DeepHalo Extension: Instance Segmentation of Dark Matter Halos

Rishav Das and Ipsita Rout
National Institute of Science Education and Research

Abstract

Our aim is to instance segmentation on point cloud data using Graph Neural Networks (GNNs). Employing DynamicEdgeConv as the primary tool, we endeavored to delineate distinct instances within the point cloud. We used snapshot 99 of TNG50-4-Dark catalogue. Initially, we explored the conversion of points to octree structures; however, this approach was deemed unnecessary. Despite our efforts, severe class imbalance thwarted our model's efficacy, leading to unreliable predictions. While several strategies were attempted to address this issue, none yielded satisfactory results. Ultimately, we employed focal sigmoid loss as the loss function and Intersection over Union (IOU) as the evaluation metric. Despite these efforts, significant challenges persist, highlighting the complexity of instance segmentation tasks in the context of heavily imbalanced datasets.

1 Introduction

1.1 Subfinders and Point Cloud Data

Subfinders in point cloud data refer to algorithms or techniques used to identify and extract specific features or substructures within a larger point cloud dataset. Point clouds are collections of data points in a three-dimensional coordinate system, often generated by 3D scanning technologies like LiDAR (Light Detection and Ranging) or photogrammetry.

In point cloud processing, subfinders play a crucial role in various applications such as object recognition, scene understanding, and geometric analysis.

An example of such a task involves object detection and segmentation. This entails employing subfinders to discern specific objects or substructures within a point cloud, including buildings, vehicles, trees, or other environmental elements. Through an analysis of the geometric properties and spatial interconnections among points, these algorithms effectively partition the point cloud into separate objects or areas of significance.

Instance segmentation and semantic segmentation are two fundamental tasks in computer vision, each serving distinct purposes in image analysis.

Semantic segmentation involves labeling each point in a point cloud with a corresponding class label, such as "object type 1," "object type 2" "object type 3," etc. The goal is to partition the point cloud into semantically meaningful regions, providing a holistic understanding of its content. Unlike object detection, semantic segmentation does not differentiate between individual instances of the same class; instead, it focuses on categorizing points based on their shared characteristics.

On the other hand, instance segmentation extends semantic segmentation by not only labeling each point with a class but also distinguishing between individual object instances within the same class. This means that in addition to identifying the category of each point (e.g., "object type"), instance segmentation algorithms also delineate the boundaries of each distinct object (e.g., differentiating between two separate objects of the same object type).

For our specific task, we are employing dark matter halos, which we instance segment into their sub-halos.

2 DATA

TNG-Illustris, short for "The Next Generation Illustris," is a cutting-edge cosmological simulation project aimed at modeling the evolution of the universe and the formation of cosmic structures, such as galaxies and galaxy clusters.

The TNG Illustris simulation incorporates advanced numerical techniques and high-performance computing to simulate the complex interplay of dark matter, gas, stars, and black holes over cosmic time scales.

2.1 TNG50-4-Dark

For our investigation, we have selected the 99th snapshot derived from the TNGdark-50-4 model, which represents a distinct variant or iteration within the TNG (The Next Generation) cosmological simulation series. This particular simulation is meticulously crafted to delve into the intricate properties and dynamic behaviors inherent to dark matter within the cosmic landscape.

2.2 Data Specification

- There are 100 snapshots stored for every run. These include all particles/cells in the whole volume.
- The snapshot data is structured based on group/subgroup memberships assigned by the FoF or Subfind algorithms, rather than spatial position. Within the snapshot files, particles are sorted by their group and subgroup assignments. The sorting order within each particle type is: GroupNumber, SubgroupNumber, BindingEnergy. Additionally, particles belonging to a group but not to any subgroups ("fuzz") are placed after the last subgroup.
- Each HDF5 snapshot comprises multiple groups, including "Header," "Parameters," "Configuration," and five "PartTypeX" groups. **We are using PartType1 only, i.e. dark matter only group.**

Features used for our model were positions, velocities and gravitational potential energy of each particle.

Overall, features from the snapshot used for both data preprocessing and machine learning are as follows:

- **Coordinates:** Spatial position within the periodic simulation domain of BoxSize. Comoving coordinate.
- **ParticleIDs:**The unique ID (uint64) of this gas cell. Constant for the duration of the simulation. May cease to exist (as gas) in a future snapshot due to conversion into a star/wind particle, accretion into a BH, or a derefinement event
- **Potential:**Gravitational potential energy.
- **Velocities:**Spatial velocity.

3 Methodologies

Initially, we formulated an array structure dedicated to each subhalo, wherein each array possesses a standardized shape denoted as (N, 8). In this context, the variable N symbolizes the count of data points encapsulated within the confines of the particular subhalo under consideration. Within each array instance, the eight distinct columns delineate specific parameters or attributes associated with the constituent data points.

- Coordinates in 3 directions
- Velocities in 3 directions

- Gravitational potential of the particle
- A label indicating the subhalo to which the particle belongs.

In our pursuit of employing Graph Neural Networks (GNNs) for semantic segmentation tasks on point cloud data, we embarked on a crucial initial step: converting the raw point cloud data into a graph format amenable to GNN processing. To accomplish this conversion, we turned our attention to the exploration of octree structures, which offer a sophisticated framework for organizing and representing point cloud data in a hierarchical manner.

Octree structures, renowned for their efficacy in partitioning three-dimensional space, present an elegant solution for converting point cloud data into a graph format. By recursively subdividing the spatial domain into octants, each potentially containing points, octrees enable the representation of complex geometric structures with varying densities of data points. This hierarchical organization not only facilitates efficient storage and retrieval of point cloud information but also lends itself naturally to graph-based representations, where nodes correspond to octants and edges capture spatial relationships between them.

In our endeavor to harness octree structures for point cloud conversion, we developed bespoke octree conversion code tailored to our specific requirements. This bespoke solution allowed us to finely control the octree generation process, accommodating nuances and intricacies inherent to our dataset and segmentation objectives. By implementing our custom conversion code, we ensured flexibility and adaptability, crucial factors in the pursuit of optimal performance in semantic segmentation tasks.

Furthermore, in our quest for comprehensive exploration and experimentation, we augmented our efforts by integrating existing tools and libraries into our workflow. Among these resources, the PyOctree library, curated by the esteemed Michael Hogg, emerged as a valuable asset. Leveraging the functionalities provided by the PyOctree library augmented our capabilities, offering additional perspectives and avenues for experimentation in our quest for refined semantic segmentation solutions.

Through the judicious combination of innovative custom code development and strategic integration of external resources such as the PyOctree library, we fortified our approach to point cloud conversion, laying a robust foundation for subsequent GNN-based semantic segmentation endeavors. Our commitment to methodical exploration and integration exemplifies our dedication to pushing the boundaries of knowledge and technology in the dynamic field of point cloud analysis and graph-based machine learning.

However, our efforts took a turn when we discovered the DynamicEdgeConvWang et al. 2019 Team 2024 method.

- **Handling Irregular Data:** Point cloud data lacks the regular grid structure present in images or volumetric data. DynamicEdgeConv addresses this challenge by dynamically creating edges between neighboring points based on their spatial proximity.
- **Edge Features:** DynamicEdgeConv computes edge features by aggregating information from neighboring points within a specified radius. Unlike traditional convolutional operations, which operate on fixed-size kernels, DynamicEdgeConv adapts the receptive field dynamically based on the local point density.
- **Robust to Point Order:** DynamicEdgeConv is inherently invariant to the order of points in the input point cloud, making it robust to permutations or random shuffling of point coordinates.
- **Integration with CNNs:** DynamicEdgeConv can be seamlessly integrated into existing CNN architectures designed for point cloud processing, such as PointNet and its variants. By replacing traditional convolutional layers with DynamicEdgeConv layers, these architectures gain the ability to capture local spatial relationships and semantic features more effectively.

DynamicEdgeConv rendered the octree conversion unnecessary for our purposes. DynamicEdgeConv provided an alternative approach that effectively addressed our requirements for semantic segmentation on point cloud data, thus obviating the need for octree-based solutions.

4 The Model

:

We have established a PyTorch model, denoted as PointNetInstanceSeg, which encapsulates our architectural design for semantic segmentation tasks tailored to point cloud data. This model, crafted within the PyTorch framework, embodies our specialized implementation aimed at addressing the intricacies and challenges inherent to instance segmentation within the context of point cloud analysis.

PointNetInstanceSeg is a subclass of nn.Module, the base class for PyTorch neural network modules. This model is tailored for instance segmentation tasks on point clouds.

The model comprises two primary components:

1. DynamicEdgeConv layers: These are graph convolution layers specialized for point clouds. They compute features for each point based on its K nearest neighbors. The model includes two such layers, each followed by a SiLU (Sigmoid Linear Unit) activation function.

2. A fully connected (nn.Linear) layer: This layer predicts the instance mask for each point. It accepts 64 input features and outputs 21 features.

The forward method defines the model's forward pass. It accepts a data object with pos and edge_index attributes. Traditionally, pos should represent point positions, but we have also included velocities and gravitational potential since those are also features that we use. edge_index represents graph edges. These are passed through the DynamicEdgeConv layers and the fully connected layer, and the output is returned.

5 PROBLEMS

We encountered several challenges during our model training process. One major issue was the heavy class imbalance in the dataset, with the largest subhalo dominating and resulting in 80% accuracy initially. To address this, we experimented with weighted loss functions such as cross-entropy, using class frequencies as weights. Additionally, we attempted to improve results by one-hot encoding labels and switching to BCEwithlogits loss, but observed limited improvement.

We also explored binary segmentation as an alternative approach but found that the class imbalance persisted, making this avenue unviable. Eventually, we opted to employ the SMOTE technique to oversample the minority class and switched to focal sigmoid loss as a loss function, with IoU as a metric. While this approach yielded slightly improved results, the gains were modest.

The prediction were good for those halos that had small number of subhalos. But for those halos which had a large number of subhalos (/>5) the predictions were way off.

The utilization of SMOTE resulted in a notable increase in computational requirements. This substantially augmented the dataset size, leading to heightened computational demands during both training and evaluation phases.

During training, we faced computational constraints, particularly with longer epochs. Training for 20 epochs took 5 hours, and attempting 100 epochs required compromising on model architecture by removing one DynamicEdgeConv layer to complete training. These challenges underscore the complexities of dealing with class imbalance and computational limitations in training deep learning models for instance segmentation tasks on heavily imbalanced datasets.

6 RESULTS

The outcomes obtained from the experimentation exhibit a notable degree of dissatisfaction. While certain predictions demonstrated perceptible validity for select halos, the overall consistency remained conspicuously deficient, thereby implying a plausible attribution of the outcomes to chance occurrences. Our comprehensive evaluation, conducted across 23 halos, yielded a mean Intersection over Union (IOU) score of 0.775 ± 0.998. However, the pronounced variance accompanying these results signifies a lack of reliability and statistical significance, thereby compromising the integrity of the evaluation.

The underlying rationale for the suboptimal predictive performance observed in halos characterized by a substantial number of subhalos is posited to stem from the issue of class imbalance. Specifically, the presence of a multitude of subhalos results in a diminutive allocation of data points per individual subhalo, exacerbating the challenge of effectively delineating the primary halo. This imbalance

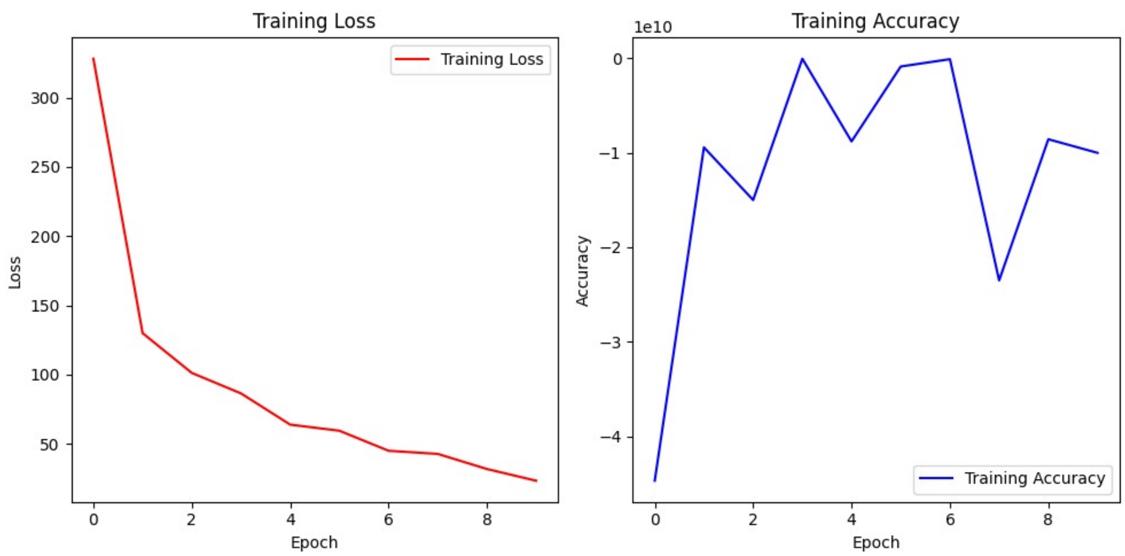


Figure 1: Training loss and accuracy

underscores a critical predicament; however, the imposition of heightened penalties for misclassifying the primary halo risks exacerbating the issue, potentially dissuading the machine learning model from predicting it altogether.

6.1 Future Plans

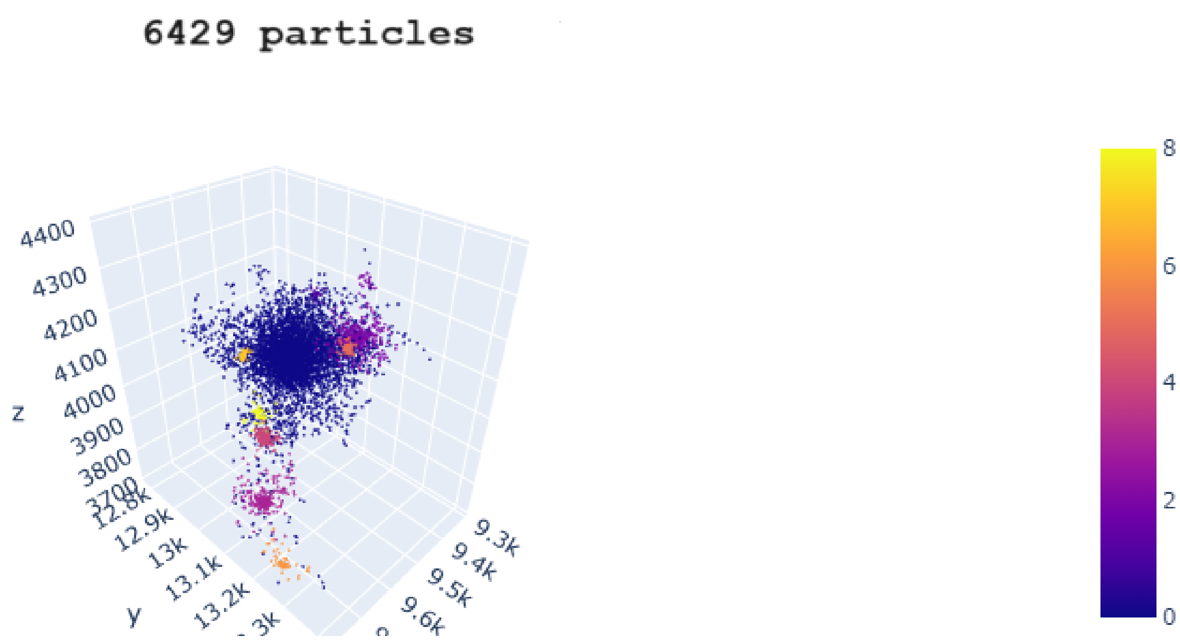
What needs to be done in the future is to overcome the significant class imbalance prevalent in the dataset without compromising the prediction of the primary halo. Initially, we explored various strategies, including weighted loss functions and binary segmentation approaches, but encountered limited success. Subsequently, we adopted the Synthetic Minority Over-sampling Technique (SMOTE) to rebalance the dataset, coupled with focal sigmoid loss as the loss function and Intersection over Union (IOU) as the evaluation metric. Despite these efforts, our results remained suboptimal, with instances of reasonable predictions being inconsistent. Furthermore, the computational demands escalated substantially due to the implementation of SMOTE, hindering efficient model training and evaluation. Our findings underscore the complexities inherent in instance segmentation tasks on heavily imbalanced datasets and highlight the need for further exploration of advanced methodologies and computational optimization techniques in this domain.

References

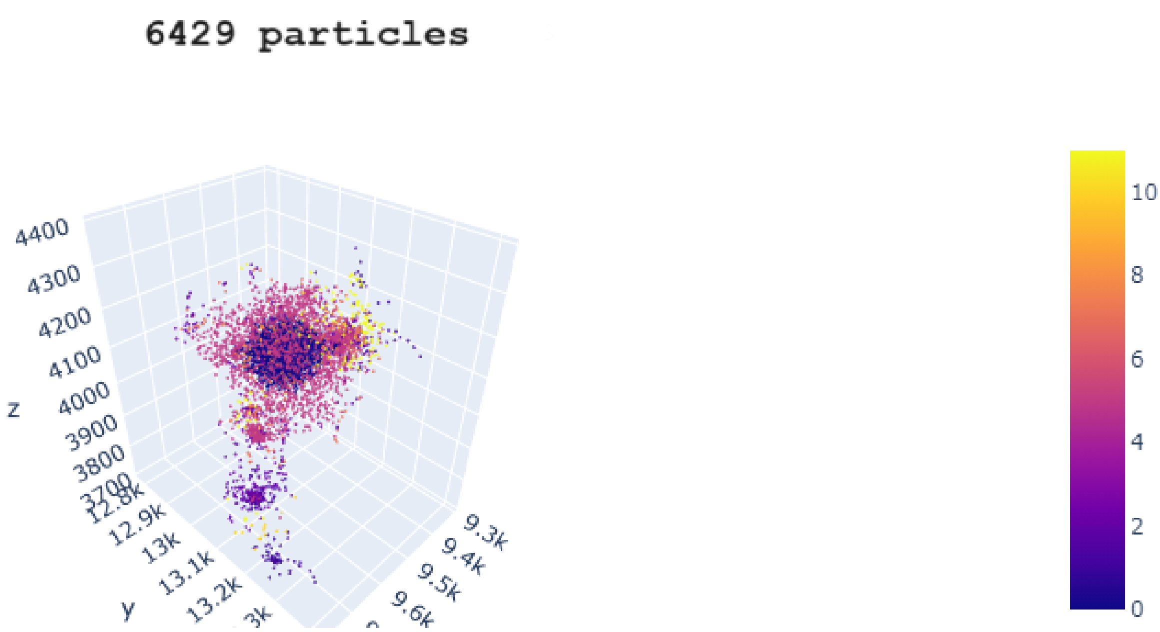
Rishav Das, Ipsita Rout (2024). $deephalo_{g}nn$. https://github.com/Rishav-Das1/deephalo_gnn.git.

Team, PyG (2024). $Torch_{g}eometric.nn.conv.dynamicedgeconv$. URL: https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.conv.DynamicEdgeConv.html.

Wang, Yue et al. (2019). "Dynamic graph CNN for learning on point clouds". In: *ACM Transactions on Graphics* 38.5, 1–12. DOI: 10.1145/3326362.



(a) ground truthRishav Das 2024



(b) predictedRishav Das 2024

Figure 2: Visual representation of ground truth and predicted labels