Comparing and improving hybrid deep learning algorithms for identifying and locating primary vertices

S Akar[1], **M Peters**[1], **H Schreiner**[2], **M D Sokoloff**[1], **W Tepe**[1]

[1] University of Cincinnati, Cincinnati, OH 45221, USA
[2] Princeton University, Princeton, NJ 08544, USA

E-mail: simon.akar@cern.ch

Abstract. Using deep neural networks to identify and locate proton-proton collision points, or primary vertices, in LHCb has been studied for several years. Preliminary results demonstrated the ability for a hybrid deep learning algorithm to achieve similar or better physics performances compared to standard heuristic approaches. The previously studied architectures relied directly on hand-calculated Kernel Density Estimators (KDEs) as input features. Calculating these KDEs was slow, making use of the DNN inference engines in the experiment's real-time analysis (trigger) system problematic. Here we present recent results from a high-performance hybrid deep learning algorithm that uses track parameters as input features rather than KDEs, opening the path to deployment in the real-time trigger system.

1. Introduction

The LHCb experiment was recently upgraded for Run 3 of the LHC. It will record proton-proton collision data at five times the instantaneous luminosity of Run 2. The average number of visible primary vertices (PVs), proton-proton collisions in the detector closest to the beam-crossing region, will increase from 1.1 to 5.6. The experiment will move to a pure software data ingestion and trigger system, eliminating the Level 0 hardware trigger altogether [1]. A conventional PV finding algorithm [2, 3] that satisfies all requirements defined in the Trigger Technical Design Report [4] serves as the baseline. In parallel, we have been developing a hybrid machine learning algorithm, designed to run in the initial stage of the LHCb upgrade trigger.

The initial algorithm defined a one-dimensional Kernel Density Estimator (KDE) histogram, plus two more one-dimensional histograms, to describe the probabilities of tracks traversing small voxels in space. These feature sets were modified to use two KDEs rather than one. [5–7]. "Classic" convolutional Neural Networks (CNNs), referred to as `KDE-to-hists` models, produce one-dimensional histograms that nominally predicts Gaussian peaks at the locations of true PVs using three (or four) input histograms as their feature sets. A hand-written clustering algorithm identifies the candidate PVs and their positions. A first set of results used a "toy Monte Carlo" with proto-tracking [5]. Using track parameters produced by the LHCb Run 3 Vertex Locator (VELO) tracking algorithm [8] leads to significantly better performance [6].

The original KDE [6] is a projection of a three-dimensional probability distribution in voxels that has contributions *only* when two tracks pass close to each other. Calculating this KDE exactly is very time-consuming. One of the goals of our project is to predict PV positions directly from tracks' parameters. A proof-of-concept was presented last year [7]. We trained a model, `tracks-to-KDE`, predicting the KDE using the tracks parameters as input and merged

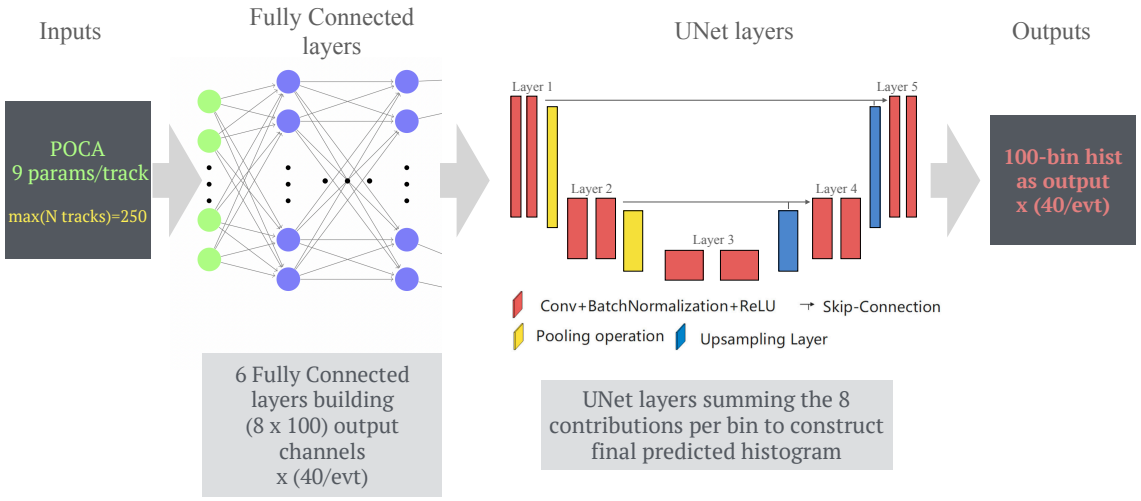

Figure 1. This diagram illustrates the end-to-end, `tracks-to-hist`, deep neural network used to predict an event's target histogram from its tracks' `poca-ellipsoid`s. Each event is now sliced in 40 independent 100-bin histograms. Six fully connected layers populate 8 100-bin channels in the last of these layers, 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.

it with a trained version of a `KDE-to-hists` model. Mediocre performance, compared to the original `KDE-to-hists` model, was achieved after a final training allowing all weights of the merged `tracks-to-hists` model to float.

In the last PV-finder update [6], we also presented studies of alternative `KDE-to-hists` model architectures. In particular, we showed that the capacity of our original CNN architecture [5] to learn increased with the number of parameters, as expected from first principles. The study of a modified version of the popular U-Net architecture [9], showed similar performances compared to the best classic CNN architectures, and it trained more quickly. Here we present improved results from an updated architecture for `tracks-to-hists`. A brief description of the model in given in Sec. 2. Performance is discussed in Sec. 3. A summary is presented in Sec. 4

2. Model architecture and training strategy

Building on our previous experience, we define a merged `tracks-to-hists` model; this architecture is based on the one used in the proof-of-concept presented in the last update [7]. The latest `tracks-to-hists` model, whose architecture is shown in Fig. 1, includes a few updates: the `tarcks-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 sets is merged to the trained `KDE-to-hists` 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 original approach, the one-dimensional feature sets 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. In place of representing each event by a single 4000-bin histogram (feature set), we now slice each event into 40 slices of 100 bins each. 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. Also, with an average of ~ 5 PVs per event, most of the bins in both the KDE and target histograms have no significant activity. The

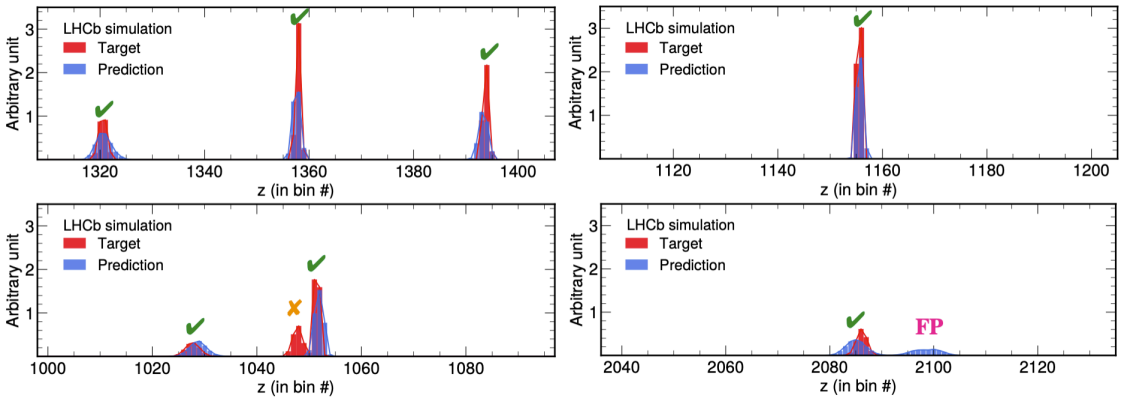

Figure 2. Plots illustrating typical examples of target (shown in red) versus predicted (shown in blue) histogram. Each of the four plots is drawn from different slices of different individual events. Each of the 100 bins in the histogram corresponds to a range in the z-direction of 100 μm. Predicted PVs that are matched to the correct true PV (majority of cases) are identified by a green tick in the plots. Example of missed true PV (orange cross label) and false positive prediction (FP label) are shown in the bottom-left and bottom-right plots, respectively.

40 slices of 100 bins are independent and homogeneous between events. Each slice is treated independently, after which the predicted 4000-bin histogram is stitched back together.

Training on nVidia RTX 2080Ti GPUs is performed in several steps. First, weights for the `tracks-to-KDE` model that uses individual tracks parameters evaluated at their points of closest approach (poca-ellipsoid) to the beamline [7] as input features to predict the KDE are determined. Then, the U-Net `KDE-to-hists` model is trained using the hand-calculated KDEs as the input feature set. Finally all weights and biases in the combined `tracks-to-hists` model are allowed to float. An asymmetry parameter between the cost of overestimating contributions to the target histograms and underestimating them [5] is used as a hyperparameter to allow higher efficiency by incurring higher false positive rates.

3. Performances

Evaluation Performance evaluation is obtained by a matching procedure done by a heuristic algorithm, based on the PV positions along the beam axis, z. Figure 2 shows typical examples of predicted and target histograms using the final `tracks-to-hists` model. In the previous report of PV-finder performances [5–7] a predicted PV was matched if the distance, Δz, between it's position, z_{pred}, and the true PV position, z_{true}, satisfied $\Delta z = |z_{\text{pred}} - z_{\text{true}}| \leq 0.5\,\text{mm}$. The false positive rate is obtained from the ratio of remaining predicted PVs after the matching procedure over the total number of true PVs. This approach suffered a few limitations, independent of the intrinsic DNN algorithm performances. For instance, it is known that a PVs resolution depends on the number of tracks originating from it. In case of low-multiplicity PVs, the predicted histogram from the DNN algorithm, similar in shape to the target histogram, can be displaced more than 0.5 mm from the true PV position. This scenario can simultaneously "produce" a missed PV (reduced efficiency) and a false positive signal (increased false positive rate). To address this issue, a new matching procedure is used. Instead of a fixed window around the predicted PV position, a resolution function is introduced in the matching procedure. The resolution for a given predicted PV, $\sigma(z)$, is a function of the standard deviation of the predicted histogram multiplied by a constant parameter that can be tuned. Using this definition, a predicted PV is now matched if $\Delta z = |z_{\text{pred}} - z_{\text{true}}| \leq 5\sigma(z)$. In Fig. 2 we observe that for matched PVs, the standard deviation of the predicted peaks (blue entries) is slightly larger than

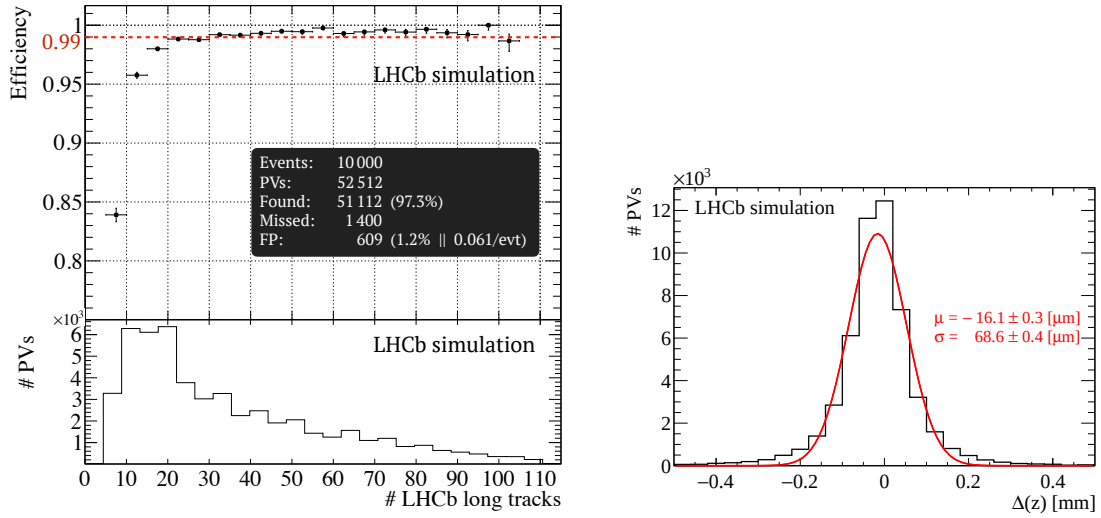

Figure 3. (left) Efficiency of finding true PVs as a function of the number of fully reconstructed originating tracks. The integrated efficiency and false positive rate are also reported. (right) Distribution of the distance, Δz, between the predicted and true PV position for matched PVs as defined in the text, overlaid by a fitted Gaussian.

that of the corresponding true PVs (red entries), estimated from the number of tracks. Peaks with larger standard deviations are typically associated with PVs with fewer tracks.

Predicted position and efficiency The reported results come from statistically independent validation samples. Using the matching procedure described above, we quantified the performances of the trained `tracks-to-hists` DNN on a sample of 10K events corresponding to slightly more than 50K true PVs. Figure 3 shows the efficiency of finding true PVs as a function of the number of fully reconstructed originating tracks, labelled LHCb long tracks, defined as tracks with hits in all the elements of the LHCb tracking system. We observe that the efficiency is very high ($> 99\%$) and stable for PVs with more than 20 tracks, and rapidly decreasing for PVs with fewer tracks. Also shown in Fig. 3 is the distribution of the distance, Δz, between the predicted and true PV position for matched PVs. From a fit using a Gaussian distribution, we observe a small bias of $-16.1\,\mu$m on the predicted PV position. Most notably, the majority of matched PVs have a Δz value below the bin width of the target histogram of 0.1 mm.

Performances evolution Figure 4 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-hists` model described at ACAT-2019 [5]. The green squares show the performances of a `KDE-to-hists` described at Connecting-the-Dots in 2020 [6]. Both of the above models were trained using "toy Monte Carlo" with proto-tracking. All subsequent DNN models were trained using full VELO tracking algorithm [8], 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-hists` models using either a classical CNN architecture or a U-Net model described at CHEP-2021 [7]. The performances of all above models were obtained using the "old" matching procedure with a fixed searching window of 0.5 mm. The magenta diamonds show the performance of the `tracks-to-hist` model described in Sec. 2. These performances are obtained using the improved matching procedure described above. The

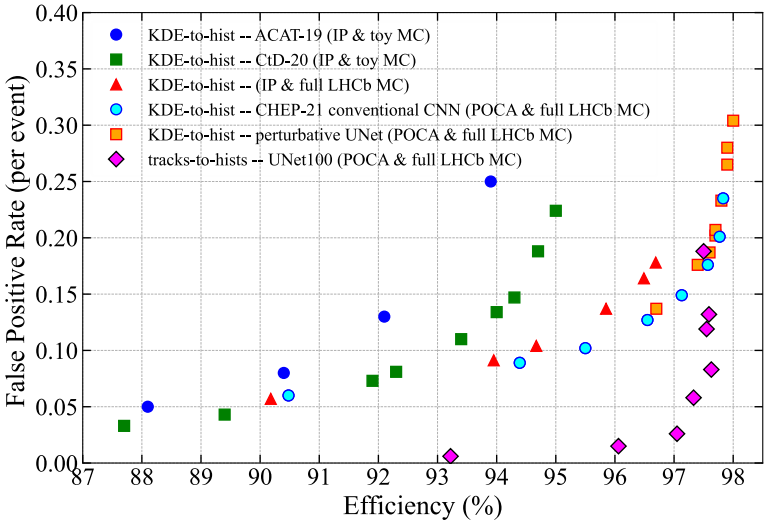

Figure 4. Comparison between the performances of models reported in previous years and the newest model (magenta diamonds). An asymmetry parameter described in the text is varied to produce the families of points observed.

performance of 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).

4. Summary and Conclusions

Since we presented results at CHEP 2021, we have updated the `tracks-to-hists` model to produce the first competitive end-to-end hybrid deep learning algorithms for identifying and locating primary vertices based solely on tracks' parameters as input features. The performance of this model reaches very high efficiencies with low false positive rates. By removing the (very slow) KDE calculation used as part of the hybrid `KDE-to-hists` models, we have opened the path to include a PV finding DNN in the real-time trigger system of the experiment. Studies are currently ongoing to validate the feasibility of including a version of the `tracks-to-hists` model to execute in the CUDA cores (or in the tensor cores) of the GPUs constituting the LHCb trigger system. We also plan on studying the effect of quantization on the model performances, as well as the size of the model itself (number of nodes). Our specific algorithms are being designed for use in LHCb with its Run 3 detector. Developing and deploying machine learning inference engines that are highly performant and satisfy computing system constraints requires sustained effort. The results reported here should encourage work focused on using DNNs for identifying vertices in other high energy physics experiments as well.

5. Acknowledgments

The authors thank the LHCb computing and simulation teams for their support and for producing the simulated LHCb samples used in this paper. The authors also thank the full LHCb Real Time Analysis team, especially the developers of the VELO tracking algorithm [8] used to generate the "full LHCb MC" results presented in Fig 4.

This work was supported, by the U.S. National Science Foundation under Cooperative Agreement OAC-1836650 and awards PHY-1806260, and OAC-1450319.

References

[1] Aaij R *et al.* (LHCb) 2014 LHCb Trigger and Online Upgrade Technical Design Report URL https://cds.cern.ch/record/1701361

[2] Reiss F 2020 Excerpts from the LHCb cookbook — RECEPTS for testing lepton universality and reconstructing primary vertices presented 18 12 2020 URL https://cds.cern.ch/record/2749592

[3] Reiss F *et al.* (LHCb) 2020 Fast parallel Primary Vertex reconstruction for the LHCb Upgrade talk presented at Connecting-the-Dots 2020 URL https://indico.cern.ch/event/831165/timetable/?view=standard

[4] Aaij R *et al.* (LHCb) 2019 *JINST* **14** P04013 (*Preprint* 1812.10790)

[5] Fang R, Schreiner H F, Sokoloff M D, Weisser C and Williams M 2020 *J. Phys. Conf. Ser.* **1525** 012079 (*Preprint* 1906.08306)

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

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

[8] Hennequin A, Couturier B, Gligorov V, Ponce S, Quagliani R and Lacassagne L 2020 *JINST* **15** P06018 (*Preprint* 1912.09901)

[9] Ronneberger O, Fischer P and Brox T 2015 *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* ed Navab N, Hornegger J, Wells W M and Frangi A F (Cham: Springer International Publishing) pp 234–241 ISBN 978-3-319-24574-4 (*Preprint* 1505.04597)