NUCLEI AND NUCLEAR REACTIONS

M.Y. EL-BAKRY,[1,2] EL-SAYED A. EL-DAHSHAN,[3,4] E.F. ABD EL-HAMIED[1]

[1] Department of Physics, Faculty of Education, Ain Shams University
   (Roxy, Cairo, Egypt)
[2] Buraydah Tabouk University, Faculty of Science, Department of Physics
   (Tabouk, KSA)
[3] Dept. of Phys., Faculty of Science, Ain Shams University
   (Abbasia, Cairo, Egypt; e-mail: esldahshan@eelu.edu.eg)
[4] Egyptian E-Learning University
   (33, El-Mesah Str., El-Dokki-Giza-Postal code 1261, Egypt)

CHARGED PARTICLE PSEUDORAPIDITY DISTRIBUTIONS FOR Pb–Pb AND Au–Au COLLISIONS USING NEURAL NETWORK MODEL

The artificial neural network (ANN) approach is used to model the Pb–Pb and Au–Au collisions on the basis of the Levenberg–Marquardt learning algorithm. We simulate the rapidity distribution for π^- and $\kappa^\pm$ produced in Pb–Pb collisions at different energies and the pseudorapidity distribution of charged particles in Au–Au collisions. Our functions obtained within the ANN model show a very good agreement with the experimental data for both types of collisions, which indicates that the trained network takes on the optimal generalization performance. Thus, the ANN model can be widely applied to the modeling of heavy-ion collisions.

K e y w o r d s: charged particles, neural network, pseudorapidity distribution, Pb–Pb and Au–Au collisions, simulation.

1. Introduction

High-energy nucleon-nucleon and nucleus-nucleus collisions are an excellent tool to study nuclear matter [1]. The quark-gluon plasma (quark matter) predicted by various theories has been studied in high-energy nucleus-nucleus collisions. On the other hand, some properties obtained from nuclear reactions have been explained by the knowledge of current physics. The multiparticle production is an important experimental phenomenon in high-energy nucleus-nucleus collisions. One can use multiplicity, pseudorapidity (rapidity), transverse energy, *etc.*, to describe the characteristics of multiparticle production [2]. In the investigation of multiparticle produc-

tion, the pseudorapidity distributions of charged particles $(dN_{\rm ch}/d\eta)$ produced in heavy ion collisions are a powerful tool to describe the global properties of the collision system, where $N_{\rm ch}$ is the charged multiplicity, the pseudorapidity is defined by the formula $[\eta = -\ln[\tan(\theta/2)]$, and θ is the emission angle of the concerned charged particle. The pseudorapidity distributions have been measured in a wide range of energies at the GSI, AGS, SPS, and RHIC accelerators [3, 4]. Many models have been introduced in the field of high-energy heavy-ion collisions such as the one-dimensional string model [5], fireball model [6], multisource ideal gas model (and its predecessor, the thermalized cylinder model) [7–13], quark-gluon string model (QGSM) [14–18], string percolation model [19], heavy-ion jet interaction generator model (HIJING) [20–22], relativistic transport model
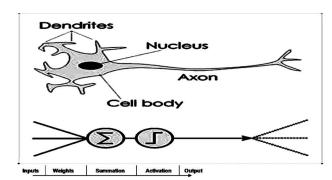
Fig. 1. Biological neuron versus artificial neurons

(ART) [23], ZPC parton cascade model [24], color glass condensate model (CGC) [25], relativistic quantum molecular dynamic model (RQMD) [26–29], hydrodynamic model [30, 31 and relativistic cascade model (ARC)[32–34]. In parallel to theoretical approaches based on different views, the development in the artificial intelligence (AI) field has indicated the strong presence of neural networks in high-energy physics [35–38].

Neural networks are composed of simple interconnected computational elements operating in parallel. These artificial neural networks (ANN) are obtained so that a particular inputs leads to a specific target output. Trained neural networks are able to perform complex functions in various fields of application including the pattern recognition, modeling, identification, classifications, speech, and vision and control systems [35–38].

Our present work uses ANN to model the nucleus-nucleus (A-A) collision features at high energies and proposes to calculate the rapidity distributions for π^-, K^+, and K^- productions in central Pb–Pb collisions at CERN SPS energies [39–41] and pseudorapidity distributions of charged particles for various c.m. energies in Au–Au central collisions at $\sqrt{s} = 19.6$–200 GeV (PHOBOS experiments) [1, 42–44].

The ANN model has been applied successfully to explore heavy ion collisions from CERN-SPS energies ($E_{\mathrm{lab}} = 20 - 160A$ GeV) up to the full BNL-RHIC energy ($\sqrt{s_{NN}} = 200$ GeV).

The paper is organized in three sections. Section 2 introduces the artificial neural network (ANN) model, and Section 3 describes the manner of modeling and simulating the A-A collisions (Pb–Pb and Au–Au). The following sections provide the results, discussion, and conclusion.

2. Artificial Neural Networks (ANNs)

Artificial Neural Network is a general mathematical computing paradigm that models the operations of biological neural systems. ANNs are a powerful general-purpose technique, and they have been widely applied to a variety of physical problems. In some fields such as high energy physics, they are the most widely applied computational intelligence technique.

ANNs are electrical analogues of the biological neural networks. Biological nerve cells, called neurons, receive signals from neighboring neurons or receptors through dendrites, process the received electrical pulses at the cell body and transmit signals through a large thick nerve fiber, called an axon (see Fig. 1). In a similar way, the electrical model of a typical biological neuron consists of a linear activator followed by a nonlinear inhibiting function. The linear activation function yields the sum of weighted input excitations, while the nonlinear inhibiting function attempts to culture the signal levels of the sum [45].

ANN is a collection of such electrical neurons connected in various topologies. The most common application of an ANN is that to machine learning. In a learning problem, the weights and/or nonlinearities in an ANN undergo an adaptation (or learning) cycle. The adaptation cycle is required for updating the parameters of the network, until a state of equilibrium is reached. ANN supports both supervised and unsupervised types of machine learning. ANNs can be considered as simplified mathematical model soft brain-like systems, and they function as parallel-distributed computing networks. However, in contrast to conventional computers, which are programmed to perform a specific task, most neural networks must be taught or trained. They can learn new associations, new functional dependences, and new patterns. Neural networks obviate the need to use complex mathematically explicit formulas, computer models, and impractical and costly physical models.

2.1. The mathematical model

When creating a function model of the biological neuron, there are three basic components of importance. First, the synapses of the neuron are modeled as weights. The strength of the connection between an input and a neuron is noted by the value of weight. Negative weight values reflect inhibitory

710

connections, while positive values designate excitatory connections. The next two components model the actual activity within the neuron cell. An adder sums up all the inputs modified by their respective weights. This activity is referred to as a linear combination. Finally, an activation function controls the amplitude of the neuron. An acceptable range of outputs is usually between 0 and 1 or − 1 and 1. Mathematically, this process is described in Fig. 2.

From this model, the interval activity of a neuron can be shown to be

$$Y_k = f\left(\sum_{j=1}^{m} W_{kj} + b_k\right). \tag{1}$$

The output of the neuron, Y_k, would be, therefore, the outcome of some activation on the value of ν_k. The hidden neurons (neurons of hidden layers) and the weight factors of the links between them play a critical role during the learning processing. In the case of supervised training, the numerical values of weight factors change according to the training data sets, in order to minimize the difference between the actual outputs and the target values. Thus, the relationship between causal factors and the response is mapped during the learning process. The transfer function of processing nodes is used to determine the output value of the node based on the total net input from nodes in the prior layer. The most widely used transfer functions are a sigmoid and tan sigmoid function, which are shown in the following relations [48]:

$$Y_s = \frac{1}{1 + e^{-x}}, \tag{2}$$

$$Y_{\text{tans}} = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{3}$$

3. Modeling the Nucleus-Nucleus Rapidity Distributions Using ANN

The objective of this study is the modeling of the rapidity distributions of charged particles produced from nucleus-nucleus (A-A) collisions. In particular, we will model the rapidity distributions of π^-, K$^+$ and K$^-$ mesons produced in central Pb–Pb collisions and the pseudorapidity distributions of Au–Au collisions at various energies. The feed forward
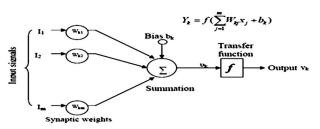

Fig. 2. Architecture of a typical artificial neural network [46, 47]
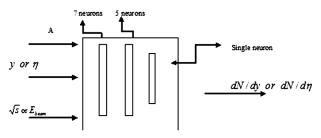

Fig. 3. Generic block diagram for charged particles produced in A-A collisions based on the ANN model

neural networks are used to build the models of rapidity distributions for charged particles produced in A-A collisions. The proposed ANN models of rapidity distributions have three layers: input layer (with tree inputs), output layer (with one output), and one hidden layer. The inputs involve the center of mass energy ($\sqrt{s}$), mass number of the projectile nucleus (A), and the rapidity (y) or the pseudorapidity distributions (η). The output is the rapidity distribution dN/dy of π^- and $\kappa^\pm$ mesons for Pb–Pb collisions and the pseudorapidity distributions for Au–Au collisions ($dN/d\eta$). The function of the hidden layer is to intervene between the external input and the network output in some useful manner. The general configuration (input, output, and hidden layer) of the proposed ANN model is shown in Fig. 3. Using this input–output arrangement, different network configurations were tried to achieve a good mean square error (MSE) and a good performance for the network. In order to achieve these goals, the proposed feed forward neural network was trained using the Levenberg–Marquardt (LM) learning algorithm [49–51].

Neural network training is usually formulated as a nonlinear optimization problem. The LM method is a modification of the classic Newton algorithm for finding the optimum solution to a minimization prob-
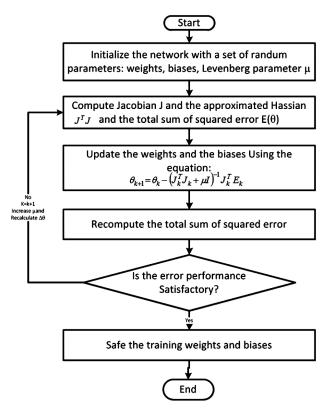
```
                    ┌─────────┐
                    │  Start  │
                    └─────────┘
                         │
   ┌─────────────────────────────────────────────┐
   │ Initialize the network with a set of randum  │
   │ parameters: weights, biases, Levenberg        │
   │ parameter μ                                    │
   └─────────────────────────────────────────────┘
                         │
   ┌─────────────────────────────────────────────┐
   │ Compute Jacobian J and the approximated       │
   │ Hassian  JᵀJ  and the total sum of squared     │
   │ error E(θ)                                     │
   └─────────────────────────────────────────────┘
                         │
   ┌─────────────────────────────────────────────┐
   │ Update the weights and the biases Using the   │
   │ equation:                                      │
   │ θ_{k+1}=θ_k−(J_k^T J_k+μI)^{-1} J_k^T E_k     │
   └─────────────────────────────────────────────┘
                         │
   ┌─────────────────────────────────────────────┐
   │ Recompute the total sum of squared error      │
   └─────────────────────────────────────────────┘
                         │
              ◇ Is the error performance
                   Satisfactory? ◇
                         │ Yes
   ┌─────────────────────────────────────────────┐
   │ Safe the training weights and biases          │
   └─────────────────────────────────────────────┘
                         │
                    ┌─────────┐
                    │   End   │
                    └─────────┘
```

No
K=k+1
Increase μ and
Recalculate Δθ

Fig. 4. Flow-chart of the Levenberg–Marquardt optimization method for the neural-network training

lem. This method minimizes the error function E (the squares of residuals, i.e. the squares of the differences between the desired outputs and the outputs of the network) by modifying the network weights w and biases b. This optimization technique is more powerful than the gradient descent technique and the back-propagation with momentum. It also prevents the network from the falling into local minima. The only disadvantage with this algorithm is the need for a larger memory and a greater number of hidden neurons. This algorithm adapts the parameters $\theta = \{w_{ji}, b_j\}$ with the use of the following expressions:

$$\theta_{k+1} = \theta_k - (J_k^T J_k + \mu I)^{-1} J_k^T E_k, \qquad (4)$$

$$E_k = \sum_{i=1}^{N} (d_i - y_i)^2, \qquad (5)$$

where N denotes the sample number of the learning process, d_i is the expected value for the cost estimation, y_i is the actual value, J represents the Jaco-

bian matrix of the error vector $E(\theta)$ assessed in θ; J^T is the transposed matrix of J; I is the identity matrix that has the same dimension with the approximated Hessian matrix $J^T J$; $J^T E(\theta)$ is the gradient of the error function E with respect to the weight and the bias parameters $\theta = \{b_j, w_{ji}\}$; and the adjusting parameter μ (damping factor) is increased or reduced along each learning iteration to guide the optimization process ($\mu = 0.001$ as the initial learning parameter). When the scalar μ is very large, the Levenberg–Marquardt method approximates the gradient descent method. However, when μ is small, it is the same as the Gauss–Newton method. This method switches between the gradient descent and the Gauss–Newton techniques. The advantage of this method is in that it converges faster around the minimum and gives more accurate results.

This adjustment for μ is done by using the adjustment factor β. If μ needs to increase, it is multiplied by β. If it needs to decrease, then it is divided by β ($\beta = 0.1$ as the decrement factor, and $\beta = 10$ as the increment factor). The process is repeated until the error decreases. When this happens, the current iteration ends. Therefore, the training process using the Levenberg–Marquardt algorithm could be designed as follows:

1. With the initial biases and weights (randomly generated), compute the Jacobian J and the error gradient $J^T E(\theta)$ and approximate the Hessian matrix $J^T J$.

2. Do an update as directed by Eq. (4) to adjust weights.

3. With the new weights, evaluate the total error.

4. If the error has not decreased, discard the new weights, increase μ using β, and go to step 2, else decrease μ using β, and stop.

5. Go to step 2 with the new weights until the current total error is smaller than the required value.

The flow-chart of the above procedure is shown in Fig. 4.

The Levenberg–Marquardt optimization method for the neural-network training is described in [50, 51] with more details.

3.1. Rapidity distributions for the π^- production in central Pb–Pb collisions

The proposed neural network model of rapidity distributions for π^- produced in central Pb–Pb colli-
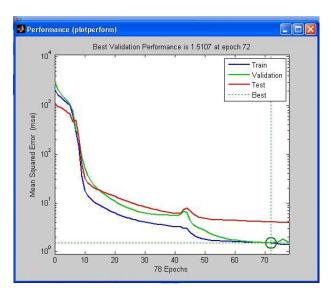
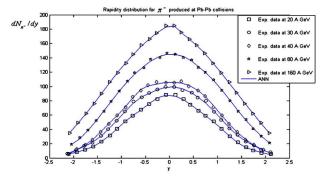**Fig. 5.** Training procedure of the ANN model

Fig. 6. Rapidity distribution of π^- in Pb–Pb collisions at SPS energies from 20 to $158A$ GeV (7% most central collisions for 20–$80A$ GeV, 5% most central collisions for $158A$ GeV) measured by the NA49 collaboration. [($\circ$) – experimental data, (-) – ANN]

sions at (20, 30, 40, 80, and $160A$ GeV) has three inputs $(A, y, \sqrt{s})$, one output(dN/dy), and two hidden layers (one layer consists of 7 neurons, and the second consists of 5 neurons). The transfer functions of the first layer and two hidden layers were chosen to be a tan sigmoid, while the output was chosen to be a pure line. The number of epochs = 72. In this case, the center-of-mass energies of 20, 30, 40, and $160A$ GeV are used to train the neural network model.

The training procedure is shown in Fig. 5. The plot shows the mean squared error of the network starting at a large value and decreasing to a smaller
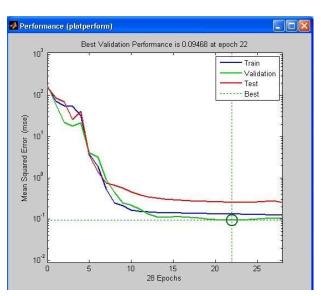

Fig. 7. Training procedure of the ANN model with the best validation performance of 0.09468
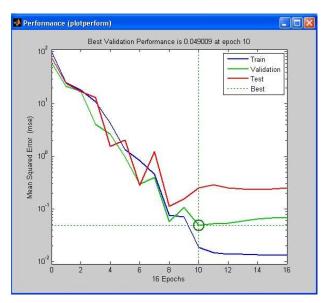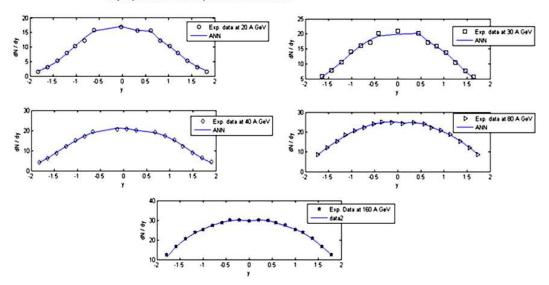

Fig. 8. Training procedure of the ANN model with the best validation performance of 0.049009

value. It shows that the network is learning. The plot has three lines, because the input and target vectors are randomly divided into three sets. 80% of the vectors are used to train the network, and 20% of the vectors are used to validate how well the network is generalized. Training on the training vectors continues as long the training reduces

Fig. 9. Rapidity distribution of K^+ produced in central Pb–Pb collisions at SPS energies from $(20$ to $158)A$ GeV (7% most central collisions for 20–80A GeV and 5% most central collisions for 158A GeV measured by NA49 collaboration. [(∘) – experimental data, (-) – ANN model]

the network's error by the validation vector. After the network memorizes the training set, training is stopped. This technique automatically avoids the problem of overfitting, which plagues many optimization and learning algorithms. Finally, the last 20% of the vectors provide an independent test of the network generalization to data that the network has never seen.

Figure 6 presents the rapidity distribution of π^- produced in central Pb–Pb collisions (7% at 20–80A GeV), (5% at 158A GeV) at SPS energies (experimental data), and the calculated results (solid lines) obtained with the use of the ANN model. The calculated results are in good agreement with the experimental data [39–41].

The rapidity distribution of π^- produced in central Pb–Pb collisions which obtained from the ANN model is

$$dN/dy = W(3,2)Y_{\text{tans}}\{W(2,1)Y_{\text{tans}}\times$$

$$\times[W(1,1)P + b(1)] + b(2)\} + b(3).] \tag{6}$$

Here, P is the input which is (A, y, E_{beam}), $W(3,2)$ – linked weight between the second hidden layer and the output, $W(2,1)$ – linked weight between the first and second hidden layers, $W(1,1)$ – linked weights

between the input layer and the first hidden layer, $b(1)$ is the bias of the first hidden layer, $b(2)$ is the bias of the second hidden layer, and $b(3)$ is the bias of the output layer. The weights and biases in relation (6) are given in Appendix A.

3.2. Rapidity distribution of $\kappa^\pm$ produced at different energies in central Pb–Pb collisions

The proposed neural network model of the rapidity distribution of $\kappa^\pm$ produced in central Pb-Pb collisions (20, 30, 40, 80, and 160A GeV) have three inputs $(A, y, \sqrt{s})$, one output (dN/dy), and two hidden layers, which consist of 9 and 5 neurons, respectively. The configuration of the proposed ANN model is shown in Fig. 3. The transfer functions of the first layer and two hidden layers were chosen to be tan sigmoid, while that for the output layer was chosen to be a pure line. The number of epochs was 22 and 10 for $\kappa^\pm$. In this case, the energies of 20, 30, 40, 80, and 160A GeV were used to train the neural network with performances of 0.094 and 0.049 for $\kappa^\pm$ (see Figs. 7 and 8). The results in Figs. 7 and 8 are reasonable because of the following considerations: the final mean-squared error is small, the test set error and the validation set error have similar character-

istics, and no significant overfitting has occurred by the 28- and 17-th iterations.

The rapidity distributions dN/dy of $\kappa^{\pm}$ produced in central Pb–Pb collisions (7% at 20–$80A$ GeV and 5% at $158A$ GeV) at SPS energies are simulated. The comparison between dN/dy calculated by employing the ANN Model and the corresponding experimental data is shown in Figs. 9 and 10. The calculated results are in good agreement with the experimental data [39–41].

The obtained function describes the rapidity distributions of $\kappa^{\pm}$ produced in central collisions as in Eq. (6), but different in weights and biases. The weights and biases in Eq. (6) for $\kappa^{\pm}$ produced in Pb–Pb collisions are given in Appendices B and C.

3.3. Pseudorapidity distributions of charged particles in Au–Au collisions

The proposed neural network model of the pseudorapidity distributions of Au–Au collisions at RHIC energies from 19.6 to 200 GeV performed by the PHOBOS collaboration [30, 31, 32, 39] has three inputs $(A, \eta, \sqrt{s})$, one output $(dN/d\eta)$, and two hidden layers (they consist of 9 and 8 neurons, respectively). The configuration of the proposed ANN model is shown in Fig. 3. The transfer functions of the first and second hidden layers were chosen to be tan sigmoid, while that for the output layer was chosen to be a pure line. The feed forward neural network back propagation is used to the modeling. The number of epochs is 42. In this case, the energies of 19.6, 62.4, 130, and 200 GeV were used to train the neural network. Training was terminated after 16 iterations as in Fig. 11.

The following function describes the pseudorapidity distributions of charged particles for Au–Au collisions:

$$dN/d\eta = W(3,2)Y_{\text{tans}}\{W(2,1)Y_{\text{tans}}\times$$

$$\times[W(1,1)B + b(1)] + b(2)\} + b(3). \qquad (7)$$

Here, B is the input which is $(A, \eta, \sqrt{s})$, $W(1,1)$ – linked weights between the input layer and the first hidden layer, $W(2,1)$ – linked weights between two hidden layers, $W(3,2)$ – linked weights between the second layer and the output layer, $b(1)$ is the bias of the first hidden layer, $b(2)$ is the bias of the second layer, and $b()$ is the bias of the output layer.
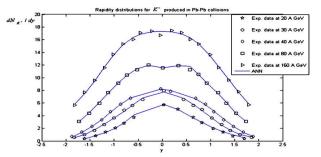
Fig. 10. Rapidity distributions of K^- produced in central Pb–Pb collisions at SPS energies from $(20$ to $158)A$ GeV, (7% most central collisions for 20–$80A$ GeV and 5% most central collisions for $158A$ GeV) measured by NA49 collaborations. [(∘) – experimental data, (-) – ANN model]
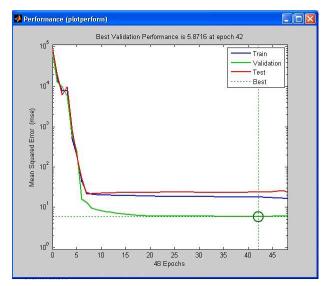

Fig. 11. Training procedure of the ANN model with the best validation performance of 12.1
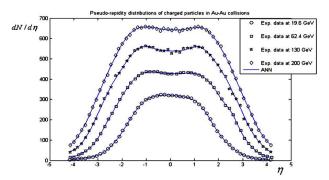

Fig. 12. Pseudorapidity distributions of charged particles for Au–Au collisions at RHIC energies from 19.6 to 200 GeV performed by the PHOBOS collaborations. [(∘) – experimental data, (-) – ANN model]

The weights and biases in Eq. (7) are given in Appendix D.

The simulation results based on the ANN approach to modeling the pseudorapidity distribution $(dN/d\eta)$ of charged particles in Au–Au collisions at different energies (from 19.6 to 200 GeV) are given in Fig. 12. It can be seen from these figures that the trained ANN model shows the exact fitting with experimental data [1, 42–44]. The match is still good, indicating that the designed neural network is robust.

4. Discussion and Conclusion

We developed a neural network approach to simulate and to predict the rapidity distribution (dN/dy) of π^- and $\kappa^\pm$ for Pb–Pb collisions at various energies. The ANN model also allowed us to simulate the pseudorapidity distribution $(dN/d\eta)$ of charged particles for Au–Au at various energies. Within this model, we have obtained a formula describing these collisions. Some different configurations of a network structure were investigated. A network structure with hidden layers and different neurons for $(\pi^-$ and $\kappa^\pm)$ for Pb–Pb and Au–Au collisions has given the best mean square error. The weights and biases used for the designed network are presented in the appendix. A very good agreement between the predicted values from the trained neural network and the validating data is achieved, which indicates that the trained network takes on optimal generalization performance. This also demonstrates how a typical data fitting technique based on neural networks can find the basic pattern information implied in a great number of experimental data, extract useful rules, and then apply these rules to obtain reasonable forecasting results. The results of the ANN model have showed a good agreement with the experimental data. The results demonstrate the feasibility of such technique in extracting the collision features and prove its efficiency.

Thus, we conclude that the ANN model based on the LM learning technique is able to perfectly model and simulate the pseudorapidity distribution $(dN/d\eta)$ of charged particles for A-A at various energies.

APPENDIX A

Weights and biases for the rapidity distribution of π^- in central Pb–Pb collisions:

$$W(1,1) = \begin{bmatrix} 2.0651 & 3.0749 \\ 2.9074 & -2.2949 \\ -3.1145 & -2.0049 \\ -2.7102 & 2.5249 \\ 3.5910 & -0.9081 \\ 3.3609 & 1.5570 \\ -2.4075 & -2.8149 \end{bmatrix},$$

$$W(2,1) = \begin{bmatrix} 0.4992 \\ -0.7003 \\ 0.3404 \\ 0.8429 \\ 0.1122 \\ 0.8653 \\ 0.8815 \\ 0.9431 \\ 0.7609 \\ 0.3467 \\ 0.1378 \\ 0.7072 \end{bmatrix}, \quad W(3,2) = \begin{bmatrix} 0.9289 \\ -0.7013 \\ 0.1944 \\ -0.0697 \\ -0.1773 \end{bmatrix},$$

$$b(1) = \begin{bmatrix} -3.7041 \\ -2.4694 \\ 1.2347 \\ 0 \\ 1.2347 \\ 2.4694 \\ -3.7041 \end{bmatrix}, \quad b(2) = \begin{bmatrix} -1.7619 \\ -0.8809 \\ 0 \\ 0.8809 \\ 1.7619 \end{bmatrix},$$

$$b(3) = [0.7610].$$

APPENDIX B

Weights and biases for the rapidity distribution of K$^+$ in central Pb–Pb collisions:

$$W(1,1) = \begin{bmatrix} 1.2636 & 3.4819 \\ 2.3892 & 2.8305 \\ -3.6687 & -0.5103 \\ 3.6139 & -0.8123 \\ -1.2393 & 3.4906 \\ -2.5993 & -2.6389 \\ 2.6320 & -2.6063 \end{bmatrix},$$

$$W(2,1) = \begin{bmatrix} -0.1599 \\ 0.9119 \\ 0.0703 \\ -0.0485 \\ 0.7800 \\ -0.8176 \\ -0.9814 \\ -0.1894 \\ -0.9780 \\ 0.7995 \\ -0.6799 \\ 0.6038 \end{bmatrix}, \quad W(3,2) = \begin{bmatrix} -0.9581 \\ -0.5805 \\ 0.3548 \\ -0.0937 \\ -0.0136 \end{bmatrix},$$

$$b(1) = \begin{bmatrix} -3.7041 \\ -2.4694 \\ 0 \\ -1.2347 \\ -2.4694 \\ 3.7041 \end{bmatrix}, \quad b(2) = \begin{bmatrix} 1.7619 \\ 0.8809 \\ 0 \\ -0.8809 \\ -1.7619 \end{bmatrix},$$

$$b(3) = [0.2946].$$

APPENDIX C

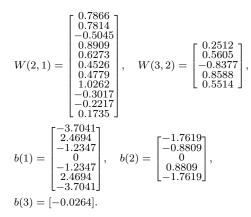Weights and biases for the rapidity distribution of K$^-$ in central Pb–Pb collisions:
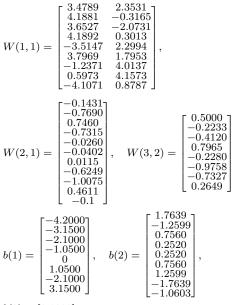
$$W(1,1) = \begin{bmatrix} 0.4326 & -3.6787 \\ -2.9881 & -2.1889 \\ 2.5329 & -2.7026 \\ 1.3343 & -3.4554 \\ -1.8419 & -3.2136 \\ 0.5857 & -3.6574 \\ -0.7891 & -3.6190 \end{bmatrix},$$

$$
W(2,1) = \begin{bmatrix} 0.7866 \\ 0.7814 \\ -0.5045 \\ 0.8909 \\ 0.6273 \\ 0.4526 \\ 0.4779 \\ 1.0262 \\ -0.3017 \\ -0.2217 \\ 0.1735 \end{bmatrix}, \quad W(3,2) = \begin{bmatrix} 0.2512 \\ 0.5605 \\ -0.8377 \\ 0.8588 \\ 0.5514 \end{bmatrix},
$$

$$
b(1) = \begin{bmatrix} -3.7041 \\ 2.4694 \\ -1.2347 \\ 0 \\ -1.2347 \\ 2.4694 \\ -3.7041 \end{bmatrix}, \quad b(2) = \begin{bmatrix} -1.7619 \\ -0.8809 \\ 0 \\ 0.8809 \\ -1.7619 \end{bmatrix},
$$

$$
b(3) = [-0.0264].
$$

APPENDIX D

Weights and biases for the pseudorapidity distributions of charged particles for Au–Au collisions:

$$
W(1,1) = \begin{bmatrix} 3.4789 & 2.3531 \\ 4.1881 & -0.3165 \\ 3.6527 & -2.0731 \\ 4.1892 & 0.3013 \\ -3.5147 & 2.2994 \\ 3.7969 & 1.7953 \\ -1.2371 & 4.0137 \\ 0.5973 & 4.1573 \\ -4.1071 & 0.8787 \end{bmatrix},
$$

$$
W(2,1) = \begin{bmatrix} -0.1431 \\ -0.7690 \\ 0.7460 \\ -0.7315 \\ -0.0260 \\ -0.0402 \\ 0.0115 \\ -0.6249 \\ -1.0075 \\ 0.4611 \\ -0.1 \end{bmatrix}, \quad W(3,2) = \begin{bmatrix} 0.5000 \\ -0.2233 \\ -0.4120 \\ 0.7965 \\ -0.2280 \\ -0.9758 \\ -0.7327 \\ 0.2649 \end{bmatrix},
$$

$$
b(1) = \begin{bmatrix} -4.2000 \\ -3.1500 \\ -2.1000 \\ -1.0500 \\ 0 \\ 1.0500 \\ -2.1000 \\ 3.1500 \end{bmatrix}, \quad b(2) = \begin{bmatrix} 1.7639 \\ -1.2599 \\ 0.7560 \\ 0.2520 \\ 0.2520 \\ 0.7560 \\ 1.2599 \\ -1.7639 \\ -1.0603 \end{bmatrix},
$$

$$
b(3) = [0.1032].
$$

1. M. Mitrovski, T. Schuster, G. Gräf, H. Petersen, and M. Bleicher, arXiv: nucl-th/ 08122041.
2. F.H. Liu, Chin. J. Phys. **38**, 42 (2000).
3. C.R. Meng, Chin. Phys. Lett. **26**, 102501 (2009).
4. F.H. Liu, Jain-Xin Sun, and Er-Qin Wang, Chin. Phys. Lett. **27**, 032503 (2010).
5. K. Werner, Phys. Rep. **232**, 87 (1995).
6. G.D. Westfall *et al.*, Phys. Rev. Lett. **37**, 1202 (1976).
7. F.H. Liu, Acta Phys. Sin. **7**, 321 (1998).
8. F.H. Liu and Y.A. Panebratsev, Nucl. Phys. A **641**, 379 (1998).
9. F.H. Liu and Y.A. Panebratsev, Phys. Rev. C **59**, 1193 (1999).
10. F.H. Liu and Y.A. Panebratsev, Phys. Rev. C textbf59, 1798 (1999).
11. F.H. Liu, Phys. Lett. B **583**, 68 (2004).
12. F.H. Liu, D.H. Zhang, and M.Y. Duan, Europhys. Lett. **61**, 736 (2003).
13. F.H. Liu, X.Y. Yin, J.L. Tian, and N.N. Abd Allah, Phys. Rev. C **69**, 034905 (2004).
14. A.B. Kaidalov, Phys. Lett. B **116**, 459 (1982).
15. A.B. Kaidalov and K.A. Ter-Martirosyan, Phys. Lett. B **117**, 247 (1982).
16. N.S. Amelin *et al.*, Phys. Rev. C**47**, 2299 (1993).
17. G. Burau *et al.*, Phys. Rev. C **71**, 054905 (2005).
18. E.E. Zabrodin *et al.*, J. Phys. G **31**, S995 (2005).
19. J. Dias de Deus and J.G. Milhano, Nucl. Phys. A **795**, 8 (2007).
20. X.N. Wang, Phys. Rev. D **43**, 104 (1991).
21. X.N. Wang and M. Gyulassy, Phys. Rev. D **44**, 3501 (1991).
22. X.N. Wang and M. Gyulassy, Phys. Rev. Lett. **68**, 1480 (1992).
23. B.A. Li and C.M. Ko, Phys. Rev. C **52**, 2037 (1995).
24. B. Zhang, Comput. Phys. Commun. **109**, 193 (1998).
25. D. Kharzeev, E. Levin, and L. McLerran, Phys. Lett. B **561**, 93 (2003).
26. H. Sorge, H. Stocker, and W. Greiner, Nucl. Phys. A **498**, 567 (1989).
27. H. Sorge, A. von Keitz, R. Mattiello, H. Stocker, and W. Greiner, Nucl. Phys. A **525**, 95 (1991).
28. A. Jahns *et al.*, Nucl. Phys. A **566**, 483 (1994).
29. K. Tywoniuk *et al.*, Phys. Lett. B **657**, 170 (2007).
30. R.B. Clare and D. Strottmann, hys. Rep. **141**, 177 (1986).
31. U. Ornik, R.M. Weiner, and G. Wilk, Nucl. Phys. A **566**, 469 (1994).
32. Y. Pang, T.J. Schlagel, and S.H. Kahana, Nucl. Phys. A **544**, 453 (1992).
33. Y. Pang, T.J. Schlagel, and S.H. Kahana, Phys. Rev. Lett. **68**, 2743 (1992).
34. S.H. Kahana, T.J. Schlagel, and Y. Pang, Nucl. Phys. A **566**, 465 (1994).
35. A.K. Hamid, Can J. Phys. 76-63-7 (1998).
36. P. Bhatet *et al.*, *Proceedings of the Summer Study on HEP, Snowmass, Colorado, 1990.*
37. R.P. Lippman, IEEE Acoust. Speech Signal Process. Mag., No. 4, 4-22 (1987).
38. M.Y. El-Bakry and K.A. El-Metwally, Solit. Fract. **16**, 279 (2003).
39. S.V. Afanasiev *et al.*, Phys. Rev. C **66**, 054902 (2002); [arXiv:nucl-ex/0205002].
40. M. Ga'zdzicki, C. Alt *et al.*, J. Phys. G **30**, S119 (2004); [arXiv:nucl-ex/0403023].
41. C. Alt *et al.*, [NA49 Collaboration], Phys. Rev. C **77**, 024903 (2008).
42. B. Alver *et al.*, arXiv: 0709.4008 [nucl-ex].

43. B.B. Back *et al.*, Phys. Rev. C **74**, 021901 (2006).

44. M.C. Abreu *et al.*, Phys. Lett. B **530**, 43 (2002).

45. K. Metaxiotis, *Intelligent Information Systems and Knowledge Management for Energy: Applications for Decision Support, Usage, and Environmental Protection* (Nat. Techn. Univ. of Athens, Athens, 2010).

46. M.Y. El-Bakry, A.M. Basha, N. Rashed, A. Radi, and M.A. Mahmoud, *6th Conference on Nuclear and Particle Physics, Luxor, Egypt, November 17–21, 2007*.

47. S. Haykin, *Neural Network: A Comprehensive Foundation* (Pearson Education, Upper Saddle River, NJ, 2005).

48. E. El-Dahshan, A. Radi, M. Y. El-Bakry, and M. El-Mashad, *6th Conference on Nuclear Particle Physics, Luxor, Egypt, November 17–21, 2007*.

49. F.M. Dias *et al.*, Eng. App. of Artif. Intell. **19**, 1 (2006).

50. M.T. Hagan and M.B. Menhaj, IEEE Trans. on Neural Networks **6**, 861 (1994).

М.І. Ель-Бакрі, Ель-Саед А. Ель-Дашан, Е.Ф. Абд Ель-Хамід

РОЗПОДІЛ ПСЕВДОШВИДКОСТЕЙ ЗАРЯДЖЕНИХ ЧАСТИНОК У Pb–Pb I Au–Au ЗІТКНЕННЯХ ЗА МОДЕЛЛЮ НЕЙРОННИХ МЕРЕЖ

Р е з ю м е

Моделюються Pb–Pb і Au–Au зіткнення в методі штучних нейронних мереж (ШНС) на основі навчального алгоритму Левенберга–Маркардта. Розраховано розподіл швидкостей для π^- і $\kappa^{\pm}$, народжених у Pb–Pb зіткненнях при різних енергіях і розподілу псевдошвидкостей заряджених частинок у Au–Au зіткненнях. Функції, отримані у ШНС моделі, дають дуже гарне узгодження з експериментом для обох типів зіткнень. Це свідчить про те, що навчена мережа дає оптимальні загальні характеристики, а ШНС модель може знайти широке застосування для опису зіткнень важких іонів.