IMPROVEMENTS OF THE LOOT MODEL FOR PRIMARY VERTEX FINDING BASED ON THE ANALYSIS OF DEVELOPMENT RESULTS

E. Rezvaya[1,a], P. Goncharov[1], Y. Nefedov[1], G. Ososkov[1], A. Zhemchugov[1]

[1] *Joint Institute for Nuclear Research, 6 Joliot-Curie, 141980, Dubna, Moscow region, Russia*

E-mail: [a] rezvaya2016@gmail.com

The recognition of particle trajectories (tracks) from experimental measurements plays a key role in the reconstruction of events in experimental high-energy physics. Knowledge about the primary vertex of an event can significantly improve the quality of track reconstruction. To solve the problem of primary vertex finding in the BESIII inner tracking detector we applied the LOOT program which is a deep convolutional neural network that processes all event hits at once, like a three-dimensional image. We used mean absolute error to measure the quality of the trained model, but a thorough analysis of the results showed that this metric by itself is inadequate without considering output distributions of the vertex coordinates. Correcting all errors allowed us to propose special corrections to the loss function that gave quite acceptable results. The process of our problem investigation and its outcomes are presented.

High energy physics, event reconstruction, deep learning, BESIII, LOOT

1. Introduction

One of the most important problems of modern high-energy physics is the reconstruction of events occurring during the interaction of particles in experimental detectors. Reconstruction consists in recognizing for each event all trajectories of particles - tracks, determining their parameters, finding the coordinates of the event vertex in order to determine the type of event and identify interacting particles from this data. It should be noted that most reconstruction methods, such as the Kalman filter or some modern deep neural network methods, are local, i.e. do not use information about the entire picture of the event, but reconstruct each track separately, in order to find the coordinates of the event vertex later, as the point closest to all tracks. This does not allow us to evaluate the global picture of an event, to see the dependence between individual tracks or groups of tracks, to observe such phenomena as secondary vertices, as it is done in global tracking methods that perform track recognition across the entire picture of the event. Such a global approach is implemented in a new neural network model LOOT (Look Once On Tracks) [1] with an u-shaped architecture [2] and Dice loss function [3]. The resulting model with slight modifications was applied for direct reconstruction of event vertices in the internal detector of the BES-III experiment [4].

The inner tracker detector of the BESIII experiment has a total of three cylindrical GEM type stations. This means not only that there are many fake hits due to strip readout of information in the GEM detector, but also that if a particle is registered only at two out of three stations, then its track in a magnetic field cannot be restored without knowing additional information about the coordinates of the vertex. In addition, knowledge of the coordinates of the vertex would give a significant reduction in the algorithmic complexity in the search for two-hit seeds – from O (n ^ 2) to O (n), because the primary vertex can be considered as a first hit of every track candidate, and no longer need to sort out all combinations of hits at the first two stations. Also, the knowledge about primary vertex is important for determining the momenta of particles, their spatial angles of emission, and recognizing the trajectories of short-lived and neutral particles.

In [4], we proposed a method of modification of the original LOOT architecture in order to use it for vertex finding. The initial model has a form of autoencoder, because it needs to extract useful features for the tracking in a contracting path, and then use these features in an expansive path to disentangle tracks in the original dimension. We have removed the decoder part, since a set of features that is extracted in the bottleneck of the network is sufficient to predict the position of the primary vertex. And for predicting the vertex location, an output layer with three neurons (for X, Y and Z) coordinates was added. The resulting model should solve the regression task, so the common metrics for evaluation of the regression models were used to report the quality of the algorithm. The metrics looked satisfactory at the first sight, nevertheless we continued interpreting the results and eventually found out that the low value of mean absolute error (0.009) does not mean that the model works correctly.

This work is a continuation of the research begun in [4]. It consists of three steps, which ultimately made it possible to improve the loss function and achieve physically acceptable results. The second section of the article describes the ways of how to better interpret the model result and correctly evaluate the quality. The third section is devoted to our first attempt to gain the better results from the existing pipeline. The fourth section narrates the steps of improving the loss function that allow us to obtain the acceptable results in the primary vertex finding.

2. Evaluation and interpretation of the model

Metrics are used to assess the quality of a model in machine learning problems. The metric used is determined by the type of the problem being solved. For regression or prediction, the most commonly used metrics are MAE (mean absolute error) and MSE (mean square error).

The first results of the model's operation were measured using the sum of the distances in all three coordinates from the real vertex to the vertex predicted by the network. This is called the mean absolute error and for the first version of the model, it was 0.009. The main problem was that the value of the metric in this case turned out to be uninformative. This value does not make it possible to understand the real picture of the location of the predicted vertex relative to the true one.

The fact is that the initial data were presented in the Cartesian coordinate system, but the LOOT model can only work with data from detector stations located consistently one after another, representing them as separate feature channels in the convolution operations. Since the detector is cylindrical, when preparing the data, we transformed the coordinates into a cylindrical system. Next, the input coordinates should be normalized to simplify the process of the model weights optimization. Therefore, initially, normalized values of coordinates in a cylindrical system were fed to the input of the metric.

To get the metric in the required units namely centimeters, it is necessary to translate the coordinates values to the original format to the input of the function that calculates the metric. Therefore, before calculating the metric, a new stage was added to reverse the transformation of the normalized values into the original units. It consists of two stages: denormalization and conversion from a cylindrical coordinate system to a Cartesian one (see the formula 1), and is applied to the network prediction while the corresponding real locations of the primary vertices remain the same – in the original cartesian coordinates. After adding this backward conversion, the metric value became 0.4 cm.

$$\begin{gathered} x = X * \sin \mathrm{Y}, \\ y = X * \cos Y, \\ z = Z. \end{gathered} \tag{1}$$

But the metric still gives an idea of the sum of the distances along all coordinates and it is impossible to understand which coordinate is predicted worse or better than the others by the value of the metric. Therefore, for a better understanding of the operation of the model, a graph of the distribution of the coordinates of the true vertex and the predicted one was built (Fig. 1).

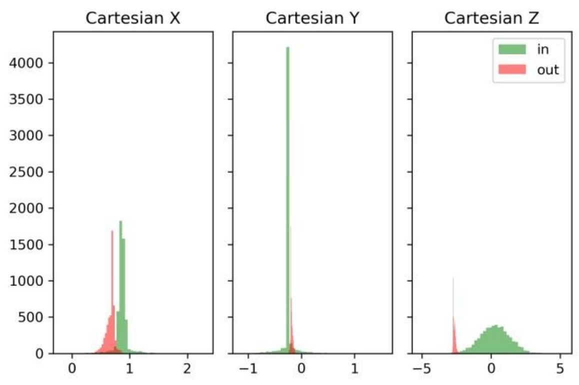

Figure 1. Distribution of the coordinates of the true and the predicted vertex

The graph shows that the distribution does not correspond to each other, which indicates the incorrect operation of the model. The measures taken do not allow obtaining the correct accuracy, therefore, in the next stage, we used the structural features of the detector for a detailed analysis.

3. Using features of the structure of the BES-III detector

The BES-III detector is designed in such a way that X and Y coordinates of the event's primary vertex are known. Therefore, an intermediate verification step was undertaken: predicting Z coordinate of the primary vertex separately, while taking X and Y coordinates as a fixed value. This stage showed a decrease in the value of the network error, but at the same time the plot of the distribution of Z coordinate shows that the predicted values do not correspond to the true ones (Fig. 2).

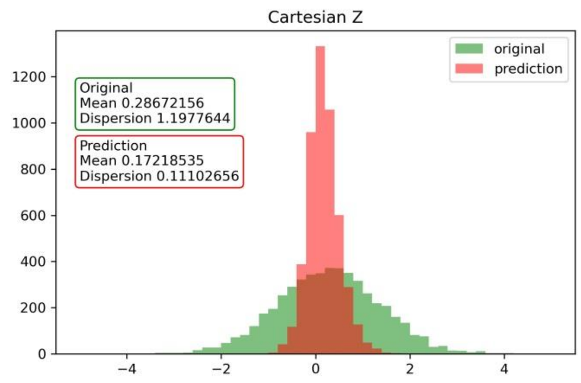

Figure 2. Distribution of Z coordinate of the true and the predicted vertex

Since the model is wrong, predicting even one coordinate, the problem may lie in the learning process. In the next step, we checked the loss function, which is used to train the model.

4. Loss function improvement

Loss function takes two parameters as input - true and predicted values. The function shows how well the model performs during prediction. The higher its value, the worse the model works.

Since we are trying to solve the regression problem the MSE-loss function was chosen for optimization. At the same time, we ran into a problem with model training. Predicted values are actually normalized coordinates from a cylindrical system. In order to calculate the loss function, the true vertex coordinates were converted to the same format as the prediction. The whole point of MSE is in the squaring the distance between real and predicted values, but squaring such small values in the loss function gives the function value close to zero. Besides the normalized coordinates make equal contribution to the loss function despite that the dispersion of original Z coordinate is much greater than for X or Y ones. All the above hampers the network from making accurate predictions.

To improve the quality of network predictions we changed the process of calculation of the loss function by adding the same step as for metrics calculation namely denormalization and conversion to the cartesian system before the loss estimation. This made it possible to avoid falling into the local minimum during training.

We retrained the model with the improved loss function and with prediction of all coordinates, not only Z, and observed that the distribution of the predicted coordinates is much closer to the real ones than it was before all optimizations (see the figure 3).

To train the LOOT model with modified loss function, we used 200K training events and 50K test events generated by the Monte Carlo for the BESIII inner detector. The model was trained for 150 epochs using Adam optimizer [5] with a learning rate of 0.001. The batch size is 64.

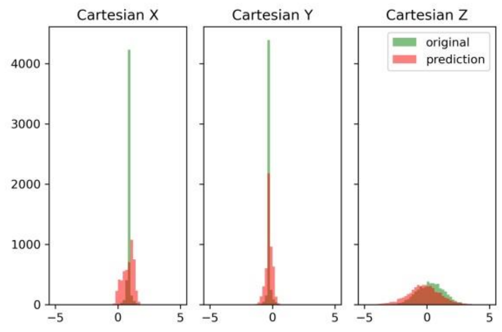

Figure 3. Result distribution of the coordinates of the true and the predicted vertex.

The model was trained with the help of Ariadne library [6]. The goal of this library is to offer a researcher a handy framework for rapid prototyping a new NN model for the tasks related to events reconstruction and provide a step-by-step fully reproducible pipeline including data preparation, training and evaluation phases. It is planned that the LOOT model will be part of the library in the future.

5. Conclusion

We have modified the loss calculation and significantly improved the quality of prediction. The value of MAE after all changes to the loss and being computed on the denormalized and translated to cartesian system predictions equals 1.9 cm. One can see that the new metric value is almost three times greater than the value of the first poor working version of the model. Thus, we showed that the metric selected at the beginning is confusing and we can't rely on it in our decision which model is better. The distribution of the coordinates of the true and the predicted vertices is much more informative and demonstrates that the model now operates well and gives acceptable results in the primary vertex finding.

6. Acknowledgement

The reported study was funded by RFBR according to the research project № 18-02-40101.

The calculations were carried out on the basis of the HybriLIT heterogeneous computing platform (LIT, JINR).

References

[1] P. Goncharov, G. Ososkov, D. Baranov, S. Shengsen, and Z. Yao, CEUR Workshop Proc. – Vol. 2507. – pp. 130-134 (2019)

[2] O. Ronneberger. P. Fischer and T. Brox, MICCAI 2015. Part of the Lecture Notes in Computer Science book series (LNCS, volume 9351), pp. 234-241 (2015).

[3] Sudre C. H. et al., Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations, Springer, Cham. – pp. 240-248 (2017).

[4] E. Rezvaya, P. Goncharov, E. Schavelev, I. Denisenko, G. Ososkov, and A. Zhemchugov, AIP Conference Proc. – Vol. 2377. – No. 1. – pp. 060005 (2021).

[5] S. Nikolenko, A. Kadurin and E. Arhangelskaya, Deep Learning //Spb: Piter. – 480 p. (2018).

[6] Ariadne, https://github.com/t3hseus/ariadne.