Development and performance of track reconstruction algorithms at the energy frontier with the ATLAS detector

Louis-Guillaume Gagnon\textsuperscript{1} on behalf of the ATLAS collaboration

\textsuperscript{1}Université de Montréal - Département de physique, C.P. 6128, Succ. centre-ville, Montréal, Quebec, H3C 3J7, Canada

E-mail: louis.guillaume.gagnon@cern.ch

Abstract. ATLAS track reconstruction software is continuously evolving to match the demands from the increasing instantaneous luminosity of the LHC, as well as the increased center-of-mass energy. These conditions result in a higher abundance of events with dense track environments, such as the core of jets or boosted tau leptons undergoing three-prong decays. These environments are characterised by charged particle separations on the order of the ATLAS inner detector sensor dimensions and are created by the decay of boosted objects. Significant upgrades were made to the track reconstruction software to cope with the expected conditions during LHC Run 2. In particular, new algorithms targeting dense environments were developed. These changes lead to a substantial reduction of reconstruction time while at the same time improving physics performance. The employed methods are presented and physics performance studies are shown, including a measurement of the fraction of lost tracks in jets with high transverse momentum.

1. Introduction

The unprecedented center-of-mass energy of the Large Hadron Collider (LHC), which now runs at $\sqrt{s} = 13$ TeV, has led to an increased prevalence of so-called dense environments, such as the core of high-pt jets, in which the distance between charged particles becomes comparable to the detector resolution.

At the heart of the ATLAS detector \cite{1} lies a state-of-the-art tracking detector called the inner detector (ID) which enables the reconstruction of tracks left by charged particles in the region of pseudorapidity $|\eta| < 2.5$. A silicon pixel tracker is installed at the center of the ID. Since the installation of the insertable B-layer (IBL) \cite{2} in 2015, it comprises four layers of sensors assembled in coaxial cylinders in the central regions and two sets of three disks installed in the forward and backward regions.

Due to electron-hole diffusion in the silicon bulk, charge drift in the axial B-field of the ID and $\delta$-rays, ionising particles typically deposit charge in more than one pixel as seen in figure 1. Charge interpolation can then be used to precisely estimate the location of the particle hit within the cluster. However, in dense environments where the distance between particles is in the order of the pixel size, the charge clusters can merge, compromising the measurement precision which is crucial to the tracking and vertexing performance.
Figure 1. Different causes of charge clusters in the silicon pixel detector. Left: charge drift in the solenoid B-field. Center: cluster merging due to close-by particles. Right: δ-ray. [3]

To recover optimal performance in the dense regime, a machine learning approach using three sets of neural networks has been chosen to identify and split clusters originating from more than one charged particle [3]. The ATLAS tracking chain is described in section 2 along with a discussion of the networks used to recover optimal performance at the energy frontier. Performance measurement are presented in section 3.

2. Tracking in the ATLAS inner detector

2.1. Track finding

The first step in the track reconstruction chain is to find an initial list of track candidates. Track seeds are defined as sets of three space-points (3D coordinates of hits measured in the inner detector) passing $p_T$ and impact parameter cuts. Starting with a seed, a combinatorial Kalman filter [4] is then used to find an actual track candidate.

2.2. Ambiguity solving

Multiple track candidates can be produced from a given seed if at any step in the filter, more than one space-point is compatible with the filter’s estimate. This means that in general, the track finding step will yield an excess of track candidates. An ambiguity solving step is thus required in which the tracks are scored according to a number of metrics, including the number of disconnected track segments, the fit $\chi^2$ and $\log(p_T)$. In order to classify the clusters as comprising charge from one, two, or three or more particles, a neural network is used. If a cluster is used by more than one track and is identified by the neural network as originating from one particle, a penalty is applied to its ambiguity solving score. If the cluster is correctly identified as originating from multiple particles, it is defined as a shared cluster and no penalty is applied.

The neural network takes as input:

- A 7x7 discretized charge matrix, obtained from the calibration of time-over-threshold values measured by the pixel sensors, centered on the charge centroid. As all the neural networks do not accept matrices as inputs, the charge matrix is flattened into a length 49 vector in row-major order.
- A length-7 vector of pixel pitches in the local Y direction $^1$, since the pixel size is not constant in this direction.

$^1$ The positions are measured in a frame of reference local to the pixel sensor considered, in which the local X and Y directions correspond to the transverse and longitudinal directions respectively.
• A binary variable encoding the ID region (endcap or barrel)
• The layer number
• $\phi$ and $\theta$ angles of incidence of track candidates.

The two last quantities have been shown to improve significantly the performance of the neural network [3]. Since the neural network’s performance is robust with respect to smearing of the angle of incidence [5], the true angle of incidence of the MC-generated particle is currently used as a proxy for the actual track measurements.

A dijet Monte Carlo sample generated with Pythia 8.186 [6] using the A14 set of tuned parameters [7] and the NNPDF2.3LO parton distribution function set [8] is employed to produce the training set. A filter that keeps only jets with transverse momentum between 1.8 and 2.5 TeV is applied, resulting in a high fraction of multi-particle clusters. The dataset is subsampled to 12 million clusters containing 22%, 26% and 52% of 1, 2 and $\geq 3$ particle clusters, respectively.

This network as well as those described later are trained with the Keras [9] python package using the theano [10] backend, employing stochastic gradient descent combined with backpropagation to adjust the weights. The training uses a patience-based early-stopping strategy, in which the number of remaining epochs is increased by 1.75 for every epoch that sees a loss decrease of at least 5% for epochs above the 50th one. The required validation loss is computed on 10% of the training set. The hyperparameters used for all networks are listed in table 1. Due to a limitation of the C++ package used to run the neural network during reconstruction of ATLAS data and MC, the output layer for the neural networks used to estimate the charged particle multiplicity and measurement errors use independent sigmoid functions for each output node. An optimization pass using a random search over hyperparameter combinations [11] has shown that these relatively small, shallow networks are a good tradeoff between performance and evaluation speed.

| Hyperparameter         | Number network | Position networks | Error networks |
|------------------------|----------------|-------------------|---------------|
| Structure              | (60)-25-20-(3) | (60)-40-20-(2/4/6) | (62/64/66)-15-10-(30/50/60) |
| Activation             | Sigmoid        | Sigmoid           | Sigmoid       |
| Output activation      | Sigmoid        | Linear            | Sigmoid       |
| Learning rate          | 0.08           | 0.04              | 0.3           |
| L2 regularizer         | 1e-7           | 1e-7              | 1e-6          |
| Momentum               | 0.4            | 0.3               | 0.7           |
| Batch size             | 60             | 30                | 50            |
| Loss function          | cross-entropy  | mean absolute error | cross-entropy |

Table 1. Hyperparameters used to train the three sets of neural networks. In the Structure row, the numbers in parenthesis denote the input and output layer sizes. The sigmoid function is defined here as $f(x) = 1/(1 + e^{-2x})$.

A validation receiver operating characteristic (ROC) curve, produced on a statistically independent subsample taken from the same Monte Carlo dataset as the training set is shown in figure 2.

2.3. Track fitting
After ambiguity solving, all tracks are fitted and their final parameters are measured. However, clusters containing more than one particle are still only assigned a single hit position which compromises the track reconstruction performance, as seen in figure 3. To mitigate this problem, a second set of neural networks is used to measure the true hit position of all particles.
Figure 2. Pairwise receiver operating characteristic (ROC) curves for the neural network used to estimate the number of charged particles contributing to a pixel cluster, in the two particles vs one particle and three particles vs one particle cluster cases, for all the pixel layers including the IBL. The curve in black is produced using the two particles class score and its integral gives the probability that a randomly chosen two particles cluster is scored lower in its true class than a randomly chosen one particle cluster, while the curve in red is produced using the three particles class score and its integral gives the probability that a randomly chosen three particle cluster is scored lower in its true class than a randomly chosen one particle cluster. The random line is obtained by randomly assigning a class to a cluster with a variable bias and represents a universal baseline for the two cases. The curves are produced using a PYTHIA8 dijet sample with a filter selecting jets with $p_T$ between 1.8 and 2.5 TeV. [12]

Figure 3. The cluster residuals in the local x direction for clusters with a width of four pixels in the x-direction reconstructed with the connected component analysis (CCA) clustering algorithm (dashed line) and the NN clustering algorithm (solid line). [3]

Separate trainings are performed for one, two and three particle clusters (clusters with more than three particles are not considered). They all use the same input set as the neural network.
used to identify the particle multiplicity and output one, two or three length-2 position vectors containing the estimated hit position of the particles within the clusters. The training set is produced from the same sample as before but retaining 12 million clusters of the appropriate class for each neural network. The hyperparameters are the same across the set and are listed in table 1. The residual distribution for two particle clusters from a statistically independent subsample taken from the same Monte Carlo dataset as the training set is shown in figure 4.

Finally, a third set of neural networks is used to estimate the uncertainties on the position measurements. There are six neural networks in the set, one per direction / number of particles pair. The input set is the same as for the previous two sets, augmented with the position estimates as given by the previous neural network set. To produce a target, an interval corresponding to the typical residuals range is quantized, and the bin corresponding to the actual residual for a given particle is set to 1, which essentially turns the problem solved by the network into a classification task. After training, for each particle in a given cluster, the networks produce binned probability densities over the residuals. Since the goal is to learn the uncertainty of the underlying distribution, the square root of the second moment of each particle’s distribution in a given direction is defined as the point estimate of the error instead of the most probable value. The pull distribution, where the pull variable is defined as $(\text{pos}_{\text{NN}} - \text{pos}_\text{truth})/\sigma_{\text{NN}}$, in the X direction for clusters from a statistically independent subsample taken from the same Monte Carlo dataset as the training set is shown in figure 4. If the error is correctly estimated, the distributions should have zero mean and unit variance. As can be seen, the standard deviations obtained from truncated gaussian fits are less than one for all cluster types, meaning that the error neural networks typically overestimate the true uncertainty. The shape of the one particle clusters pull is affected by the non-gaussian tail of the associated residual distribution.

**Figure 4.** Left: Difference between the neural network position estimation and the true hit position, in local Y direction, for two particles clusters. The nominal pixel size is of 400 $\mu$m in the local Y direction. Pixels on the front-end chips on a module have a size in local Y of 600 $\mu$m ($\approx$ 10% of the pixels). In the IBL, the pixel size is of 250 $\mu$m in this direction. Right: The difference between the neural network estimation and the true hit position divided by the error estimation, in the local X direction for all pixel layers including the IBL. All distributions are computed using a PYTHIA8 dijet sample with a filter selecting jets with $p_T$ between 1.8 and 2.5 TeV. [12]
3. Performance

3.1. Track reconstruction efficiency

To characterize the performance of the algorithms optimized for dense environments, the shared cluster multiplicity is measured in simulations according to three different selections [13]. An ideal selection is performed at truth-level and is compared to an optimized algorithm (“TIDE”) taking into account shared clusters as identified by the neural networks presented in section 2. Additionally, these selections are compared to a previous unoptimized version of the cluster splitting algorithm (“Baseline”), showing that the optimization of tracking for dense environments allows to recover the trend of the ideal selection (figure 5). Ref. [13] lists the full set of differences between the optimized and unoptimized algorithms.

![Figure 5](image_url)

**Figure 5.** A comparison of the average number of merged pixel clusters on truth-based reconstruction tracks and split (shareable) pixel clusters is shown as a function of the $\rho$-meson and $\tau$-lepton transverse momentum. The trend of the average number of shareable clusters on tracks reconstructed by the TIDE optimized chain follows the true average number of merged clusters as shown on the points labeled Ideal better than the Baseline reconstruction chain. Ref. [13] lists the full set of differences between the Baseline and TIDE reconstruction chain.

3.2. Lost track fraction

A data-driven performance measurement can be obtained by performing a $dE/dX$ fit for a single-particle cluster selection to single and multiple particle templates. The method is described in detail in ref. [14].

First, a single-particle cluster template is obtained by selecting clusters away from a jet core. The multiple-particle clusters template is obtained by selecting clusters used by more than one track within the jet core. Then, the $dE/dX$ distribution measured from clusters used by only one track within the jet core is fit to the two templates as shown in figure 6. In the ideal regime where the multiple particle clusters are perfectly identified, only the single-particle template should be used by the fit and the weight of the multiple-particle template in the fit result, called $F_{lost}$, measures the inefficiency of tracking in dense environments. This inefficiency is related to the fraction of clusters wrongly identified as comprising charge from only one particle by the neural network presented in section 2.2. As can be seen in figure 7, this fraction grows with
the jet $p_T$ but remains small, at around 3% for a 1.5 TeV jet, and the data and Monte Carlo agreement is within 25% for all considered $p_T$ bins.

![Figure 6. Left: Single-track and multiple-track templates Right: Data $dE/dx$ distributions (black circles) with fit results (red solid line) in two bins of jet $p_T$. The Single-Track template scaled by $1 - F_{\text{lost}}$ is shown as the Single-Track Contribution (blue dashed line) and the Multiple-Track template scaled by $F_{\text{lost}}$ is shown as the Multiple-Track Contribution (green dashed line). The bottom panel shows the ratio of fit/data within the fit range (0.67-3.07 MeV g$^{-1}$ cm$^2$). [14]](image)

![Figure 7. Left: The measured fraction of lost tracks, $F_{\text{lost}}$, in the jet core ($\Delta R(jet, trk) < 0.05$) as a function of jet $p_T$ for data (black circles) and simulation (red squares). Black error bars indicate statistical uncertainty while the grey and red error bands indicate the total uncertainty for data and simulation respectively. Right: The ratio of the fraction of lost tracks, $F_{\text{lost}}$, in data with respect to simulation (PYTHIA8) as a function of jet $p_T$. Black error bars indicate the combined statistical uncertainty of data and simulation while the grey error band indicates the total uncertainty, taking into account the statistical and systematic uncertainties of both data and simulation. [14]](image)
4. Conclusion
ATLAS track reconstruction algorithms that were optimized for dense environments such as the core of high $p_T$ jets have been presented. In particular, a neural network algorithm used to identify and split charge clusters originating from more than one charged particle in the pixel detector has been described in detail.

Such dense environments are more and more frequent at the energy frontier, and it has been shown that the optimization of ATLAS tracking algorithms leads to a better shared cluster identification efficiency in such difficult cases. A data-driven measurement has also been presented, showing that the fraction of lost tracks near the core of high-$p_T$ jets remains small, even at very high $p_T$.

References
[1] ATLAS Collaboration 2008 *J. Instrum.* 3 S08003
[2] ATLAS Collaboration 2010 ATLAS-TDR-19 URL http://cds.cern.ch/record/1291633
[3] ATLAS Collaboration 2014 *J. Instrum.* 9 P09009 (Preprint 1406.7690)
[4] Frühwirth R 1987 *Nucl. Instrum. Meth.* A 262 444 – 450
[5] ATLAS Collaboration ATL-PHYS-PUB-2015-052 URL https://cds.cern.ch/record/2116350
[6] Sjostrand T, Mrenna S and Skands P Z 2008 *Comput. Phys. Commun.* 178 852 (Preprint 0710.3820)
[7] ATLAS Collaboration 2014 ATL-PHYS-PUB-2014-021 URL https://cds.cern.ch/record/1966419
[8] Ball R D et al. 2013 *Nucl. Phys.* B867 244–289 (Preprint 1207.1303)
[9] Chollet F 2015 Keras https://github.com/fchollet/keras
[10] Theano Development Team 2016 (Preprint 1605.02688)
[11] Bergstra J and Bengio Y 2012 *J. Mach. Learn. Res.* 13 281–305
[12] ATLAS Collaboration 2016 IDTR-2016-010 URL https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PLOTS/IDTR-2016-010/
[13] ATLAS Collaboration 2015 ATL-PHYS-PUB-2015-006 URL http://cds.cern.ch/record/2002609
[14] ATLAS Collaboration 2016 ATL-PHYS-PUB-2016-007 URL http://cds.cern.ch/record/2140460