Implementation and performance of the ATLAS pixel clustering neural networks

Louis-Guillaume Gagnon[1,*], on behalf of the ATLAS Collaboration

[1]Group of Particle Physics, University of Montreal

Abstract. The high particle densities produced at the Large Hadron Collider (LHC) mean that in the ATLAS pixel detector the clusters of deposited charge start to merge. A neural network-based approach is used to estimate the number of particles contributing to each cluster, and to accurately estimate the hit positions even in the presence of multiple particles. The algorithm and its implementation are thoroughly described and a set of benchmark measurements is presented. The problem is most acute in the core of high-momentum jets where the average separation between particles becomes comparable to the detector granularity. This is further complicated by the high number of interactions per bunch crossing. Both these issues will become more acute as the Run 3 and HL-LHC program require analysis of higher and higher p_{T} jets, while the interaction multiplicity rises. Future prospects in the context of LHC Run 3 and the upcoming ATLAS inner detector upgrade are also discussed.

1 Introduction

At the center of the ATLAS detector [1] is installed a high-performance tracker comprised of three subdetectors. From the outside-in, these different components are a gaseous straw-tube detector (TRT), a silicon microstrip detector (SCT) and a silicon pixel detector [2]. With the installation of the Insertable B-Layer (IBL) as the innermost layer in 2014 [3], the pixel subsystem generally provides four high-precision measurements per-track. The nominal pixel pitches are of 50 μm (400 μm) in the transverse (longitudinal) direction, while pixels in IBL have a pitch of 250 μm in the longitudinal direction. The intrinsic accuracies, estimated in Monte Carlo simulations, are of 8.5 μm (47 μm) in the transverse (longitudinal) directions in the IBL and 9.4 μm (89 μm) in the second layer [4]. The combination of the closeness of the pixel measurements from the interaction point and their precision makes the performance of the pixel detector absolutely crucial for optimal tracking and vertexing performance.

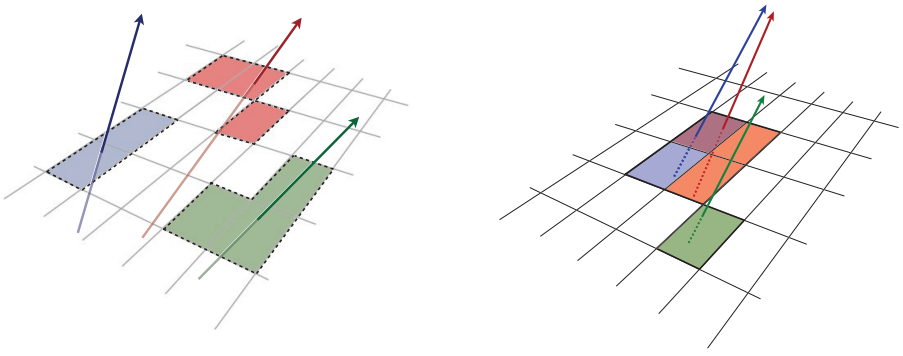
To perform track reconstruction in ATLAS [5, 6], first a set of track seeds is defined by finding collections of three hits in three-dimensional space ("space-points") satisfying transverse momentum and impact parameter requirements. Then, a combinatorial Kalman filter [7] is used to update these track seeds by iteratively incorporating hits from other layers.

*e-mail: louis.guillaume.gagnon@cern.ch

While building a track, if at any step multiple space-points are compatible with the filter's estimate, multiple track candidates are formed in order to obtain a high track finding efficiency. However, this also means that the number of excess track candidates [1] will typically be high and that an ambiguity solving stage stage must be implemented to reduce the reconstruction ambiguities to an acceptable level. This is done by scoring the tracks according to metrics related to the track quality, such as the number of disconnected track segments ("holes"), the χ^2 of the fit and $\log(p_{\mathrm{T}})$.

Ionizing particles will typically deposit charge in more than one pixel because of charge diffusion in the silicon bulk, charge drift in direction of the Lorentz angle caused by the axial magnetic field, and creation of δ-rays (Figure 1a), for example. Nevertheless, precise estimation of hit positions within clusters can be obtained using charge interpolation techniques parameterized on the incidence angle of the track candidate and the number of pixel rows and columns [2] contained in the cluster. However, the high centre-of-mass energy of the LHC [8] leads to a significant cross section for very boosted objects such as high-p_{T} jets, in which the separation between constituent particles becomes comparable to the pixel detector granularity. In such dense environments, the charge clusters can merge (Figure 1b) which degrades the hit position estimation precision and the overall track reconstruction performance.

To recover near-optimal performance in dense environments, three sets of neural networks, collectively referred to as the "pixel clustering neural networks", are used to (1) estimate the particle multiplicity in a cluster, (2) measure the hit positions, and (3) measure the associated uncertainties. The method has originally been described in Ref. [9] and an updated description and performance measurement is presented in Ref. [10]. This algorithm is also discussed in the broader context of tracking in dense environments in Ref. [11].



(a) (b)

Figure 1: Illustration of (a)single-particle pixel clusters on a pixel sensor and (b)a merged pixel cluster due to very collimated charged particles [11]. Different colours represent energy deposits from different charged particles traversing the sensor and the particles trajectories are shown as arrows.

[1] Excess track candidates stems from multiply-used or incorrectly-assigned space-points.

[2] Rows of pixels are defined along the transverse direction while columns are longitudinal to the beam.

2 The pixel clustering neural networks

2.1 Neural network training

The dataset used for training is a Monte Carlo (MC) dijet sample generated by Pythia 8.186 [12] using the A14 set of tuned parameters [13] and the NNPDF2.3LO parton distribution function set [14]. A filter that keeps only events where the leading jet has transverse momentum between 1.8 and 2.5 TeV is used to ensure a high fraction of multiple-particle clusters. The selected events are then used as inputs to a detailed simulation of the ATLAS detector implemented using the GEANT4 software and which includes an accurate description of the charge clustering in the pixel detector [15–17].

The inputs for the neural networks used to estimate the particle multiplicity and the hit positions are:

- A 7×7 digitized charge matrix centered on the charge centroid of the cluster. As the neural networks do not accept matrices as inputs, the charge matrix is flattened into a length 49 vector in row-major order. Clusters are built by finding groups of neighboring pixels with non-zero ToT based on an 8-cell connectivity scheme [18].
- A length-7 vector of pixel pitches in the longitudinal direction.
- A binary variable encoding the inner detector region (endcap or barrel).
- An integer variable representing the cylinder (barrel) or disc (endcap) number.
- ϕ and θ angles of incidence of the track candidate being scored.

Since the pixel clustering neural networks have been shown to be robust against smearing of the angles of incidence [19], the true angles of incidence of the MC-generated particles are used as proxies for the track angles to avoid circular dependencies with the track reconstruction chain when creating the dataset. After training is completed, reconstructed track angles are always used.

The neural networks used to estimate the hit position uncertainties use the same inputs as the other networks, with the addition of the network-estimated positions.

The dataset is further split into training and test sets comprising 12 million and 5 million clusters respectively. For the neural network used to estimate the particle multiplicity, the parent dataset is sampled to obtain a composition of approximately 25% 1-particle, 25% 2-particle and 50% 3-particle clusters. For the other networks, the dataset is constructed to contain only clusters from the appropriate multiplicity class.

All networks are trained [3] with stochastic gradient descent, using a patience-base early stopping strategy in which the number of remaining training epochs, set at a minimum of 50, is increased by 1.75 for each epoch where the loss value computed on a statistically independent dataset decrease by at least 0.5%. The hyperparameters for the three network sets are summarized in Table 1. An optimization pass using a random search over hyperparameter combinations [21] has shown that these relatively small, shallow networks are a good trade-off between performance and evaluation speed.

2.2 Particle multiplicity estimation

A single neural network is trained to classify clusters as comprising charge deposited by 1, 2 or ≥ 3 particles. The output of the network is a length-3 vector representing the categorical distribution of the estimated particle multiplicity. The multiplicity-conditional joint distributions of P_2 and P_3 are shown in Figure 2. Pairwise receiver operating characteristic (ROC)

[3] For a introduction to machine learning with neural networks, see Ref. [20].

Table 1: Hyperparameters used to train the three sets of neural networks [10]. In the row "Structure", the numbers in parenthesis denote the input and output layer sizes, with numbers separated by slashes corresponding to different sizes in datasets with 1, 2 or 3 particles per cluster respectively, while the numbers in-between represent the hidden layer sizes.

Hyperparameter	Multiplicity network	Position networks	Uncertainty networks
Structure	(60)-25-20-(3)	(60)-40-20-(2/4/6)	(62/64/66)-15-10-(30/50/60)
Hidden activation	Sigmoid	Sigmoid	Sigmoid
Output activation	Sigmoid	Identity	Sigmoid
Learning rate	0.08	0.04	0.3
Weight decay	10^{-7}	10^{-7}	10^{-6}
Momentum	0.4	0.3	0.7
Minibatch size	60	30	50
Loss function	cross-entropy	mean squared error	cross-entropy

curves, probing the trade-off between efficiency and fake rate of the estimator, are shown in Figure 3. Since there are three target classes, there is no one-to-one correspondence between any two output probabilities, and so two different pairwise ROC curves have to be considered for each pair of classes, depending on the probability that is used to score the clusters [22].

To transform the output distribution into a point estimate of the multiplicity, the following decision rule based on the 2- and $\geq$ 3-particle bins of the output vector (P_2 and P_3) is employed:

- If $P_2 < 60\%$ and $P_3 < 20\%$: classify as 1-particle cluster
- If $P_2 > 60\%$ and $P_3 < 20\%$: classify as 2-particles cluster
- If $P_3 > 20\%$: classify as $\geq$ 3-particles cluster

The thresholds are chosen to obtain an optimal trade-off between efficiency of tracking inside jets and production of fake tracks arising from shared clusters.

2.3 Hit position and uncertainty estimation

Three networks are used to estimate the hit positions in clusters arising from 1, 2 or 3 particle respectively (clusters with more than 3 particles are treated as 3-particle clusters). These networks are trained to output one length-2 vector for each particle, corresponding to the transverse ("local x") and longitudinal ("local y") positions of the hit.

Residual distributions, where the residual is defined per-particle as the difference between the estimated and true positions, are shown separately in the local x and y directions for each particle multiplicity in Figure 4.

To perform the track fit, an estimation of the uncertainty associated to the hit positions is required. These are supplied by a set of six neural networks, one per multiplicity–direction pair. Since the uncertainties are *a priori* unknown, the task is setup as an unsupervised learning problem in which the supervised target which drives the learning process is a binned probability density over the residual of a given cluster (see Ref. [10] for a detailed description). To obtain a point estimate of the uncertainty, the root-mean-square of the distribution is used. An example of the uncertainty estimation task is shown in Figure 5.

Pull distributions, where the pull is defined per-particle as the residual divided by the estimated uncertainty, are shown separately in the local x and y directions for each particle

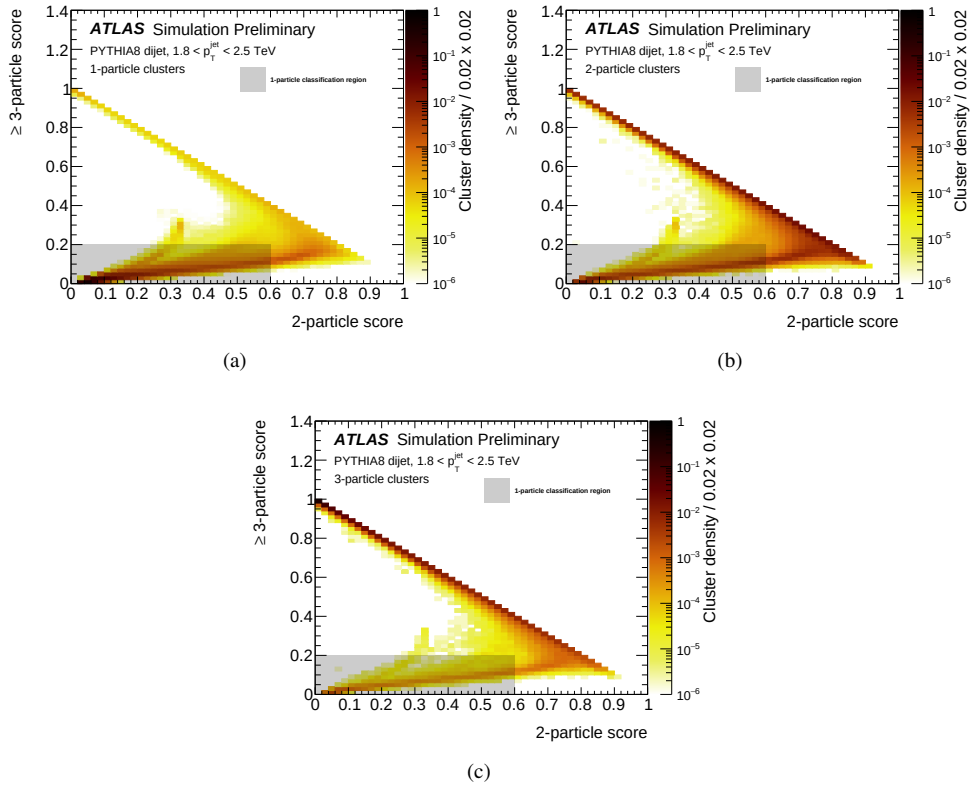


Figure 2: Joint distribution of the 2-particle and ≥ 3-particle neural network score, for (a) 1-particle, (b) 2-particle and (c) ≥ 3-particle clusters [10]. The shaded regions highlight the 1-particle cluster classification region.

multiplicity in Figure 6 after application of scaling factors accounting for detector calibration and alignment effects. The means and standard deviations of the pull distributions are obtained by Gaussian fits truncated to $\pm 3\sigma$. If the residual distribution is Gaussian and the error is properly estimated, the pull distribution should be consistent with a normal distribution with zero mean and unit variance. However, some of the pull distributions shown in Figure 6 deviate from such unit Gaussians; this can be caused by the non-Gaussian nature of the residual distributions, or by the error networks being unable to perfectly capture all the underlying residual distributions. The latter problem could be alleviated by using a different, more powerful regression model; see Section 3.

The η-dependence of the residual widths and the pull standard deviations can be seen in Figure 7.

3 Future directions

In 2026, the LHC will enter its high-luminosity phase of operation (HL-LHC) [23]. The steep increase in occupancy as well as the accumulated radiation damage will necessitate the complete replacement of the current inner detector [24]. The retained design is an all-silicon tracker with microstrips in the outermost layers and pixels in the innermost layers. The

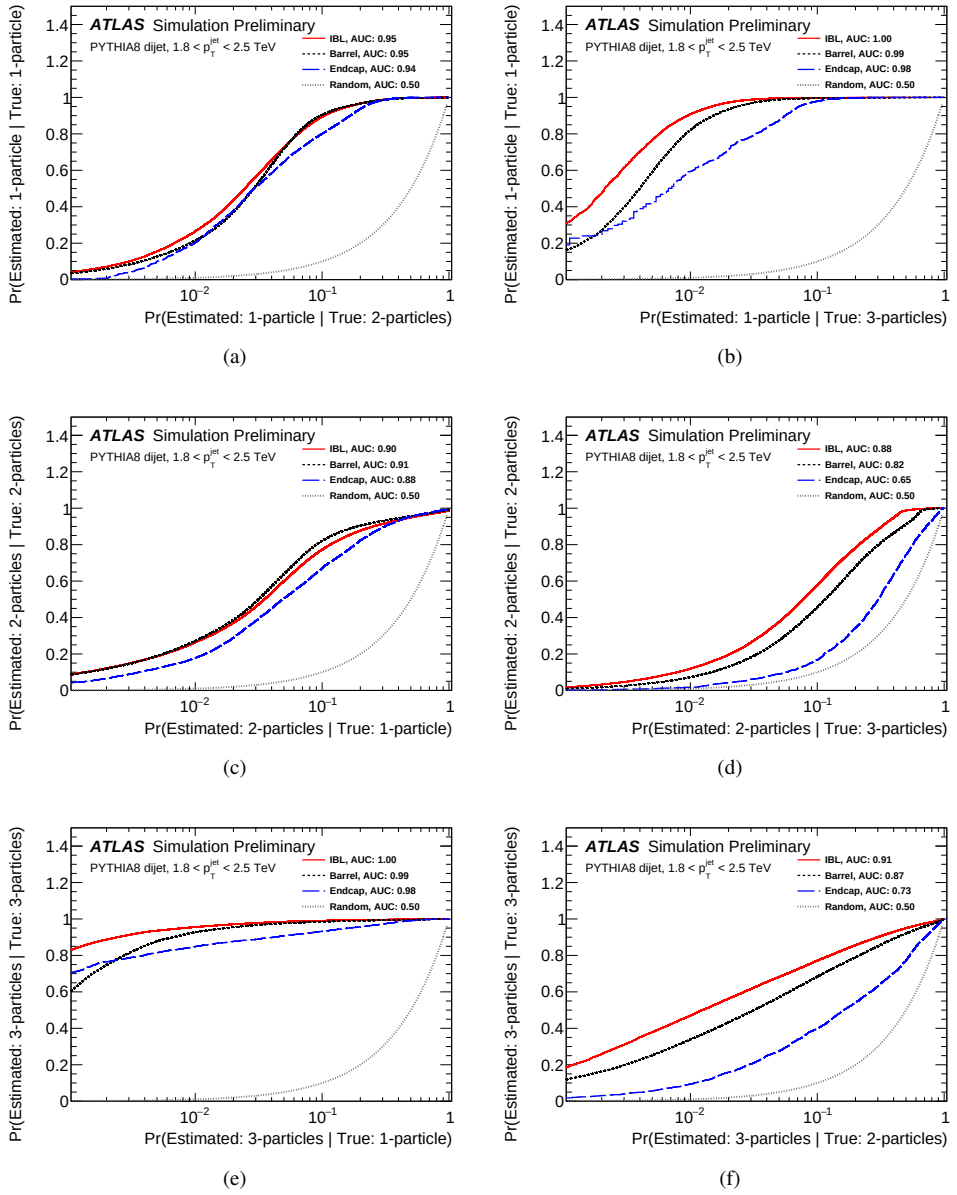


Figure 3: Pairwise ROC curves for the network used to estimate the particle multiplicity [10]. All figures follow an m vs n convention, where m- and n-particle clusters are scored according to the m-particle bin of the network output. The corresponding area-under-curve (AUC) measures the probability that an m-particle cluster is scored higher than an n-particle cluster in the m-particle bin. (a) 1-particle vs 2-particle clusters. (b) 1-particle vs $\geq$ 3-particle clusters. (c) 2-particle vs 1-particle clusters. (d) 2-particle vs $\geq$ 3-particle clusters. (e) $\geq$ 3-particle vs 1-particle clusters. (f) $\geq$ 3-particle vs 2-particle clusters. The ROC curves are produced separately for clusters in the IBL (solid line), barrel (medium-dashed line) and endcap (long-dashed line) regions. The small-dashed lines corresponds the a random classifier with variable bias (AUC = 0.5) and constitutes a universal baseline.

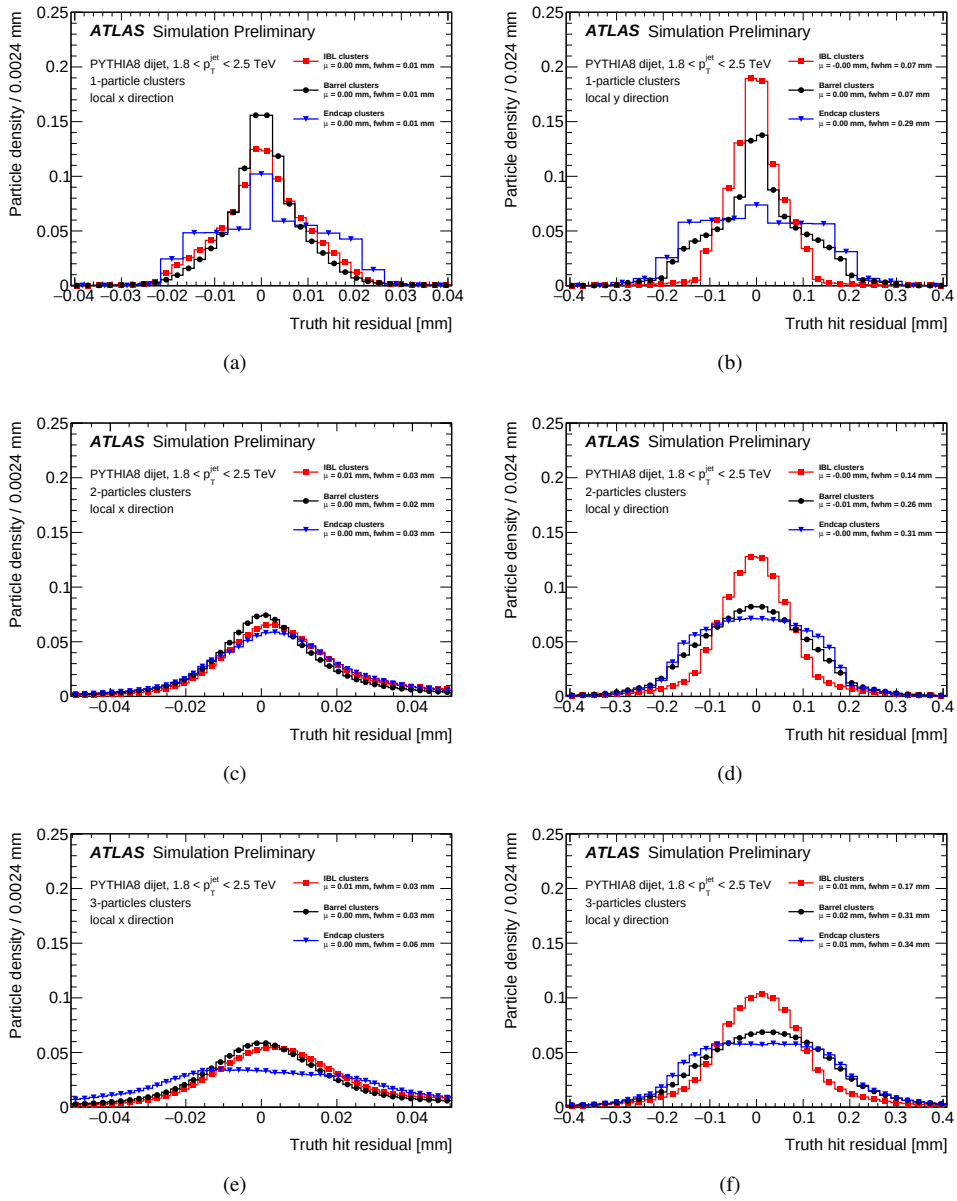


Figure 4: Difference between the neural network position estimation and the true hit position in the (left) local x and (right) local y directions for true (a), (b) 1-particle, (c), (d) 2-particles and (e), (f) 3-particles clusters [10]. These residual distributions are produced separately for clusters in the IBL (square), barrel (circle) and endcap (inverted triangle) regions. All sample means have negligible uncertainties while the full width at half maximum values have relative uncertainties of less than 5%.

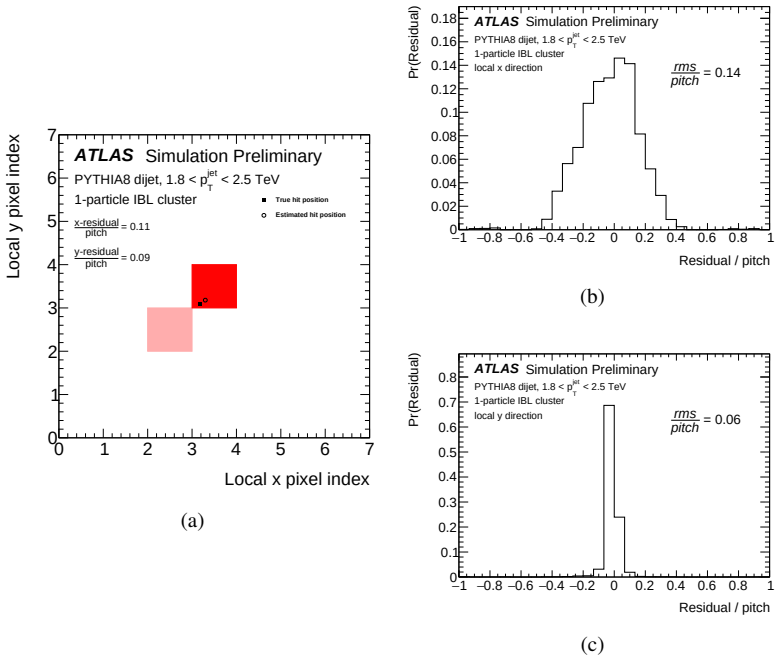


Figure 5: Example use case of the neural network used to estimate the uncertainty for a 1-particle IBL cluster [10]. (a)1-particle cluster with true hit position marked by the full square and hit position estimated by the neural network marked by the open circle. The cluster is fed to the two neural networks that estimate the probability distribution of this cluster's residual in the local x and y directions. These estimated probability distributions are shown in (b) and (c) respectively. The neural networks output node are directly mapped to bins of the residual distributions, and the *rms* of these distributions are used as point estimates of the uncertainties. In order to compare the performance in both directions, the residuals and *rms* values are divided by the pitches (50 μm and 250 μm in the local x and y directions respectively). As expected, the estimated rms/pitch is biggest for the direction where the residual/pitch is biggest.

pixel sensors will have different thickness and pitches than in the current detector, and the pixel clustering neural networks will correspondingly have to be adapted to the new design. The effects of the increase in number of interactions per bunch crossing will also have to be assessed and mitigated. This necessary phase of research and development represents an opportunity to explore more radical changes to the algorithm. One such possible algorithmic change would replace the sets of networks used to estimate the hit position and uncertainties by *mixture density networks* [25]. In this approach, a probability distribution over the true hit positions within a cluster is defined as a linear combination of Gaussians:

$$P(\mathbf{x}|\mathbf{C},\boldsymbol{\mu},\boldsymbol{\sigma}^2) = \sum_i C_i \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i,\boldsymbol{\sigma}_i^2)$$

Mixture density networks are trained to output the $\mathbf{C}$, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ parameters, directly using the mixture as the loss function. Estimates of the positions and uncertainties can be obtained

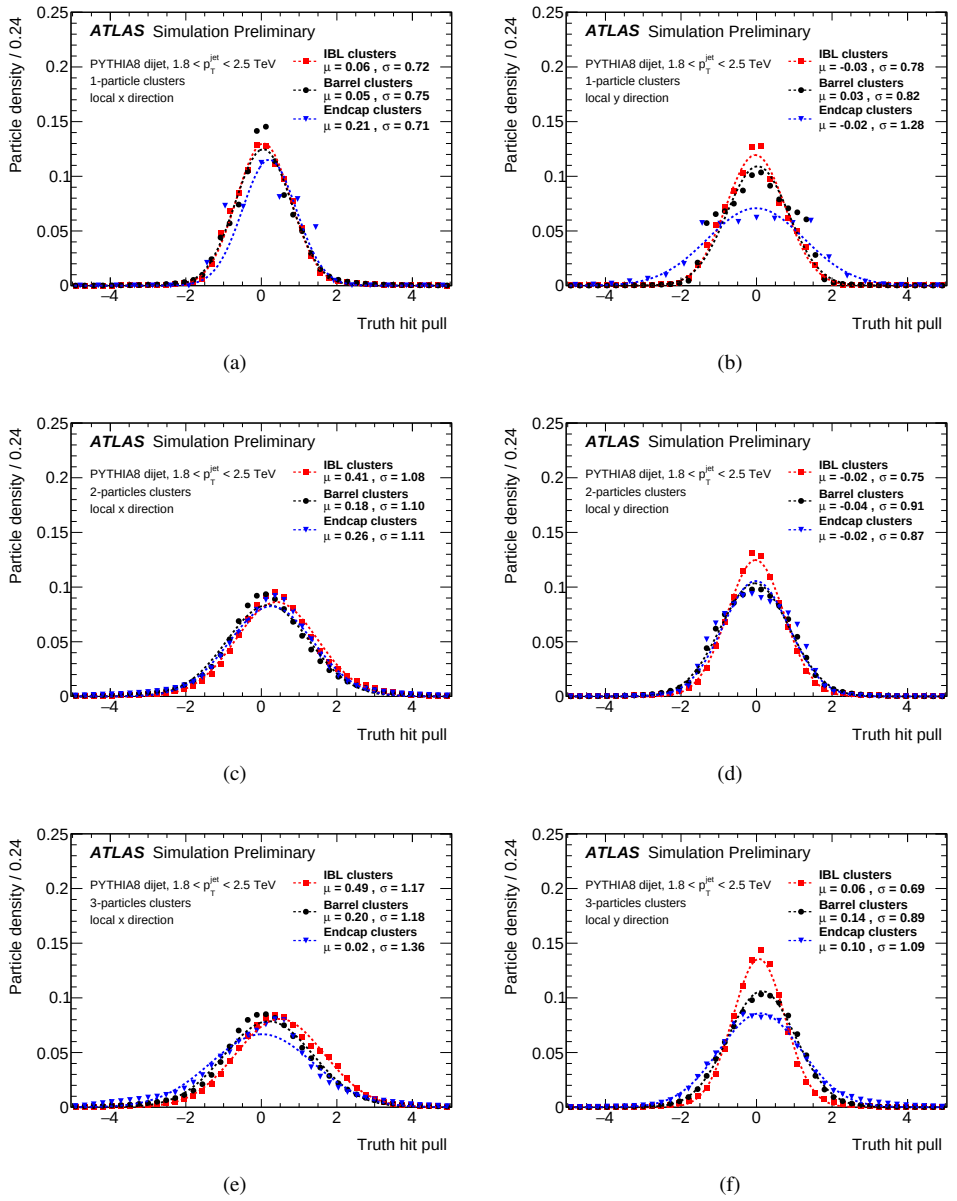


Figure 6: Difference between the neural network position estimation and the true hit position divided by the estimated uncertainty in the (left) local x and (right) local y directions for true (a), (b) 1-particle, (c), (d) 2-particles and (e), (f) 3-particles clusters [10]. These pull distributions are produced separately for clusters in the IBL (square), barrel (circle) and endcap (inverted triangle) regions. The means and standard deviations are estimated with truncated Gaussian fits and are represented as dashed lines. All means and standard deviations have negligible uncertainties.

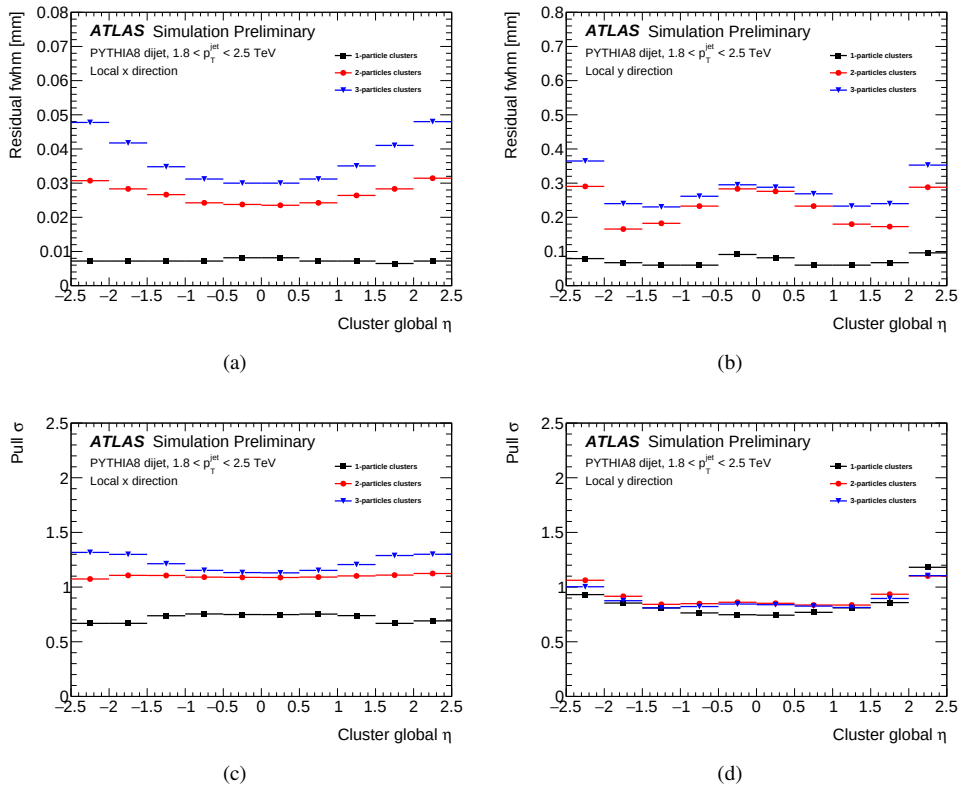


Figure 7: (a),(b) Residual widths and (c),(d)pull standard deviations in the (left) local x and (right) local y directions as a function of the cluster global η [10]. The standard deviations are obtained with truncated Gaussian fits.

by using the parameters of the component with highest prior probability C_i or by taking a weighted average of the parameters.

This model has at least three advantages over the current scheme. First of all, such linear combinations of Guaussians can approximate almost any continuous density [26], therefore this approach uses a well-motivated probability model instead of an ad-hoc loss function for training. Also, the error estimation task is made simpler since the network only has to estimate the variance instead of reconstructing the residual distributions from scratch for each cluster. Finally, such networks allow obtaining the positions and uncertainties for a given cluster in both directions using a single network instead of three which would allow to use bigger and deeper networks, which are more powerful, while keeping execution speed at acceptable levels.

4 Conclusion

A description and a set of performance measurements of the neural networks used to estimate the cluster particle multiplicity and hit positions in the pixel detector have been presented. This machine learning algorithm is an integral part of the ATLAS strategy for recovering

optimal tracking and vertexing performance in dense environments such as the core of high-p_{T} jets which have non-negligible cross-section at the LHC. In particular, the algorithm is able to efficiently separate clusters with different multiplicities of charged particles, as shown by the ROC curves with high area-under-curve in Figure 3. Moreover, the set of network used to estimate the hit positions allows the detector intrinsic accuracies to be nearly recovered as shown in the residual distributions of Figure 4. Possible algorithmic changes have been discussed in the context of the the upcoming phase II upgrade of the ATLAS detector, in which the current inner detector will be replaced by a new tracker with different pixel depth and pitches.

References

[1] ATLAS Collaboration, J. Instrum. **3**, S08003 (2008)
[2] G. Aad et al., J. Instrum. **3**, P07007 (2008)
[3] B. Abbott et al., J. Instrum. **13**, T05008 (2018), 1803.00844
[4] ATLAS Collaboration, *ATLAS Insertable B-Layer Technical Design Report*, ATLAS-TDR-19 (2010), https://cds.cern.ch/record/1291633
[5] ATLAS Collaboration, ATL-PHYS-PUB-2015-018 (2015), https://cdsweb.cern.ch/record/2037683
[6] ATLAS Collaboration, ATL-PHYS-PUB-2015-051 (2015), https://cds.cern.ch/record/2110140
[7] R. Frühwirth, Nucl. Instrum. Methods **A262**, 444 (1987)
[8] L.R. Evans, P. Bryant, J. Instrum. **3**, S08001 (2008)
[9] ATLAS Collaboration, J. Instrum. **9**, P09009 (2014), 1406.7690
[10] ATLAS Collaboration, ATL-PHYS-PUB-2018-002 (2018), https://cds.cern.ch/record/2309474
[11] ATLAS Collaboration, Eur. Phys. J. **C77**, 673 (2017), 1704.07983
[12] T. Sjostrand, S. Mrenna, P.Z. Skands, Comput. Phys. Commun. **178**, 852 (2008), 0710.3820
[13] ATLAS Collaboration, ATL-PHYS-PUB-2014-021 (2014), https://cds.cern.ch/record/1966419
[14] R.D. Ball et al., Nucl. Phys. **B867**, 244 (2013), 1207.1303
[15] H. Bichsel, Rev. Mod. Phys. **60**, 663 (1988)
[16] J. Allison et al., Nucl. Instrum. Meth. **A835**, 186 (2016)
[17] ATLAS Collaboration, Eur. Phys. J. **C70**, 823 (2010), 1005.4568
[18] A. Rosenfeld, J.L. Pfaltz, J. ACM **13**, 471 (1966)
[19] ATLAS Collaboration, ATL-PHYS-PUB-2015-052 (2015), https://cds.cern.ch/record/2116350
[20] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning* (MIT Press, 2016), http://www.deeplearningbook.org
[21] J. Bergstra, Y. Bengio, J. Mach. Learn. Res. **13**, 281 (2012)
[22] D.J. Hand, R.J. Till, Mach. Learn. **45**, 171 (2001)
[23] G. Apollinari, I. Béjar Alonso, O. Brüning, M. Lamont, L. Rossi, CERN-2015-005 (2015), https://cds.cern.ch/record/2116337
[24] ATLAS Collaboration, ATL-PHYS-PUB-2016-025 (2016), https://cdsweb.cern.ch/record/2222304
[25] C.M. Bishop (1994), http://publications.aston.ac.uk/373/

[26] C.M. Bishop, *Pattern Recognition and Machine Learning*, Information science and statistics (Springer, New York, 2006), ISBN 978-0-387-31073-2