Studying a New Primary Vertex (PV) Identification Algorithm Within ACTS Framework

Oct. 12th 2023

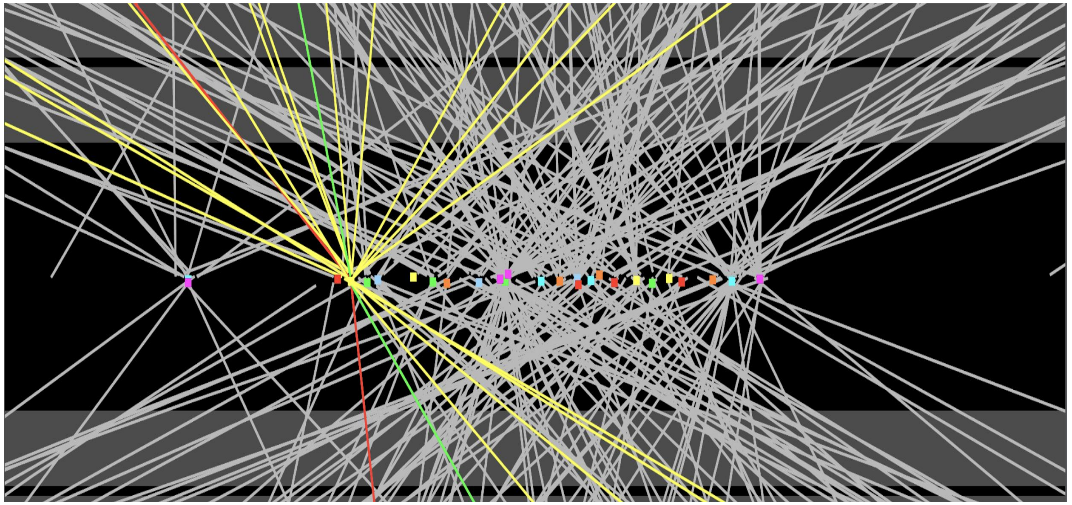
Presenter: Layan AlSarayra

Authors: Layan AlSarayra[1], Rocky Bala Garg[2], Lauren Tompkins[2], Ananya Singha[3]

[1] IRIS-HEP Fellow
[2] Stanford University
[3] HSF-India Fellow

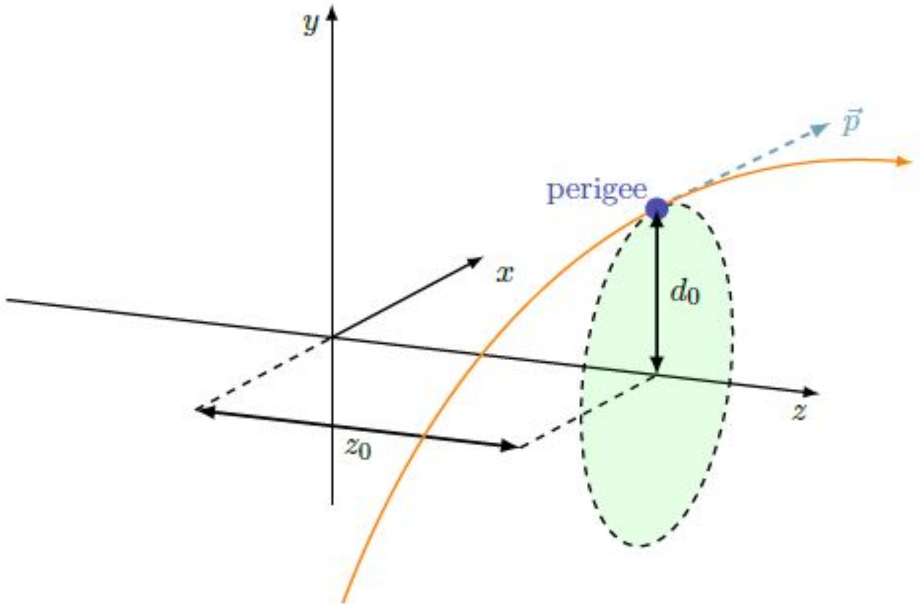
Primary Vertices

- A Primary Vertex (PV) is the accurate estimation of the inelastic p-p interaction point
- 50-70 collisions per event in Run3 of LHC, 150-200 expected at HL-LHC
- A fast and efficient algorithm is needed to identify PVs from the reconstructed tracks


An actual collision environment in ATLAS

Tracks and Track Reconstruction

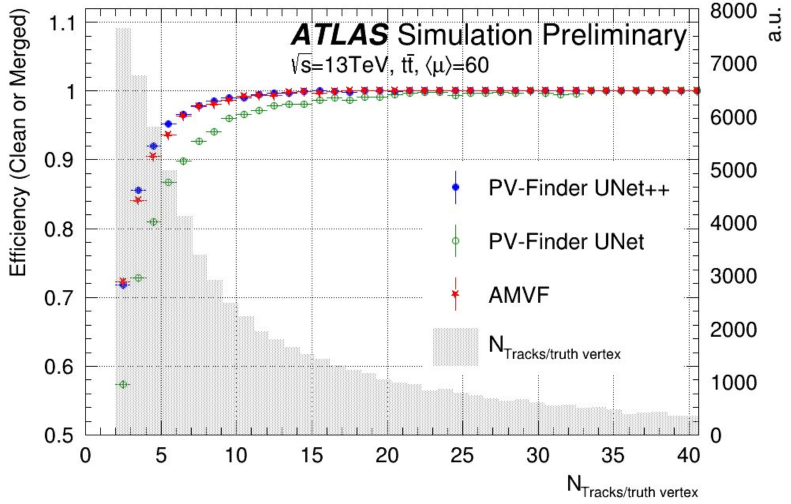
- ➢ Tracks are the paths taken by charged particles while traversing through the detector
- ➢ Track Reconstruction: Finding sets of measurement coming from one charged particle and building the associated trajectory

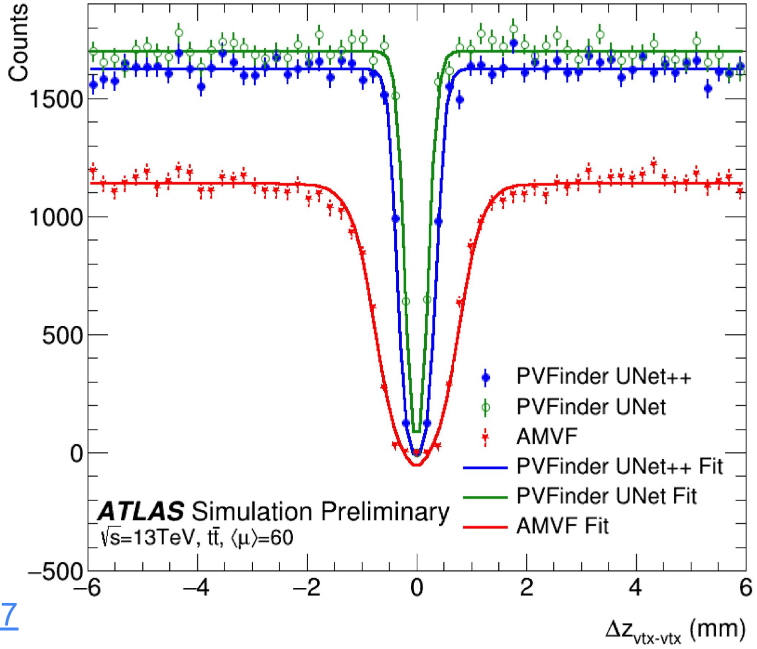


A helical track can be defined by **5** parameters

For our study we are using **2**; **d_0** and **z_0** and their uncertainties

A Previous Study on a PV Identification Algorithm on ATLAS Data





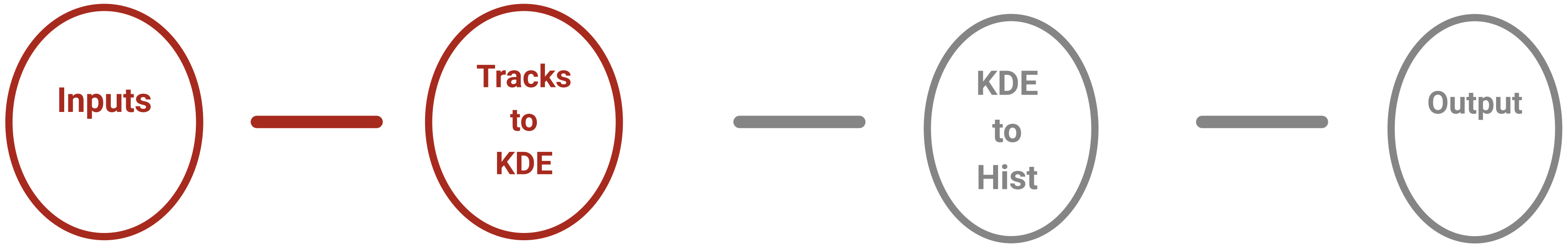
https://cds.cern.ch/record/2858348

https://inspirehep.net/literature/1740667

Promising Results:

The PV-Finder algorithm first developed for LHCb, was adapted for ATLAS simulated data and achieved comparable performance to Adaptive Multi-Vertex Finder (AMVF), and obtained better efficiency and more than two times better resolution with a slight increase in fake rate over all the pile-up range

A Deep Learning Based Approach



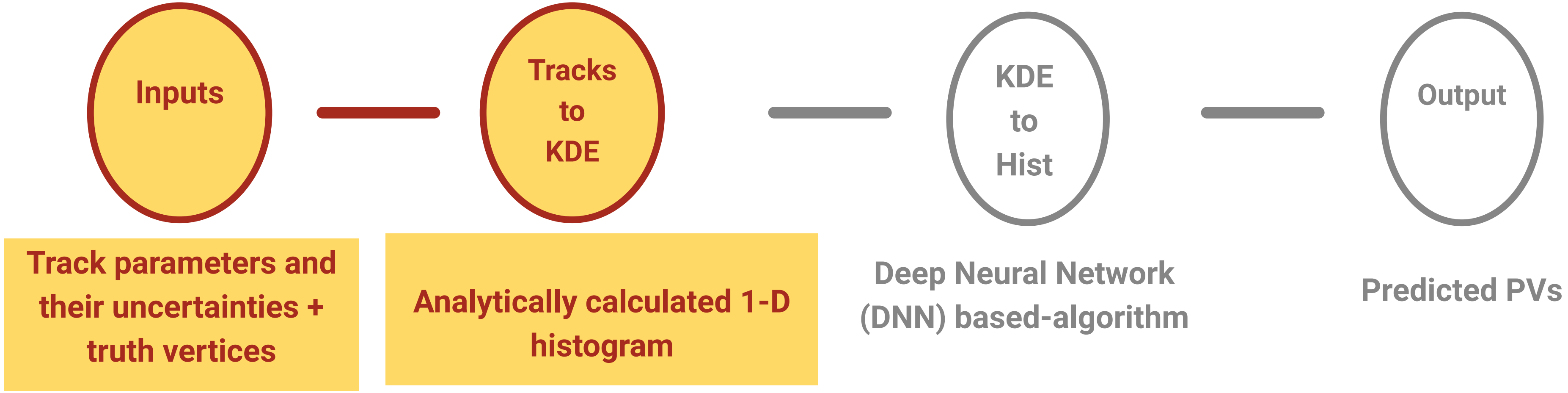
Track parameters and their uncertainties + truth vertices

Analytically calculated 1-D histogram

Deep Neural Network (DNN) based-algorithm

Predicted PVs

A Deep Learning Based Approach


Inputs
Tracks to KDE
KDE to Hist
Output
Track parameters and their uncertainties + truth vertices
Analytically calculated 1-D histogram
Deep Neural Network (DNN) based-algorithm
Predicted PVs

By taking reconstructed tracks and their associated uncertainties, we can calculate KDEs $\longrightarrow$ which is sufficient to feed it to NN to estimate PVs locations

Tracks to KDEs : Analytical Approach to Kernel Density Estimation

- KDE: 1D probability distribution estimation technique that transforms the tracks and their measured resolutions into representations of the track density
- A track's density in the (d0,z0)-plane is given by a transverse and longitudinally correlated Gaussian probability distribution centered around (d_{0i}, z_{0i})

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

$$\Sigma_i = \begin{pmatrix} \sigma^2(d_{0,i}) & \sigma(d_{0,i},z_{0,i}) \\ \sigma(d_{0,i},z_{0,i}) & \sigma^2(z_{0,i}) \end{pmatrix}$$

$$W(z) = \sum_{i\in\text{Tracks}} P_i(0,z)$$

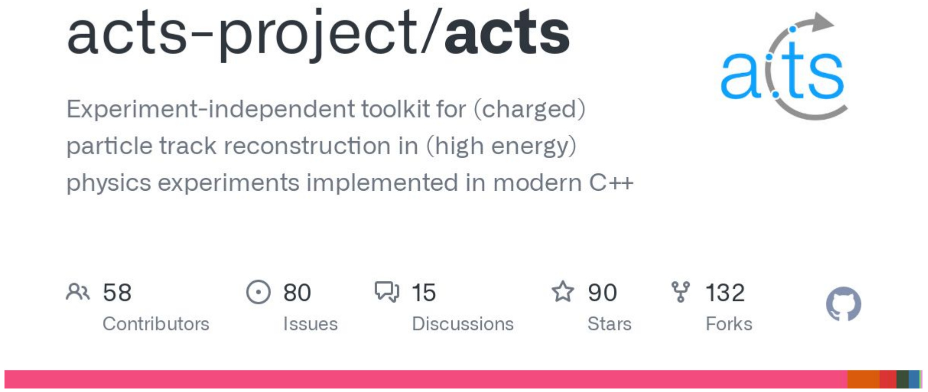
ACTS - A Common Tracking Software

- ACTS is an independent, free, open-source software project for track reconstruction in particle physics experiments
- It's designed to be easily adapting to specific experiment's needs
- Provides essential components, such as:

- Track models (e.g. helical, linear)
- Fitting algorithms
- Geometry components (surfaces, volumes, detector elements, etc)
- Propagation components
- Visualization

And much more!

https://link.springer.com/article/10.1007/s41781-021-00078-8

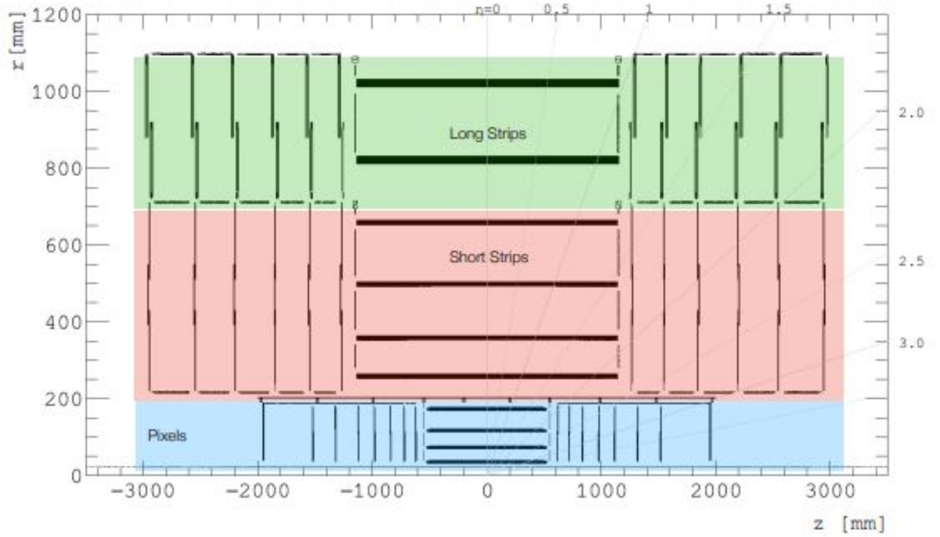

acts-project/**acts**

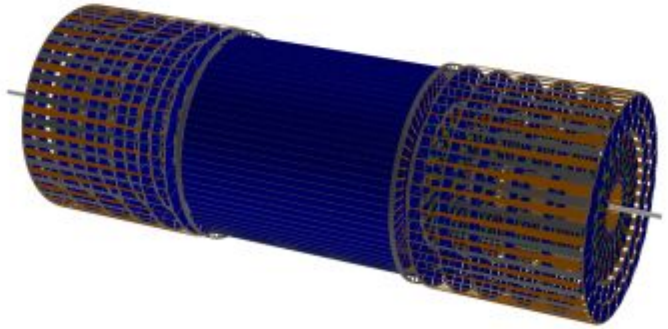
Experiment-independent toolkit for (charged) particle track reconstruction in (high energy) physics experiments implemented in modern C++

58 Contributors · 80 Issues · 15 Discussions · 90 Stars · 132 Forks

Setup in ACTS

1. Detector used: Open Data Detector (ODD)
2. Event generation: Pythia8 with ttbar
3. Reference PVs: AMVF algorithm
4. Pile-up: 50
5. Events Number: 3000
6. Number of tracks/event ~ 300 tracks





https://iopscience.iop.org/article/10.1088/1742-6596/2438/1/012110

Implementing PDFs in ACTS to generate KDE histograms

Gather track parameters and their covariances from tracks generated by CKF algorithm

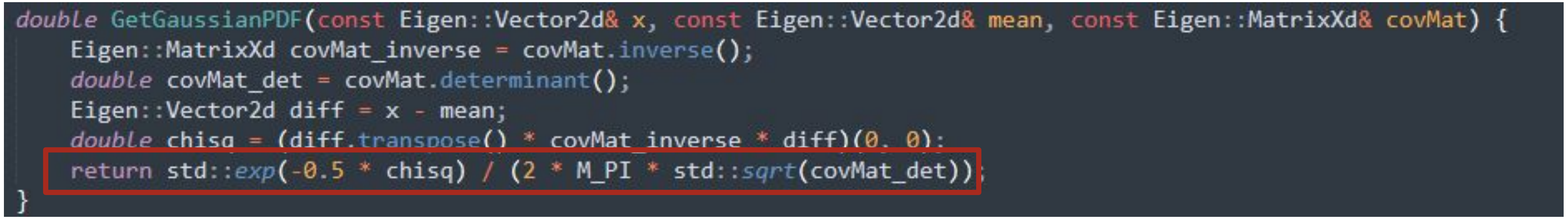
$(d_0, z_0, \boldsymbol{\sigma}\_d_0, \boldsymbol{\sigma}\_z_0, \boldsymbol{\sigma}\_d_0z_0)$

KDEgenerator

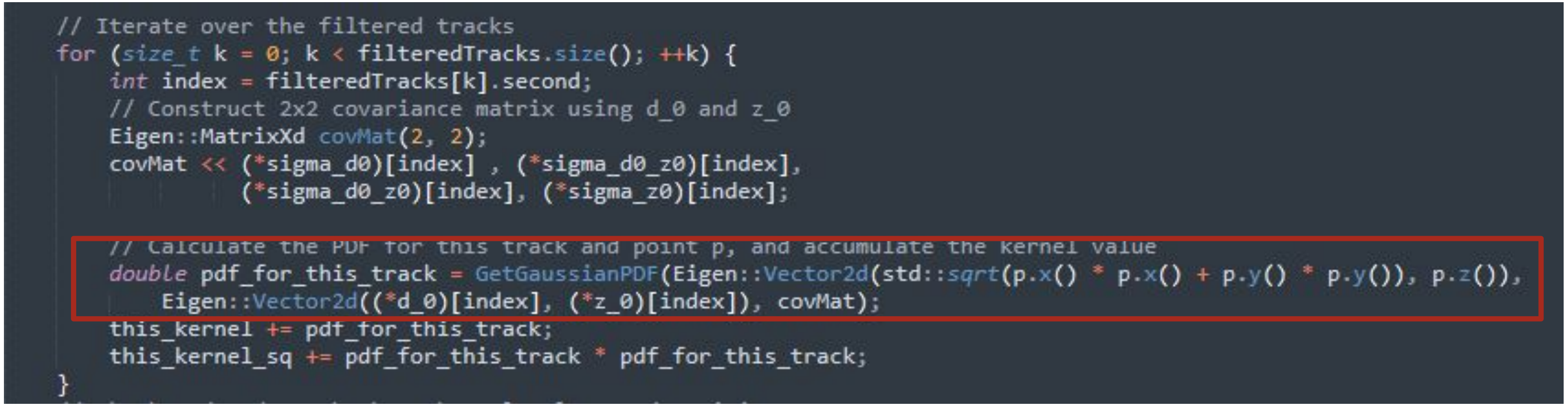
Constructor{ Initializing I/O ROOT files, TTrees, Branches, vectors, etc. }
Execute { sorting tracks, defining lambda functions for binning and pdf evaluation, filtering tracks, calculating covariance matrix, grid search, outputs best kernel value for that bin, etc }
Finalize { write data to output ROOT file, clean-up }

Run KDEgenerator on 3000 events
and generate 1-D binned histogram with 12,000 bins with z-range [-24,24] cm (40μm wide bins)

We encoded the PDFs calculation analytically



```cpp
double GetGaussianPDF(const Eigen::Vector2d& x, const Eigen::Vector2d& mean, const Eigen::MatrixXd& covMat) {
    Eigen::MatrixXd covMat_inverse = covMat.inverse();
    double covMat_det = covMat.determinant();
    Eigen::Vector2d diff = x - mean;
    double chisq = (diff.transpose() * covMat_inverse * diff)(0, 0);
    return std::exp(-0.5 * chisq) / (2 * M_PI * std::sqrt(covMat_det));
}
```

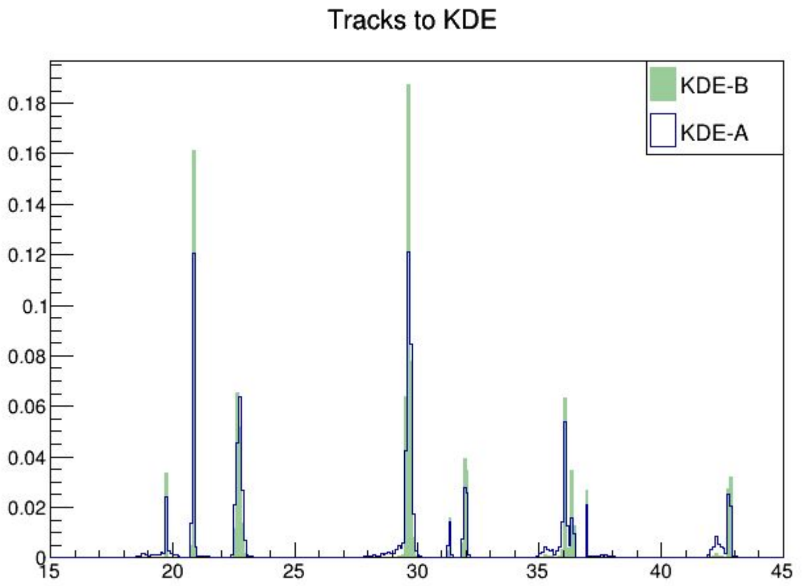


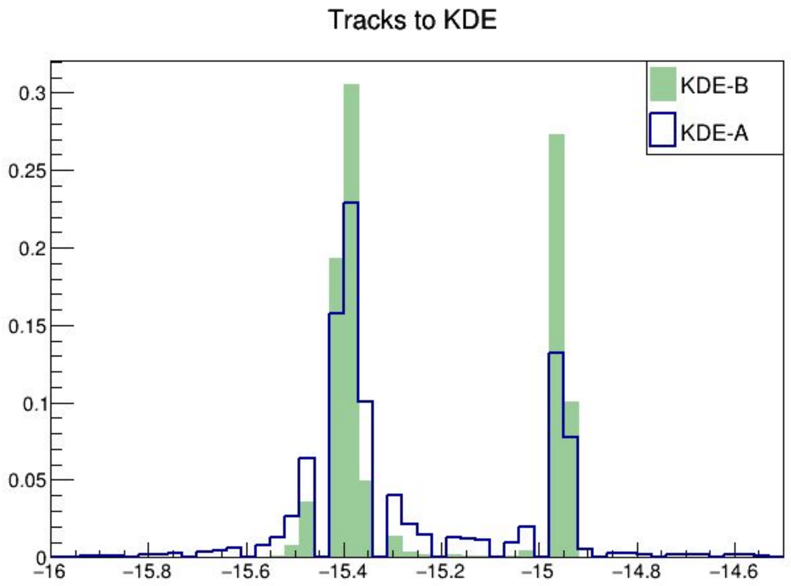
```cpp
    // Iterate over the filtered tracks
    for (size_t k = 0; k < filteredTracks.size(); ++k) {
        int index = filteredTracks[k].second;
        // Construct 2x2 covariance matrix using d_0 and z_0
        Eigen::MatrixXd covMat(2, 2);
        covMat << (*sigma_d0)[index] , (*sigma_d0_z0)[index],
                  (*sigma_d0_z0)[index], (*sigma_z0)[index];

        // Calculate the PDF for this track and point p, and accumulate the kernel value
        double pdf_for_this_track = GetGaussianPDF(Eigen::Vector2d(std::sqrt(p.x() * p.x() + p.y() * p.y()), p.z()),
            Eigen::Vector2d((*d_0)[index], (*z_0)[index]), covMat);
        this_kernel += pdf_for_this_track;
        this_kernel_sq += pdf_for_this_track * pdf_for_this_track;
    }
```

https://github.com/Layan-Sarayra/acts/blob/ACTS_Layan/Examples/Algorithms/Vertexing/src/KDEgenerator.cpp

Results from Analytical Approach

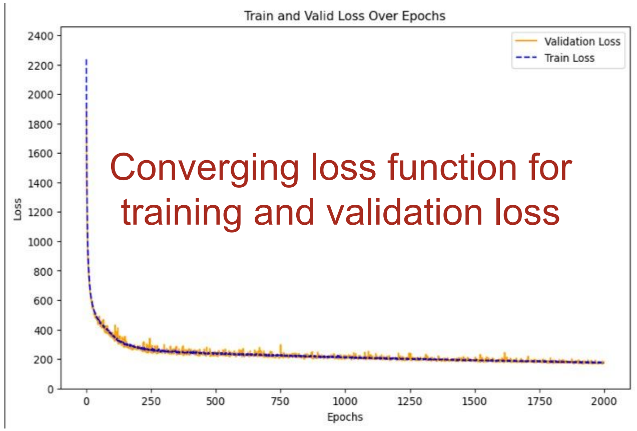
KDE-A represents the kernel value = pdf of this track
KDE-B represents the square of the kernel value = the squared value of pdf of this track





These KDE distributions will be fed into DNN to get PV predictions
A small z-range is shown to better observe the structure

NN Architecture for KDE



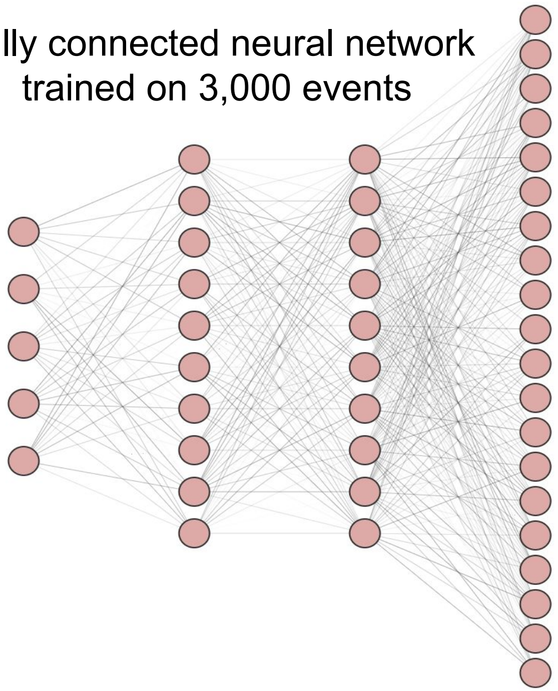
Converging loss function for training and validation loss

Fully connected neural network trained on 3,000 events

Track parameters and their uncertainties

+

Analytically calculated KDEs



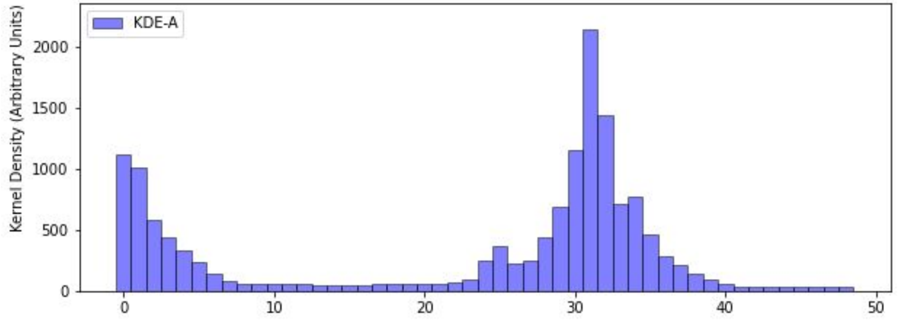
KDE predictions

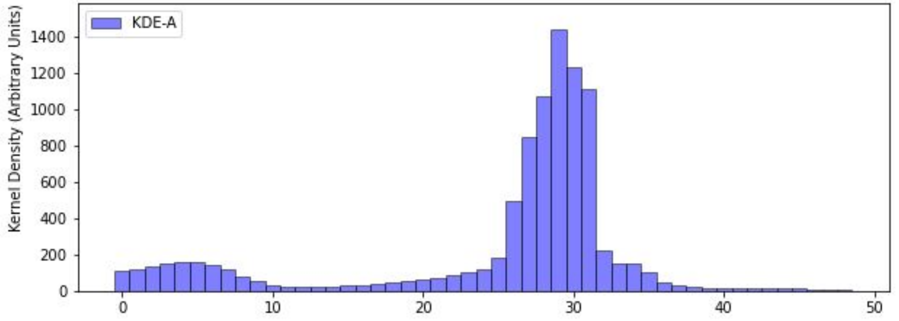
Results from NN Based-Approach

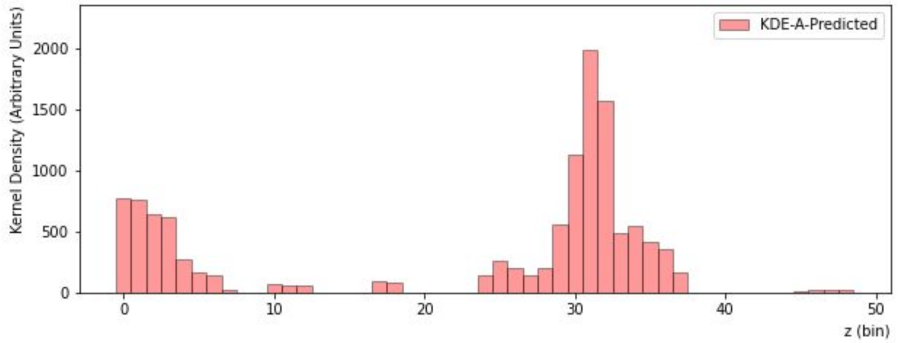
By Ananya Singha, HSF-India Fellow

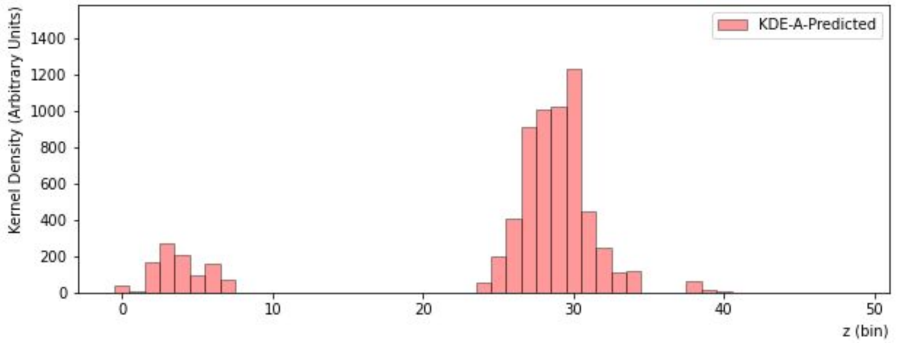
Analytically calculated KDE-A

NN predicted KDE-A









Conclusion and Future Work

- ❑ Completed the first step in the project; to build a DL-based PV Identification algorithm within ACTS
- ❑ Compared the results from analytical vs NN approach
- ❑ A drawback is our analytical method to produce KDEs takes large run-time

- ❖ Next step is to feed KDEs into DNN to get PV predictions
- ❖ Use of neural network for KDE generation (Ananya is working on it)
- ❖ We are Integrating this algorithm within ACTS to enable adoption by other experiments for broader uses

Thank you!

My gratitude to Rocky Bala Garg and Lauren Tompkins,

and to all the help and support from Bastian Schlag and Rory O’Dwyer

Questions?