4.1

A single CNN layer can learn a more global feature than the previous one. This means that when stacking them, the earlier ones can learn local features, whereas the deeper ones can learn more global features. As they follow a hierarchy, this property is known as heirarchical approach.

2.4.2 Edge Convolution

As CNNs are very useful and effective against images, a similar approach for learning point cloud data is promoted, but as standard CNNs cannot be applied on the point cloud representation, some modifications must be made to the algorithm to make it usable.

One of the major hurdles is that point cloud representation does not have a regular matrix like shape, but is rather irregular distribution. Hence, the basis for the convolution (local patch of each point on which the kernel operates), is to be defined.

Another problem is that the regular convolution operation, in the form of $\sum \mathcal{K}_j x_j$ ($\mathcal{K}_j$: j^{th} kernel, x_j : j^{th} feature) is not permutation invariant. Hence, this also needs to be modified.

Edge Convolution is a modification of the standard convolution operation for point clouds. This operation requires the point cloud as a graph, with each point represting a vertex and the edges are constructed as connections between the k nearest neighbours of the point.

Definition 4.0: Edge Convolution

The EdgeConv operation is as follows :

$$x'_i \triangleq \square_{j=1}^{k} h_\theta(x_i, x_{i_j})$$

In this defintion, the variables and operations are as below :
$\square$: Channel-wise symmetric aggregation operation
$h_\theta : \mathcal{R}^F \times \mathcal{R}^F \to \mathcal{R}^F$
x_i : Feature Vector of point x_i

In the defintion of EdgeConv given above, the function is parametrized by the set of learnable parametes θ. In this paper, a specialized form of edge function is used.

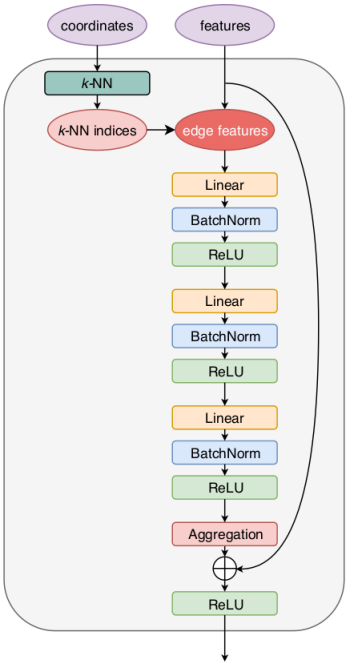
The h function used is defined as :

$$h_\theta(x_i, x_{i_j}) = \overline{h}_\theta(x_i, x_i - x_{i_j})$$

In this, the function $\overline{h}_\theta$ can be implemented as a multilayer perceprtron, whose parameters are shared among all the edges. Also in this paper, the aggregation operation used is mean, rather than max which was the function used in the original paper. The EdgeConv operation thus is :

$$x'_i = \frac{1}{k} \sum_{j=1}^{k} \overline{h}_\theta(x_i, x_i - x_{i_j})$$

An EdgeConv block is implemented according to the diagram given below :



The hyperparameters of the EdgeConv operation are the number of neighbours k, and the number of channels $C =< C_1, C_2, C_3 >$, each corresponding to number of units in the linear transformation layer.

As this operation preserves the dataset as a point cloud, they can be stacked just as easily as standard convolutions.

Note 4.2

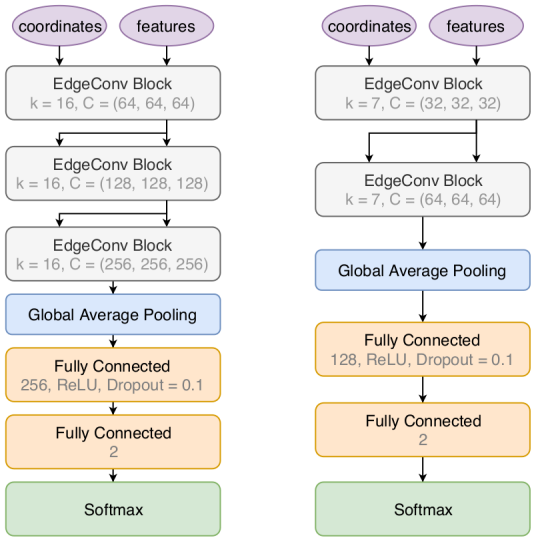
ultilayer Perceptron is a class of feedforward ANNs. The term, though is used ambiguously to classify any feedforward ANN.

The stackability also allows a us to view the feature vectors learnt as new coordinates of the original points in the latent space. This allows determination of distance between points in this new space (Computed while determining h_θ). This means that proximitiy of points can be dynamically learnt using EdgeConv. This results in the DGCNN, in which the graph describing the point clouds are dynamically updated to reflect the changes in the edges.

2.4.3 ParticleNet

ParticleNet is a sort of modified and specialized version of DGCNN, which has different design choices to suit the jet tagging task. It makes extensive use of EdgeConv operation, but has specialized configuration to make it suit better.

The ParticleNet algorithm is given in the diagram given below :



ParticleNet ParticleNet-Lite

These are the two implementations of ParticleNet which were used in the paper. All the results published in the paper were using this algorithm.