NOVEL APPROACH TO THE PARTICLE TRACK RECONSTRUCTION BASED ON DEEP LEARNING METHODS

D. Baranov[1], S. Mitsyn[1], G. Ososkov[1,a], P. Goncharov[2], A. Tsytrinov[2]

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

[2] *Sukhoi State Technical University of Gomel, October Ave. 48, Gomel, 246746 Republic of Belarus*

E-mail: [a] ososkov@jinr.ru

A fundamental problem of data processing for high energy and nuclear physics (HENP) experiments is the event reconstruction. The main part of it is finding tracks among a great number of so-called hits produced on sequential co-ordinate planes of tracking detectors. The track recognition problem consists in joining these hits into clusters, each of them joins all hits belonging to the same track, one of many others, discarding noise and fake hits. Such a procedure named tracking is especially difficult for modern HENP experiments with heavy ions where detectors register events with very high multiplicity. Besides, this problem is seriously aggravated due to the famous shortcoming of quite popular multiwired, strip and GEM detectors where the appearance of fake hits is caused by extra spurious crossings of wires or strips, while the number of those fakes is greater for some order of magnitude than for true hits. Here we discuss the novel two steps technique based on hit preprocessing by a sophisticated directed search followed by applying a deep learning neural network. Preliminary results of our approach for simulated events of the BM@N GEM detector are presented.

Keywords: tracking, BM@n experiment, GEM detectors, directed search, recurrent neural networks

1. Introduction

Event reconstruction is one of the most important problems in the modern high energy and nuclear physics (HENP). It consists of the determination of parameters of vertices and particle tracks for each event. The main part of it is finding tracks among a great number of so-called hits produced by charged particle passage through sequential co-ordinate planes of tracking detectors. The track recognition problem consists in joining these hits into clusters, each of them joins all hits belonging to the same track, one of many others, discarding noise and fake hits. Such a procedure named tracking is especially difficult for modern HENP experiments with heavy ions where detectors register events with very high multiplicity. Besides, this problem is considerably aggravated due to the famous shortcoming of quite popular multiwired, strip and GEM detectors where the appearance of fake hits is caused by extra spurious crossings of wires or strips, while the number of those fakes is greater for some order of magnitude than for true hits (see [1], for example).

Traditionally tracking algorithms based on the combinatorial Kalman Filter have been used with great success in HENP experiments for years [2]. However, the initialization procedure needed to start Kalman filtering requires a really vast search of hits needed to obtain so-called "seeds", i.e. initial approximations of track parameters of charged particles. Besides these state-of-the-art techniques are inherently sequential and scale poorly with the expected increases in detector occupancy in new conditions as for planned NICA experiments. In frames of the NICA project the Baryonic Matter at Nuclotron (BM@N) experiment is already running since 2015. One of methods to overcome the seed search problem for data of the BM@N GEM detector was proposed in [1] on the basis of a special coordinate transformation, but it is still also not always acceptably efficient. The actuality of data processing problems for BM@N experiment inspired us to apply our efforts to improve the tracking efficiency of its GEM microstrip detector on a new machine learning way.

Machine learning algorithms could make a great contribution to this problem due to their capability to model complex non-linear data dependencies, to learn effective representations of high-dimensional data through training, and to parallelize on high-performance computing means such as GPUs.

2. BM@N GEM tracking system

BM@N is a fixed target experiment aimed to study dense baryonic matter which can be created as a result of heavy-ion collisions [3]. The core of the central tracking system in the BM@N is represented by the GEM (Gas Electron Multiplier) detector that consists of a fixed number of gas-filled chambers combined into independent parts named «stations». The stations are positioned at some distances from one another along the beam axis. Moreover, in order to measure charged particles momenta, the stations are placed inside the large aperture dipole magnet. The last of BM@N experimental runs was carried out in spring 2017 with the setup comprised six stations. Namely this setup is considered in this article.

The operation of the GEM chamber is founded on principles of gas ionization and electron multiplication. The chamber, as a basic element of the station, is filled with a certain gaseous mixture to reach effective gas ionization. A charged particle, interacting with atoms in the gas mixture, produces lots of electron-ion pairs. Then the produced electrons are transmitted through the GEM's foils, resulting in electron avalanches. These avalanches are transferred by electric field toward the readout plane where they can be registered as a signal by the set of parallel microstrips as sensitive elements. Depending on the particle trajectory inclination not only one, but several adjacent strips can be activated forming a cluster. As usual, a clustering procedure is applied to get the central strip number with its total charge of all cluster strips.

Detectors with a microstrip-based readout system are widely used in many physics experiments because such readout electronics have a small number of readout channels in comparison with other types of readouts, such as pads or pixels. Using the smaller quantity of readout channels simplifies the wire bonding process and make assembling cheaper. It also allows the designers to use a more simple cooling system. These features help to reduce the amount of scattering materials on the way of particles.

Microstrip readout problems. Activated strips of one layer, crossing strips of another layer, form intersections in space called «hits». We deal with two types of the intersections: the first of these are real points through which charged particles passed. The second, named "fakes" produces spurious coordinates of non-existent tracks of the particles. The fakes are the principal disadvantage of using strips as a readout in track reconstruction problems. The case when strips of two layers cross at right angle (orthogonal strips) gives us the largest number of fakes. One of the ways to decrease the fake number is to rotate strips of one layer on a small angle with respect to another layer. As it is shown in fig.1, such rotation removes the majority of fake crossings out of the sensitive area (although lots of them are still left). Therefore the GEM chambers used in the BM@N have two-dimensional microstrip readout boards with two sets of strips rotated by stereo-angle of 15 degrees relatively to each other.

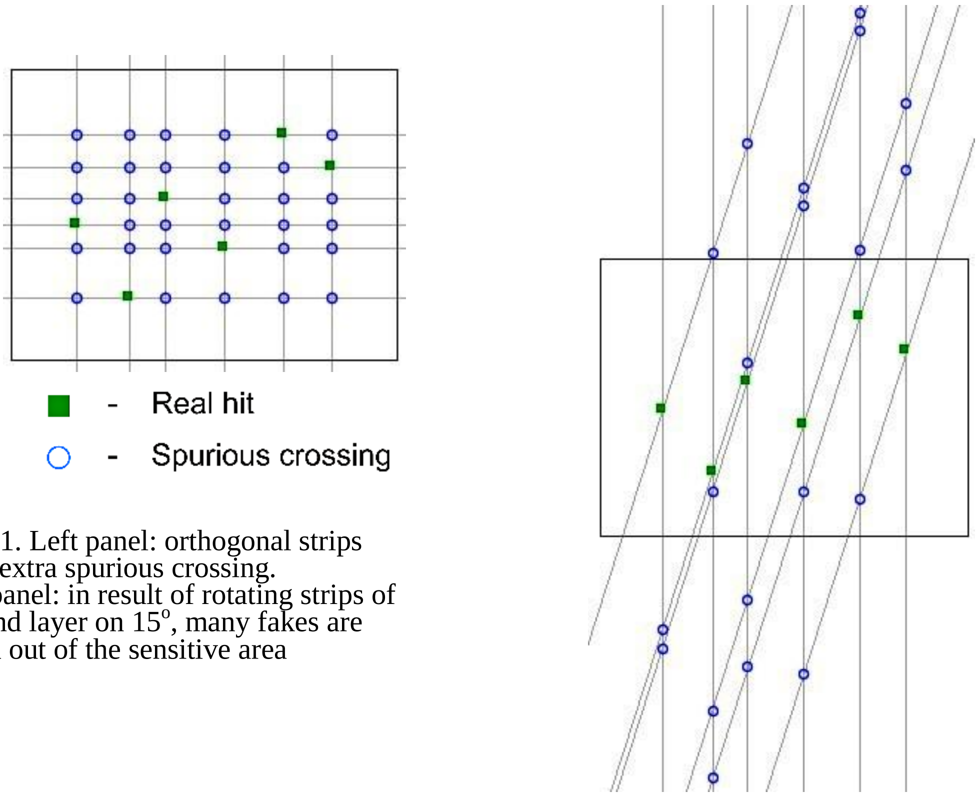


Figure 1. Left panel: orthogonal strips produce extra spurious crossing.
Right panel: in result of rotating strips of the second layer on 15^{o}, many fakes are swopped out of the sensitive area

Nevertheless, a significant quantity of fake hits seriously contaminates data needed for further processing and diminishes the efficiency of track finding. Carrying out the next BM@N runs planned for the short term implies increasing the multiplicity of events. Thus, the fake rejection procedure remains an important problem. In this paper we present a new approach to track recognition based on using deep learning networks.

3. Two-step tracking strategy

BM@N tracking implies a combinatorial search through many hits and thousands of fakes situated on sequential stations for namely such hits that belong to some of tracks, i.e. lying on a smooth curve. Starting from every hit on some of stations one should search for a corresponding (nearest) hit on the adjacent station. One of ways to reduce immense combinatorics is to use the curve smoothness to predict some smaller area for searching hits on the next station.

Therefore we choose two-step tracking starting from a preprocessing intended to find all possible track-candidates by a directed search followed then by applying a deep learning recurrent neural network**.** On the first step, track seeds are yielded using a simple and computationally effective algorithm that has relatively low efficiency, i.e. it produces a lot of fake seeds along with true seeds. After that on the second step a neural network is applied in order to effectively filter out fake tracks.

3.1 Tracking by directed search

The so-called directed search algorithm can be simply described as a procedure that goes through reconstructed hit points at every station, starting from the first one and extending current track candidates by one hit at every station also taking into account possible target coordinates. Having a model for the target greatly increases performance because a lot of tracks can be filtered out that, according to a simple track model, at a large distance from a target. Also, for this algorithm to be computationally simple, it is very important to utilize a lot of speeding up techniques.

As the main characteristic of the algorithm is a search of possible hits for extending a trajectory of the current track candidate to the next station, an effective search is crucial for the computing performance. Thus, the algorithm itself is divided into two sub-algorithms. The starting one does the preliminary search - for each track it filters out hits that just cannot possibly be an extension of the track, but it actually utilizes spatial indexes to extract hits of interest – these that have even the smallest possibility to be possible candidates for the extension. The next sub-algorithm extends the track by the extracted hits (thus making one extended track for each hit) and filters out tracks by a weak χ^2 criterion.

3.2. Preliminary search

BM@N magnetic field is not homogenous, but due to its overwhelming vertical component it is possible to work simultaneously in two projections: YoZ and XoZ (fig. 2).

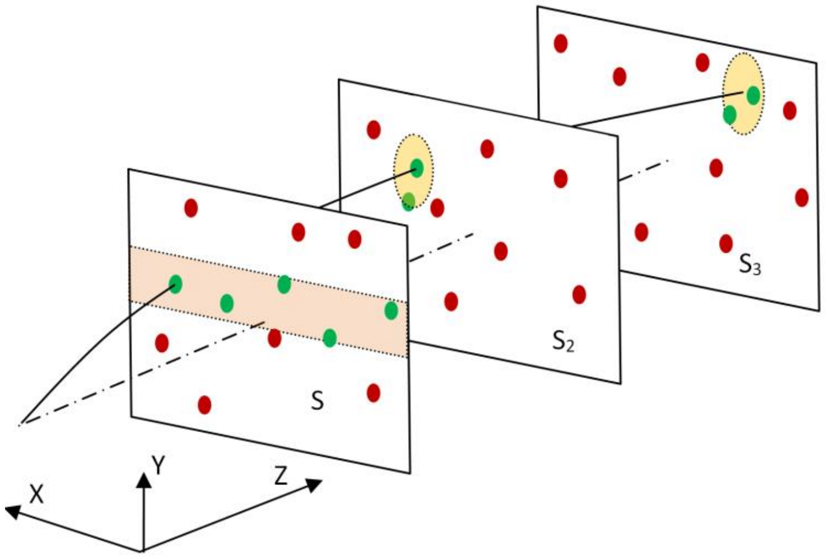


Figure 2. Spatial indices for algorithm speed-up

On YoZ projection, tracks are almost straight lines. It makes it possible to predict ***y*** coordinate starting with the second station (stationId=1). Thus, merely by sorting hit indices array by y coordinate, making a confidence interval and executing a binary search we can exclude all hits that are not in the confidence interval. A possible approach to deduce a confidence interval that we use is estimation of slope and intercept parameters of weighted linear regression (given errors of hit reconstruction), but while they seem to be normally distributed (as by simple Monte-Carlo modeling experiment given normal hit reconstruction errors), they are not IID and thus ***y*** estimation is not normal. At the moment we believe that the confidence interval parameters should be computed by a linear function of slope and intercept (non-linear if considering the distance from track mean z coordinate) with coefficients deduced by Monte-Carlo method for each event type) independently.

Thus confidence interval width $\omega = Ab + B(z - \bar{z})k$ with A and B being Monte-Carlo deduced, intercept b and slope k, $\bar{z}$ is track hits' mean $\boldsymbol{z}$ coordinate (intercept is **y** coordinate of the line at $\bar{z}$).

The XoZ projection case is more complicated. The track projection looks as a curve, but is not exactly a circle. Thus we enable limited rotational component. The limitation is on change in rotation – it should not change substantially. To account the field inhomogeneity the criterion is local, so only four adjacent hits are considered. Given hits' projections h_0, h_1, h_2, h_3 we define vectors $v_i = h_{i+1} - h_i$ and define that angle between these vectors should not change a lot. The angle can be deduced from easily computable sines, and we avoid arcsine computation by working directly with sines – thus we call the XoZ projection criterion the *sine criterion*. Using this criterion an X coordinate can be deduced starting from the third station.

Again, as in the case of the YoZ projection filtering, the parameters of the confidence interval is difficult to deduce, and again we are going to obtain the best value for confidence interval coefficients by Monte-Carlo experiments. At the moment, we utilize the admissiblity function $\text{admissible}(\{h_0, h_1, h_2, h_3\}) = \left[|\sin(\beta) - \sin(\alpha)| < \Delta\sin(\alpha) + W_{field}\right]$ where $\Delta\sin(\alpha)$ should be deduced from hit errors; α is an angle between v_0 and v_1, β is an angle between v_1 and v_2. The W_{field} coefficient takes into account field inhomogeneity and enables slight changes in rotation, and we believe it is very difficult to deduce it in any other way besides Monte-Carlo. That being said, the specific form of $\Delta\sin(\alpha)$ is an open question.

Starting with third station, both hit search criterion has to be utilized. We use kd-tree [4] data structure, but in reality any type of spatial index structure should be utilized.

3.3. Postfiltering

The postfiltering by χ^2 criterion is also difficult to perform. The problem is that errors given on hit coordinates are suffered from the fact that two close intersections with strip layer can be reconstructed as one hit, which has error comparable to cluster width, while isolated hit produces cluster of large width but small error. Thus, if taking cluster width for error, the $\chi^2$is underestimated and a lot of fake tracks are not filtered, while in the other case a lot of real tracks are filtered out. Again, we choose, as the most efficient way, the Monte-Carlo determination of a coefficient of the scaling cluster width.

4. Track-candidate classification via deep recurrent network

After doing the first step of our tracking we obtain a bunch of track-candidates, which should be divided into two groups: real tracks and, so named, ghost tracks formed by fakes and, possibly, by parts of different tracks. Thus we have a classification task, what can be solved by using artificial neural networks (NNs), which have become on top of classification over the past decade.

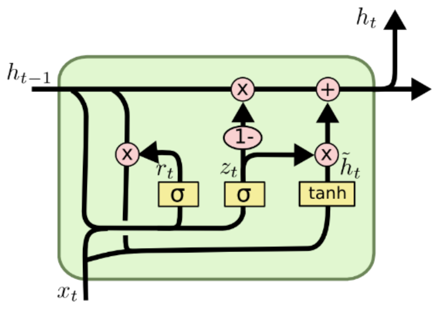
Hits belonging to some of tracks are indicated by three coordinates – features and situated on sequential stations along a particle way through the detector. Thus, a track-candidate presents as a sequence of hits. Each time step of such sequences is the specific point on k-th station of the detector, hence for 6 stations we have a vector of 3*6=18 features to define the track-candidate. They form a dynamical system like a movie, but unfortunately, ordinary neural networks, even deep ones with many layers, deep belief networks and such advanced nets as convolutional NNs are designed to manipulate with static objects, predicting a probability distribution over all observable classes. To handle dynamic objects the neural net should possess a kind of memory. It turns out that processing such kind of data refers us to dynamical systems as recurrent neural networks (RNNs), which are able to process sequential data.

4.1. Gated recurrent unit

The main difference of RNNs from other neural nets is their ability to memorize and extract previous states to operate on problems going through time. A simple block of RNN represents as a loop over inputs divided into time steps, in the heart of the loop there is a feedforward neural network, which we will refer to as «RNN cell». The unrolled block, can be thought of as multiple copies of the same network, each passing a message to a successor.

However, operating on memory should be restricted to prevent from information morphing and vanishing and exploding sensitivity. For example, humans do not remember every moment in their life, they determine for themselves what information is needed for memorization, and which one should be forgotten. So, the main principle of RNNs family is that new information should be incrementally added to memory, herewith not all of it, but only such its important part that can be useful in the future.

Gated recurrent unit (GRU) [5] presented in fig.3 is a specific type of RNN. The core idea of GRU is a kind of memory named the cell state that works like a conveyor belt for running information (line on the top of GRU model, fig.3). Instead of having a single neural layer, GRU includes three layers interacting in a very special way. These layers are capable to protect and control the cell state with the mechanism of gates – filters that optionally let information through. They are composed out of a sigmoidal layer and a pointwise multiplication operation and have the ability to write or forget information with atrainable degree of selectivity. GRU combines the «input» and «forget» gates into a single «update» gate.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 3. GRU unit schema with definition of used functions

Denotations in fig. 3 are as follows: **"."** is a matrix multiplication, **"*"** means element-wise multiplication, "[H, X]" is a concatenation of matrices H and X and "σ" – is sigmoid activation; *t* is a current time step, h_{t-1} determines the state on the previous time step and *x* is an input; rectangles are neural network layers with specific activations and circles are the pointwise operations.

However, a construction of using only the GRU layer is not enough to build a classifier. To perform classification on the output of our model, we need to stack on top of our network a so-called fully-connected layer, consisting of a single neuron with logistic activation function connected to all GRU cells on the previous layer.

This single neuron outputs the probability that input sample belongs to positive class (track-candidate is the real track). If it is true, then the output value is close to 1, otherwise it will be tagged as a member of negative class (track-candidate is a ghost). To train our NN classifier a binary cross-entropy error, also known as Kullback-Leibler divergence [6], should be minimized. The binary cross-entropy formula is defined as follows:

$$L(y, \hat{y}) = -y log\hat{y} - (1 - y)\log(1 - \hat{y}), \qquad (1)$$

where y determines a label and $\hat{y}$ is an output of RNN.

4.2 Model selection

The first part of study is to select which method of optimization should be used to minimize the network error function. We tried to train four RNNs each with one hidden layer consisting of 64 GRU neurons by different optimizers. Four gradient and subgradient methods were applied: Adagrad [7], Adadelta [8], Adam [9] and RMSProp [10] to compare validation efficiencies. We found that RMSProp method performs better than others.

The key idea of RMSProp gradient descent is to use for weight updating the inversely proportional value of the gradient root mean square (hence RMS) averaged by recent gradients.

We consider the gradient of the objective function g_t with respect to the weight at the time step t, and denote averaged value ${g^2}_t$ as follows:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)g_t^2, \tag{2}$$

where γ is a weights decay parameter, which we set equals to 0.9. Then the weights update rule is:

$$W_{t+1} = W_t - \frac{\eta g_t}{\sqrt{E[g^2]_t + \epsilon}}, \tag{3}$$

where η is a learning rate (we set it to 0.001) and epsilon is a small value of order **1E-10**.

The next step of the model selection is to choose the number and type of hidden layers and neurons per layer. We made several attempts combining such layers as convolutional [11], maximum pooling layer (MaxPool) [11] and GRU, trying to stack them in different ways varying numbers of hidden neurons in them.

The main reason for using the convolutional layer after the input is that it can produce a greater number of new meaningful features for RNN layer than original 18 input features only. Each convolutional filter acts as a feature detector, defined to respond maximally to specific temporal sequences within the timespan of the kernel.

Since we have input data as vectors, we use one dimensional convolutions (Conv1D) and MaxPool. Each 1D kernel is actually 3×3 filter, which performs only in one dimension across 6×3 matrix of features of particular track-candidate. As we applied padding, i.e., adding zeros across bound of features matrix, each 1D filter convolution gives a 6×1 vector. The convolutional layer outputs eventually a matrix of $6 \times N\_filters$, where *N_filters* means the number of specified convolutional kernels.

We found in our study that the best results of validation efficiency (fig. 4) was obtained by the combination of one convolutional layer and two GRU layers one after another. Besides, we applied 30% dropout [12] to each of GRU layers to prevent our model from getting overfitted and made the first GRU layer to extract features in bidirectional mode.

Eventually, the final setup of our model has five hidden layers, except input and output:

– 32 Conv1D 3×3 filters;

– bidirectional GRU with the output size equals to 64;

– 30% dropout layer;

– forward GRU with 64 hidden neurons;

– 30% dropout layer.

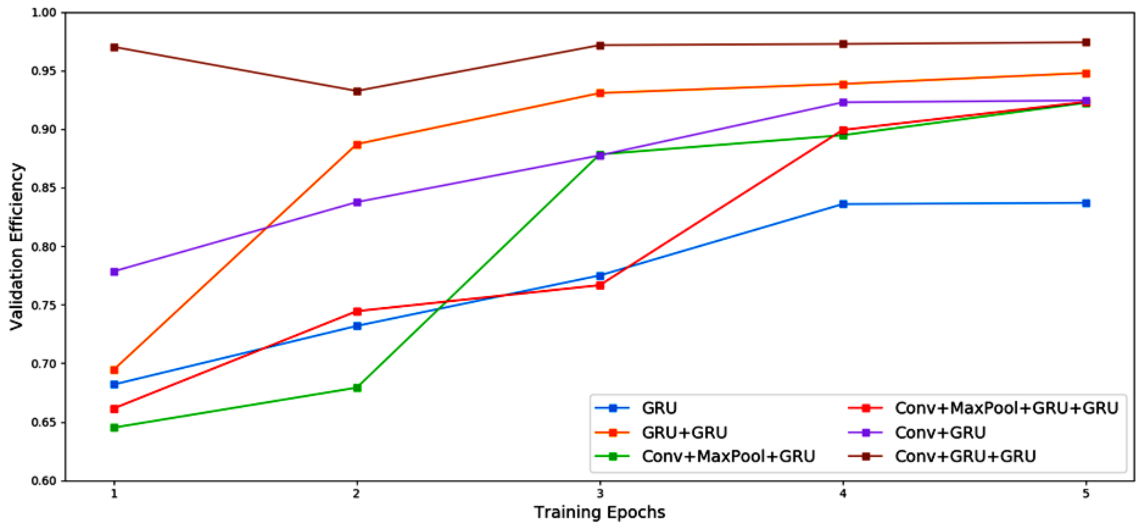


Figure 4. Validation efficiency value vs training epoch number for 6 different models of NN

4.3 Track-candidates classification results

To evaluate our NN classifier we generated two data sets: (1) the small data set with of 80 thousand track-candidates marked as real tracks and 80 thousand ghosts; (2) another one with 82 677 real tracks and 695 887 false track-candidates labeled as ghosts. All data was created using BOX Monte-Carlo generator of BM@N ROOT library. We divided each of data sets in two parts: train and test, in the ratio of 70 to 30.

Testing efficiency is the same for both samples, trained on small and big dataset, and equals to 97.5%. Trained RNN can process 6500 track-candidates in one second on the single Nvidia Tesla M60. All computations were performed on HybriLIT cluster [13].

5. Acknowledgement

This research was supported by the State Committee on Science and Technology of the Republic of Belarus (contract No. 202/17(039/64) of 25/07/2017).

6. Conclusion

Two-step tracking algorithm was proposed in frames of the BM@N data processing project intended to improve the tracking efficiency of the GEM microstrip detector on a new machine learning way. Preprocessing based on the greed KD-tree 2D search allows to extract possible track-candidates on the first stage of tracking. Then deep recurrent network on the second stage classifies track-candidates in two groups: true tracks and ghosts. The main efforts were applied to find such types and structures of our NN model which should provide the highest tracking efficiency being at the same time suitable for parallelization. As the result, we choose the neural network with five layers: one convolutional, two layers of gated recurrent units alternating with two dropout layers.

Test classification efficiency of this network is on the level of 97.5%. The trained network can process 6500 track-candidates in one second on the single Nvidia Tesla M60.

We are going to speed up the preprocessing stage by using either parallel computations or proper neural network.

References

[1] D. Baranov, S. Merts, G. Ososkov, O. Rogachevsky, New Algorithm of Seed Finding for Track Reconstruction, EPJ Web of Conferences 108, 02012 (2016).

[2] R. Fruhwirth,, Application of Kalman filtering to track and vertex fitting, Nucl. Instrum. Meth. A 262, 444 (1987)

[3] BM@N Conceptual Design Report (BM@N collaboration), http://nica.jinr.ru/files/BM@N/BMN_CDR.pdf

[4] S. Maneewongvatana, D. M. Mount Analysis of approximate nearest neighbor searching with clustered point sets // arXiv preprint arXiv:cs/9901013, 1999.

[5] Cho K. et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation //arXiv preprint arXiv:1406.1078, 2014.

[6] Kullback S., Leibler R. A. On information and sufficiency, The annals of mathematical statistics, 1951, v. 22, №. 1, 79-86.

[7] Duchi J., Hazan E., Singer Y. Adaptive subgradient methods for online learning and stochastic optimization //Journal of Machine Learning Research. – 2011. – T. 12. – №. Jul. – C. 2121-2159.

[8] Zeiler M. D. ADADELTA: an adaptive learning rate method, arXiv preprint arXiv:1212.5701, 2012.

[9] Kingma D. P., Ba J. L. 12 2015. Adam: A Method for Stochastic Optimization, arXiv preprint arXiv:1412.6980, 2014

[10] Tieleman T., Hinton G., Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude ,COURSERA: Neural networks for machine learning, 2012, v. 4, №. 2, 26-31.

[11] LeCun Y. et al. Gradient-based learning applied to document recognition, Proceedings of the IEEE, 1998, v. 86, №. 11, 2278-2324.

[12] Srivastava N. et al, Dropout: a simple way to prevent neural networks from overfitting, Journal of machine learning research, 2014, v. 15, №. 1, 1929-1958.

[13] Web-portal of HybriLIT JINR computation facility [Electronic resource]. – Mode of access: http://hybrilit.jinr.ru