CS230

Deep learning jet clustering algorithm for analysis of particle collisions at the Large Hadron Collider

Camila Cendra, Justin Chen, Khaled Jedoui, and Professor Ariel Schwartzman*
Stanford University
ccendra@stanford.edu, jyc100@stanford.edu, thekej@stanford.edu, sch@slac.stanford.edu

Abstract

The Large Hadron Collider makes tens of millions of proton-proton collisions per second. Decisions about which events to save for further analysis depend on the number and intensity of jets, sprays of particles that are produced from the collisions. We propose a novel approach of clustering jets using convolutional neural networks. We base our architecture off of Mask R-CNN [2] to both produce region of interest and segmentation predictions of jets. We achieve a precision of 0.86 and recall of 0.55 with better performance in predicting higher intensity jets.

1 Introduction

The Large Hadron Collider (LHC) makes proton-proton collisions 40 million times per second. However, only 1,000 proton-proton collision events can be saved every second. Current simple algorithms attempt to cluster jets, sprays of particles produced from proton-proton collisions (Figure 1a), in order to decide which events to save. There exist theory-based jet clustering algorithms that can be used for jet clustering analysis, but they are too slow to implement on-site and for every proton-proton collision event. One of the most popular clustering algorithms utilized in particle physics is the anti-kt Fastjet clustering algorithm [1] (Figure 1b), capable of detecting the energy and location of jets within an image.

In this project, we evaluate the performance of a Mask R-CNN [2] for its use as an alternative decision-maker algorithm that can identify and locate jets in proton-proton collisions. The advantages of implementing a Convolutional Neural Network (CNN) for particle detection is two-fold: i) the CNN can determine whether an event will be saved, therefore performing the function of the simple algorithms currently utilized on-site for this purpose ; ii) the CNN can identify jets and estimate their energy, performing the role of complex clustering operations such as the anti-kt Fastjet clustering algorithm.

The input data for our Mask R-CNN network are 'raw' images of 64 x 64 pixels collected by a 2D detector during a single proton-proton event. The ground truth input for each example is obtained by implementing the anti-kt Fastjet clustering algorithm. We train a modified Mask R-CNN and evaluate its performance in learning the rules of the anti-kt jet clustering algorithm, achieving a precision of 86% with conditional post-processing filters.

*Professor Schwartzman is a faculty member with SLAC and served as the advisor for this project.

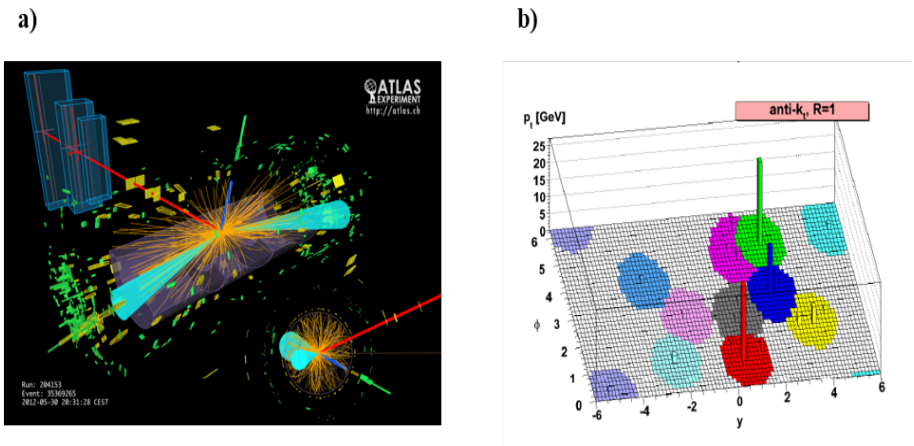

Figure 1: Representation of a proton-proton collision event, in which the blue cones are the jets containing groups of particles (a). A collision event in which the jets of particles have been clustered using the anti-kt Fasjet clustering algorithm [2] (b).

2 Dataset and Features

Our dataset is a set of a 100,000 images (64 x 64 pixels) taken during the proton-proton collision events, where the value of each pixel corresponds to the energy (pT) measured at that location (Figure 2a). The dataset was provided by Prof. Ariel Schwartzmann, and it comes from a single experiment performed at the LHC. The dataset was split using a 99 % / 1 % split for training and development (dev) sets.

In order to obtain the ground truths of our dataset; that is, the pixels in each image corresponding to a jet, we use the anti-kt FastJet clustering algorithm [1]. This processing step allows us to identify the number of jets in each image as well as their corresponding total energy and pixel-by-pixel spatial location. Identification of jets is performed using a recursive approach as described by Cacciari et al.[1]. Clustering algorithms are defined by a set of parameters, which condition the shape, distribution and minimum energy of detected jets. In our case, given the interest of this project to detect high energy jets, and as suggested by Prof. Schartzmann, we set $R = 0.4$ and a minimum energy threshold of $20GeV$.

We then assign labels to the identified jets, with the pixels comprising each jet assigned the integer value of the jet number (i.e. for Jet1 identified by the fastjet clustering algorithm, the pixels comprising this jet will be set to 1) and the background pixels set to 0. Figure 1 depicts an example input image and its associated ground-truth label. We also calculate bounding boxes around each labeled jet (based on the most extreme pixels of each jet) for training the object detection portion of our algorithm.

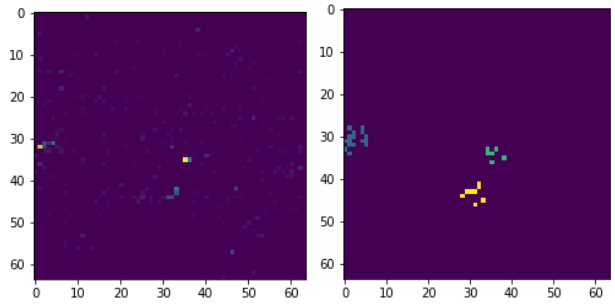

Figure 2: Example input (a) and labeling of jets via anti-kt FastJet clustering algorithm [2] (b).

3 Model

3.1 Mask R-CNN

Our model is based on Mask R-CNN [2], implemented in Keras [3]. This model detects and classifies objects in images and also outputs a segmentation of the objects. The model generates bounding boxes and segmentation masks for each instance of an object in the image. It's based on Region Proposal Network (RPN) and a ResNet101 backbone. Through training, the RPN learns the attention mechanism of proposing regions of interest (ROIs). The ROIs are fed into the ResNet portion of the network to learn classification of the regions and into a series of convolution layers to learn the segmentation.

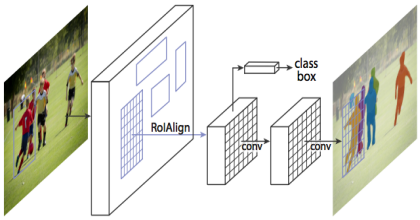

Figure 3: The Mask R-CNN framework [2]

3.2 Loss function

Mask R-CNN uses multi-task loss, meaning that the loss function is the sum of multiple losses. This is because we are trying to accomplish region proposal, classification, and segmentation.

$$\mathcal{L} = \mathcal{L}_{class} + \mathcal{L}_{box} + \mathcal{L}_{mask}$$

$\mathcal{L}_{class}$ is cross entropy loss.

$\mathcal{L}_{box}$ is L_1 loss over the box coordinates.

$\mathcal{L}_{mask}$ is per-pixel cross entropy loss.

3.3 Modifications to Mask R-CNN

To train the model to detect jets, we made several modifications to the original Mask R-CNN model. First, we reduced the output class space to two classes: jet or background. In order to speed up training and forward propagation, we reduced the size of the classification portion of the network, using ResNet50 instead of ResNet101. Finally, due to the smaller image size compared to the types of images the original Mask R-CNN was trained on, we reduced both the size of the proposed regions as well as the number of proposals per event.

4 Results

We used early stopping to retrieve the weights at the brink of over-fitting. The results were obtained through analysis on the development set. Generally, we had good performance in predicting high energy jets. The confusion matrix (Figure 6) shows that we tend to predict the same number of jets with $pT > 20GeV$ as there are in the ground truth for each event and sometimes over-predict the number of jets. However, we rarely under-predict the number of jets. This is promising as it means that our algorithm is unlikely to miss interesting events with a large number of high energy jets.

Figure 4 and Figure 5 highlight specific events where our algorithm performed well and poorly, respectively. In Figure 4, we retrieved predicted jets that overlapped significantly with the ground truth jets. The segmentation, however, seems to over-predict the number of jet pixels. In Figure 5, we predicted two jets that were not in the ground truth. These jets were not of very high pT values and

this likely means that a lower pT threshold on the FastJet algorithm would have revealed similar jets. Many of our missed predictions occur in predicting low energy jets that do not exist in the ground truth.

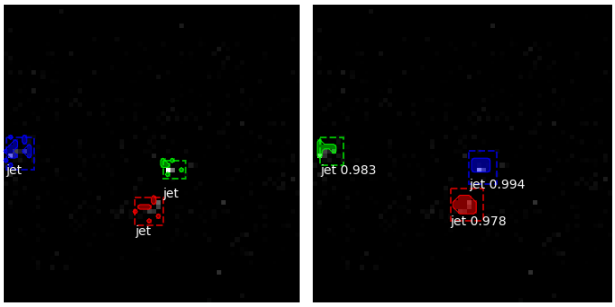

Figure 4: Success: Example ground truth (a) and prediction [2] (b).

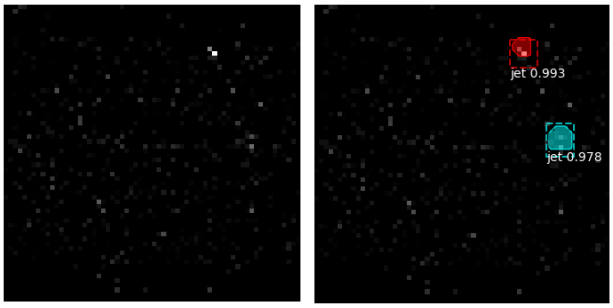

Figure 5: Failure: Example ground truth (a) and prediction [2] (b).

In order to account for this over-prediction, we implement post-processing filters to only keep the best predictions. We use the F1-Score as our metric to compare model performance. Table 1 shows the performance of our model across different post-processing filters.

Post-processing filters	Precision	Recall	F1-Score
None	0.2459	0.3005	0.2705
$pT > 20$	0.6433	0.4245	0.5115
$pT > 20$ or confidence> 0.98	0.5086	0.4135	0.4562
$pT > 20$ and confidence> 0.98	0.8592	0.5516	0.6719

Table 1: F1-Score over various post-processing filters

These analyses are performed comparing prediction versus ground truth of the bounding boxes of the jets. Similar comparisons over the segmentation of the jets resulted in worse performance. However, as this model will be used to classify interesting events, we are not overly concerned with the worse performance in segmentation.

Forward propagation for a single event takes 64ms on an Nvidia GTX 1060-6GB. The algorithms used at LHC run on specialized FPGAs that should run in faster time.

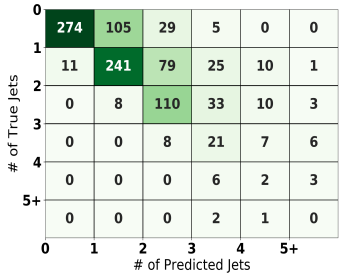

Figure 6: Confusion matrix of number of predicted jets (with $pT > 20GeV$).

5 Conclusion/Future Work

Using the Mask R-CNN architecture, we were able to produce a model that accurately predicted high energy jets. Our model performed significantly better in detecting jets (with bounding boxes) as opposed to per-pixel segmentation. Prediction performance was significantly improved by applying post-processing filters to the model output to only keep the highest confidence predictions of high energy jets. Based on these results, this novel deep learning approach to jet classification is promising.

For future work, there are several steps to take to improve model performance. We had to employ early stopping as the model over-fit the training data set. Increasing the size of the training dataset would be a good way to address this problem. In addition, all of the events used to train the model come from a single experiment. While events should not differ drastically between experiments, in order to get a representative dataset, more data should be drawn from varied experiments. From the analysis of our model, we found that detection performance was higher than segmentation performance. It would be interesting to experiment with limiting or staggering the multi-task learning in order to focus solely on jet detection as opposed to both detection and segmentation. Finally, it would be useful to test throughput of the model using the hardware used at the LHC in order to gauge the ability of the model to run in real-time.

6 Code

Code for this project can be found at:
`https://github.com/ccendra/cs230-jetRecognition_finalProject`.
The main file used for training is train_jets_100k_20GeV_scratchTrain_newJets.ipynb and for analysis is plots.ipynb. The backbone of the code is based off of an open-source implementation of Mask R-CNN in Keras [3].

7 Contributions

Work was shared equally among our team. Each team member contributed in writing code, writing reports, and creating/presenting the poster. Within those tasks, Khaled led model architecture research and design, Camila led the training and iteration of the model, and Justin led analysis of results.

We would like to give special thanks to Lucio Dery for his guidance as our CS 230 teaching assistant and to Murtaza Safdari for his guidance and for helping to provide the data used in this project.

References

[1] M. Cacciari, G.P. Salam and G. Soyez, Eur.Phys.J. C72. FastJet user manual. arXiv:1111.6097, 2012

[2] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask R-CNN. arXiv:1703.06870, 2017

[3] `https://github.com/matterport/Mask_RCNN`

Appendix

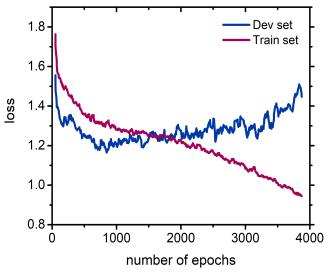

Figure 7: Training and validation loss.

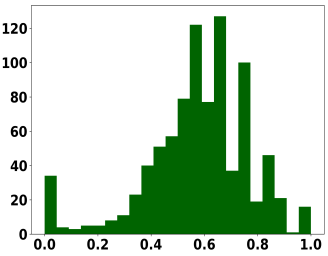

Figure 8: Histogram of IoU between ground truths and closest predictions.

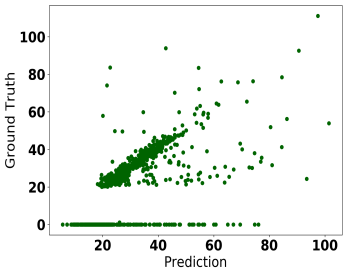

Figure 9: Comparison of pT of highest energy jets in each event between ground truths and predictions (GeV).