Dynamic Graph Convolutional Neural Network with ParticleNet

Noam Tal - Department of physics, Ben Gurion University

November 4, 2020

Abstract

In this report I will present an implementation of supervised deep neural network to identify semi-visible jets. The network that I will use is the ParticleNet, a customized neural network architecture using Dynamic Graph Convolutional Neural Network which is a convolutional neural network that operates on particle clouds, similar to the concept of point clouds in computer vision, rather than analyze jet images. This approach considers a jet as an unordered set of particles, a "particle cloud", that respects the permutation symmetry and has been proved to be afficient when dealing with particle jets tagging.

Contents

1 Introduction

1.1 Jets

A jet can be described as a shower of hadrons and other particles produced by the hadronization process that occur as a consequence of collision between accelerated particles. During a collision, high energy quarks are produced and as a consequence of color confinement, they do not propagate freely and are observed as jets of colorless particles. Color confinement is the phenomenon that color-chaged particles (such as quarks and gluons) cannot be isolated and therefore cannot be directly observed in normal conditions. In the QCD (quantum chromodynamics) theory, which is the theory of the strong interaction between quarks and gluons, the analog (in quantum electrodynamics) of electric-charge is color-charge and the analog to photon as the force carrier is the Gluon. The hadronisation process can be described by five stages:

1. Quark and antiquark produced in an interaction initially separate at high velocities.
2. As they separate the color field is restricted to a tube with energy density of approximately 1 GeV/fm.
3. As the quarks separate further, the energy stored in the color field is sufficient to provide the energy necessary to form new qq pairs and breaking the color field into smaller "strings" is energetically favorable.
4. This process continues and further qq pairs are produced until
5. All the quarks and antiquarks have sufficiently low energy to combine to form colorless hadrons.

The result is two jets of hadrons, one following the initial quark direction and the other in the initial antiquark direction. Hence, in high-energy experiments, quarks and gluons are always observed as jets of hadrons.

1.2 Detector

The Large Hadron Collider (LHC) at CERN is a pp collider designed to operate at a centre-of-mass energy of 14 TeV. Since the colliding partons (model of hadrons, such as protons and neutrons) have no momentum transverse to the beam axis, the jets are produce back to back in the transverse plane and have equal and opposite transverse momenta, p_T.

The main detectors of the LHC are ATLAS and CMS. both detectors have the same basic design, an inner detector close to the beam line to measure tracks of charged particles, followed by an electromagnetic calorimeter (ecal) which measures (mostly) the energy of electrons and photons, then a hadronic calorimeter (hcal) which measures the energies of neutrons and protons, then at the outside a muon detector.

The tracking system in the inner detector has different parts. For example, on ATLAS, the region closest to the beam line has a set of silicon pixel detectors. Outside of this is a semiconductor tracker, also made of silicon, followed by a a transition radiation tracker. These systems have decreasing resolution away from the beam but combine together to give a qualitative picture of what charged particles were produced by the collision and what direction they went in.

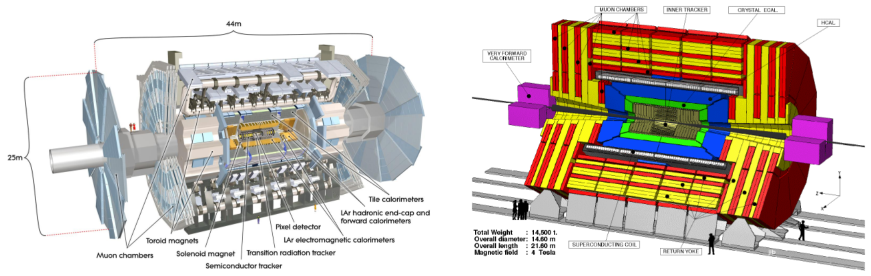


Figure 1: The Atlas and CMS detectors.

In practice, jets must be defined from experimental observables, namely the 4-momenta of the observed particles in the event. The simplest way to define a jet is to draw a cone of size $R = \sqrt{(\Delta\eta)^2 + (\Delta\varphi)^2}$ around some particle and include everything in that cone as the jet. Such a procedure depends strongly on which particle is chosen as the center of the cone.

Most popular jet algorithms are based on the idea of iterative clustering:

1. Calculate the pairwise distance $d_{ij} = R_{ij} = \sqrt{(\eta_i - \eta_j)^2 + (\varphi_i - \varphi_j)^2}$ (Aachen algorithm) between every pair of objects.
2. Merge the two closest particles.
3. Repeat until no two particles are closer than some given R.

Such an algorithm will result in n jets of size R.

The "objects" on which the jet algorithm operates can be quarks and gluons (for theory calculations), they can be all measured particles (pions, protons, electrons, etc.). Or they can be energy deposits in the calorimeters or topoclusters (aggregate energy deposits in the ATLAS detector) or particle-flow candidates, as reconstructed by CMS.

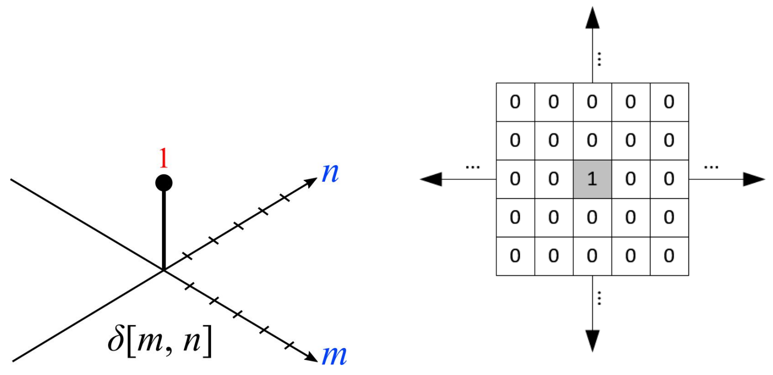
1.3 Dark Jets

The huge wealth of data taken at the LHC offers a unique opportunity to explore the properties of dark sectors and uncover the nature of dark matter (DM). At the same time, such an amount of data poses an unprecedented challenge to precisely determine and efficiently suppress backgrounds in order to identify potential signals of new physics. As the complexity of experimental analyses increases, there has been rapidly growing interest in using machine learning techniques to distinguish signal from background. For example, deep neural networks are powerful tools for the classification of jets, which can significantly improve the sensitivity to new physics signals hidden in QCD background. A particularly interesting and well-motivated case are so-called dark showers, which may resemble QCD jets even though they result from new interactions and contain exotic particles. A dark parton shower can be composed of both invisible dark matter particles as well as dark sector states that decay to Standard Model particles via a portal. The focus here is on the specific case of 'semi-visible jets', where the visible states in the shower are Standard Model hadrons.

1.4 2D Convolution

We would like to adopt a similar approach to CNN's for learning on point cloud data (jet graph), however, regular convolution operation cannot be applied on point clouds, as the points there can be distributed irregularly, rather than following some uniform grids as the pixels in an image. Therefore, DGCNN starts with a block called EdgeConv that refer to a jet as a point cloud which is represented as a graph, whose vertices are the points (particles) themselves and the edges are the connections between each point to its k nearest neighboring points. This way, a local patch needed for convolution is defined for each point as the k nearest neighboring points connected to it. For each particle the convolution is then performed over its nearest neighbours.

2D convolution is an extension of previous 1D convolution by convolving both horizontal and vertical directions in 2 dimensional spatial domain. Convolution is frequently used for image processing, such as smoothing, sharpening, and edge detection of images. The impulse (delta) function is also in 2D space, so $\delta[m, n]$ has 1 where m and n is zero and zeros at $m, n \neq 0$. The impulse response in 2D is usually called "kernel" or "filter" in image processing.



(a) The impulse (delta) function

(b) 2D matrix representation of impulse function. The shaded center point is the origin where m=n=0

Figure 2: Explanation of the term "kernel"

A signal can be decomposed into a sum of scaled and shifted impulse (delta) functions

$$x[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j] \cdot \delta[m-i, n-j]$$

where m, i and $m-i$ denote the column indexes of a matrix and n, j and $n-j$ denote the row indexes of a matrix. An output for a convolution between an input signal $x[m,n]$ and an impulse response $h[m,n]$ is:

$$y[m,n] = x[m,n] * h[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j] \cdot h[m-i, n-j]$$

The center point of a kernel is h[0, 0]. For example, if the kernel size is 5, then the array index of 5 elements will be -2, -1, 0, 1, and 2. The origin is located at the middle of kernel.

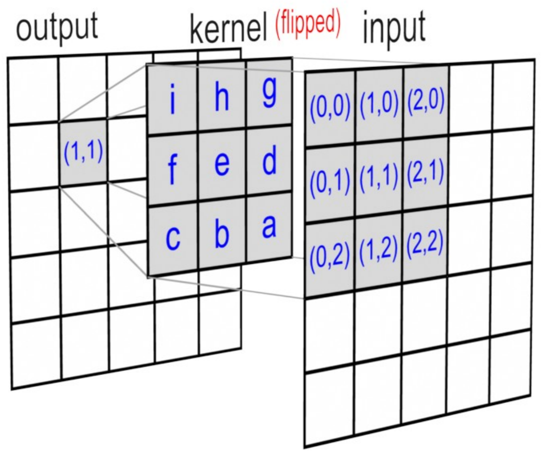


Figure 3: 2D Convolution

In DGCNN the kernel $h[m,n]$ is implemented as a dense layer and calculates edge features for a point and each of its k neighbours.

The DGCNN is Dynamic in the sense that for each EdgeConv layer in the network's architecture, new k nearest neighbors are calculated according to the distances between the particles that changes when moving from one layer to the next. Basically, the network learns new parameters according to the new distances calculated from the last layer's output.

2 Data

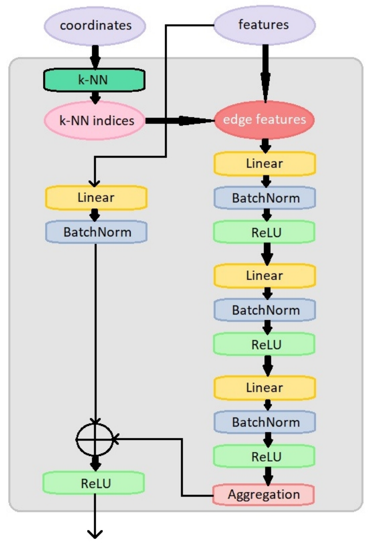
The data implemented in the network is constructed from simulated data of particles originated in jets that have pT between 125 Gev and 175 Gev. Each jet contain 50 tracks, each track contain 5 features: $log(pt)$, $\delta\eta$, $\delta\varphi$, δR and $log(xpt)$. The data is organized in a dataframe table such that each row contain all the information about a single jet. The last column of the table is labeled by the jet's label - 0 for signal and 1 for background. For training I used 40k samples, For validation 4k samples and for testing 36k samples. From the dataframe table I created a 'awkd' format file which contain 4 dictionaries that specify the particles features:

- Points - contain only $\delta\eta$ and $\delta\varphi$ of each particle.
- Features - contain all features data: $log(pt)$, $\delta\eta$, $\delta\varphi$, δR and $log(xpt)$.
- Mask - contain only $log(pt)$.

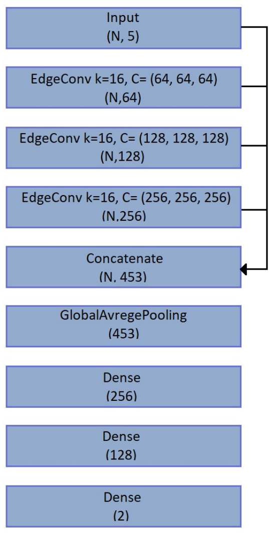
The 'awkd' files are the input to the first layer of the neural network.

3 Method and Architecture

The first layer of the network is an EdgeConv layer. An EdgeConv layer is described in figure 4a, it starts with finding the k nearest neighbours tracks to each track in a jet according to the distance parameters, $\delta\eta$ and $\delta\varphi$. Masking



(a) The structure of a single EdgeConv block.



(b) The structure of the DGCNN.

Figure 4: Architecture of the model

is required to pad the empty features of the missing tracks of the jet: If there are less then 50 tracks in a jet, very large values (999) are added to the features of the missing tracks such that the distances between the existed tracks to the padded ones are very large. The distance between a track i to his neighbor j is defined as:

$$d_{ij} = \sqrt{(\eta_i - \eta_j)^2 + (\varphi_i - \varphi_j)^2}$$

The result is a distance set of 50 records, each record contain the information about the distance between a specific track to the other 49 tracks.

The features of the k nearest neighbours are then subtracted from the features of all tracks and concatenated with the original values of the features for every track. For example, if there were 50 tracks with 16 nearest neighbours and 5 features for each track, we'll get 50 tracks with 16 nearest neighbours and 10 features (5 original features + 5 subtracted features).

The output of the knn is the input for the first convolution layer. The convolution in the ParticleNet is applied on the local patch constructed from the edges of the k nearest neighbours particles that has been found earlier in the process to the point cloud for each channel (feature), followed by batch normalization and activation function 'Relu'. A jet graph is analog to an image and a the local patch input has the same size as the kernel.

Training deep neural networks is complicated in the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layer change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it hard to train models with saturating nonlinearitiesd. We adress the problem by apllying batch normalization on the layer inputs.

The operation of convolution, batch normalization and activation occur two more times with the same number of channels followed by aggregation operation (mean over the k nearest neighbors dimention). As a result, the output is 50 tracks with a new number of features corresponding to the number of neurons in the layer. A shortcut passing this operation allows the original features to pass near and apply convolution and batch normalization on them and then being added to the output of the operation specified before. Finally, an activation function 'Relu' is applied on the final output. Two more EdgeConv blocks are stacked below the first with different number of channels corresponding to the number of neurons in each convolutional layer. The first EdgeConv correspond to 64 channels, the second corresponds to 128 channels and the last correspond to 256 channels. After the EdgeConv blocks, a concatenation is applied on the outputs from all the EdgeConv blocks and the initial input such that we end up with 5+64+128+256=453 features for each jet. After that, a channel-wise global average pooling operation is applied to aggregate over the k edge features to reduce (by meaning over the tracks dimention) the output to one feature vector for each jet. This is followed by 3 fully connected dense layers with 256, 128 and 2 nodes and adopt 'Relu' activation function for each layer exept for the classification output, where 'softmax' is applied. To prevent the dense part of the network from overfitting, we use dropout layer with a drop probability of 0.1 in front of the first 2 dense layers.

We use a learning rate schedule during training. The initial learning rate is set to $3 \cdot 10^{-4}$. We increase it linearly for 8 epochs to $3 \cdot 10^{-3}$ and decrease it to its initial value within another 8 epochs. The next 4 epochs we reduce the learning rate further to $5 \cdot 10^{-7}$. The training finishes after 20 epochs.

4 Results

I used k=16 nearest neighbours to define the local patch for the convolution layers.

If we look at the training loss we can see fast convergence according to the learning rate schedule:

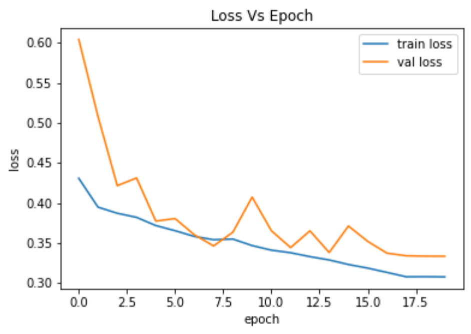
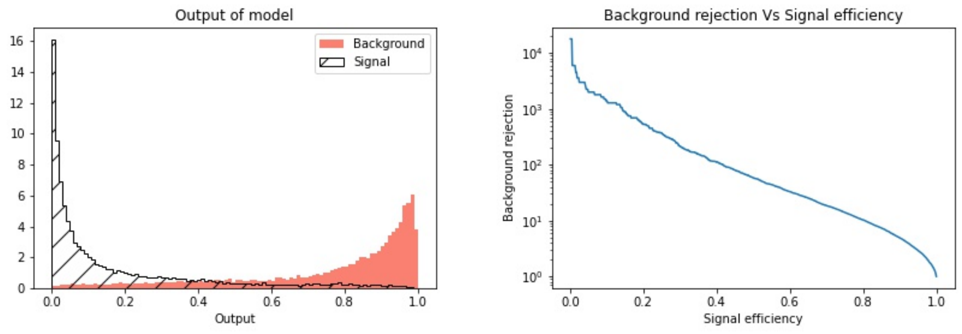


Figure 5: Loss Vs Epoch.

Looking at the output of the network for the seperated signal and background datasets we can see a good seperation of the outputs:



(a) Histogram of output probability.

(b) ROC curve.

Figure 6: Output of the model and the corresponding ROC curve

The ROC curve of these results defined as the background rejection as function of the signal efficiency such that the area below each of the background histogram and the signal histogram is normalized to 1 and presented in logarithm scale.

5 Refferences

1. H. Qu and L. Gouskos, ParticleNet: Jet Tagging via Particle Clouds, Phys. Rev. D 101 (2020), no. 5 056019, [1902.08570].

2. Elias Bernreuther, Thorben Finke, Felix Kahlhoefer, Michael Krämer, Alexander Mück, Casting a graph net to catch dark showers, report no. TTK-20-17, P3H-20-025, [2006.08639].

3. Mark Thomson, Modern Particle Physics, University of Cambridge, 2013.

4. Matthew D. Schwartz, TASI Lectures on Collider Physics, Department of Physics Harvard University, 2017.

5. http://www.songho.ca/dsp/convolution/convolution.html#convolution_2d