PROCEEDINGS OF SCIENCE

Applying and optimizing the Exa.TrkX Pipeline on the OpenDataDetector with ACTS

Paolo Calafiura,[a] **Lukas Heinrich,**[b] **Benjamin Huth,**[c,*] **Xiangyang Ju,**[a] **Alina Lazar,**[d] **Daniel Murnane,**[a] **Andreas Salzburger**[e] **and Tilo Wettig**[c]

[a] *Lawrence Berkeley National Laboratory, 1 Cyclotron Rd, Berkeley, CA 94720, USA*
[b] *Technische Universität München, Arcisstraße 21, 80333 München, Germany*
[c] *Universität Regensburg, Universitätsstraße 31, 93053 Regensburg, Germany*
[d] *Youngstown State University, One University Plaza, Youngstown, OH 44555, USA*
[e] *CERN, Espl. des Particules 1, 1211 Meyrin, Switzerland*

E-mail: benjamin.huth@ur.de

Machine learning is a promising field to augment and potentially replace part of the event reconstruction of high-energy physics experiments. This is partly due to the fact that many machine-learning algorithms offer relatively easy portability to heterogeneous hardware and thus could play an important role in controlling the computing budget of future experiments. In addition, the capability of machine-learning-based approaches to tackle nonlinear problems can improve performance. Particularly, the track reconstruction problem has been addressed in the past with several machine-learning-based attempts, largely facilitated by the two highly resonant machine-learning challenges (TrackML). The Exa.TrkX project has developed a track-finding pipeline based on graph neural networks that has shown good performance when applied to the TrackML detector. We present the technical integration of the Exa.TrkX pipeline into the framework of the ACTS (A Common Tracking Software) project. We further present our efforts to apply the pipeline to the OpenDataDetector, a model of a more realistic detector that supersedes the TrackML detector. The tracking performance in this setup is compared to that of the ACTS standard track finder, the Combinatorial Kalman Filter.

41st International Conference on High Energy Physics - ICHEP2022
6-13 July, 2022
Bologna, Italy

*Speaker

PoS(ICHEP2022)227

1. Introduction

In order to cope with the challenges of the upcoming high-luminosity upgrade of the LHC (HL-LHC), much research is being done on new track-reconstruction algorithms based on novel methods like machine learning. To help with its transition from development to physics testing and optimization, the *Exa.TrkX* pipeline [1] was implemented in the tracking-software package *ACTS* [2] and used in a realistic virtual detector, the *OpenDataDetector* (ODD) [3]

2. The Exa.TrkX pipeline

The *Exa.TrkX* project has developed a multi-step machine-learning-based algorithm for track finding. It builds a graph whose nodes are hits in the 3D space and uses a *graph neural network* (GNN) to label the edges such that connected hits correspond to a track:

1. Graph building: A fully-connected graph is not manageable for O(100K) nodes. Therefore, a *metric-learning* approach in combination with a fixed-radius nearest-neighbor search is used to create the initial graph.
2. Graph filtering: Because the initial graph is still too large to be processed by a GNN on the available hardware, the graph size is further reduced by an edge-classifying dense network.
3. Edge classification: The final edge scoring is done by a GNN. There exist several GNN architectures. For this work, an *Interaction GNN* was used.
4. Track building: As a final step, a weakly-connected-components algorithm forms track candidates from the edge scores.

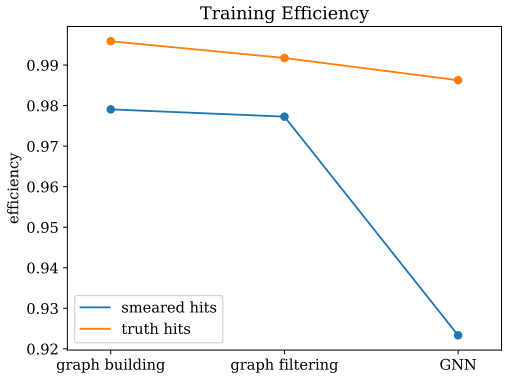
3. ACTS and the OpenDataDetector

ACTS is a software toolkit that aims at providing both production-ready implementations of standard tracking algorithms like the *Kalman Filter* (KF) or the *Combinatorial Kalman Filter* (CKF) and being a highly flexible platform for R&D. ACTS therefore allows for assembling tracking chains from stable and tested standard algorithms and experimental components like the Exa.TrkX pipeline and also provides standard tools to evaluate and compare tracking performance.

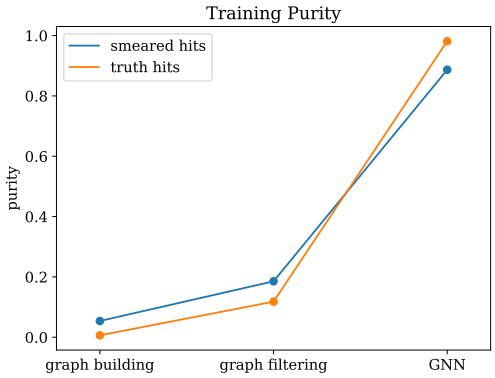
The *OpenDataDetector* is a realistic model of a cylindrical tracking detector. Originating from the detector used for the TrackML challenge, it was enhanced by an increasingly realistic description of detector components and material. However, a realistic geometric digitization model is not yet available for the ODD. Therefore we rely on a more straightforward approach: A 2D measurement

detector part	distance from beam pipe	σ_x	σ_y
pixel	$r < 20\,\text{cm}$	$10.0\,\mu\text{m}$	$10.0\,\mu\text{m}$
short strip	$20\,\text{cm} < r < 70\,\text{cm}$	$45.0\,\mu\text{m}$	$1.2\,\text{mm}$
long strip	$r > 70\,\text{cm}$	$60.0\,\mu\text{m}$	$3.6\,\text{mm}$

Table 1: The uncertainties assumed for the measurements in the different regions of the OpenDataDetector.

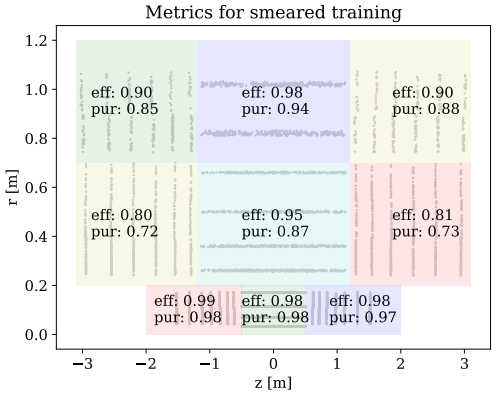


(a) Efficiency for true and smeared hits in the three steps.

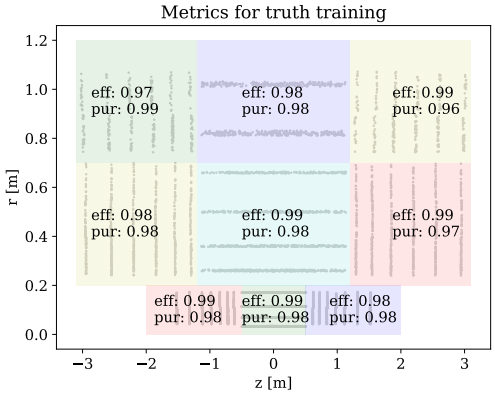


(b) Purity for true and smeared hits in the three steps.

Figure 1: Efficiency and purity of the training phases of steps 1-3 (see Sec. 2) for one selected event.



(a) Training with smeared hits.



(b) Training with true hits.

Figure 2: Efficiency and purity in different detector regions after the GNN step for one selected event.

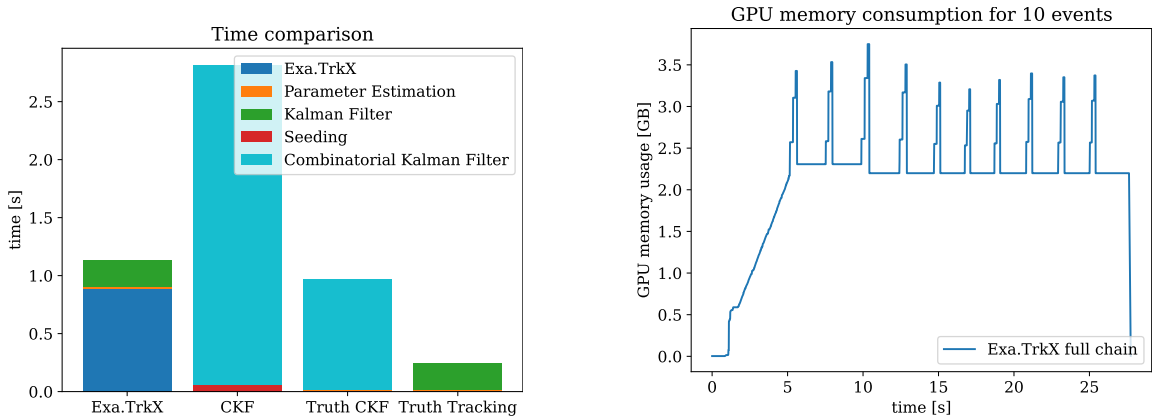
on a detector module is smeared with a two-dimensional Gaussian distribution $\mathcal{N}(0,\sigma)$, where the covariance matrix σ reflects the properties of the module (see Tab. 1).

To use the CKF as a benchmark for track finding performance, we tuned a few critical parameters to the ODD using a hyperparameter optimization framework [4]. This did not lead to cutting-edge performance but provided a baseline for track-finding performance.

4. Training the Pipeline

The Exa.TrkX training pipeline is implemented using *pytorch* [5]. The training code was forked from Exa.TrkX and modified to be compatible with ACTS. It can be found in [6].

The training data consist of 1000 events generated using *Pythia8* [7] and simulated with the ACTS fast simulation, emulating the conditions foreseen at HL-LHC. 950 events are used for training and 25 each for testing and validation. Hyperparameters and scripts that steer the training can be found in [8]. An NVIDIA A100 GPU with 40 GB of memory was used for the training.



(a) Average runtime of the different full chains.

(b) GPU memory usage profiled with *nvidia-smi*.

Figure 3: Computational performance for ten events with particle selection applied.

The pipeline was trained twice with different input data: Once with the hits given directly by the simulation (referred to as *true hits* later on) and once with the hits after applying the simplified digitization model described in Sec. 3 (referred to as *smeared hits* later on). A training run takes about 18 h and uses up to 90% of the available GPU memory.

To qualify the performance of the training, we define two edge-based metrics:

$$\text{efficiency} = \frac{\text{\# true edges in graph}}{\text{\# all true edges}} \qquad \text{purity} = \frac{\text{\# true edges in graph}}{\text{\# all edges in graph}}$$

As shown in Fig. 1, we obtain about 99% efficiency and purity with the true hit data but lower performance with the more realistic smeared ones. Especially the efficiency of the GNN step drops significantly. As shown in Fig. 2, this is mainly caused by the outer parts of the detector with lower measurement resolution. We are currently investigating how to improve the performance here.

5. Inference within ACTS

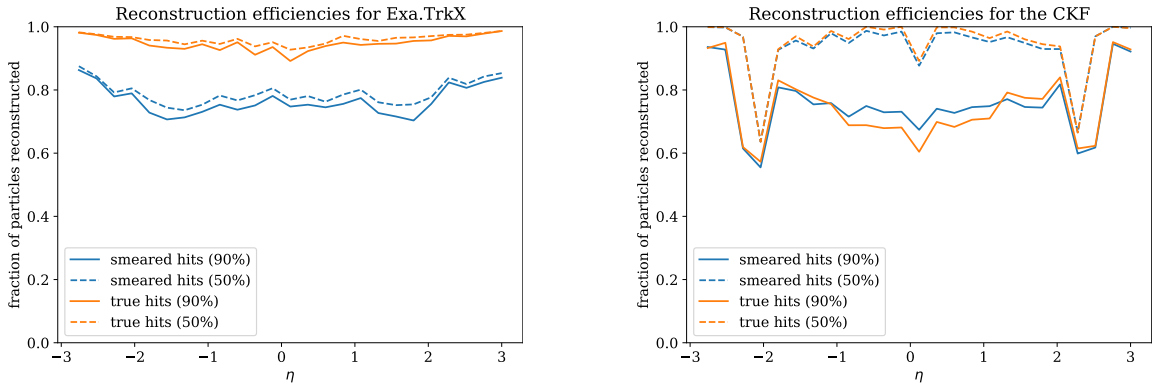
For the inference, a selection has been applied to cover particles of interest: we only consider particles with energy above 500 MeV, pseudo rapidity $|\eta| < 3$, and a transverse distance from the beam pipe $\rho < 2$ mm.

The Exa.TrkX pipeline in ACTS is implemented with the *torchscript* backend with the help of *FRNN* [9] and *Boost.Graph* [10]. The inference was run on the same system as the training (GPU: NVIDIA A100, CPU: AMD EPYC 7662). There is also a standalone implementation of the Exa.TrkX pipeline in C++ available [11].

5.1 Computing performance

We have assembled four different full chains to compare their performance characteristics:

1. GNN-based tracking chain (*Exa.TrkX + parameter estimation + KF*).
2. State-of-the-art full chain (*Seeding + CKF*).
3. *Truth-seeded CKF*, a best-case scenario for the CKF.



(a) Efficiencies of the Exa.TrkX-based full chain.

(b) Efficiencies of the CKF-based full chain.

Figure 4: Comparison of the reconstruction efficiencies of the Exa.TrkX pipeline and the CKF for ten events with particle selection applied. This is shown for two different matching criteria: one that requires 90% of a track's hits to have been found correctly to count as reconstructed, and one only requiring 50%.

4. General baseline with *Truth track finding + KF*.

The runtime of these four chains is summarized in Fig. 3a. Note that the Exa.TrkX pipeline can exploit a whole GPU, whereas all other modules run single-threaded on a CPU. On the other hand, the CPU algorithms are extensively optimized for performance, whereas optimization efforts for the Exa.TrkX pipeline have just started. A final performance evaluation must also include monetary and energy consumption aspects, however, a similar per-event timing is already promising.

There are two contributions to the observed GPU memory consumption (see Fig. 3b): The baseline memory usage of the torchscript runtime (roughly 2 GB) and the memory usage of the algorithm itself. It is expected that optimizations of the models and the ACTS implementation of the pipeline can reduce the memory footprint.

5.2 Track finding performance

The *reconstruction efficiency* (the fraction of tracks that was reconstructed by a track-finding algorithm) for both the truth-based and the smeared models are shown in Fig. 4a. The performance in the more realistic scenario with the smeared input and the respective models is significantly lower than in the truth-based case, which reaches almost optimal efficiency. This is in line with the observed metrics of the training (see Fig. 1) and indicates that the Exa.TrkX pipeline performance is sensitive to the detector resolution. Related work has shown that better performance can be reached in similar cases [12]. Therefore, we assume these issues can be solved by further optimization.

Even though the CKF does not show state-of-the-art performance in our setup, its performance does not depend strongly on the input type (see Fig. 4b). This is expected because the CKF performs a least-square estimate and thus can take into account the measurement uncertainty.

6. Conclusion

We successfully trained the Exa.TrkX pipeline and were able to run a full track-reconstruction chain within ACTS consisting of both traditional algorithms like the KF and novel modules based

on GNNs. This chain can reach good track-reconstruction efficiency in general, but we observe a performance decrease in detector regions with low measurement resolution that is currently under investigation. In addition, our setup still uses several simplified components, especially with respect to the simulation and the digitization model. We plan to improve upon these aspects in future work.

Note that we only investigated one realization of a GNN-based tracking chain. There are many developments in this field regarding alternative graph-building approaches, different GNN architectures, track-building methods, or even hybrid algorithms combining, e.g., classical algorithms like the CKF and modern GNN-based solutions. Therefore, this work can also serve as a baseline for future research on this topic within ACTS.

Acknowledgment

LH is supported by the Excellence Cluster ORIGINS, funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC-2094-390783311. GPU resources were made available by the ORIGINS Data Science Lab (ODSL).

This research was partly supported by the U.S. Department of Energy's Office of Science, Office of High Energy Physics, under Contract No. DE-AC02-05CH11231 (CompHEP Exa.TrkX); and by the National Science Foundation under Cooperative Agreement OAC-1836650.

References

[1] X. Ju et al., *Performance of a geometric deep learning pipeline for HL-LHC particle tracking*, *The European Physical Journal C* **81** (2021) 876.

[2] X. Ai et al., *A Common Tracking Software Project*, *Computing and Software for Big Science* **6** (2022) 8.

[3] C. Allaire et al., *OpenDataDetector*, *Zenodo* (2022) .

[4] R. Bala et al., *Exploration of different parameter optimization algorithms within the context of ACTS software framework*, 2022.

[5] A. Paszke et al., *Pytorch: An imperative style, high-performance deep learning library*, arXiv:2203.11601.

[6] B. Huth, *Forked Exa.TrkX repository with the modified training pipeline*, 2022.

[7] C. Bierlich et al., *A comprehensive guide to the physics and usage of PYTHIA 8.3*, arXiv:2203.11601.

[8] B. Huth, *Training and inference scripts*, 2022.

[9] L. Xue, *FRNN library*, 2022.

[10] J. Siek et al., *Boost.Graph library*, 2001.

[11] X. Ju, *Standalone C++ implementation of the Exa.TrkX pipeline*, 2022.

[12] ATLAS collaboration, *GNN tracking performance for ATLAS ITk simulated data*, 2022.