ATLAS PUB Note

ATL-PHYS-PUB-2021-002

6th August 2021

Physics Object Localization with Point Cloud Segmentation Networks

The ATLAS Collaboration [1]

In modern particle physics experiments, the identification and trajectory of physics object, e.g. leptons and jets, depends on a complex pipeline of feature extraction, search, and machine learning algorithms. Deep neural networks (DNNs) offer a possibility of streamlining this process. This note describes the production of a dataset that the ATLAS collaboration is releasing for community use. It is formatted like popular point-cloud datasets, e.g. ShapeNet or S3DIS, which have been used to research new network topologies and applications. This dataset broadens the domain of these public datasets to include high energy physics detectors and encourages new deep learning research. The note also describes the application and performance of the PointNet++ and DGCNN networks, as well as the GravNet and GarNet networks developed by the HEP community. These networks predict to what type of physics object each detector readout channel is associated, e.g. electron, jet, or neither in this case. In the analysis, the performance is measured using the mean Intersection over Union scores. The result suggests that these networks could be used as a method to narrow the search space in a traditional reconstruction pipeline.

The dataset is derived from simulated $Z \to e^+e^-$ production in association with jets. Each event is represented as a list of non-zero ATLAS detector readouts, including the detector channel readout position, raw readout value, and some truth information. The truth information indicates if a channel is associated to an electron, jet, or neither. There is also tracking information included to facilitate track studies.

[1] The full author list can be found at:
https://atlaspo-eos.web.cern.ch/authorlists/PUB-EGAM-2020-01/atlas_authlist.pdf

1 Introduction

Reconstructing and identifying objects, like particle jets and leptons, is an integral piece of both trigger and physics object reconstruction with data collected by the ATLAS detector [1] at the LHC [2]. Current algorithms have been developed and honed over the last two decades leading to very high precision particle identification [3–5]. These algorithms perform a step-wise reduction and refinement transforming raw detector readout channels to energy clusters and particle tracks, which are used in identifying final state particles and measuring their four momenta.

The ATLAS collaboration is releasing the dataset described herein to encourage cross-domain research. The first study motivating this effort will be described below. It aims to understand if deep machine learning methods applied to low-level ATLAS detector data can effectively label detector channels as being associated with a jet or electron. The dataset contains detector readout channels represented as a point-cloud with each channel represented by the central position of its active detector, and a scalar value that represents its readout. Channels with zero reading (no signal output or below noise threshold) are omitted to reduce the storage footprint and computational intensity.

An approach referred to as point-cloud semantic-segmentation in the deep learning community is applied here. This means classifying the non-zero readout from each channel in one collision event as belonging to a jet, electron, or neither. Great strides on point-cloud segmentation have been made recently, usually evaluated on benchmark datasets such as ShapeNet [6], S3DIS [7], and ScanNet [8]. These are datasets focused on computer vision tasks like segmenting rooms, city streets, and everyday objects. They can often be solved by only looking at a small handful of points at a time.

Deep neural networks have been shown to be the most effective tool for performing point-cloud segmentation [9–22]. These networks can fit complex datasets and generalize well to unseen samples. The immediate question is how one should process point-cloud data. Convolutional Neural Networks (CNNs) are a class of neural network that can learn features with spacial extent making them a logical choice for this problem. CNNs [23–25] have achieved state-of-the-art performance in many computer vision tasks that require efficient learning of image, video, audio, and voxel data. However, a core assumption of CNNs is that the data is arranged on a lattice. While ATLAS data is arranged geometrically based on the detector design, it does not follow a pattern easily represented by an multi-dimensional array. Point clouds have no such structure.

The application of machine learning techniques to particle physics detector data is a very active field of research. References [26, 27] are examples of training CNNs using calorimeter data, formatted as two dimensional images arranged in pseudorapidity (η) and azimuthal (ϕ) angles (see Section 2 for description), to perform jet classification. Reference [28] is an example of a study using Graph Neural Networks, which thoroughly explains the challenges of translating irregularly shaped and spaced detectors to the CNN lattice data structure used in image classification tasks. This study goes beyond past results, that typically have used simplified detector models and only data from calorimeters, by using a fully operational LHC detector simulation that includes the silicon tracker.

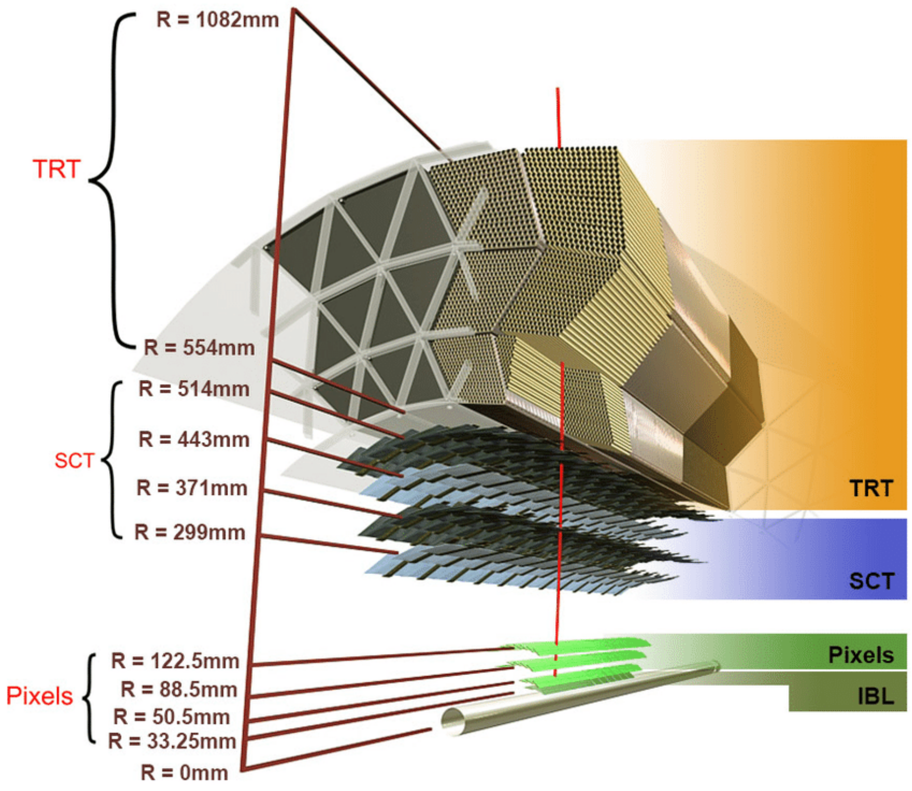


Figure 1: The Inner Detector structure inside the barrel region $|\eta| < 2.5$ [1].

2 The ATLAS Detector

The ATLAS experiment [1] at the LHC [2] is a multi-purpose particle detector with a forward-backward symmetric cylindrical geometry and a near 4π coverage in solid angle[2]. It consists of an inner tracking detector (ID) surrounded by a thin superconducting solenoid providing a 2 Tesla axial magnetic field, electromagnetic (EM) and hadron (HAD) calorimeters, and a muon spectrometer.

The ID (Figure 1) covers the pseudo rapidity range $|\eta| < 2.5$. It is divided into two regions, the barrel region where the cylinders around the beam pipe provide full coverage for $|\eta| < 1.5$, and the end-cap region where the disks, placed perpendicular to the beam axis, extend the η coverage to 2.5. The ID consists of three subdetectors, silicon pixel (Pixel), silicon microstrip (SCT), and transition radiation tracking (TRT) detectors. The Pixel detector consists of four layers located at a mean radius of 33, 50.5, 88.5, and 122.5 mm in the barrel and three disks in each of the end-caps. The innermost layer (the insertable B-layer [29, 30]) has the highest granularity, with a typical pixel size of 50 μm $\times$ 250 μm $\times$ 200 μm. The next three layers have a typical pixel size of 50 μm $\times$ 400 μm $\times$ 200 μm. Each individual pixel measures the charge collected, and outputs the time duration it exceeds a given threshold (referred to as time over threshold or TOT). There are about 80 million readout channels in the Pixel detector. Outside of the Pixel detector, the SCT detector consists of four double strip layers in the barrel at radii of 299, 371, 443, and

[2] ATLAS uses a right-handed coordinate system with its origin at the nominal interaction point (IP) in the centre of the detector and the z-axis along the beam pipe. The x-axis points from the IP to the centre of the LHC ring, and the y-axis points upwards. Cylindrical coordinates (r, ϕ) are used in the transverse plane, ϕ being the azimuthal angle around the z-axis. The pseudorapidity is defined in terms of the polar angle θ as $\eta = -\ln\tan(\theta/2)$. Angular distance is measured in units of $\Delta R \equiv \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$.

514 mm, with nine disks in each of the end-caps. A typical strip size is 126 mm $\times$ 80 μm $\times$ 285 μm in barrel. Unlike pixels, strips only provide the number of consecutive strips recording a hit as output. There are about six million channels in the SCT detector.

The high-granularity EM and HAD calorimeters measure the EM and hadronic energy up to $|\eta| = 4.9$. The EM barrel and EM end-cap are liquid-argon (LAr) sampling calorimeters covering $|\eta| < 1.475$ in the barrel, and $1.375 < |\eta| < 3.2$ in the end-caps. These calorimeters are segmented into three layers (barrel shown in Figure 2). Layer 1 is finely segmented in the η direction, typical cell size is 0.003×0.1 in $\Delta\eta \times \Delta\phi$. Layer 2 has a granularity of 0.025×0.025 in $\Delta\eta \times \Delta\phi$. Layer 3 has a cell size of 0.05×0.025 in $\Delta\eta \times \Delta\phi$.' The thickness varies in each layer to maintain a constant radiation length X_0 as shown in Figure 2. In front of the Layer 1, there is a thin presampler (PS) layer, covering the pseudorapidity interval $\eta < 1.8$, with a granularity of $\Delta\eta \times \Delta\phi = 0.025 \times 0.1$. The HAD end-cap is also a LAr sampling calorimeter covering $1.5 < |\eta| < 3.2$ with 4 sampling layers separated into two granularity regions of 0.1×0.1 in $\Delta\eta \times \Delta\phi$ for $1.5 < |\eta| < 2.5$ and 0.2×0.2 in $\Delta\eta \times \Delta\phi$ for $2.5 < |\eta| < 3.2$. The barrel HAD (steel/scintillator-tile) calorimeter consists of 64 modules of size $\Delta\phi \sim 0.1$. The barrel covers the region $|\eta| < 1.0$, and the two extended barrels cover the region $0.8 < |\eta| < 1.7$. The central hadronic calorimeter system is segmented into three layers as shown in Figure 3 with granularity of 0.1×0.1 in $\Delta\eta \times \Delta\phi$ for the first two layers and 0.2×0.1 in $\Delta\eta \times \Delta\phi$ for the third. In total, the calorimeters have approximately 200,000 readout channels.

The muon spectrometer surrounds the calorimeters and is based on three large air-core toroidal superconducting magnets with eight coils each. The field integral of the toroids ranges between 2.0 and 6.0 T·m across most of the detector. The muon spectrometer includes a system of precision tracking chambers and fast detectors for triggering.

A two-level trigger system is used to select events [31]. The first-level trigger is implemented in hardware and uses a subset of the detector information with lower resolution to reduce the accepted rate to at most nearly 100 kHz. This is followed by a software-based trigger with access to the full event data that reduces the accepted event rate to 1 kHz on average depending on the data-taking conditions.

3 Method

Simulated collisions in the ATLAS detector are used to generate a training dataset on which the following point-cloud networks are evaluated: PointNet++ [10], DGCNN [13], GravNet, and GarNet [28]. PointNet++ is a network that learns functions only on individual points, DGCNN is a state-of-the-art computer vision network that operates on graphs, and GravNet and GarNet are state-of-the-art graph networks in the High-Energy Physics domain. The goal is to characterize the existing literature against this novel dataset. Each of the reference networks have published software implementations which are used as written for these studies.

3.1 Dataset

The dataset is composed of three different samples, each with the following processes generated for pp collisions at $\sqrt{s} = 13$ TeV

1. $Z \rightarrow e^+e^-$ + 2-jets: with jet $p_T > 20$ GeV and electron $p_T > 10$ GeV,

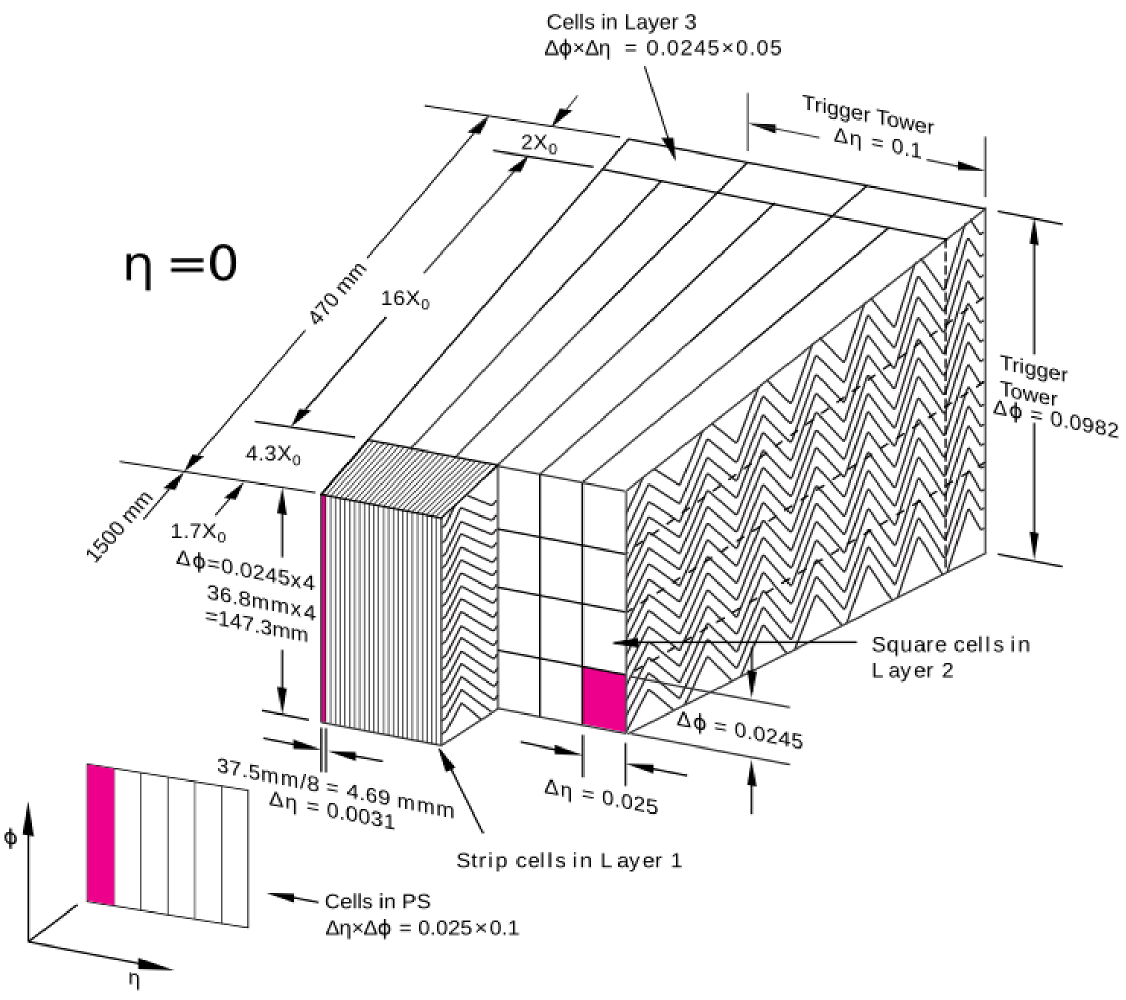


Figure 2: Sketch of a barrel EM calorimeter module [1].

2. $Z \rightarrow e^+e^-$ + 2-jets: with jet p_T > 700 GeV and electron p_T > 10 GeV, and
3. $ZZ \rightarrow e^+e^-e^+e^-$ with electron p_T > 10 GeV and events with reconstructed jets are vetoed,

with the relevant kinematic distributions in truth object p_T, η, and ϕ shown in Figures 4, 5, and 6, respectively. p_T in this context refers to the transverse momentum. In addition, the distribution of channels per truth object is shown. Dataset (1) covers the full p_T spectrum, dataset (2) aims to provide more jet examples at higher energy, and dataset (3) provides more training examples in the electron class to address the imbalance in detector channels associated with electrons versus jets.

These collision events were generated at leading order with MadGraph v2.6.7 [32] and showered in Pythia v8.235 [33] using the default tune. The hard process contains a fixed jet multiplicity. The parton distribution function NNPDF2.3 LO [34] was used. The events are fully simulated using the ATLAS Geant4 [35] detector simulation. The ATLAS simulation infrastructure [36] was employed on the Theta supercomputer at Argonne National Laboratory to generate these samples.

Only signals from the Pixel, SCT, and calorimeters are included in the output dataset. The TRT and muon spectrometer were not included in this dataset. Each event contains all non-zero inputs from these detectors

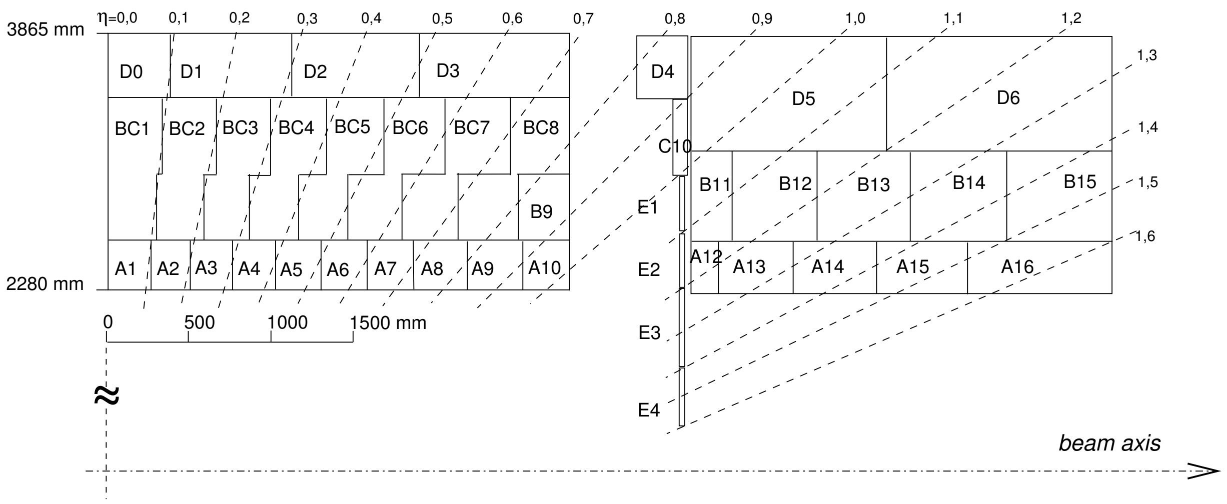


Figure 3: Segmentation in depth and η of the tile-calorimeter modules in the central (left) and extended (right) barrels [1].

including their physical position and the number of hits in a tracking detector or the deposited cell energy in a calorimeter cell (as described in [37, 38]). Inputs from the LAr calorimeter had a noise suppression applied requiring values to be greater than 250 MeV. Only signals from detector channels within the range $|\eta| < 3.0$ are included in this dataset. Effects due to pile-up, the result of multiple protons interacting in the same or neighboring bunch crossing, are not included in this sample. There is vertex smearing applied, though the vertex location was not varied. All collisions occur at $(x, y, z) = (-0.5, -0.5, 0)$ mm in the detector coordinates, with a spread of $(10^{-2}, 10^{-2}, 42)$ mm.

For each detector input, a unique identification number (id) is assigned. Their three dimensional geometric center position is extracted from the Run 2 ATLAS detector geometry with alignment correction. The "raw" detector value for each channel is taken from the time over threshold (ToT) in the Pixel detector, the number of consecutive strips that recorded a hit in the SCT, and the digitized energy deposition in the HAD/EM calorimeters.

Truth labels for each detector input are derived using the (η, ϕ, r)-position from generated leptons, reconstructed jets, and reconstructed tracks. Generated leptons were used as their location is correlated to their energy deposition in the detector and does not change significantly through the showering and reconstruction process. However, the location of generated partons is not necessarily correlated to the energy deposition in the detector as it may go through hard decays during showering resulting in multiple distinct jets. Only truth leptons, reconstructed jets and detector channels that fall within the range $|\eta| < 3.0$ are included in this dataset omitting the more complicated geometry in the forward region.

The lepton label is assigned to inputs based on the position of the generated truth leptons. The jet label uses the ATLAS Flavor tagging scheme [39], which is quickly reviewed here. ATLAS defines truth jets, derived from hadronic parton showers, and reconstructed jets, derived from reconstructed energy clusters based on calorimeter measurements. The truth jets are matched to reconstructed jets based on proximity. The proximity between two objects with $(\eta_i, \phi_i)|_{i=1,2}$ is measured with $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$ where $\Delta\eta = \eta_1 - \eta_2$ and $\Delta\phi = \phi_1 - \phi_2$. Reconstructed jets must match a truth jet within $\Delta R < 0.3$. Reconstructed jets within $\Delta R < 0.3$ of a truth lepton are removed to suppress reconstructed jets caused by electrons. Finally, detector inputs in the tracking detectors are assigned a value that identifies which unique track with which

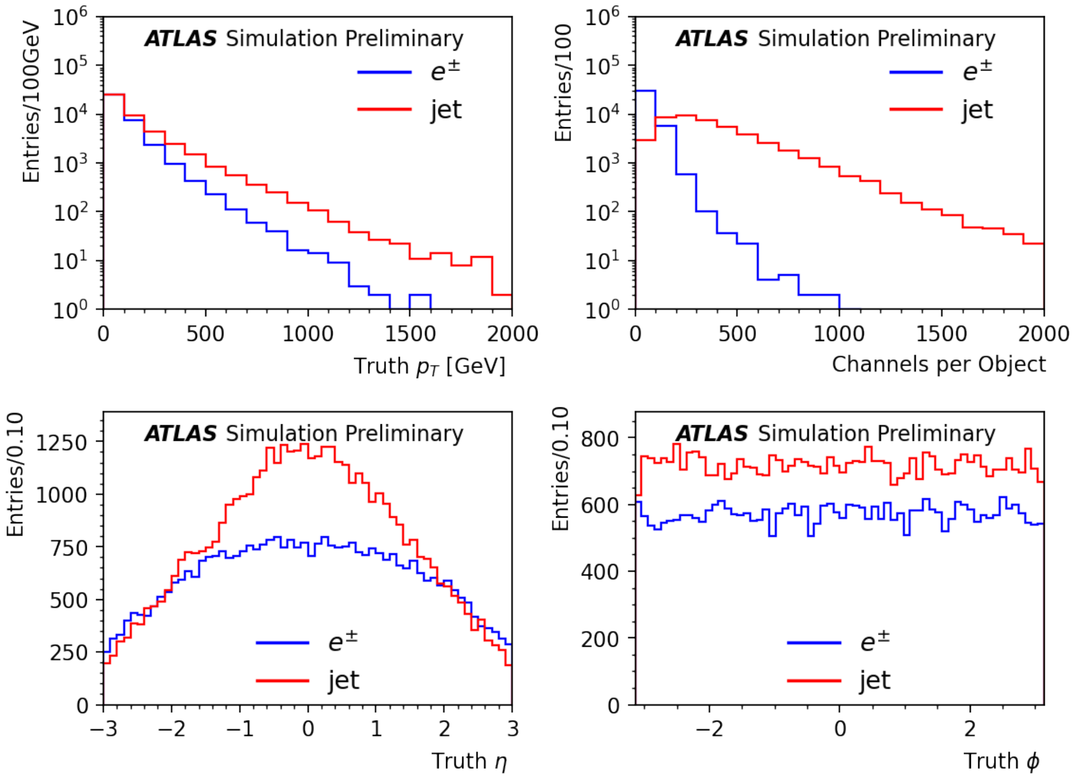


Figure 4: $Z \rightarrow e^{+}e^{-}$ + 2-jets: with jet $p_T > 20$ GeV and electron $p_T > 10$ GeV. Showing truth p_T (top-left), η (bottom-left), ϕ (bottom-right), and number of detector channels contained (top-right) for each object

it was associated during track reconstruction. Tracks are defined as generated stable particles from the decay process with hits in more than two layers in the Pixel and SCT detectors. This additional piece of information is not used in the training described below, but included to encourage track reconstruction studies.

A simple procedure is used to label detector channels as belonging to a jet or electron. Only reconstructed jets passing the criteria above are used. If a detector channel is within a configurable ΔR region of the object, it is labeled with that object's particle ID. For leptons $\Delta R < 0.2$ is used, while for jets from b, c or light quarks $\Delta R < 0.4$ is used. Anything not labeled as jet or lepton is labeled as background. In cases of overlap, the jet label takes priority, however this occurs in less than 0.6% of the objects.

Averaging across the three physics process, without pile-up included, each event contains roughly 3000 non-zero channels of the possible 86.2 million channels, but extreme events contain up to 23,000 non-zero channels in dataset (2). A total of 117,000 events were produced (20,000 in dataset (1), 39,000 in dataset (2), 58,000 in dataset (3)) containing about 150,000 electrons and 330,000 jets. The dataset contains 330 million channels in total, 30% are points labeled as jets, 7% are labeled as electrons, and the remaining are background which is dominated by the detector noise and truth failing the selection described above. Figure 7 shows the distribution of non-zero channels per event for the three datasets generated. The highest bin contains overflow in which there are only two events.

As described in Section 2, it would be prohibitive to store and process dense representations of these events so a sparse representation is used which is described in the Appendix A.

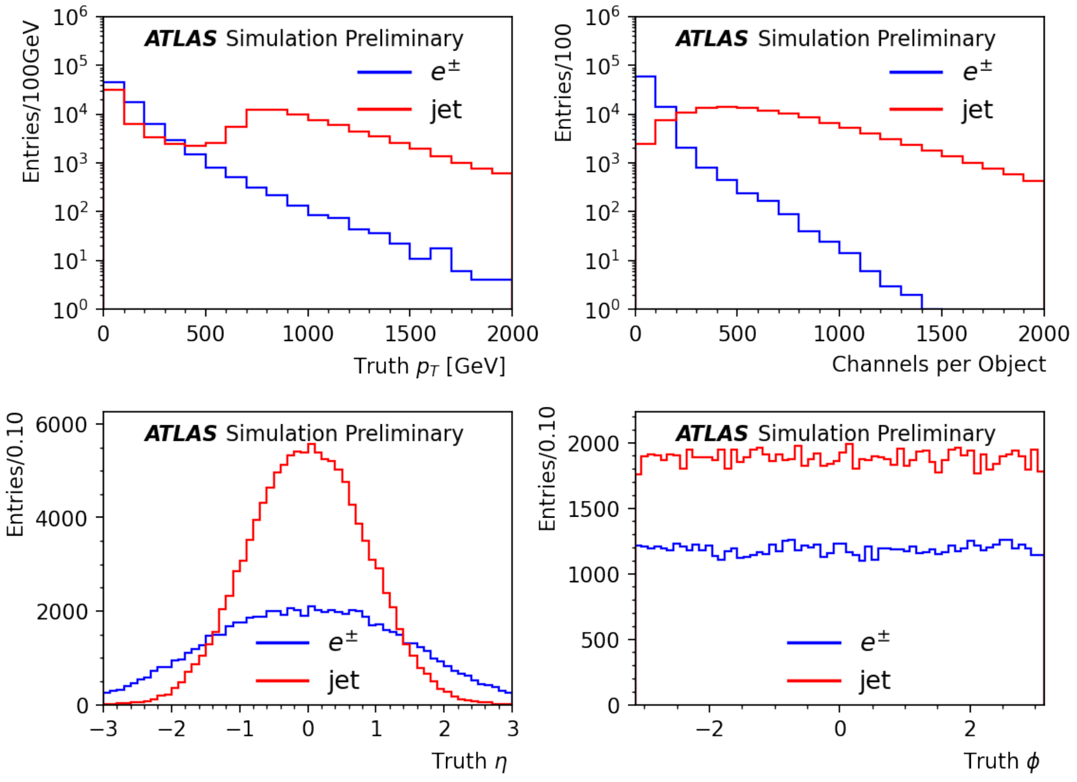


Figure 5: $Z \rightarrow e^{+}e^{-}$ + 2-jets: with jet $p_T > 700$ GeV and electron $p_T > 10$ GeV. Showing truth p_T (top-left), η (bottom-left), ϕ (bottom-right), and number of detector channels contained (top-right) for each object

3.2 Dealing with Bias in a Dataset

The dataset contains points labeled with three classes, background, jet, and electron, with 63%, 30%, and 7% of all points having these respective labels. Due to the imbalanced representation of classes, a machine learning model trained with unweighted cross-entropy loss would learn to accurately predict background points at the expense of the other classes. To that end, a process of balanced learning is used similar to the method in Reference [40].

In an object classification problem, where one input is labeled with a single class, imbalanced datasets can be managed by sampling inputs of rarer classes more often such that on average the batched training data are balanced. In an object segmentation problem, where one input has a fixed distribution of classes, the balance of classes in a single input cannot be changed without changing the sample itself.

To address this the loss is not calculated on every point in the event. Instead the loss is only calculated on equal numbers of points from each class. For example, given N_e electron points, N_j jet points, and N_{bg} background points in a single event where $N_e < N_j < N_{bg}$, all points in the event will be processed through the network to get a prediction on every point. The loss will randomly mask out the point-wise loss of some of the jet and background points such that the number of points considered in the loss are $3N_e$ (N_e electron points, N_e jet points, and N_e background points). Figure 8 illustrates this method. The left figure represents the entire ground truth, and the right represents one possible random subsampling such that each class is equally represented.

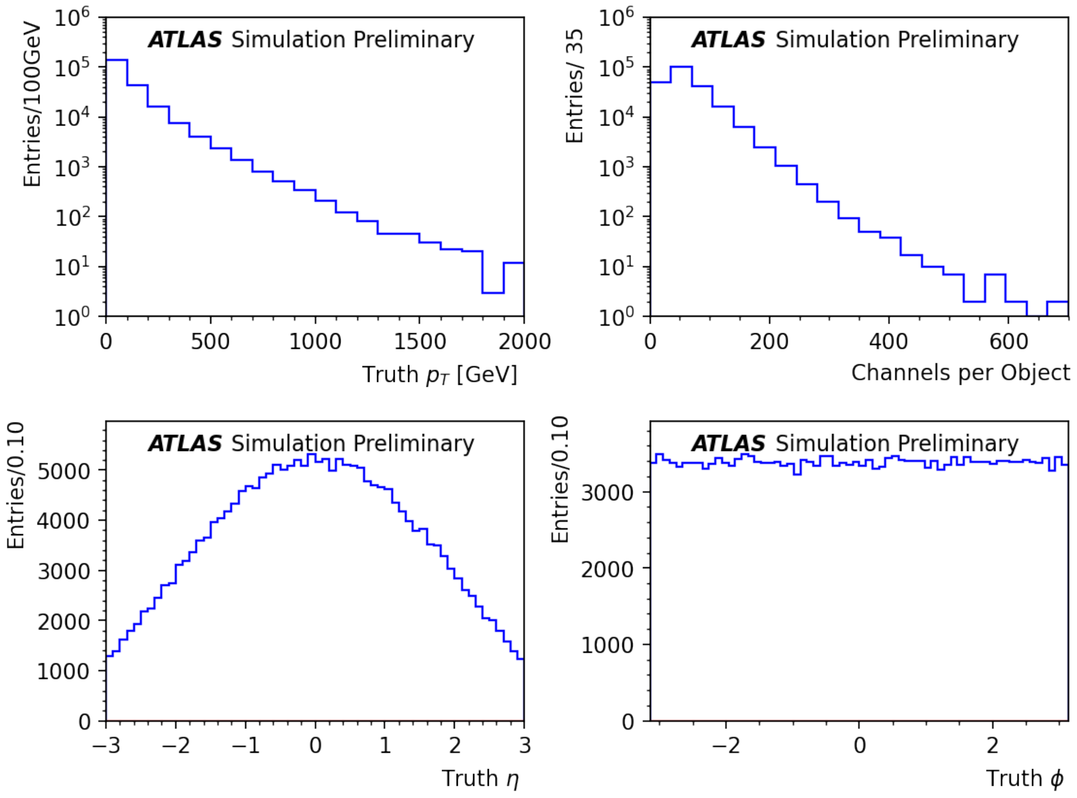


Figure 6: $ZZ \rightarrow e^+e^-e^+e^-$ with electron $p_T > 10$ GeV. Showing truth p_T (top-left), η (bottom-left), ϕ (bottom-right), and number of detector channels contained (top-right) for each object

3.3 Training Parameters

Each network is trained on 95,000 events with a batch size of 16 for 10 epochs with the Adam optimizer with an initial learning rate of 0.001 and an exponential decay parameter of 0.7 applied every 200, 000 iterations. The input features are the (x, y, z) coordinates of each detector readout channel, the (r, η, ϕ) representation of the same coordinates, and the readout value. The output is one of three classes: jet, electron, or background. Since the events all have different numbers of non-zero readout channels, the input vector is zero padded to 15, 000 during training which is above the maximum number for any single event in the training set. The zero pad is set to 25, 000 in the evaluation as there is an event of this magnitude in the test set.

4 Results

The networks are evaluated on object segmentation, a point-wise classification task, measured with 24, 000 events that are independent of the training events. The standard metric for evaluating these tasks is

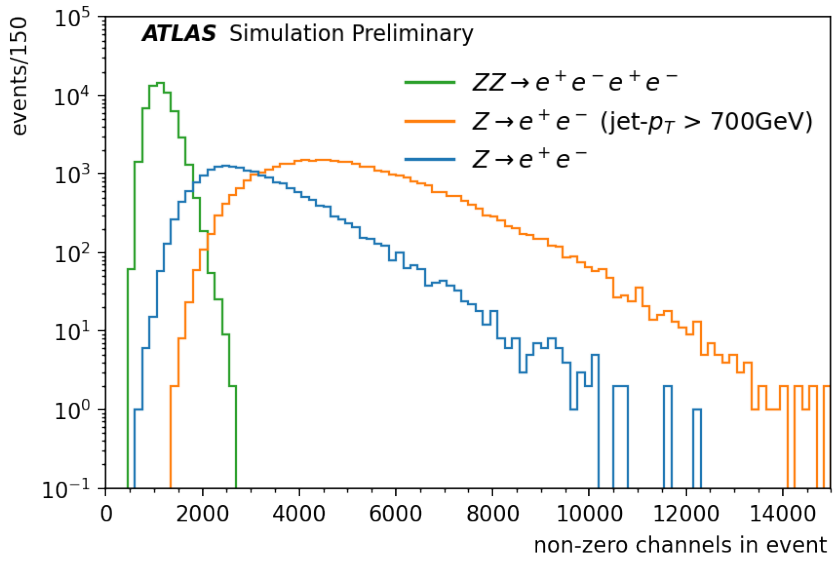


Figure 7: Distribution of non-zero channels included in each event image in the three datasets. The last bin is overflow.

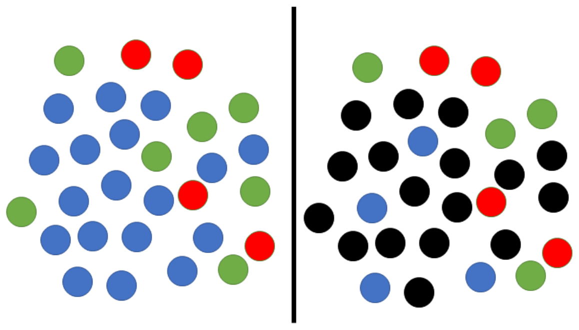


Figure 8: This diagrams the balancing of the loss calculation during the training process. The left figure represents the entire ground truth with class labels (jet, electrons, background) represented as colors. The right represents one possible random subsampling such that each class is equally represented with black representing points not included for the total loss calculation. This removes the effect of bias from the unbalanced training dataset.

Intersection over Union (IoU). For a single class prediction, the IoU is defined as

$$IoU_c = \frac{\sum_{n=0}^{N} \mathbf{1}[y_n = c \;\&\; \hat{y}_n = c]}{\sum_{n=0}^{N} \mathbf{1}[y_n = c \text{ or } \hat{y}_n = c]}, \tag{1}$$

where **1** is the indicator function, which means that it has a value of one if the condition in brackets is met and zero otherwise. y_n is the truth level label and $\hat{y}_n$ is the label predicted by the model, both of which take on class identification values, $c = 0, 1, 2$ (these are numerical representations of electron, jet, or background). The sum is over n points. The mean IoU (mIoU) is the average of all the individual class IoUs. Maximizing this score requires both precision and recall to be high. The IoU results for the networks evaluated are in Table 1.

	mIoU	Jet IoU	Electron IoU	Background IoU
PointNet++	0.776 ± 0.009	0.842 ± 0.007	0.61 ± 0.01	0.882 ± 0.007
GravNet	0.60 ± 0.02	0.74 ± 0.01	0.32 ± 0.02	0.75 ± 0.02
GarNet	0.43 ± 0.02	0.45 ± 0.04	0.13 ± 0.02	0.70 ± 0.02
DGCNN	**0.826 ± 0.005**	**0.885 ± 0.002**	**0.69 ± 0.0.01**	**0.904 ± 0.002**

Table 1: The Intersection over Union (IoU), a standard metric for semantic segmentation learning tasks, calculated for each object class separately (jet, electron, and background) and the mean IoU (mIoU) of all classes. Each row contains the metrics for each test model: PointNet++, GravNet, GarNet, and DGCNN. This table shows the mean and standard deviations of the IoUs over 3 identical runs.

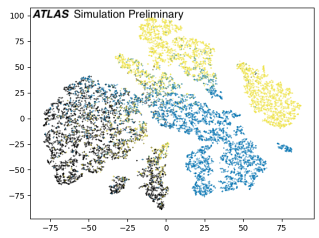
DGCNN outperforms all the other networks under evaluation, achieving a 6.4% higher mIoU than PointNet++ in these trials. GravNet and GarNet performance was unexpectedly lower.

The T-SNE plots [41] in Figure 9 translate the multidimensional feature vector to an arbitrary two dimensional space. The feature vector is the input vector to the last layer of the neural network that outputs the point label prediction. The distance between points is proportional to the distance in the higher dimensional space in order to give a visible representation of separation strength between classes. It shows the DGCNN and PointNet++ networks separate the three classes (jet, electron, background) well in feature space. GravNet and GarNet out of the box seem to be able to tell apart jets and electrons from one another. However, they frequently confuse the background class with non-background classes, and vice versa. This suggests that GravNet and GarNet are not able to precisely predict the shape of jets and electrons.

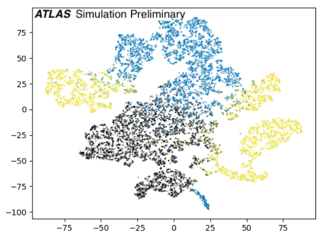
One difference between GravNet/GarNet and PointNet/DGCNN is that the latter use ReLU activation functions instead of tanh used in the former. ReLU is a common machine learning activation function that simply evaluates $f(x) = \max(0, x)$ per x input value, and has been found to lead to faster training than $f(x) = \tanh(x)$ [23]. One issue with tanh is that it can saturate an output for both large positive and negative values, which can reduce learning over time. ReLU only saturates for negative values. Table 2 shows the previous results for GravNet/GarNet as well as results using ReLU for activation functions (labeled with +ReLU). GravNet+ReLU achieves state-of-the-art mIoU over DGCNN by a margin of 2.2%, improving by 39.7% over the original version. GarNet sees an 7.8% improvement in mIoU.

The structure of GarNet's convolution, in which a hidden point-cloud is generated and associated with the real point-cloud (referred to as a bipartite graph), may be too much indirection for the network to learn effectively. Removing the bipartite graph structure and simply applying a PointNet++-style multilayer-perceptron per point improves mIoU performance by 25.5% over GarNet + ReLU. This is seen in Table 2 labeled as '-Bipartite'. Not modeling relationships between points at all is more effective than modeling them using the proposed bipartite graph structure.

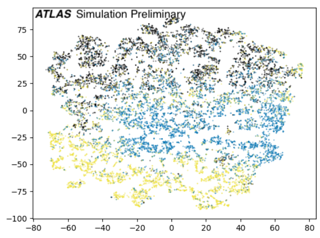
The last difference between PointNet++/DGCNN and GravNet/GarNet is that GravNet/GarNet do not have global feature vectors (GFVs) concatenated with local features before the final classification layer. PointNet++ pools its point clouds multiple times down to a single point feature vector which represents global information before 'unpooling' back to the original point cloud. DGCNN, after all of its convolutions, performs maximum pooling on the point cloud down to a single feature vector, and concatenates this global feature vector to each of the individual point feature vectors. Conceptually, having global features in addition to local features when training the classifier at the end gives it context beyond the local neighborhoods. Without it, there is a risk that the classifier attempts to learn how to predict for an 'average' sample with the given local features. To test this, a global feature vector was created from the last feature vector before the fully-connected layers and concatenated with each point-wise feature vector. Table 2



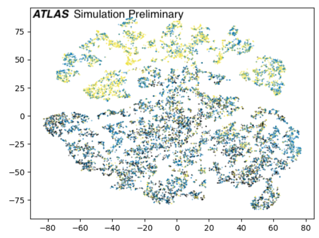
(a) PointNet++



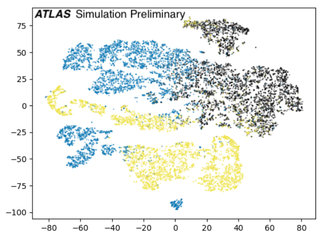
(b) DGCNN



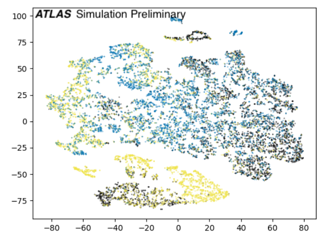
(c) GravNet



(d) GarNet



(e) GravNet+ReLU + GFV



(f) GarNet+ReLU-Bipartite+GFV

Figure 9: The T-SNE algorithm translates the multidimensional feature vector (the input to the last network layer) to an arbitrary two dimensional space. The feature vector is the last input used by the networks (a) PointNet++, (b) DGCNN, (c) GravNet, (d) GarNet, (e) GravNet+ReLU, and (f) GarNet+GFV to classify a single detector input as belonging to jet (black), electron (yellow), or background (blue). The distance between points is proportional of the distance in the higher dimensional space in order to give a visible representation of separation strength between classes. Good performance of the network is reflected by clear delineation of class points. The x- and y-axes are arbitrary units with no translatable label.

	mIoU	Jet IoU	Electron IoU	Background IoU
GravNet	0.60 ± 0.02	0.74 ± 0.01	0.32 ± 0.02	0.75 ± 0.02
+ ReLU	0.84 ± 0.02	0.86 ± 0.03	0.74 ± 0.02	0.933 ± 0.003
+ GFV	**0.866 ± 0.002**	**0.878 ± 0.006**	**0.782 ± 0.004**	**0.938 ± 0.001**
GarNet	0.43 ± 0.02	0.45 ± 0.04	0.13 ± 0.02	0.70 ± 0.02
+ ReLU	0.46 ± 0.1	0.47 ± 0.05	0.19 ± 0.02	0.72 ± 0.02
- Bipartite	0.575 ± 0.001	0.654 ± 0.003	0.305 ± 0.001	0.765 ± 0.002
+ GFV	0.577 ± 0.004	0.670 ± 0.006	0.305 ± 0.004	0.756 ± 0.004

Table 2: The Intersection over Union (IoU), a standard metric for semantic segmentation learning tasks, calculated for each object class separately (jet, electron, and background) and the mean IoU (mIoU) of all classes. Each row contains the metrics for the original and improved GarNet and GravNet models (see text for details). Training was run for 10 epochs in each case. This table shows the mean and standard deviation of each IoU over 3 identical runs.

shows the results of this addition (labeled '+GFV'). GravNet mIoU performance improves by 2.6%, while GarNet sees another 0.35% increase in mIoU performance, or 44% overall, over the original version. Figure 9 shows the T-SNE class distribution for the best performing modified GravNet and GarNet models.

T-SNE and mIoU metrics are useful tools for generally characterizing segmentation networks. However, it must be determined whether the network is finding all of the unique objects in the data. This is not immediately evident from the predictions, which are per-point instead of per-object. The ground truth in this dataset labels not only the per point class but to which object instance the point is associated, e.g. 1st electron in the event, 2nd jet in the event. This could be used to train an instance-segmentation network to identify collections of points belonging to individual physics objects in future work.

5 Discussion

This dataset offers several research opportunities. Point-cloud segmentation algorithms can be developed tuned to this physics localization domain. The same ground truth can be used to develop detection or instance segmentation algorithms, which can more directly predict the number of objects in a sample.

While DGCNN is a state-of-the-art network in computer vision tasks, and performed the best out of the baseline networks evaluated, it still does have trouble separating the classes in many cases, as the T-SNE plot shows. It is also a relatively simple network, with only four convolutional layers with 64 filters each. This means that the convolutions have limited receptive field, and only a limited amount of capacity to capture all of the different cases represented in the dataset. GravNet with modifications outperforms DGCNN, but its based on the same principles of sharing a single set of weights. This is a strong regularizer that prevents overfitting, but it may keep the network from modeling all of the complexity of the physics data. Here is a recent example of the potential for using GravNet for instance segmentation [42].

6 Summary

In this work physics object identification and localization is posed as a point-cloud segmentation problem. ATLAS detector data can be posed as a point cloud where each activated detector channel is a point that is associated with a jet, electron, or neither. This method does not quantize the data as voxel/image CNN methods do, and does not require a canonical ordering as recurrent/LSTM methods do. Some baseline point-cloud and graph network methods were evaluated to set an initial watermark for research on this dataset.

Using a point-cloud segmentation network for physics object localization is not a turnkey solution. The network makes predictions per point, but does not actually predict discrete physics objects. That said, this network could still be valuable in a physics localization pipeline by narrowing the search space to only locations predicted as likely locations for a jet or electron by the network. A natural extension for future work is to add a module to the network for instance segmentation. Previous studies have shown that this is feasible for point clouds in traditional computer vision problems [43–45]. These approaches rely on PointNets as a learning machine and could benefit from the extra learning capacity of graph neural network. They may also require modification to address the large bias toward the background class that full-detector data often contains.

Point-cloud neural networks are a potential tool to accelerate physics object localization. This study shows that point-cloud segmentation networks are able to recognize which detector channels in the full detector are associated with a jet or electron with high accuracy, even though the dataset is highly biased. It is able to perform this task quickly, with a single forward pass. Future refinements of these networks will look at increasing accuracy, clustering classified points into single objects, and using regression networks to estimate four momenta. This could lead to improved hardware level triggers and reconstruction algorithms.

Follow up analyses will also include effects due to pile-up interactions, the TRT and Muon detector information, and muon data.

References

[1] ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider*, JINST **3** (2008) S08003 (cit. on pp. 2, 3, 5, 6).

[2] L. Evans and P. Bryant, *LHC Machine*, JINST **3** (2008) S08001 (cit. on pp. 2, 3).

[3] ATLAS Collaboration, *Electron and photon performance measurements with the ATLAS detector using the 2015–2017 LHC proton–proton collision data*, JINST **14** (2019) P12006, arXiv: `1908.00005 [hep-ex]` (cit. on p. 2).

[4] ATLAS Collaboration, *Topological cell clustering in the ATLAS calorimeters and its performance in LHC Run 1*, Eur. Phys. J. C **77** (2017) 490, arXiv: `1603.02934 [hep-ex]` (cit. on p. 2).

[5] ATLAS Collaboration, *Muon reconstruction efficiency and momentum resolution of the ATLAS experiment in proton–proton collisions at $\sqrt{s} = 7$ TeV in 2010*, Eur. Phys. J. C **74** (2014) 3034, arXiv: `1404.4562 [hep-ex]` (cit. on p. 2).

[6] L. Yi et al., *A Scalable Active Framework for Region Annotation in 3D Shape Collections*, ACM Trans. Graph. **35** (2016) 210:1, ISSN: 0730-0301 (cit. on p. 2).

[7] I. Armeni, A. Sax, A. R. Zamir and S. Savarese, *Joint 2D-3D-Semantic Data for Indoor Scene Understanding*, ArXiv e-prints (2017), arXiv: `1702.01105 [cs.CV]` (cit. on p. 2).

[8] A. Dai et al., 'ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes', *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017 2432, URL: `https://doi.org/10.1109/CVPR.2017.261` (cit. on p. 2).

[9] R. Q. Charles, H. Su, M. Kaichun and L. J. Guibas, 'PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation', *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017 77, URL: `https://ieeexplore.ieee.org/document/8099499` (cit. on p. 2).

[10] C. R. Qi, L. Yi, H. Su and L. J. Guibas, 'PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space', *NIPS 30*, ed. by I. Guyon et al., Curran Associates, Inc., 2017 5099, URL: `http://papers.nips.cc/paper/7095-pointnet-deep-hierarchical-feature-learning-on-point-sets-in-a-metric-space.pdf` (cit. on pp. 2, 4).

[11] Y. Li et al., 'PointCNN: Convolution On X-Transformed Points', *NIPS 31*, ed. by S. Bengio et al., Curran Associates, Inc., 2018 826, URL: `http://papers.nips.cc/paper/7362-pointcnn-convolution-on-x-transformed-points.pdf` (cit. on p. 2).

[12] Y. Xu, T. Fan, M. Xu, L. Zeng and Y. Qiao,
'SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters',
Computer Vision – ECCV 2018, ed. by V. Ferrari, M. Hebert, C. Sminchisescu and Y. Weiss,
Cham: Springer International Publishing, 2018 90,
URL: https://link.springer.com/chapter/10.1007/978-3-030-01237-3_6
(cit. on p. 2).

[13] Y. Wang et al., *Dynamic Graph CNN for Learning on Point Clouds*, ACM Trans. Graph. **38** (2019),
ISSN: 0730-0301 (cit. on pp. 2, 4).

[14] L. Pan, P. Wang and C. Chew, *PointAtrousNet: Point Atrous Convolution for Point Cloud Analysis*,
IEEE Robotics and Automation Letters **4** (2019) 4035, ISSN: 2377-3774 (cit. on p. 2).

[15] S. Lan, R. Yu, G. Yu and L. S. Davis,
'Modeling Local Geometric Structure of 3D Point Clouds Using Geo-CNN',
2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019 998,
URL: https://doi.org/10.1109/CVPR.2019.00109 (cit. on p. 2).

[16] Y. Rao, J. Lu and J. Zhou,
'Spherical Fractal Convolutional Neural Networks for Point Cloud Recognition',
2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019 452,
URL: https://doi.org/10.1109/CVPR.2019.00054 (cit. on p. 2).

[17] J. Mao, X. Wang and H. Li,
'Interpolated Convolutional Networks for 3D Point Cloud Understanding',
2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019 1578,
URL: https://doi.org/10.1109/ICCV.2019.00166 (cit. on p. 2).

[18] Z. Zhang, B. Hua and S. Yeung,
'ShellNet: Efficient Point Cloud Convolutional Neural Networks Using Concentric Shells Statistics',
2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019 1607,
URL: https://doi.org/10.1109/ICCV.2019.00169 (cit. on p. 2).

[19] S. Prokudin, C. Lassner and J. Romero, 'Efficient Learning on Point Clouds With Basis Point Sets',
2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019 4331,
URL: https://doi.org/10.1109/ICCV.2019.00443 (cit. on p. 2).

[20] Y. Liu et al.,
'DensePoint: Learning Densely Contextual Representation for Efficient Point Cloud Processing',
2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019 5238,
URL: https://doi.org/10.1109/ICCV.2019.00534 (cit. on p. 2).

[21] L. Jiang et al.,
'Hierarchical Point-Edge Interaction Network for Point Cloud Semantic Segmentation',
2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019 10432,
URL: https://doi.org/10.1109/ICCV.2019.01053 (cit. on p. 2).

[22] A. Komarichev, Z. Zhong and J. Hua,
'A-CNN: Annularly Convolutional Neural Networks on Point Clouds',
2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019 7413,
URL: https://doi.org/10.1109/CVPR.2019.00760 (cit. on p. 2).

[23] A. Krizhevsky, I. Sutskever and G. E. Hinton,
'ImageNet Classification with Deep Convolutional Neural Networks',
Advances in Neural Information Processing Systems 25,
ed. by F. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger, Curran Associates, Inc., 2012
1097, URL: `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf` (cit. on pp. 2, 11).

[24] K. Simonyan and A. Zisserman,
'Very Deep Convolutional Networks for Large-Scale Image Recognition',
International Conference on Learning Representations, 2015, arXiv: `1409.1556 [cs.CV]`
(cit. on p. 2).

[25] K. He, X. Zhang, S. Ren and J. Sun, 'Deep Residual Learning for Image Recognition',
2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016 770,
URL: `https://ieeexplore.ieee.org/document/7780459` (cit. on p. 2).

[26] L. de Oliveira, M. Kagan, L. Mackey, B. Nachman and A. Schwartzman,
Jet-images — deep learning edition, Journal of High Energy Physics **2016** (2016) 69,
ISSN: 1029-8479 (cit. on p. 2).

[27] P. Baldi, K. Bauer, C. Eng, P. Sadowski and D. Whiteson,
Jet substructure classification in high-energy physics with deep neural networks,
Phys. Rev. D **93** (9 2016) 094034 (cit. on p. 2).

[28] S. R. Qasim, J. Kieseler, Y. Iiyama and M. Pierini, *Learning representations of irregular particle-detector geometry with distance-weighted graph networks*,
The European Physical Journal C **79** (2019) 608, ISSN: 1434-6052 (cit. on pp. 2, 4).

[29] ATLAS Collaboration, *ATLAS Insertable B-Layer Technical Design Report*,
ATLAS-TDR-19; CERN-LHCC-2010-013, 2010,
URL: `https://cds.cern.ch/record/1291633` (cit. on p. 3),
Addendum: ATLAS-TDR-19-ADD-1; CERN-LHCC-2012-009, 2012, URL:
`https://cds.cern.ch/record/1451888`.

[30] B. Abbott et al., *Production and integration of the ATLAS Insertable B-Layer*,
JINST **13** (2018) T05008, arXiv: `1803.00844 [physics.ins-det]` (cit. on p. 3).

[31] ATLAS Collaboration, *Performance of the ATLAS muon triggers in Run 2*,
JINST **15** (2020) P09015, arXiv: `2004.13447 [hep-ex]` (cit. on p. 4).

[32] J. Alwall et al., *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*, JHEP **07** (2014) 079,
arXiv: `1405.0301 [hep-ph]` (cit. on p. 5).

[33] T. Sjöstrand et al., *An introduction to PYTHIA 8.2*, Comput. Phys. Commun. **191** (2015) 159,
arXiv: `1410.3012 [hep-ph]` (cit. on p. 5).

[34] R. D. Ball et al., *Parton distributions with LHC data*, Nuclear Physics B **867** (2013) 244,
ISSN: 0550-3213 (cit. on p. 5).

[35] GEANT4 Collaboration, S. Agostinelli et al., *Geant4 – a simulation toolkit*,
Nucl. Instrum. Meth. A **506** (2003) 250 (cit. on p. 5).

[36] ATLAS Collaboration, *The ATLAS Simulation Infrastructure*, Eur. Phys. J. C **70** (2010) 823,
arXiv: `1005.4568 [physics.ins-det]` (cit. on p. 5).

[37] ATLAS Collaboration, *Readiness of the ATLAS Tile Calorimeter for LHC collisions*, Eur. Phys. J. C **70** (2010) 1193, arXiv: 1007.5423 [hep-ex] (cit. on p. 6).

[38] ATLAS Collaboration, *Readiness of the ATLAS liquid argon calorimeter for LHC collisions*, Eur. Phys. J. C **70** (2010) 723, arXiv: 0912.2642 [hep-ex] (cit. on p. 6).

[39] ATLAS Collaboration, *ATLAS b-jet identification performance and efficiency measurement with $t\bar{t}$ events in pp collisions at $\sqrt{s} = 13$ TeV*, Eur. Phys. J. C **79** (2019) 970, arXiv: 1907.05120 [hep-ex] (cit. on p. 6).

[40] H. Small and J. Ventura, *Handling Unbalanced Data in Deep Image Segmentation*, University of Colorado (2017), URL: http://cs.uccs.edu/~jkalita/work/reu/REU2017/16Small.pdf (cit. on p. 8).

[41] L. van der Maaten and G. Hinton, *Visualizing Data using t-SNE*, Journal of Machine Learning Research **9** (2008) 2579, URL: http://www.jmlr.org/papers/v9/vandermaaten08a.html (cit. on p. 11).

[42] J. Kieseler, *Object condensation: one-stage grid-free multi-object reconstruction in physics detectors, graph, and image data*, The European Physical Journal C **80** (2020) (cit. on p. 13).

[43] W. Wang, R. Yu, Q. Huang and U. Neumann, 'SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation', *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018 2569, URL: https://doi.org/10.1109/CVPR.2018.00272 (cit. on p. 13).

[44] L. Yi, W. Zhao, H. Wang, M. Sung and L. J. Guibas, 'GSPN: Generative Shape Proposal Network for 3D Instance Segmentation in Point Cloud', *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019 3942, URL: https://doi.org/10.1109/CVPR.2019.00407 (cit. on p. 13).

[45] B. Yang et al., 'Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds', *Advances in Neural Information Processing Systems 32*, ed. by H. Wallach et al., Curran Associates, Inc., 2019 6737, URL: http://papers.nips.cc/paper/8899-learning-object-bounding-boxes-for-3d-instance-segmentation-on-point-clouds.pdf (cit. on p. 13).

Appendices

A Data format for training

For each event a text file is created with each row representing one detector channel. For each channels these values are included: [`hwid`, `idx`, `x`, `y`, `z`, `r`, `eta`, `phi`, `raw`, `pid`, `n`, `truth_eta`, `truth_phi`, `truth_pt`, `trk_good`, `trk_barcode`, `trk_pt`]. They are defined as below:

- **hwid**: unique hardware ID based on a bit mask that can be used to select specific detector components (unsigned 64-bit integer). To select subdetector channels use these cuts: Pixel < 5e17 < SCT < 3e18 < LAr < 4.8e18 < Tile.
- **idx**: unique incremental index for each point that orders the detector channels into a one dimensional vector (unsigned 32-bit integer).
- **[x, y, z]** and **[r, eta, phi]**: central position of the active detector material corresponding to the readout channel in the ATLAS right-handed coordinate system (32-bit float).
- **raw**: a value that represents the detector channel readout, which corresponding to ToT (Time over Threshold) in case of the Pixel detector, the number of consecutive strips recording a hit in case of the SCT, and the digitized energy deposition in GeV (from CaloCell object) in case of the HAD/EM calorimeter (32-bit float).
- **pid**: truth label of the object, electrons are labeled as 11, positrons as -11, jets as 0, and background as -99 (32-bit integer).
- **n**: unique index of the truth object, e.g. if there are two electrons in the event, detector inputs will be assigned to the first (0) or second (1) electron (32-bit integer).
- **truth_eta**, **truth_phi**, **truth_pt**: truth object η, ϕ, and p_T (in GeV), taken from reconstructed jets and leptons in the hard process as described above (32-bit float).
- **trk_good**: good track flag, set to 1 if this detector input was associated with a reconstructed track and is a candidate of the truth object, otherwise is 0 if the reconstructed track was not associated to a truth object (32-bit float).
- **trk_barcode**: if point was included in a reconstructed track this is a unique index of the track and set to 0 otherwise (32-bit float). Therefore, all detector channels associated to the same reconstructed track will have the same barcode index.
- **trk_pt**: transverse momentum in MeV of the reconstructed track.

Note, the `trk_good`, `trk_barcode`, `trk_pt` columns are only included for points in Pixel and SCT detectors. Only non-zero points are included in the output for storage efficiency.

B Example Python Code to Read CSV File Format

The follow example code can be used to load the files into a `pandas DataFrame` object for processing in popular machine learning frameworks like Tensorflow or PyTorch.

```
import pandas as pd
import numpy as np
col_names    = ['hwid', 'idx',
                'x', 'y', 'z',
                'r', 'eta', 'phi',
                'raw', 'pid', 'n',
                'truth_eta', 'truth_phi','truth_pt',
                'trk_good', 'trk_barcode', 'trk_pt']
col_dtype    = {'hwid': np.int64, 'idx': np.int32,
                'x': np.float32,'y': np.float32, 'z': np.float32,
                'r': np.float32, 'eta': np.float32,
                'phi': np.float32,
                'raw': np.float32, 'pid': np.int32, 'n': np.int32,
                'truth_eta': np.float32, 'truth_phi': np.float32,
                'truth_pt': np.float32,
                'trk_good': np.float32, 'trk_barcode': np.float32,
                'trk_pt': np.float32}
filename = '/path/to/filename.csv'
data = pd.read_csv(filename,
                   header=None,
                   names=col_names,
                   dtype=col_dtype,
                   sep='\t')
```