Advances in developing deep neural networks for finding primary vertices in proton-proton collisions at the LHC

Simon Akar[1], *Mohamed* Elashri[1], *Rocky Bala* Garg[2,*], *Elliott* Kauffman[3], *Michael* Peters[1], *Henry* Schreiner[3], *Michael* Sokoloff[1,**], *William* Tepe[1], and *Lauren* Tompkins[2]

[1]University of Cincinnati
[2]Stanford University
[3]Princeton University

Abstract. We are studying the use of deep neural networks (DNNs) to identify and locate primary vertices (PVs) in proton-proton collisions at the LHC. Earlier work focused on finding primary vertices in simulated LHCb data using a hybrid approach that started with kernel density estimators (KDEs) derived heuristically from the ensemble of charged track parameters and predicted "target histogram" proxies, from which the actual PV positions are extracted. We have recently demonstrated that using a UNet architecture performs indistinguishably from a "flat" convolutional neural network model. We have developed an "end-to-end" tracks-to-hist DNN that predicts target histograms directly from track parameters using simulated LHCb data that provides better performance (a lower false positive rate for the same high efficiency) than the best KDE-to-hists model studied. This DNN also provides better efficiency than the default heuristic algorithm for the same low false positive rate. "Quantization" of this model, using FP16 rather than FP32 arithmetic, degrades its performance minimally. Reducing the number of UNet channels degrades performance more substantially. We have demonstrated that the KDE-to-hists algorithm developed for LHCb data can be adapted to ATLAS and ACTS data using two variations of the UNet architecture. Within ATLAS/ACTS, these algorithms have been validated against the standard vertex finder algorithm. Both variations produce PV-finding efficiencies similar to that of the standard algorithm and vertex-vertex separation resolutions that are significantly better.

1 Introduction

Reconstruction of proton-proton collision points, referred to as primary vertices (PVs), is critical for physics analyses conducted by all experiments at the Large Hadron Collider (LHC) and for triggering in LHCb. The precise identification of the PV locations, and their other characteristics, enables the complete reconstruction of final states under investigation. Moreover, it provides crucial information about the collision environment, which is essential for obtaining accurate measurements. The task of PV reconstruction poses a significant challenge across the experiments conducted at the LHC.

The LHCb detector has been upgraded for Run 3 of the LHC so that it can process a five-fold increase in its instantaneous luminosity compared to Run 2 and it has removed its

*e-mail: rocky.bala.garg@cern.ch
**e-mail: sokoloff@ucmail.uc.edu

hardware-level trigger in favor of a pure software trigger [1]. The average number of visible PVs detected in the vicinity of the beam crossing area has increased from 1.1 to 5.6. In contrast, the ATLAS experiment has observed an average of 40-60 simultaneous collisions (known as pile-up, μ) during Run 3 in 2023 and is expected to see 140-200 simultaneous collisions during the coming high-luminosity phase of the LHC. These demanding conditions invite development of new PV reconstruction algorithms to address these challenges.

This document presents the implementation and performance of a family of machine learning PV reconstruction algorithms known as `PV-Finder` for both LHCb and ATLAS. Conceptually, these algorithms compute one-dimensional Kernel Density Estimators (KDEs) that describe where charged track trajectories overlap in the vicinity of the beamline and use these as input feature sets for convolutional neural networks (CNNs) that predict `target histograms` that are proxies for the PV positions. LHCb has traditionally used heuristically computed KDEs with its CNNs; in this papers it reports merging a fully connected neural network for KDE computation with a CNN to produce an "end-to-end" `tracks-to-hist` deep neural network (DNN) model and compares its performance with that of older models. ATLAS currently uses an analytical approach for KDE computation (referred to as a `KDE-to-hist` model) and compares the performance with the Adaptive Multi-Vertex Finder (AMVF) algorithm [2], the heuristic PV identification algorithm currently used in ATLAS.

2 `PV-Finder` in LHCb

The original LHCb DNN for reconstructing PVs used a single kernel density estimator (KDE) calculated using a heuristic algorithm as the input feature set for each event (each beam crossing) and produced a target histogram from which PV positions were deduced. We refer to this class of algorithms as `KDE-to-hist` algorithms. The results of the initial proof-of-principal project, and some details of the “toy Monte Carlo" and KDE used for that study are reported in Ref. [3]. Using track parameters produced by the LHCb Run 3 Vertex Locator (VELO) tracking algorithm [4] leads to significantly better performance [5]. Since then, our research has advanced in several directions. We replaced our original input feature set with four input feature sets: a first KDE based on summed probabilities in voxels projected onto the beam axis, a second KDE based on summed (probability-squared values) in voxels projected onto the beam axis, plus the $x-$ and $y-$ coordinates of the maximum summed probability at each value of z (along the beam axis). We found that using a modified U-Net architecture [6] in place of our original CNN architecture provided equally good fidelity and trained much more quickly. We also investigated using a fully connected network to calculate a KDE from track parameters (a `tracks-to-KDE` model) and merging this model with a `KDE-to-hist` model to produce an “end-to-end" `tracks-to-hist` neural network. The fidelity of the `tracks-to-hist` studied then was inferior to that of the `KDE-to-hist` models. The results of these studies were presented at CHEP-2021 [7].

A major advance reported at this conference (CHEP-2023) is that we have produced a `tracks-to-hist` model that produces efficiencies very similar to the best produced by our `KDE-to-hist` models *and* produces significantly lower false positive (FP) rates. These results were reported previously at ACAT-2022 [8]. Below, we summarize the most salient features. Brand new for this conference are results using FP16 arithmetic rather than FP32 arithmetic for the `tracks-to-hist` model and results using smaller U-Net components in the FP16 `tracks-to-hist` models.

The current `tracks-to-hist` model, whose architecture is shown in Fig. 1, includes a few updates relative to the original version described in Ref. [7]: the `tracks-to-KDE` part of the model consists of 6 fully connected layers that are initially trained to produce a KDE and the weights of the first 5 layers are temporarily frozen; a variation with 8 latent feature

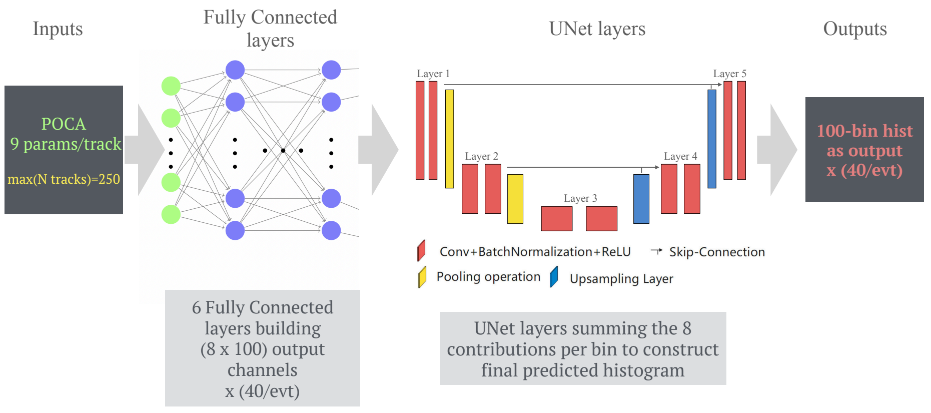
Inputs
Fully Connected layers
UNet layers
Outputs
POCA
9 params/track
max(N tracks)=250
Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
100-bin hist as output x (40/evt)
Conv+BatchNormalization+ReLU
Skip-Connection
Pooling operation
Upsampling Layer
6 Fully Connected layers building (8 x 100) output channels x (40/evt)
UNet layers summing the 8 contributions per bin to construct final predicted histogram

Figure 1: This diagram illustrates the end-to-end, `tracks-to-hist`, DNN Each event is now sliced into 40 independent 100-bin intervals. Six fully connected layers populate 8 100-bin channels in sixth layer, for each track. These contributions are summed and processed by a U-Net model with 5 convolutional layers to construct the final 100-bin histogram.

sets is merged to a `KDE-to-hist`-like DNN where the classical CNN layers are replaced by a U-Net model. Critically, we also updated the structure of the input data for training and inference. In the earlier approach [7], the target histograms consisted of 4000 bins along the z-direction (beamline), each 100 μm wide, spanning the active area of the VELO around the interaction point, such that $z \in [-100, 300]$ mm. Parameters describing all tracks served as input features. In place of describing the true PVs using a single 4000-bin histogram, we now slice each event into 40 intervals of 100 bins each. For each interval, parameters of tracks whose points of closest approach to the beamline lie within 2.5 mm of the interval edges are used as input features. This approach is motivated by the fact that the shapes of the target histogram are expected to be invariant as a function of the true PV position and it is easier for a DNN to learn to predict target histograms over a smaller range of bins. In particular, the fully connected layers that calculate the KDE-like latent features used as input features by the U-Net layers predict heuristic KDEs as the ground truth much more effectively when training 100-bin intervals rather than the full 4000-bin range. Additionally, the depth of the U-Net part of the DNN can be lower when processing a 100-bin feature set rather than a 4000-bin feature set. With an average of $\sim$ 5 PVs per event, most of the bins in both the KDE and target histograms have no significant activity. We expect this will allow us to eventually build a more performant inference engine in the LHCb software stack. The 40 intervals of 100 bins are independent and homogeneous between events. Each interval is treated independently, after which the predicted 4000-bin histogram is stitched back together. As in past studies, an asymmetry parameter between the cost of overestimating contributions to the target histograms and underestimating them [3] is used as a hyperparameter to allow higher efficiency by incurring higher false positive rates.

Performance is evaluated using a heuristic algorithm, based on the PV positions along the beam axis, z. Exactly how efficiencies and FP rates are calculated is described in Ref. [7]. The left-hand plot in Fig. 2 shows how the performance of the DNN algorithms have evolved over time. The efficiency is shown on the horizontal axis and the false positive rate per event is shown on the vertical axis. The solid blue circles show the performance of any early `KDE-to-hist` model described at ACAT-2019 [3]. The green squares show the performances of a `KDE-to-hist` described at Connecting-the-Dots in 2020 [5]. Both of the above models were trained using "toy Monte Carlo" with proto-tracking. All subsequent DNN models were

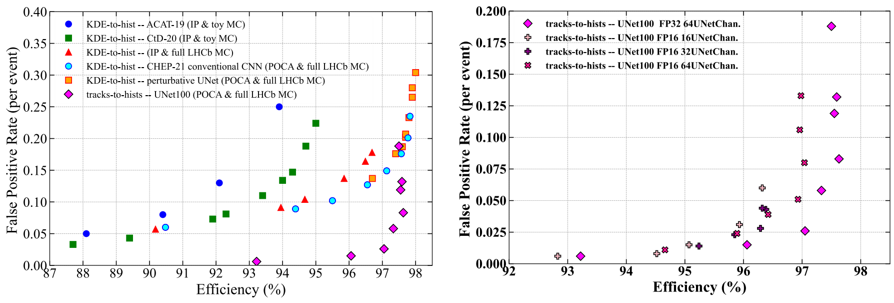

Figure 2: (left) Comparison between the performances of models reported in previous years and the new `tracks-to-hist` model (magenta diamonds). A cost asymmetry parameter described in Ref. [3] is varied to produce the families of points observed. (right) Comparison between `tracks-to-hist` models. The magenta diamonds here are the same as in the plot on the left. The other models have U-Net architectures but use FP16 arithmetic rather than FP32. Two of the FP16 models have smaller U-Net components than the FP32 model. NB: the horizontal and vertical scales on the right cover more limited ranges than those on the left.

trained using full VELO tracking algorithm [4], leading to significantly better performances (red triangles to be compared to green squares). The cyan circles and the yellow squares correspond to the best achieved performances for `KDE-to-hist` models using either a classical CNN architecture or the U-Net model described at CHEP-2021

The performances of all above models were obtained using an “older" matching procedure with a fixed search window of 0.5 mm. The magenta diamonds show the performance of the `tracks-to-hist` model described above using the matching procedure described in Ref. [7]. The new `tracks-to-hist` model enables the DNN to simultaneously reach high efficiencies ($> 97\%$) and low false positive rates (0.03 per event or 0.6% per reconstructed PV).

Running an inference engine inside a software stack adds another “knob to turn" – throughput versus fidelity. Computing resources are finite, especially in LHCb’s first level software trigger which processes 30 MHz of beam crossing data, about 40 Tbit/s, in a GPU application [1]. Modern GPUs provide FP16 performance that can be about twice as fast as FP32 arithmetic, so it is interesting to investigate whether using FP16 arithmetic degrades performance significantly. It similarly interesting to investigate how performance degrades as the size of the convolutional network inside our DNN is reduced. The right-hand plot in Fig. 2 shows the efficiency versus FP rate for four DNN configurations. The magenta diamonds correspond to the default `tracks-to-hist` configuration. These points are exactly the same as those in the left-hand plot; the ranges of the axes have been modified to focus on the region of interest. The purple "×" markers correspond to the same logical configuration, but using FP16 arithmetic rather than FP32. Near 96% efficiency, the FP rate has increased marginally. Near 97% efficiency, the FP rate has increased much more substantially. Reducing the number of U-Net channels from 64 to 32 or 16, while using FP16 arithmetic, (the darker and lighter crosses in the plot) additionally additionally increases the FP rate near 96% efficiency by a small amount, but increases the FP rate much more significantly near 96.5%. We have begun to code an inference engine to run in LHCb’s first level software trigger. The details of the model to be instantiated will balance fidelity of the model against throughput.

3 `PV-Finder` in ATLAS

The ATLAS experiment at the LHC is a versatile particle detector designed with a symmetric cylindrical geometry and near-complete coverage of 4π in solid angle [9]. It has a multi-layer structure with many sub-detector systems including an inner tracking detector, superconducting magnets, electromagnetic and hadronic calorimeters, and a muon spectrometer. An extensive software suite [10] facilitates its various functions such as data reconstruction and analysis, detector operations, trigger and data acquisition systems etc.

The input dataset used for studying `PV-Finder` in ATLAS has been generated using POWHEG BOX[v2] [11] interfaced with PYTHIA[8.230] [12] and processed through the ATLAS detector simulation framework [10], using the GEANT4 toolkit [13]. The hard-scatter (HS) process involves the production of semi-leptonically decaying top quark pairs ($t\bar{t}$) from proton-proton collisions at a center-of-mass energy of 13 TeV, overlaid with simulated minimum-bias events with an average pile-up of 60.

3.1 `PV-Finder` algorithm and model architecture

The flowchart representing work-flow of the `PV-Finder` algorithm for ATLAS is shown in figure 3. More details about the architecture can be found at the ATLAS PubNote [14]. Truth-matched reconstructed tracks passing tight quality selection cuts [15] and $p_T > 500$ MeV are used for the preparation of input features for the neural network. A track's signed radial and longitudinal impact parameters, d_0 and z_0, measured at the point of closest approach (POCA) to the beamline, and their uncertainties, $\sigma(d_0)$ and $\sigma(z_0)$, are used as input to generate KDEs. Each KDE feature is a one-dimensional binned histogram with 12,000 bins in $z \in [-240, 240]$ mm, corresponding to a bin-size of 40 μm.

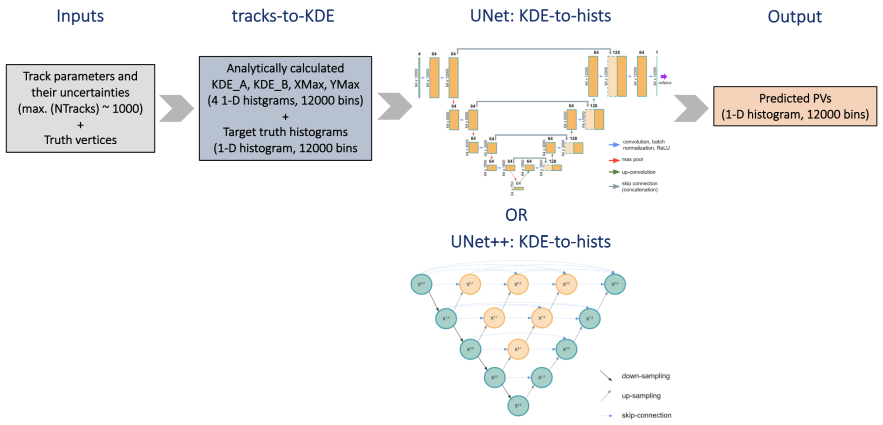
Inputs
tracks-to-KDE
UNet: KDE-to-hists
Output
Track parameters and their uncertainties (max. (NTracks) ~ 1000) + Truth vertices
Analytically calculated KDE_A, KDE_B, XMax, YMax (4 1-D histgrams, 12000 bins) + Target truth histograms (1-D histogram, 12000 bins
Predicted PVs (1-D histogram, 12000 bins)
convolution, batch normalization, ReLU
max pool
up-convolution
skip connection (concatenation)
softplus
OR
UNet++: KDE-to-hists
down-sampling
up-sampling
skip-connection

Figure 3: Flowchart representing work-flow of the `PV-Finder` algorithm from left to right.

To compute these features, each track is modeled as a correlated radial and longitudinal Gaussian probability distribution $\mathbb{P}(d, z)$ centred at (d_0, z_0) which is defined as follows:

$$\mathbb{P}(r) = \mathbb{P}(d, z) = \frac{1}{2\pi\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}\Big((d-d_0),(z-z_0)\Big)^T \Sigma^{-1} \Big((d-d_0),(z-z_0)\Big)\right) \qquad (1)$$

where d and z are coordinates in the radial and longitudinal directions and $\Sigma = \begin{pmatrix} \sigma^2(d_0) & \sigma(d_0, z_0) \\ \sigma(d_0, z_0) & \sigma^2(z_0) \end{pmatrix}$ is the covariance matrix. The sum of probabilities from all the contributing tracks is considered in each z-bin and four KDE features are constructed: `KDE-A` (sum of track probability values), `KDE-B` (sum of the squares of track probability values), `XMax` (`YMax`) (location of the maximum summed track probability in $x(y)$ (mm)). An example illustrating these four features for a random event is shown in Fig. 4. The vertical grey lines in the upper plot mark the locations of true primary vertices while the horizontal grey line in the lower plot denotes the position of the beam spot in the radial direction. A restricted range of the luminous region is shown so that details can be seen.

To train the neural network, a one-dimensional target truth histogram, with the same binning as the input features and calculated by considering Gaussian probabilities around truth vertex locations, is also provided as input along with the four KDE features. A CNN is trained on these features which then outputs a distribution with approximately Gaussian peaks centered at the predicted locations of PVs. An algorithm then takes this predicted distribution and identifies the candidate PV locations on the z-axis by finding the local maxima. Two NN architectures have been considered for these studies: the UNet architecture is inspired from the original architecture developed for biomedical image segmentation [6] while the UNet++ architecture is a variation of UNet with dense skip connections.

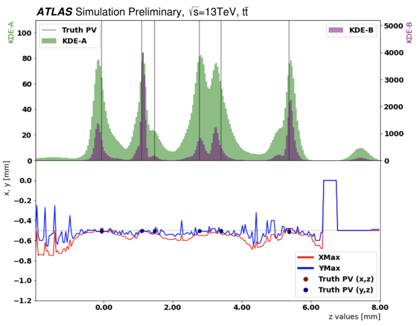

Figure 4: An example of the input KDE features. Upper plot shows `KDE-A` (green) and `KDE-B` (purple) while lower plot shows `XMax` (blue) and `YMax` (red). See text for details.

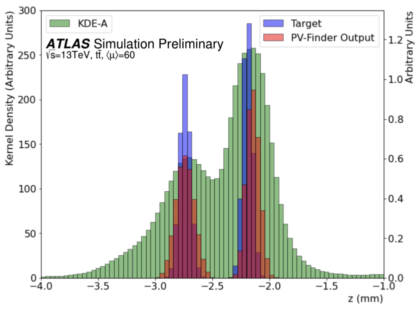

Figure 5: An example of correctly predicted `PV-Finder` output (in red) compared with KDE-A (in green) and target truth (in blue).

3.2 Performance

The `PV-Finder` algorithm's performance for UNet and UNet++ architectures has been studied and a comparative analysis is conducted with the AMVF algorithm using an independent test data sample. Figure 5 showcases an example of two adjacent vertices accurately located by the `PV-Finder` algorithm. To quantitatively evaluate the performance of the `PV-Finder`, vertex classification is performed, and efficiency and false positive rates are calculated. The classification assigns vertices into distinct categories, namely clean, merged, split, and fake based on the distance between the center of a predicted vertex and the z-location of truth vertices. The classification is illustrated in Figure 6 and demonstrated in Figure 7 for the three approaches.

The truth and reconstructed primary vertices are associated based on a vertex-vertex resolution, $\sigma_{\text{vtx-vtx}}$, which is obtained by computing the z-difference between pairs of nearby reconstructed vertices and fitting the distribution with the fit function: $y = \frac{a}{1+\exp(b\cdot(R_{cc}-|x|))} + c$, where a, b, c are free parameters, and R_{cc} is the cluster–cluster resolution referred to as $\sigma_{\text{vtx-vtx}}$. The vertex-vertex resolution for `PV-Finder` UNet, `PV-Finder` UNet++ and AMVF is presented in Figure 8 and Table 1.

The vertex finding efficiency is defined as the number of truth vertices assigned to reconstructed vertices as "clean" and "merged" divided by the total number of reconstructable truth vertices while the false positive rate is defined as the average number of predicted vertices not matched to any truth vertex. Figure 9 shows the vertex finding efficiency as a function of the number of reconstructed tracks associated to a truth vertex and Table 1 shows the average efficiency and false positive rates obtained for three cases.

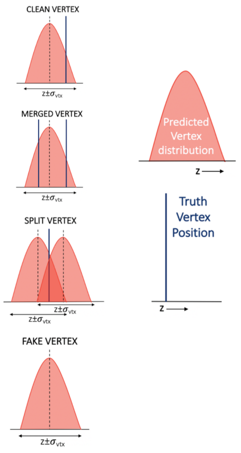

Figure 6: Vertex Classification Scheme. The dashed line represents the reconstructed vertex location, the solid line represents the truth vertex location, and σ_{vtx} represents $\sigma_{\text{vtx-vtx}}$.

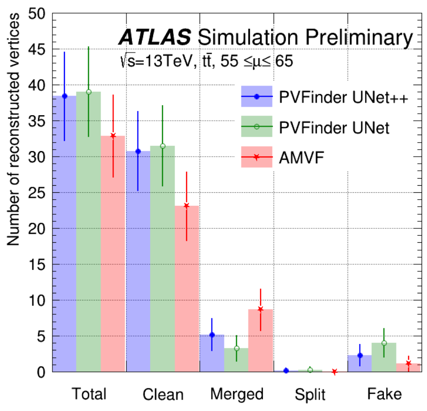

Figure 7: Vertex Classification rates for `PV-Finder` and AMVF for a $t\bar{t}$ sample with $55 \leq \mu \leq 65$, comparing AMVF (red), `PV-Finder` UNet (green) and `PV-Finder` UNet++ (blue).

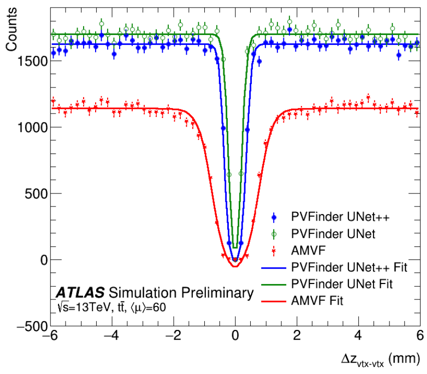

Figure 8: $\Delta z_{\text{vtx-vtx}}$ for `PV-Finder` UNet++ (blue), `PV-Finder` UNet (green) and AMVF (red). See text for discussion. More details at [14].

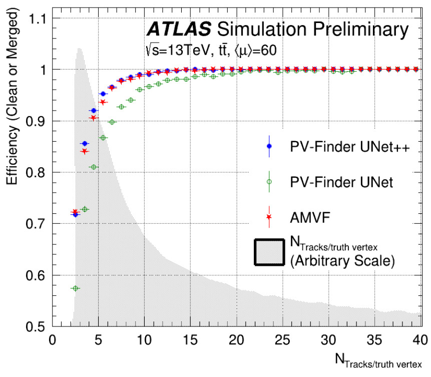

Figure 9: PV efficiencies for AMVF and `PV-Finder`'s UNet++ and UNet architectures vs the number of reconstructed tracks associated to truth vertex. More details at [14].

Method	$\sigma_{\text{vtx-vtx}}$ (mm)	Efficiency	FP Rate (per event)
`PV-Finder` UNet	0.23 ± 0.01	88.7%	2.6
`PV-Finder` UNet++	0.37 ± 0.01	94.2%	1.5
AMVF	0.76 ± 0.02	93.9%	0.8

Table 1: A comparison of the vertex-vertex separation resolutions, the efficiencies, and the false positive (FP) rates of the algorithms.

4 Conclusion

The `PV-Finder` family of algorithms has been studied by both the LHCb and ATLAS experiments. LHCb has demonstrated the performances of the end-to-end `tracks-to-hist` approach for several configurations including those that use FP16 arithmetic rather than FP32. ATLAS has demonstrated that a hybrid `KDE-to-hist` approach produces efficiencies comparable to the ATLAS AMVF algorithm while also achieving significanty improved resolution. These enhanced efficiency and resolution metrics hold significant importance, especially considering the future High Luminosity LHC program. The results are promising and motivate further studies and refinement of the `PV-Finder` algorithms across experiments.

[This work was supported by the National Science Foundation under Cooperative Agreements OAC-1836650 and PHY-2323298]

References

[1] R. Aaij et al., Comput. Softw. Big Sci. **4**, 7 (2020), `arXiv:1912.09161`

[2] The ATLAS collaboration, *Development of ATLAS Primary Vertex Reconstruction for LHC Run 3* (2019), `http://cds.cern.ch/record/2670380`

[3] R. Fang, H.F. Schreiner, M.D. Sokoloff, C. Weisser, M. Williams, J. Phys. Conf. Ser. **1525**, 012079 (2020), `arXiv:1906.08306`

[4] A. Hennequin, B. Couturier, V. Gligorov, S. Ponce, R. Quagliani, L. Lacassagne, JINST **15**, P06018 (2020), `arXiv:1912.09901`

[5] S. Akar, T.J. Boettcher, S. Carl, H.F. Schreiner, M.D. Sokoloff, M. Stahl, C. Weisser, M. Williams, *An updated hybrid deep learning algorithm for identifying and locating primary vertices* (2020), `arXiv:2007.01023`

[6] O. Ronneberger, P. Fischer, T. Brox, pp. 234–241 (2015), `arXiv:1505.04597`

[7] S. Akar, G. Atluri, T. Boettcher, M. Peters, H. Schreiner, M. Sokoloff, M. Stahl, W. Tepe, C. Weisser, M. Williams, EPJ Web Conf. **251**, 04012 (2021), `arXiv:2103.04962`

[8] S. Akar, M. Peters, H. Schreiner, M.D. Sokoloff, W. Tepe (2023), `arXiv:2304.02423`

[9] The ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider: A Description of the Detector Configuration for Run 3* (2023), `2305.16623`, `https://arxiv.org/abs/2305.16623`

[10] The ATLAS Collaboration, *The ATLAS Collaboration Software and Firmware* (2021), ATL-SOFT-PUB-2021-001, `https://cds.cern.ch/record/2767187`

[11] S. Alioli, P. Nason, C. Oleari, E. Re, JHEP **06**, 043 (2010), `arXiv:1002.2581`

[12] Sjöstrand, Torbjörn, et al., Comput. Phys. Commun. **191**, 159 (2015), `arXiv:1410.3012`

[13] S. Agostinelli et al. (GEANT4), Nucl. Instrum. Meth. A **506**, 250 (2003), `https://cds.cern.ch/record/602040`

[14] The ATLAS collaboration, *Primary Vertex identification using deep learning in ATLAS* (2023), `https://cds.cern.ch/record/2858348`

[15] The ATLAS Collaboration, *Early Inner Detector Tracking Performance in the 2015 data at $\sqrt{s}$ = 13 TeV* (2015), ATL-PHYS-PUB-2015-051, `https://cds.cern.ch/record/2110140`