Prediction of progesterone receptor inhibition by high-performance neural network algorithm

Ilyakay V. Romero Reyes, Irina V. Fedyushkina, Vladlen S. Skvortsov and Dmitry A. Filimonov

***Abstract*—** The computing model based on data of the interaction of progesterone receptor inhibitors with ligand binding domain was developed. It allowed estimating inhibition constant for probable inhibitors with high accuracy ($R^2 = 0.95$). This model is based on the feed forward back propagation neural network with the Levenberg-Marquardt method and the GPU parallel program realization was implemented. The last one speeded up in 734 times as compared with serial MATLAB realization. The parallel realization significantly increased the size of the processed data, while reducing time-consuming.

***Keywords*—** Artificial neural networks, molecular docking and dynamics, parallel computation with GPU, progesterone receptor.

I. Introduction

LAST 20 years the various methods of computer-assisted drug discovery including structure-based and *de novo* drug design etc. of pharmacologically promising compounds are used in drug development [1]. Their application allows to significantly reduce the cost and to accelerate the processes of development and implementation. One of important task is the correlation research of the internal computer-aided estimations with kinetic values obtained in biochemical experiments. One of these values is the dissociation constant (K_d) of the protein-ligand complex. It is commonly used to describe the affinity of ligand-protein complex, i.e. how tightly the ligand binds to the particular protein. In some cases, other values, such as the inhibition constant (K_i) or values half maximal inhibitory concentration (IC_{50}) are used. There are two well-known approaches for binding energy estimation. The first one is based on the famous equation relating the dissociation constant and the change in Gibbs free energy:

$$\Delta G = -RT \ln(K_d)$$

However, the computer-aided estimations of the change of Gibbs free energy are very far from the true values. In addition, they strongly depend on the way of molecule description and selected parameters. Traditional considered exact methods of thermodynamic integration are computer-intensive and have essential restrictions on molecular dimensions and degree of chemical diversity [2]. Rather reliable is the calculation of value ΔG, but just for closely related compounds. The second approach is based on development of the scoring functions associated with the target. Here we describe the new approach that combines the both ones. The affinity of ligand-protein complex is estimated based on independent terms calculated from the molecular dynamics (MD) simulation of protein-ligand complex, that were used as input data to the artificial neural networks.

A set of progesterone receptor (PR) inhibitors was chosen as object of this research. Progesterone receptors are ligand-activated transcription factors and belong to the family of steroid hormone receptors. These proteins are present in most human tissues and they control diverse physiological and pathological processes such as regulation of proliferation of breast and ovarian cells cancer, myometrium activity, central nervous system and immune homeostasis etc.

II. Problem Formulation

The main goal of this research was to develop the scoring function of the change of Gibbs free energy based on descriptors calculated from ligand structure and energy parameters of protein-ligand complex. There were three main stages in this work. The first one includes modeling of complexes of PR inhibitors with ligand binding domain (LBD) PR, MD simulation of these complexes and calculating the energy parameters. The second one was selection of the artificial neural network (ANN) and its training algorithm, development of parallel ANN training algorithm and its software implementation using graphics processing unit (GPU). The graphical accelerator usage allows for significantly reducing the general computation time. At the last stage were training, testing, and validation of ANN for pK_i value prediction.

I. V. Romero Reyes is with the Institute of Biomedical Chemistry of Russian Academy of Medical Sciences, 10, Pogodinskaya Street, Moscow, 119121, Russia (corresponding author to provide phone: +7-495-245-07-68; e-mail: ilacai@ ibmc.msk.ru)

I. V. Fedyushkina is with the Institute of Biomedical Chemistry of Russian Academy of Medical Sciences, 10, Pogodinskaya Street, Moscow, 119121, Russia (corresponding author to provide phone: +7-495-245-07-68; e-mail: ivf@ ibmh.msk.su).

V. S. Skvortsov is with the Institute of Biomedical Chemistry of Russian Academy of Medical Sciences, 10, Pogodinskaya Street, Moscow, 119121, Russia (corresponding author to provide phone: +7-495-245-07-68; e-mail: vladlen@ ibmh.msk.su).

D. A. Filimonov is with the Institute of Biomedical Chemistry of Russian Academy of Medical Sciences, 10, Pogodinskaya Street, Moscow, 119121, Russia (corresponding author to provide phone: +7-495-255-30-29; e-mail: dmitry.filimonov@ ibmc.msk.ru).

III. Problem Solution

A. Docking and molecular dynamics

A set of ligands with known pK_i values was taken from [3] and it consists of 63 compounds.

The crystal structure of PR LBD/progesterone complex (PDB code is 1A28) [4]. The modeling of complexes of PR with ligands was performed as follows:

- The models of ligands were created using SYBYL 8.1 molecular modeling suite [5].
- The optimization of structures was done by short minimizations by Powell method with TRIPOS force field. The partial atomic charges for the ligand were calculated by Gasteiger-Hückel method.
- Calculating of additional ligand descriptors: molecular weight, surface area, molecular volume, polar surface area, number of halogens atoms in a molecule was done using SYBYL 8.1 molecular modeling suite.
- The flexible ligands docking to the binding site of PR was performed using DOCK 6.5 [6]; the solvent-accessible surface of the target for docking was built on the basis of the Connolly algorithm with the probe radius 1.4 Å; electrostatic and van der Waals potential fields generated over the target were performed using the grid (spacing 0.3 Å); the non-bond distance cutoff was 12.0 Å, the parameters for van der Waals interactions were used from dw_AMBER_parm99.defn set. The best docking pose was selected based on scoring function of DOCK 6.5.
- Optimization of complex structure using Amber 9.0 program [7] as follows: 1) formation of solvent layer (explicit water) with thickness no less than 2 Å in asymmetric rectangular box; 2) energy minimization of the system in periodically boundary conditions (PBC) up to 500 steps; 3) heating of the system from 0 K to 300 K. The simulation was performed for 10 ps (in a 2 fs time step) with NVT ensemble in PBC; 4) density normalization of system at 300 K (for 10 ps, in a 2 fs time step, with NTP ensemble); 5) equilibrating of system at 300 K (for 10 ps, in a 2 fs time step, with NTP ensemble) 6) productive MD simulating at 300 K (for 10 ps, in a 2 fs time step, with NTP ensemble). The simulation time on this step is small. But the analysis of complexes shows that the equilibration of ligand position and closest amino acid residual requests about 2-3 ps. The analysis of deep reorganization in protein structure is out the scope of this work.

The productive MD simulation was the basis for calculation of the change in virtual binding energy by MM-GBSA method [8]. The values of separate components, electrostatic Coulomb and wan der Waals interactions, hydrophobic contribution to solvation free energy and reaction field energy calculated by Poisson–Boltzmann (PB) и Generalized Born methods (GB), were averaged over the set of 10 snapshots.

B. Neural network model

According to Kolmogorov's theorem any multivariate continuous function can be represented by the superposition of a small number of univariate continuous functions [9].

Theorem (Kolmogorov, 1957). There exist fixed increasing continuous functions $\varphi_{ik}(x)$ on $I=[0,1]$ so that each continuous function f on I_n can be written in form

$$f(x^1,x^2,...,x^n)=\sum_{k=1}^{2n+1} h_k\left(\sum_{i=1}^{n}\varphi_{ik}(x^i)\right),$$

where h_k are properly chosen continuous functions of one variable.

It is clear this function has the structure of the neural network with one hidden layer. The **Stone-Weierstrass Theorem** says that every multivariate continuous function can be approximated by a polynomial to any desired degree of accuracy. It is possible to use the superposition of the addition and a continuous linear function of one argument [10] instead of polynomial (the superposition of the addition and the multiplication).

There is the statement about the universal approximation capabilities of any nonlinearity [10]. It is possible to design the system for calculating of any continuous function with any desired accuracy using linear operations and the unique nonlinear element φ.

Thus neural nets are the universal approximators of functions.

The back-propagation neural network (BPNN) [11] with one sigmoid-type hidden layer and linear output layer was used during this research. This type of neural networks is very popular and is used more than other neural network types for a wide variety of tasks [23], [24]. The applied architecture is shown in Fig. 1. The BPNN is based on the supervised procedure, i.e. the network constructs a model based on examples of data with known outputs. Its learning and update procedure works as follows: if the network gives the wrong answer, the weights are corrected, and the error is lessened so future responses of the network are more likely to be correct.

The number of hidden layers is normally chosen to be only one to reduce the network complexity, and increase the computational efficiency [12].

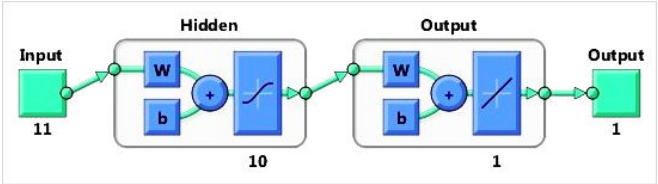

Fig. 1. Architecture of the proposed neural network.

The ANN input layer is determined from the characteristics of the application inputs. There are 11 different properties for each compound from input dataset. These properties are physical and chemical ligand properties (molecular weight, count of halogens, molecular surface, polar molecular surface, molecular volume) and energy values of complex (non-bonded electrostatic Coulomb and wan der Waals energies, hydrophobic contribution to solvation free energy for PB and GB calculations, reaction field energy calculated by PB and GB).

The hidden layer automatically extracts [12] the features of the input pattern and reduces its dimensionality. It was found

that 10 hidden neurons could accomplish the task at the hand quite reasonable. The tangent hyperbolic activation function was chosen for hidden layer. According to [12] BPNN may, in general, learn faster (in terms of the number of training iterations required) when the sigmoid activation function is nonsymmetric. An activation function $\varphi(\upsilon)$ is antisymmetric (i.e., odd function of its argument) if

$$\varphi(-\upsilon) = -\varphi(\upsilon)$$

as shown in Fig. 2.

The using function and its fast approximation are given as follows:

$$a_{i1} = \tanh(n_{i1}) = \frac{e^{n_{i1}} - e^{-n_{i1}}}{e^{n_{i1}} + e^{-n_{i1}}} \cong \frac{2}{1+e^{-2n_{i1}}} - 1$$

where a_{i1} is i^{th} element of $\mathbf{a_1}$ vector containing the outputs from hidden neurons, and n_{i1} is i^{th} element of $\mathbf{n_1}$ vector containing net-inputs going into the hidden neurons. $\mathbf{n_1}$ vector is calculated as:

$$\mathbf{n_1} = \mathbf{W_{10}p} + \mathbf{b_1}$$

where $\mathbf{p}$ is the input pattern, $\mathbf{b_1}$ is the vector of bias weighs on hidden neurons, and $\mathbf{W_{10}}$ is the weight matrix between 0^{th} (i.e. input) layer and 1^{th} (i.e. hidden) layer. Each row of $\mathbf{W_{10}}$ contains the synaptic weights of the corresponding hidden neuron.

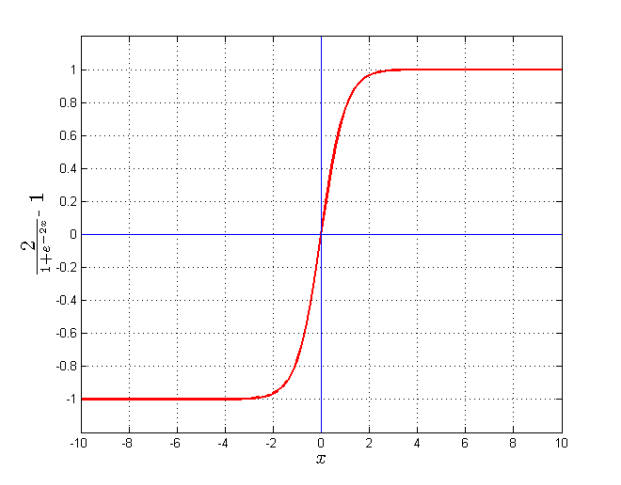

Fig. 2. Hyperbolic tangent sigmoid transfer function.

The output layer of the network is designed according to the need of the application output. Since the output of ANN is expected to produce pK_i, the number of output neurons is one.

Since the values of targets will be in the range $[5.45, 8.82]$, the pure linear activation function is selected for the outputs neurons, and expressed as:

$$\mathbf{a_2} = \mathbf{n_2}$$

where $\mathbf{a_2}$ is the column-vector coming from the second output layer, and $\mathbf{n_2}$ is the column-vector containing the net inputs going into the output layer. $\mathbf{n_2}$ is calculated as:

$$\mathbf{n_2} = \mathbf{W_{21}a_1} + \mathbf{b_2}$$

where $\mathbf{W_{21}}$ is the synaptic weight matrix between the first (i.e. hidden) layer and the second (i.e. output) layer, and $\mathbf{b_2}$ is the column-vector containing the bias inputs of the output neurons. Each row of $\mathbf{W_{21}}$ matrix contains the synaptic weights for the corresponding output neuron.

In a training stage, a set of input-target set is used for training and presented to the network many times. After the training is stopped, the performance of the network is validated. The BPNN learning algorithm involves a forward-propagating step followed by a backward-propagating step. A training set must have enough examples of data to be representative for the overall problem.

The batch mode of back-propagation learning was used in this work. In this method weights updating is performed after the presentation of all the training examples that constitute an epoch. The epoch means one full cycle of training through all training input patterns. For a particular epoch the mean squared error (MSE) reproduces here in the composite form:

$$E_{mean} = \frac{1}{2N}\sum_{n=1}^{N}\sum_{j} e_j^2(n),$$

$$e_j(n) = d_j(n) - y_j(n)$$

where the error signal $e_j(n)$ pertains to output neuron j for training example n, N is a the number of data points in the training set. The error $e_j(n)$ equals the difference between j^{th} element of desired response vector $\mathbf{d}(n)$ and the corresponding value of the network output $y_j(n)$. The inner summation with respect to j is performed over all the neurons in the output layer of the network, whereas the outer summation with respect on n is performed over the entire training set in the epoch.

Typically, for a BPNN to be applied, both training and a validation set of data are required. Both of these sets contain input and output matrices taken from real data. The first is used to train the network, and the second to assess the performance of the network after training. In the validation stage, the input set is supplied into the network and the desired output set is compared with that given by the neural network. The agreement or disagreement of these two sets gives an indication of the performance of the neural network model.

Another decision that has to be taken is the subdivision of the data set into different sub-sets which are used for training and validation the BPNN. According to [12] the best solution is to have separate data set, and to use the first set for training and validation the model, and the second independent set for testing of the model.

In our research we used all three sets: training, validation, and test sets.

The Levenberg-Marquardt method [13] was used for minimization of E_{mean} during the training. This algorithm is an iterative technique that locates the minimum of a multivariate function that is expressed as sum of squares of non-linear real-valued functions [14,15]. It can be associated as a combination of steepest descent and the Gauss-Newton method. When the

current solution is far from the correct one, the algorithm behaves like a steepest descent method. When the current solution is close to the correct solution it becomes a Gauss-Newton method. The extended description of the Levenberg-Marquardt method is also presented in [16].

The similar BPNN structure we used in research work [25].

C. Cross-validation methods

The essence of back-propagation learning is to encode an input-output mapping into synaptic weights and thresholds of a multilayer perceptron. The hope is that the network becomes well trained so that it trains enough about the past to generalize to the future. From such a perspective the learning process amounts to a choice of network selection problem as choosing, within a set of candidate model structures, the best one according to a certain criterion. In this context, a standard tool in statistics known as cross-validation provides an appealing guiding principle.

K-fold cross-validation is one way to improve over the holdout method. The dataset is divided into k subsets, and the holdout method is repeated k times. Each time, one of the k subsets is used as the test set and the other $k-1$ subsets are put together to form a training set. Then the average error across all k trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set $k-1$ times. The variance of the resulting estimate is reduced as k is increased. The disadvantage of this method is that the training algorithm has to be rerun from scratch k times, which means it takes k times as much computation to make an evaluation.

Leave-one-out cross validation is K-fold cross validation taken to its logical extreme, with K equal to M, the number of data points in the set. That means that M separate times, the function approximator is trained on all the data except for one point and a prediction is made for that point. As before the average error is computed and used to evaluate the model. The evaluation given by leave-one-out cross-validation (LOO-CV) error is good, but at first pass it seems very expensive to compute. Fortunately, locally weighted learners can make LOO predictions just as easily as they make regular predictions. That means computing the LOO-CV error takes no more time than computing the residual error and it is a much better way to evaluate models.

In our research **the following method was used**. After successful ANN training we divided the whole set randomly and independently to train, validation and test sets 100 times. The train set contained 70% of whole set, validation and test sets were for 15%. Every time ANN was trained and tested with these sets. Values of MSE were estimated. In this case the cross-validation error CV_{test} for training set type can be calculated as follows:

$$CV_{test} = \left(\frac{1}{N} \sum_{k=1}^{N} MSE_k^{test} \right)^{1/2},$$

where $N = 100$. This method is a variant of K-fold cross-validation with independent random data dividing into train, validation and test sets K different times. The advantage of doing this is that we can independently choose how many trials we average over.

D. Parallel algorithm

The runtime for parallel computing systems depends on internal structure of the used algorithm. Also it depends on the order of operations execution. It is possible to accelerate algorithm realizations using large number of processors which can execute algorithm operations in parallel mode. Just the mutually independent operations can be performed at any computing machinery. This means, the result of any of the simultaneously performed operations must not be the argument of any of them and affect by indirect way on their arguments. Considering the implementation of the algorithm in time, its operations are divided into the groups at any serial or parallel computing system. All operations of each group are mutually independent and simultaneously performed and the groups are serially performed. Thus there is a special form of algorithm presentation, which is called parallel form of the algorithm [14], where groups of operations and their sequence are fixed.

For parallel algorithm it is has to know a set of operations, which can be performed independently. Let's consider the operations of serial algorithm as graph vertexes. The partial sequence is defined on the set of vertexes and operations. It shows which operations can be performed after others ones and allows to define graph arcs. If the operation corresponding to B vertex uses the result of operation corresponding to A vertex as one of its arguments, there is an arc from A vertex to B vertex. In other case there is no arc between these vertexes. A direct acyclic graph is obtained by following this rule for all vertex pair. This graph is known as a graph of the algorithm.

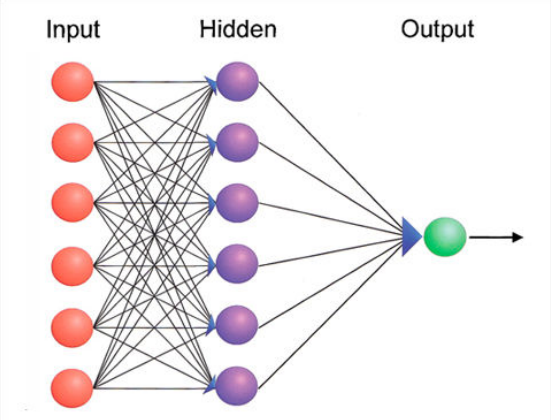

Fig. 3. The graph of the proposed neural network.

The graph of algorithm for the proposed ANN is shown in Fig. 3. The input layer is selected as the first floor, which is the group of mutually independent operations. The second floor is the hidden layer. Thus the parallel form for the graph of the algorithm for ANN is constructed.

Claim [17]. There is a fixed rounding procedure, where the round-off error depends only on the input operations. Suppose the algorithms performing one and the same set of arithmetic

operations. In order that influence of round-off errors for the same input data of the algorithms to be the same, the isomorphism for graphs of algorithms is sufficient and necessary.

According to this claim the parallel form for the algorithm saves the round-off error while performing mathematically equivalence conversions.

E. Serial program realization

The mathematical program package MATLAB 2011b was used for ANN simulation. The dataset is the sample of descriptors calculated from ligand structure and energy parameters of protein-ligand complex for 63 PR inhibitors. The input is a set of 11 values and output has one value of pK_i. The linear regression model gives $R^2 = 0.38$, indicating the bad predictive power for linear model. As mentioned above it is possible to define the nonlinear dependence in an implicit form using neural networks.

Fig. 4. Comparison of target and ANN output values of pK_i for training set.

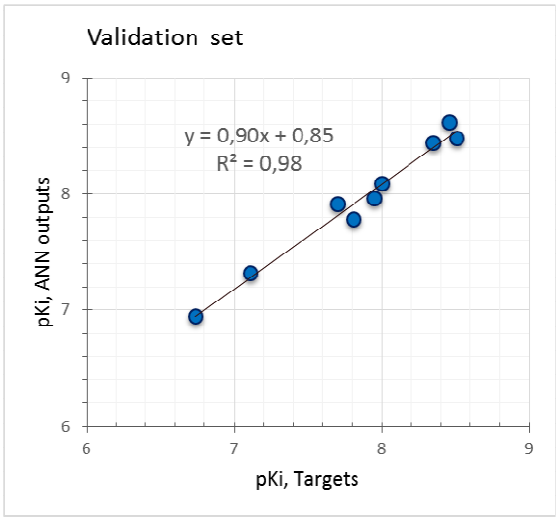

Fig. 5. Comparison of target and ANN output values of pK_i for validation set.

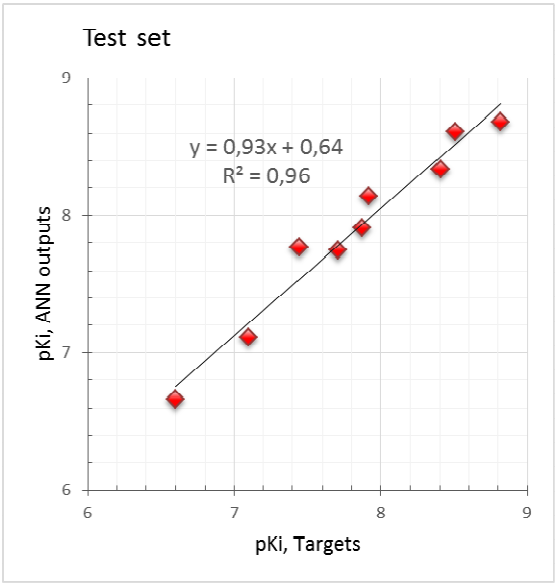

Fig. 6. Comparison of target and ANN output values of pK_i for test set.

The dataset was divided into training (45 events), validation (9 events) and test (9 events) sets. Fig. 4 shows the correlation of ANN outputs and target values with $R^2 = 0.95$ for training set. Fig. 5 shows $R^2 = 0.98$ for validation set, and Fig. 6 illustrates $R^2 = 0.96$ for test set. The error histogram for all mentioned sets is shown in Fig. 7. The root-mean-squared-error (RMSE) for test set is 0.15.

The LOO-CV procedure was used. The dependence target and ANN output values of pK_i is shown at Fig. 8. RMSE for all 63 points equals 0.17.

According to the used validation method mentioned above frequency of occurrence for MSE was calculated for 100 times of set dividing. Fig. 9 shows its cumulative distribution function.

Cross-validation error for test sets is $CV_{test} = 0.2$. The using validation method allows estimating RMSE median value as $\mathrm{RMSE}_{1/2} = 0.14$, and that RMSE is less than 0.34 with probability 0.9.

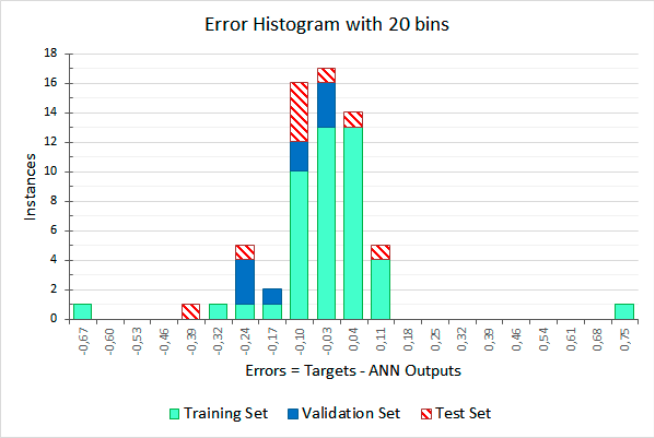

Fig. 7. Error histogram of target and ANN output values of pK_i for training set, validation and training sets.

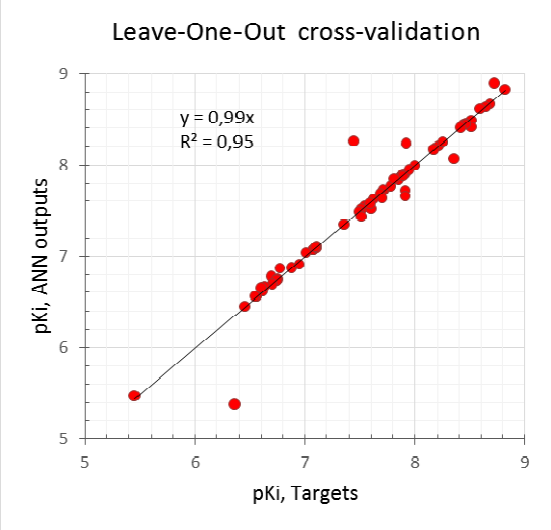

Fig. 8. Comparison of target and ANN output values of pK_i for Leave-One-Out cross-validation.

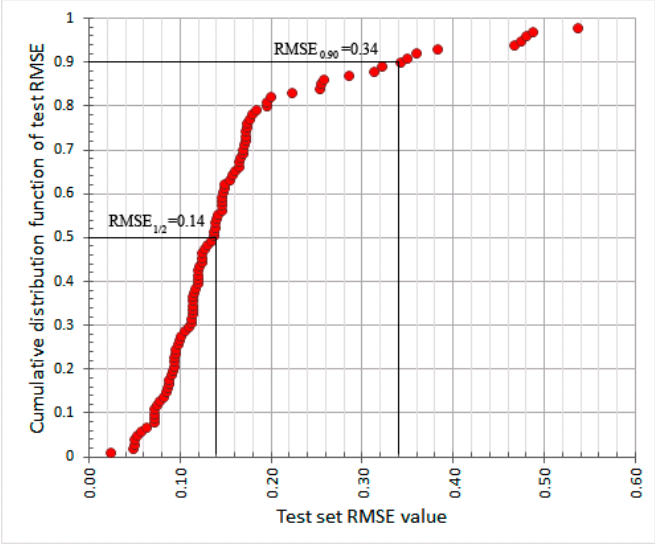

Fig. 9. Cumulative distribution function of test set RMSE.

F. Parallel program realization

The Compute Unified Device Architecture (CUDA) allows developers to use the C programming language for the development of general-purpose applications using fine-grain parallelism. CUDA is currently supported only on NVIDIA GPUs. A simple extension to C had invoked that more non-graphics developers port their existing applications to CUDA. CUDA consists of a runtime library, advanced libraries, tools, and an expanded version of C. CUDA gives developers access to the native instruction set and memory of the parallel computational elements in CUDA GPUs. It includes the CUDA Instruction Set Architecture and the parallel compute engine in the GPU.

The MATLAB serial realization was rewritten on C++ and the parallel algorithm was implemented using NVIDIA graphical processor unit and its CUDA 4.2 technology [18]. One of successful GPU using is presented in [26].

The calculations were performed using hybrid computing system based on HP Proliant G7 server platform (AMD Opteron 6100 processor) with TESLA S2050 "Fermi" GPU computing system. Fig. 10 shows a simplified hardware block diagram for the NVIDIA "Fermi" GPU architecture. Fermi contains up to 512 general purpose arithmetic units known as "streaming processors" (SP) and 64 "special function units" (SFU) for computing special transcendental and algebraic functions not provided by the SPs. Memory load/store units (LDST), texture units (TEX), fast on-chip data caches, and a high-bandwidth main memory system provide the GPU with sufficient operand bandwidth to keep the arithmetic units productive. Groups of 32 SPs, 16 LDSTs, 4 SFUs, and 4 TEXs compose a "streaming multiprocessor" (SM). One or more CUDA "thread blocks" execute concurrently on each SM, with each block containing 64–512 threads.

Three major classifications of computational kernels were identified: matrix operations, random number generation and sigmoid transfer function. These three kernels were generated and tested individually and then integrated at a final stage.

For matrix operations the CUBLAS Library [19] was used. It is an implementation of BLAS (Basic Linear Algebra Subprograms) on top of the NVIDIA CUDA runtime. It allows to access the computational resources of NVIDIA Graphics Processing Unit. To use the CUBLAS library, the application must allocate the required matrices and vectors in the GPU memory space, fill them with data, call the sequence of desired CUBLAS functions, and then upload the results from the GPU memory space back to the host. The CUBLAS library also provides helper functions for writing and retrieving data from the GPU.

Random number generation was used during ANN training and cross-validation. The CURAND Library [20] was taken for it. It provides facilities that focus on the simple and efficient generation of high-quality pseudorandom and quasirandom numbers. A pseudorandom sequence of numbers satisfies most of the statistical properties of a truly random sequence but is generated by a deterministic algorithm. A quasirandom sequence of n-dimensional points is generated by a deterministic algorithm designed to fill an n-dimensional space evenly. CURAND consists of two pieces: a library on the host (CPU) side and a device (GPU) header file. The second piece of it was used. This file defines device functions for setting up random number generator states and generating sequences of random numbers.

In our ANN the approximation of tangent hyperbolic activation function was used, which includes exponential function. According to Appendix C.2 from the CUDA Programming Guide [21], intrinsic functions (i.e. used in the kernel **__expf(-x)**)are known to be the less accurate, but faster versions of the non-underscored calls. This most probably implies that all SPs will be capable of doing the calculation. This accuracy loss is certainly acceptable trade to gain more performance.

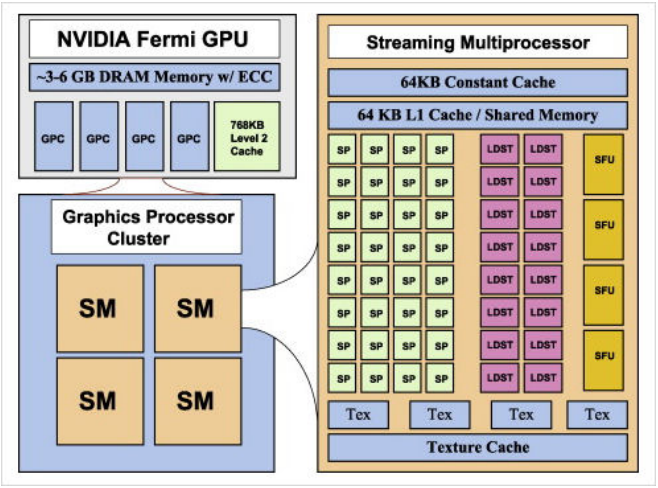

Fig. 10. A simplified hardware block diagram for the NVIDIA "Fermi" GPU architecture [22].

The separate module was developed for cross-validation methods. LOO-CV can be executed in parallel because for every event deletion the method is independent. Another method based on K-fold cross-validation was also parallelized for every random dividing.

The use of these integrations allowed achieving the maximum efficiency. Results of parallel and serial calculations gave the similar values and general speed-up was 734 as compared with serial MATLAB realization.

IV. Conclusion

The feed forward back propagation neural network for estimation of inhibition constant pK_i for probable PR inhibitors was projected and showed good predictive ability. For training set the correlation coefficient was $R^2 = 0.94$ and mean-square-error was 0.02. Various types of cross-validation methods showed the stability of used ANN. This neural network model should be used for affinity estimation with high accuracy. Additionally, this research showed that CUDA implementations can be well suited for neural network applications. The GPU parallel program realization was implemented based on serial MATLAB realization. The speed-up of it was 734. The use of graphical processor units with corresponding software implementation will significantly increase the size of the processed data, while reducing time-consuming.

Acknowledgment

The work was partially supported by the Russian Ministry of Education and Science (Grant No. 2012-1.1-12-000-2008-069, Agreement No.8274).

References

[1] A. S. Ivanov, A. V. Veselovsky, A. V. Dubanov, V. S. Skvortsov, "Bioinformatics Platform Development: From Gene to Lead Compound", *Methods Mol Biol.* Vol.316, 2006, pp. 389-431.

[2] M. Lawrenz, J. Wereszczynski, J.-M. Ortiz-Sánchez, S. E. Nichols, J. A. McCammon, "Thermodynamic integration to predict host-guest binding affinities", *Journal of Computer-Aided Molecular Design*, Vol.26, No.5, 2012, pp. 569-576.

[3] Söderholm AA, Lehtovuori PT, Nyrönen TH. Docking and three-dimensional quantitative structure-activity relationship (3D QSAR) analyses of nonsteroidal progesterone receptor ligands. *J. Med. Chem.* Vol. 49 (14), 2006, pp.4261-4268.

[4] Williams Sh. P., Sigler P. B. Atomic structure of progesterone complexed with receptor. *Nature*. Vol. 393.1998. pp. 392-396.

[5] Sybyl 8.1.Tripos Inc., St. Louis, USA.

[6] I. D. Kuntz, J. M. Blaney, S. J. Oatley, R. Langridge, T. E. Ferrin, "A geometric approach to macromolecule-ligand interactions", *J. Mol. Biol.*, Vol.161, No.2, 1982, pp. 269-288.

[7] Amber 9. Available: http://www.ambermd.org

[8] P. A. Kollman, I. Massova, C. Reyes, B. Kuhn, S. Huo, L. Chong, M. Lee, T. Lee, Y. Duan, W. Wang, O. Donini, P. Cieplak, J. Srinivasan, D. A. Case, T. E. Cheatham, "Calculating structures and free energies of complex molecules: combining molecular mechanics and continuum models", *Accts. Chem. Res.*, Vol.33, No.2, 2000, pp. 889-897.

[9] A. N. Kolmogorov, "On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition", *Dokl. Akad. Nauk SSSR*, Vol.114, No.5, 1958, pp. 953–956.

[10] A. N. Gorban, V. L. Dunin-Barkovsky, A. N. Kirdin, E. M. Mirkes, A. Yu. Novokhodko, D. A. Rossiev, S. A. Terekhov and etc. *Neiroinformatika*, Novosibirsk, Naika, 1998.

[11] D. E. Rummelhart, G. E. Hinton, R. J. Williams, "Learning internal representations by error propagation", *Vol. 1 of Computational models of cognition and perception*, Cambridge, MIT Press, 1986, pp. 319–362.

[12] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd Ed., Pearson Education, Delhi, 1999.

[13] M. Hagan, M. Menhaj, "Training Feedforward Networks with the Marquardt Algorithm", *IEEE Transactions on Neural Networks*, Vol.5, No.6, 1994. pp. 989-993.

[14] K. Levenberg, "A Method for the Solution of Certain Non-linear Problems in Least Squares", *Quarterly of Applied Mathematics*, 1944, 2(2), pp. 164-168.

[15] D. W. Marquardt, "An Algorith for the Least-Squares Estimation of Nonlinear Parameters", *SIAM Journal of Applied Mathematics*, 1963, 11(2), pp. 431-441.

[16] P. Pivoňka, J. Dohnal, "On-line Identification Based on Neural Networks Using of Levenberg-Marquardt Method and Backpropagation Algorithm", *WSEAS Transactions on Systems*, Issue 2, Vol.3, pp.381-385, April 2004.

[17] V. V. Voevodin, Vl. V. Voevodin, *Parallel calculations*. St. Petersburg, 2002.

[18] NVIDIA CUDA Technology. Available: http://www.nvidia.com/CUDA

[19] NVIDIA CUDA Toolkit 4.2 CUBLAS Guide, February 2012. Available: http://developer.download.nvidia.com/compute/DevZone/docs/html/CUDALibraries/doc/CUBLAS_Library.pdf

[20] NVIDIA CUDA Toolkit 4.2 CURAND Guide, March 2012. Available: http://developer.download.nvidia.com/compute/DevZone/docs/html/CUDALibraries/doc/CURAND_Library.pdf

[21] "NVIDIA CUDA C Programming Guide version 4.2". Available: . http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf

[22] J. E. Stone, D. J. Hardy, I. S. Ufimtsev, K. Schulten, "GPU-accelerated molecular modeling coming of age", *J Mol Graph Model.*, Vol.29, No.2, 2010, pp. 116-125.

[23] Y. Ozel, I. Guney, E. Arca, "Neural Network Solution to the Cogeneration System by Using Coal", *12th WSEAS International Conference on Circuits*, Heraklion, Greece, July 22-24, 2008.

[24] S. Sureerattanan, H. N. Phien, et. al, "The Optimal Multi-layer Structure of Backpropagation Networks", *Proceedings of the 7th WSEAS International Conference on Neural Networks*, Cavtat, Croatia, June 12-14, 2006.

[25] I. Fedyushkina, I. Romero Reyes, "Prediction of Glucocorticoid Receptor Inhibition by High-Performance Neural Network Algorithm", *Advances in Mathematical and Computational Methods. Mathematics and Computers in Science and Engineering Series*, Ed. Prof. Mihaiela Iliescu, 2012, No 4, pp. 203-208.

[26] S. Romero, M. A. Trenas, E. Gutierrez, E. L. Zapata, "Locality-Improved FFT Implementation on a Graphics Processor", *Proceedings of the 7th WSEAS Int. Conf. on Signal Processing, Computational Geometry & Artificial Vision*, Athens, Greece, August 24– 26, 2007.