Journal of Physics: Conference Series

PAPER • **OPEN ACCESS**

The HEP.TrkX Project: Deep Learning for Particle Tracking

To cite this article: Aristeidis Tsaris *et al* 2018 *J. Phys.: Conf. Ser.* **1085** 042023

IOP Conf. Series: Journal of Physics: Conf. Series **1085** (2018) 042023 doi:10.1088/1742-6596/1085/4/042023

The HEP.TrkX Project: Deep Learning for Particle Tracking

Aristeidis **Tsaris**[1,a], Dustin **Anderson**[3], Josh **Bendavid**[3], Paolo **Calafiura**[2], Giuseppe **Cerati**[1], Julien **Esseiva**[2,4], Steven **Farrell**[2], Lindsey **Gray**[1], Keshav **Kapoor**[1], Jim **Kowalkowski**[1], Mayur **Mudigonda**[2], **Prabhat**[2], Panagiotis **Spentzouris**[1], Maria **Spiropoulou**[3], Jean-Roch **Vlimant**[3], Stephan **Zheng**[3], Daniel **Zurawski**[1]

[1] Fermi National Accelerator Laboratory, Batavia, Illinois, United States of America
[2] Lawrence Berkeley National Laboratory, Berkeley, California, United States of America
[3] California Institute of Technology, Pasadena, California, United States of America
[4] Haute école spécialisée de Suisse occidentale, Fribourg, Switzerland

E-mail: [a]atsaris@fnal.gov

Abstract. Charged particle reconstruction in dense environments, such as the detectors of the High Luminosity Large Hadron Collider (HL-LHC) is a challenging pattern recognition problem. Traditional tracking algorithms, such as the combinatorial Kalman Filter, have been used with great success in HEP experiments for years. However, these state-of-the-art techniques are inherently sequential and scale quadratically or worse with increased detector occupancy. The HEP.TrkX project is a pilot project with the aim to identify and develop cross-experiment solutions based on machine learning algorithms for track reconstruction. Machine learning algorithms bring a lot of potential to this problem thanks to their capability to model complex non-linear data dependencies, to learn effective representations of high-dimensional data through training, and to parallelize easily on high-throughput architectures such as FPGAs or GPUs. In this paper we present the evolution and performance of our recurrent (LSTM) and convolutional neural networks moving from basic 2D models to more complex models and the challenges of scaling up to realistic dimensionality/sparsity.

1. Introduction

The Large Hadron Collider (LHC) [1] is the highest energy collider ever constructed. The goal of the LHC is to probe fundamental physics questions, such as the basic building blocks of matter and their interactions. Two counter-circulating proton beams collide in the center of building-sized detectors, such as ATLAS [2] and CMS [3] and by measuring the energy and momentum of the escaping particles, fundamental laws of physics can be tested. The reconstruction of the trajectories of charged particles plays a key role in identifying particles and measuring their charge and momentum. A simplified tracking work-flow consists of track seeding, track building and track fitting. Typically, a Combinatorial Kalman Filter (CKF) algorithm is being used in both track fitting and track building with the latter to be the most CPU intense process in the reconstruction work-flow.

The High-Luminosity LHC era (HL-LHC) will bring significant challenges to particle tracking. With increasing the number of interactions per beam crossing ("pile up") the detector occupancy

will also increase. Traditional tracking algorithms scale quadratically or worse [4] with detector occupancy and assuming a stable improvement in the CPU performance the next years, tracking algorithms will need to run faster and in parallel.

The goal of the HEP.TrkX project is to apply advanced machine learning algorithms to the HL-LHC track reconstruction problem. Deep learning architectures have been explored for vertex and track finding, as well as track fitting reconstruction challenges. We prototyped and developed the algorithms using datasets produced with "toy" simulations, while the robustness of the algorithms was tested with more realistic datasets. Machine learning algorithms have shown a great ability to learn effective representations of high dimensional data through training in various datasets. They also scale linearly with input data size and computationally are very regular, which makes them ideal to run efficiently on data parallel architectures on an HPC environment.

2. Deep Learning Models for Tracking with Toy Datasets

Long Short Term Memory (LSTM) [5] networks are state of the art Recurrent Neural Networks (RNN) that have seen great success in sequence prediction. We can imagine the problem of track finding as a hit prediction task, where a track has known hits in the first three layers (seed) and the neural network predicts the hit location on the other layers. The input to each LSTM layer is a detector layer in the format of pixel array followed by a single fully connected layer (FC). The FC is applied separately to each LSTM output and produces a pixel prediction for the same detector layer.

Figure 1 shows a sketch of this model architecture along with the input and the model prediction. The 2D toy dataset used here was generated using straight lines constrained within the detector volume. The target track has pixel hits in the first three layers compared to the incomplete background tracks. This model architecture was adjusted and tested to realistic dataset as described in Section 4.

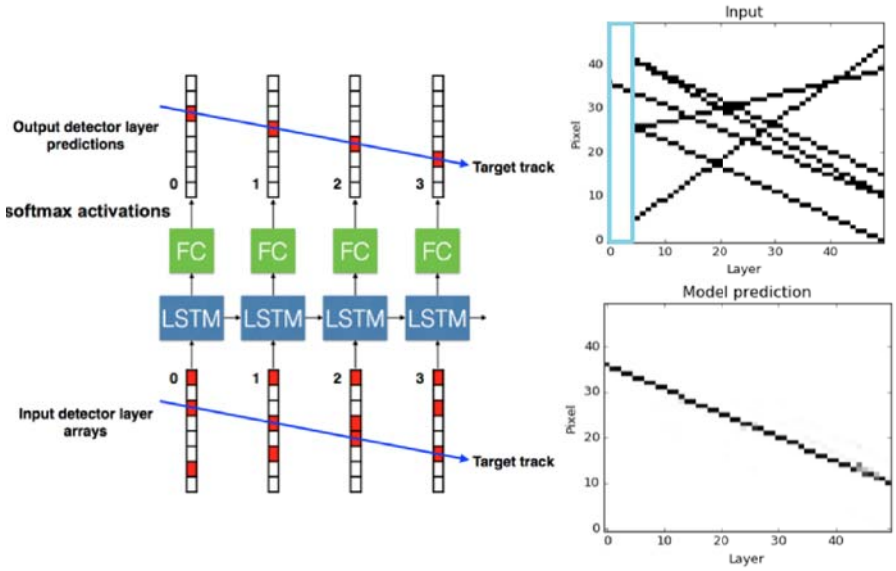

Figure 1. The plot on the left shows the LSTM model architecture [6] used to classify hits in one track. The plot on the right shows a 2D representation of the input and the output of the model. The cyan box in the upper right plot shows the seed of the target track in the first three layers.

S. Farrell et al. performed a detailed study of various model architectures and a comparison of the model predictions for 2D and 3D toy datasets [6]. The track finding problem was tested with bidirectional LSTM, deep-LSTM and next-layer LSTM (making prediction for the next detector layer). Also models that include Convolutional Neural Networks (CNNs) have been tested in toy datasets, either by modeling the track dynamics in a pixel level classification, or in an end-to-end approach where the model predicts the parameters of the track. The models that showed promising results will be tested against realistic datasets, as described in the next sections.

3. Hit Assignment to Tracks with TrackMLRamp Dataset

An alternative way to use RNN networks for track finding is to develop the model to assign hit to tracks. The input in this case is a set of hit positions (r, Φ and Z in this case) in the detector and the output is a matrix of probabilities of the hits assigned to tracks. In this case all input hits are fed in across all layers. The model architecture can be seen in the right plot of Figure 2 where three Bi-directional GRU (Gated Recurrent Units) layers were used with two dropout layers in between. GRU layers where used instead of LSTMs, since they appear to be trained faster for the shallow network architecture used.

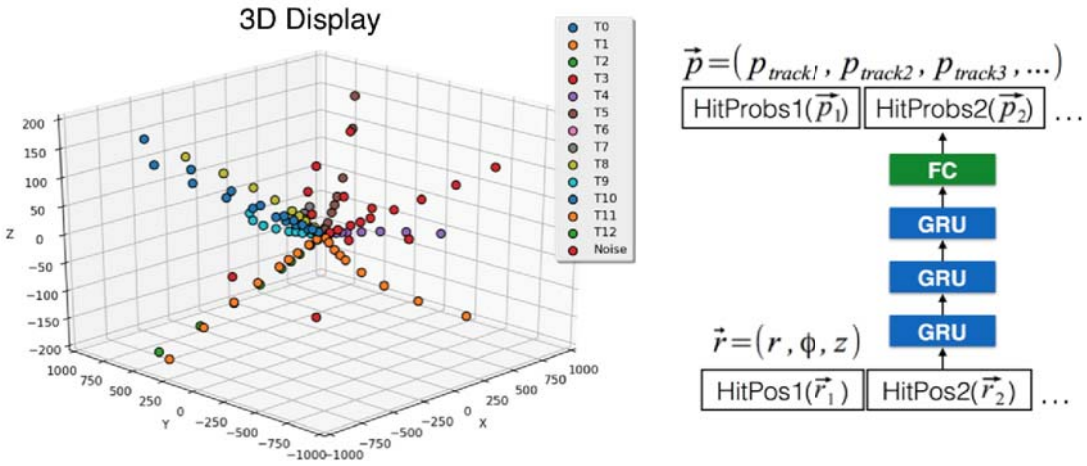

Figure 2. The plot on the left shows a 3D representation of the modified TrackMLRamp Dataset [7]. The plot on the right shows the GRU model used to assign hit to tracks.

A 3D dataset was generated to train this neural network based on the 2D dataset used in "TrackMLRamp hackathon" [7]. Events were generated with a constant magnetic field in perfectly circular detectors. A longitudinal dimension was added with respect to the beam, along with noise hits, distributed randomly across the layers. A 3D representation of an example event with 13 tracks can be seen in the left plot of Figure 2.

Figure 3 shows the accuracy as a function of the number of tracks, where the accuracy is defined as the fraction of hits in the event that is assigned to correct track. Events with one track have the lowest average accuracy, since with a flat noise rate across all events the ambiguity becomes larger compared to multi-track events. For events with higher number of tracks the accuracy seems relatively stable, and we clearly have not reached the point that the accuracy is saturated. To study this we will generate statistically larger dataset and possibly use a deeper network.

The overal hit assignment accuracy in the previous case was 87%, were detector hits assigned to only one track, the one with the highest probability. We also explored accepting multiple

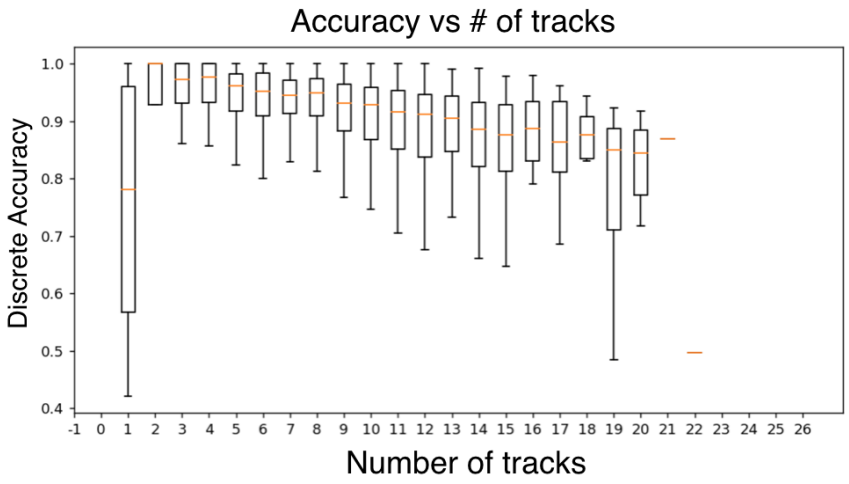

Figure 3. The plot shows the accuracy as a function of the number of tracks for the hit assignment GRU model.

hit assignments to tracks, for different threshold values of the prediction probabilities, in an effort to improve the accuracy. This study is shown in Figure 4 where the top left shows the probability of hit assignment to multiple tracks with at least a threshold value. With low enough threshold selection the probability is increased and the number of ambiguities appears not to be that high (right plot of Figure 4). For example for the particular dataset that is presented here, a threshold value of 0.7 would result a 10% more accurate hit assignment. We can still keep the events that are not covered with this threshold and resolve the ambiguities due to multiple hit assignment with another off-line algorithm downstream the reconstruction chain.

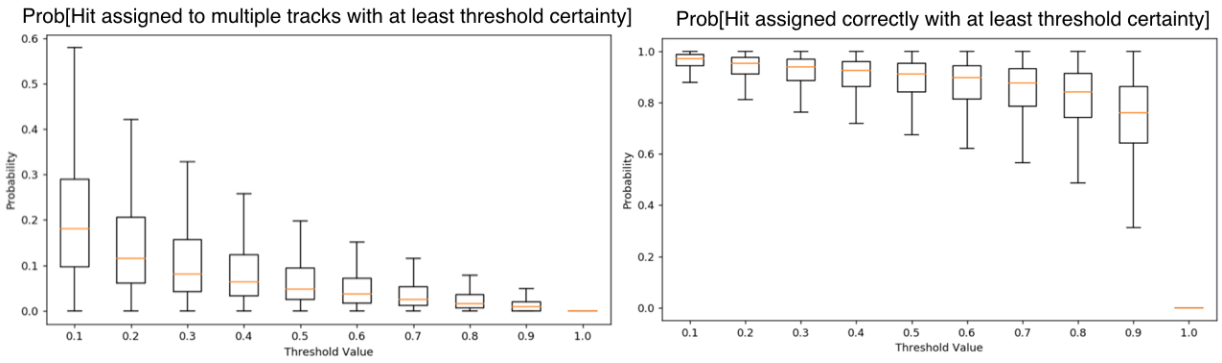

Figure 4. The plot on the left shows the probability of hit assignment to multiple tracks as a function of threshold value. The plot on the right shows the probability of correctly hit assigned to multi tracks as a function of threshold value.

4. Hit Classification and Vertex Finding with ACTS Dataset

After exploring various deep learning architectures in toy datasets, a set of those architectures is being tested against more realistic datasets simulated with A Common Tracking Software (ACTS) [8]. Deep learning algorithms described in the previous sections are tested on MC

samples from a generic LHC-like detector simulation. By increasing the complexity and realism of the dataset, we test the robustness of the tracking algorithms against different configurations. Figure 5 shows the layout of the detector geometry generated with the ACTS framework. Only hits in the barrel volume detectors are used while the hits in the end-cap detector are also available for future studies.

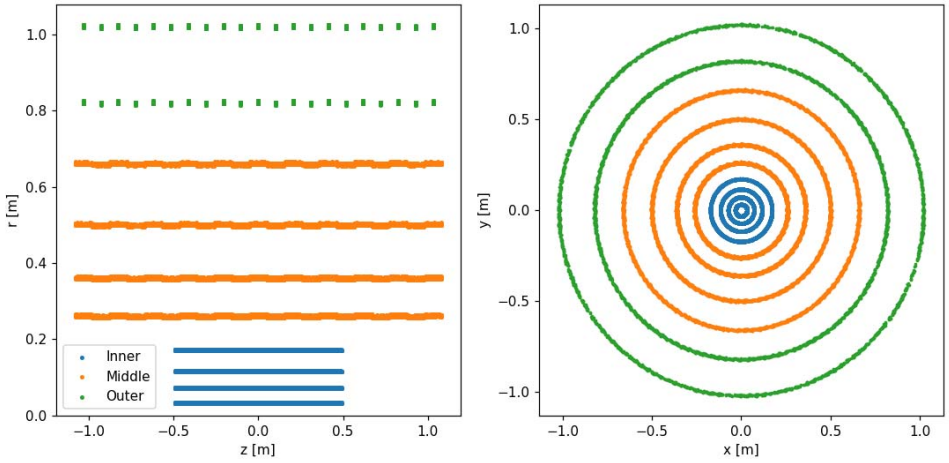

Figure 5. These scatter plots show the geometry of the ACTS dataset. Ten detector layers from three cylindrical barrel volumes are used. The left plot shows the longitudinal z coordinate vs. cylindrical radius r, while the right plot shows the x-y plane. Detector end-cap layers are not utilized and not shown.

Figure 6 shows a generated example event used by the LSTM hit classification model. The left and middle plots are projected 3D histograms which show the input data and the ground truth used to train the model. The gray color scale is setup so that dark represents the maximum number of hits. The hit classification model was used in this dataset, as described in section 2, adjusted for 3D classification using variable-sized layers and binning for the barrel-volume detector layers. As in the 2D case, the first three layers were used as a seed to constraint the target track. The right plots of Figure 6 shows the model prediction for both projections in the detector, Z and Φ and the gray scale represents the probability of the prediction. The binning was coarse and many events overwhelm with occupancy so overall accuracy was 70% but it was qualitatively seen to perform much better when occupancy is lower. This suggests that greater granularity (probably in smaller detector sub-regions) could scale up better.

The construction of track seeds from triplet-hit combinatorics can be constrained by an estimate of the interaction vertex position along the beamline (the Z axis). A deep learning solution was thus developed which takes as input the three innermost barrel pixel layer hits binned in Z and Φ as an RGB image and produces a prediction for the binned primary vertex Z position. The model is based on LeNet-5 [9] and is trained on ACTS data generated with five average interactions per sample. Figure 7 shows an example sample prediction and target as well as the thresholded accuracy as a function of bin size. An 82% accuracy can be achieved by covering 57% of the events with a resolution of 7.65 mm. Studies are ongoing to improve the performance of this model.

5. Conclusion

CNNs and LSTMs have shown great results in classification and sequential problems, but are still relatively unexplored for particle tracking problems. In this work, a variety of deep learning

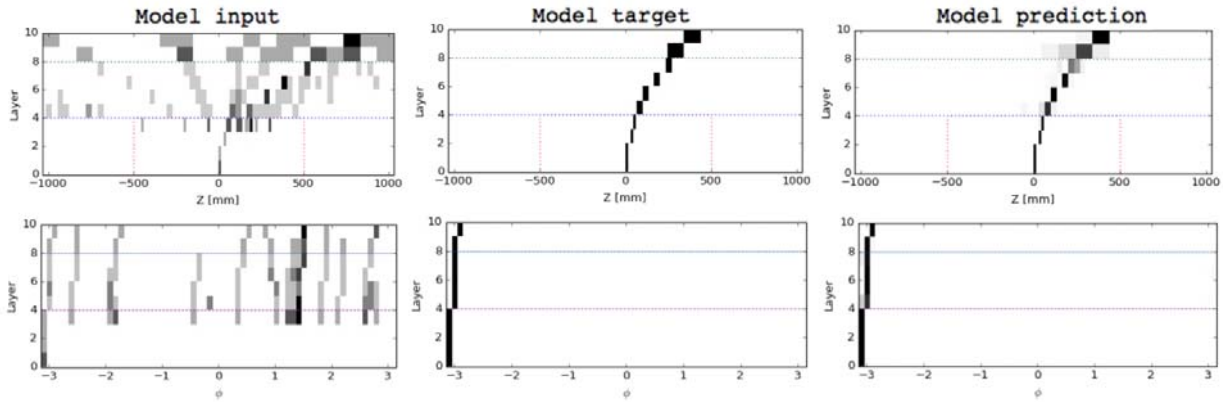

Figure 6. The plots show the input hit distribution to the LSTM network, the target tracks and the model prediction. The top plots are a projection in the Z coordinate of the detector (illustrated in Figure 5) and the bottom plots are a projection in the azimuthal angle of the detector.

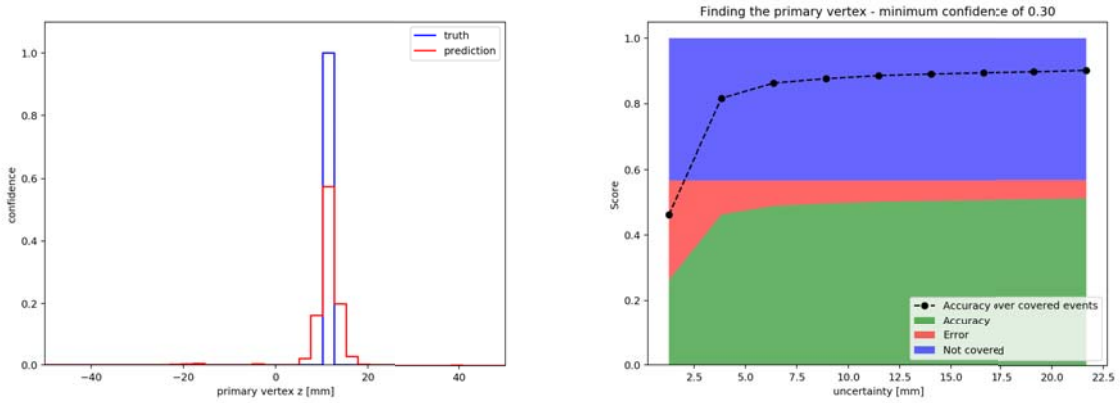

Figure 7. The plot on the left illustrates the binned Z-position of the target vertex (blue) and the output of the model (red) with a bin size of $2.55mm$. The plot on the right shows the accuracy and the error rate as a function of resolution (bin-size) with a 0.3 threshold in the confidence level of the model. The dashed line is the accuracy for events that pass this threshold (57% of the total events).

architectures was tested in semi-realistic datasets and where it was possible an optimization of the output was performed for higher efficiency of the model prediction. We have already started exploring our models with datasets generated with the ACTS framework. There are numerous architectures that have been tested with toy datasets but not yet fully adopted by the latest examples. We would like to fix this dataset (or a variation of it) and make comparisons with common tracking metrics between the different approaches described above. Also a comparison will follow with a traditional tracking algorithm such as the CKF.

The authors would like to thank the funding agencies DOE ASCR and COMP HEP for supporting this work, as well as the numerous tracking experts from ATLAS and CMS who have shared insights and experience.

References

[1] L. Evans, P. Bryant, JINST 3, S08001 (2008).
[2] G. Aad et al. (ATLAS), JINST 3, S08003 (2008)
[3] S. Chatrchyan et al. (CMS), JINST 3, S08004 (2008)
[4] G. Cerati et al., J. Phys. Conf. Ser. 664, 072008 (2015)
[5] S. Hochreiter, J. Schmidhuber, Neural Comput. 9, 1735 (1997)
[6] S. Farrell et al., EPJ Web of Conferences 150, 00003 (2017)
[7] TrackMLRamp hackathon, a 2D tracking challenge, `https://indico.cern.ch/event/577003/contributions/2476446/attachments/1423512/2183608/tr170307_davidRousseau_CTDWIT2017_trackML.pptx.pdf`
[8] A Common Tracking Software Project, `http://acts.web.cern.ch/ACTS/index.php`
[9] Y. LeCun et al., Proc. IEEE 86 (11) (1998) 22782324