SNC: THE SOFTWARE NEUROCOMPUTER WITH MODULAR ARCHITECTURE

I.MISUNO, D.RACHKOVSKIJ, E.REVUNOVA, A.SOKOLOV
V.M.Glushkov Cybernetics Center, National Ukrainian Acad. Sci., Kiev, Ukraine
dar@infrm.kiev.ua

NEURAL NETWORK INFORMATION TECHNOLOGIES AND NEUROCOMPUTERS

Since the end of 80s, we have observed a dynamic development of information technologies that make use of neural networks (NNs) for information processing. Various implementations of neural network models have been named neurocomputers (NCs) [1]. Four levels of NCs can be distinguished [18]:
Level 0. Theory and algorithms. Descriptions of various NN models.
Level 1. Software. Software implementations of various NN models using conventional computers.
Level 2. Software-hardware. Coprocessors accelerating an emulation of NNs.
Level 3. Hardware. Physical implementations of NNs.

The peculiarities of NN operations, as well as the parallelism of NN models and algorithms encourage to implement them with specialized computational devices. These are the "traditional" NCs [2] corresponding to levels 2-3. They were widely created at the end of 80s and beginning of 90s, when universal computers did not provide an adequate computational power for NN calculations. Mainly, these were vector coprocessors with the SIMD architecture, e.g., ANZA Plus from HNC or DELTA II from SAIC. A number of NCs have been created in the V.M.Glushkov Cybernetics Center [9][10][3], as well as in other institutions of the former USSR.

In practice, those software-hardware NCs turned out to be inefficient for the tasks of NN investigation, as well as for developing and running NN applications. A small volume of production did not allow the price of NCs to be reduced, whereas huge investments into the progress of usual PCs did not permit NCs to keep up with their pace of performance growth. Another drawback of traditional NCs is that the execution speed, taken alone, does not determine the expended time and cost of research, development, and application of NN technologies. Apparently, the computational complexity of the algorithms employed, the efficiency of data representations, algorithmic implementations, interactions with the user, and the efficiency of the development process are of primary importance.

We define a modern neurocomputer as a means for the efficient development and implementation of neural network information technologies. The neurocomputer developed by us is a software neurocomputer (SNC) based on the employment of power, programming technologies, and user-friendly interfaces of modern PCs.

The *efficiency of development* is achieved by the employment of the modular COM software architecture; providing a convenient user interface; standardization of the internal data formats and program blocks; providing a library of the developed and ready-to-use program blocks; providing a convenient creation and usage of new blocks; providing other resources to facilitate development (e.g., tools for testing classification algorithms, statistical processing, documenting, etc.).

The *efficiency of implementation* is achieved by the choice of NN architectures with a rather low computational complexity; an optimized algorithmic and software realization; the use of the PCs with an optimized configuration.

Thus, the created software neurocomputer will allow a wide range of tasks (from the development and investigation of novel NN architectures to the development and running of NN applications) to be solved by users who are not experts in those areas.

THE ARCHITECTURE AND IMPLEMENTATION OF THE NEUROCOMPUTER

According to the structure of information processing tasks, SNC includes units implementing the following functions: data supply; preprocessing; processing; auxiliary; user interface and system configuration.

Data supply units select input data and transform them to the formats comprehensible by subsequent units. The data can be artificial or real, and can be fed, e.g., from files. There is a generator of artificial data producing vectors with real-valued components and controlled parameters [16]. They can be used for testing classifiers.

Preprocessing units deal mainly with real-world data. For example, images are preprocessed by centering, scaling, feature extraction, etc. [1][11][3]. For acoustical signals, a multi-band digital filtration [15] is used.

Processing units process data according to their algorithms. In the current SNC version, they are mainly modifications of Random Threshold Classifiers [4][8][12]. Such a NN architecture provides a high performance and classification quality and ability to deal with data of complex class boundaries in the attribute space. Classifiers operate in two main modes - training and testing.

Auxiliary units implement statistical and other processing of the results, as well as visualization and storage tasks. The user interface unit allows a visual construction of the system configuration, the data link configuration, the parameters, and the execution algorithm. The constructed system configuration is saved into a file, and can be then executed either in the console mode or in the windowed mode.

SNC was implemented with Visual C++ 6.0. A single-thread, single-process, single-apartment COM programming architecture was used. The COM technology allows additional processing modules to be added to the program architecture without project rebuilding. There are now two kinds of user-programmable modules that can be added to the system, namely, processing modules and data format modules. Data format modules perform recognition of the input data and allow a generalized access to various input data formats via system-standard COM interfaces. Processing modules perform the actual processing of input data and are the basic system configuration units. Each processing module has a number of input and output data buffers, to be connected with other format and processing modules. To implement a new class of the processing units, a user implements a specified interface that allows the system to recognize this new object and use it in configurations along with all other objects.

SNC allows a sequence of processing units to be executed in a predefined order. A current project configuration (the used processing units, the data link configuration, the execution sequence) is stored in an external XML file providing the possibility of a convenient cross-platform exchange of implemented project configurations.

Each processing unit is considered to be a “black box” that processes a set of the input data and forms a set of the output data; has an arbitrary number of the input and output data buffers; can implement its own data transport mechanism; has two sets of parameters: initial and working set parameters, through which the unit behavior is set up and controlled by the host system; is implemented as a COM object supporting interface(s) recognizable by the host system and the other processing units.

The implemented software architecture specifies three data exchange modes: external data buffers through which data are transferred between processing units (the passive data model); internal data buffers implemented as the part of processing units (the active data model); and using processing units themselves as active data buffers allowing direct data requests from one processing unit to another (the modified active data model). As an alternative, to implement a more flexible program architecture, each processing unit can implement its own data transport interfaces (provided that the other processing units can recognize the implemented interfaces).

AN INVESTIGATION OF A REAL-VALUED ATTRIBUTE VECTOR CLASSIFICATION

The system prototype for an investigation of a real-valued vector classification includes subsystems for data generation and classification.

Data generation is carried out by the DataGen unit that allows data sets with real-valued attributes to be generated varying: the number of input attributes; the number of output classes; the complexity of class regions; the distributions of data samples (normal or random); the noise level ("measurement noise", "input errors", "teacher errors"); the number of data samples; the number of realizations of class regions [16].

Classification is made by the classification units - mainly, RTC/RSC NN classifiers [4] [12] and their modifications. The RTC classifier consists of three neuron layers: *A* input neurons in the first layer according to the number of the input attributes (components of a real-valued input vector), *N* binary neurons in the second layer, and *C* output neurons corresponding to the classes.

Binary neurons of the second layer are connected to all neurons of the input layer. Any neuron of the second layer outputs 1, if each of the input values lies between some thresholds assigned to that neuron, and 0 otherwise. Therefore, each second-layer neuron has a hyper-rectangle receptive field in the input space. Before the start of the training, all thresholds of the second-layer neurons are generated randomly and independently for each of the input dimensions (attributes). The number *N* of neurons in the second layer is large (typically, many thousands) in order to cover each point of the input space by a large enough number of different but overlapping hyper-rectangles, so that their intersection has a small volume allowing complex class regions in the input space to be approximated (see Fig. 1).

During training, only the weights of interconnections between the second layer and the output layer neurons are modified according to some version of learning rules for simple perceptrons.

The model of the RTC classifier emulated with SNC has been very efficient in: the training time, the recognition time, recognition rate, where it outperformed the nearest neighbor classifiers, the potential function classifiers, the Gaussian classifiers, and the multi-layer perceptrons trained by the standard back propagation procedure (see also [4]).

In the Random Subspace Classifier (RSC), only hyper-rectangles with the random thresholds generated for a randomly chosen subset *S* of the whole set of *A* dimensions are used as the receptive fields [12]. This allows a substantial reduction of the execution time. For example, for *A*=100 *S* can be chosen equal to 4-5 providing an acceleration by a factor of 20-25.

In the considered classifiers, the thresholds in various dimensions are chosen randomly and independently from any training set. To adapt the classifier structure to the peculiarities of a given classification problem, the distribution of the centers of hyper-rectangles for a certain dimension can be chosen as an approximation of the density distribution of the training samples for this selected dimension. The lengths of the hyper-rectangles' sides for a selected dimension can also be chosen in some relation to the distribution of the test samples.

CLASSIFICATION OF HANDWRITTEN CHARACTERS

The SNC configuration for a handwritten digit recognition is as follows. The input data are the files from the MNIST database [13]: a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image of 28x28 pixels each. They are read by the SNC unit-reader for this database format (.idx).

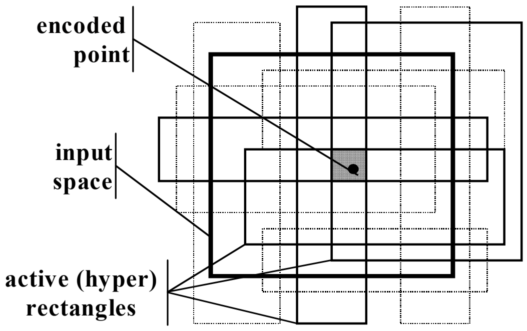

Fig. 1. Encoding by the second-layer neurons in RTC/RSC classifiers

Then, the unit implementing the binary LIRA classifier [7][6] is used. It is a modification of the RSC classifier for binary images. It may also be considered as a development of Rosenblatt's perceptron [5]. Each neuron of the second layer of this classifier detects a particular combination of 1s ("positive pixels") and 0s ("negative pixels") in the input image, thus extracting a particular feature. The dimensions of the receptive field of each neuron-detector are limited, and the number of inspected pixels is fixed and the same. Both are determined by the corresponding parameters. The receptive field size used was 10x10. Positions of receptive fields and coordinates of inspected positive and negative pixels inside the receptive field of each feature were determined randomly for each neuron of the second layer. Thus, a neuron of the second layer was only active if all image pixels corresponding to the coordinates of its positive pixels were 1 and all image pixels corresponding to the coordinates of its negative pixels were 0 (see Fig. 2).

Training of the output layer was implemented using the rule for simple perceptron. The training process comprised up to 40 passes through the training set of 60,000 examples. The number of errors on the testing set decreases with the increasing number *N* of the second-layer neurons. For example, the number of errors *E* per 10,000 test examples was *E*=1150 for *N*=4,000; *E*=330 for *N*=32,000; and *E*=175 for *N*=256,000 (see also [5]). The training time for *N*=256,000 comprised 35 passes that lasted 10 hours on a 800 MHz PC.

To decrease further the number of errors, an extended training set was formed by augmenting the original training set with artificially distorted versions of the original training samples. The distortions were shifts and skewing. This allowed 65 errors per 10,000 test examples to be reached, which is better than the results obtained with all other approaches, as far as we know.

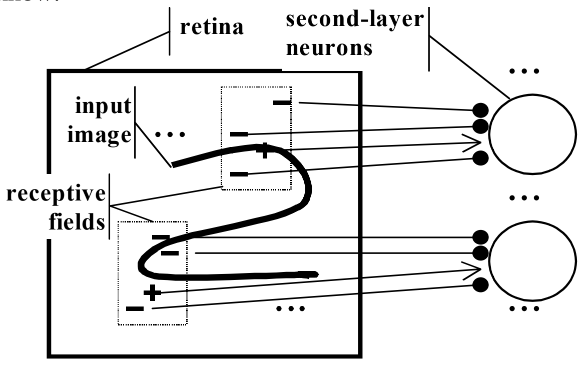

Fig 2. Encoding in LiRA classifiers

IDENTIFICATION OF A SPEAKER BY VOICE

For this task, the preprocessing unit performs a multi-band digital filtration [15]. The filtration results may be considered as a 2D gradual image, where the X axis represents time T, the Y axis represents frequency F, the value of pixel (x,y) represents the output of the corresponding filter at the corresponding point in time (see Fig. 3).

A modification of the LIRA classifier is used as the processing unit. It may be considered as a combination of binary LIRA and RSC for dealing with gradual images. In each point of the receptive field of a certain neuron of the second layer, the pixel value is compared with the threshold value for that point. If the conditions of all checks are satisfied, the feature is considered present, and the corresponding neuron is activated.

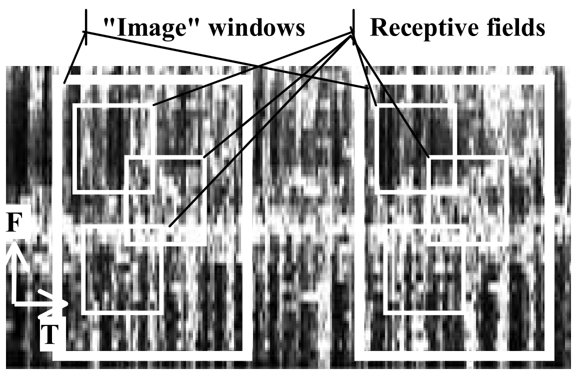

Fig. 3. Filtration results and application of LIRA classifier

The "image" (the result of filtration) is split into a number of adjacent windows. The classifier is trained on each window of each voice to be identified. The identification process of a new voice recording is implemented by "voting". The classifier outputs the recognition result (the speaker's name) for each input window of the filtered voice to be identified. The "winner" is the voice that was recognized in the majority of windows.

CONCLUSION

In conclusion, SNC provides an effective means for the development and implementation of neural network information technologies in various application areas. This has been achieved by the employment of original neural network paradigms and implementation of various processing units as COM objects. The system architecture allows a flexible configuration for solving specific problems and a straightforward integration of new units which were developed with due regard for the system requirements.

We intend to expand the range of the SNC applications through the problems of intelligent information access and management, as well as natural language processing taking into account the semantics and structure of textual data and using analogical reasoning [14]. These are based on the advantages of the structure-sensitive binary sparse distributed representations [17] in the framework of Associative-Projective Neural Networks [10][3].

REFERENCES

1. Amosov, N.M., Baidyk, T.N., Goltsev, A.D., Kasatkin, A.M., Kasatkina, L.M., Kussul, E.M., & Rachkovskij, D.A. (1991) Neurocomputers and intelligent robots. Kiev: Naukova dumka. (In Russian).
2. Hecht-Nielsen, R. (1986) Performance Limits of Optical, Electro-Optical and Electronic Artificial Neural System Processors. Proc. Soc. Photo-Opt. Instrum. Eng., 634. p. 277.
3. Kussul, E.M. (1992) Associative neuron-like structures. Kiev: Naukova Dumka. (In Russian).
4. Kussul, E.M., T.N. Baidyk, VV. Lukovich, D.A. Rachkovskij, Adaptive High Performance Classifier Based on Random Threshold Neurons, Proc. of Twelfth European Meeting on Cybernetics and Systems Research (EMCSR- 94), Austria, Vienna, 1994, pp. 1687-1694.
5. Kussul, E.M., Baidyk, T.N., Kasatkina, L.M. & Lukovich V.V. (2001). Neural network system for continuous handwritten words recognition. In: Proc. of the Intern. Joint Conference on Neural Networks (Washington, DC).
6. Kussul, E.M., Baidyk, T.N., Rachkovskij, D.A. (1995). Application of Neural Network Classifiers for the OCR of Printed Texts. In: Proc. of the Second International Symposium on Neuroinformatics and Neurocomputers, Rostov-on-Don, Russia, Sept. 20-23, pp. 1-6.
7. Kussul, E.M., Kasatkina, L.M., Lukovich, V.V. (1999) Neural classifiers for handprinted character recognition. USIM (Control Systems and Computers), N4, pp. 77-86.
8. Kussul, E.M., Kasatkina, L.M., Rackovskij, D.A., Wunsch, D.C. (1998) Application of Random Threshold Neural Networks for Diagnostics of Micro Machine Tool Condition. In: "Proceedings of IJCNN'98", IEEE, Piscataway, NJ, Vol. 3, pp. 1685-1687.
9. Kussul, E.M., Lukovich, V.V, Lutsenko, V.N. (1988) Multiprocessor computational devices for robot control in natural environment. Control systems and machines, no 5, pp.102-105.
10. Kussul, E. M., Rachkovskij, D. A., & Baidyk, T. N. (1991a) Associative-Projective Neural Networks: architecture, implementation, applications. In Proc. of the 4th Intern. Conference "Neural Networks & their Applications", Nimes, France, Nov. 4-8, 1991, pp. 463-476.
11. Kussul, E. M., Rachkovskij, D. A., & Baidyk, T. N. (1991b) On image texture recognition by associative-projective neurocomputer. In C. H. Dagli, S. Kumara, & Y. C. Shin (Eds.), Proceedings of the ANNIE'91 conference "Intelligent engineering systems through artificial neural networks", pp. 453-458. ASME Press.
12. Kussul, E.M., Rachkovskij, D.A., & Wunsch, D. (1999) The Random Subspace coarse coding scheme for real-valued vectors In: Proceedings of the International Joint Conference on Neural Networks, Washington, DC, July 10-16, 1999. 6 pp.
13. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998) Gradient-based Learning Applied to Document Recognition, Proceedings of the IEEE, v. 86, N 11, pp. 2278-2344.
14. Rachkovskij, D.A. (2001) Representation and Processing of Structures with Binary Sparse Distributed Codes. IEEE Transactions on Knowledge and Data Engineering, 13(2), pp. 261-276.
15. Rachkovskij, D. A. & Fedoseyeva, T.V. (1990) On audio signals recognition by multilevel neural network. In Proceedings of The International Symposium on Neural Networks and Neural Computing - NEURONET'90, pp. 281-283. Prague, Czechoslovakia.
16. Rachkovskij, D.A. & Kussul, E.M. (1998) DataGen: a generator of datasets for evaluation of classification algorithms. Pattern Recognition Letters 19, pp. 537-544.
17. Rachkovskij, D. A. & Kussul, E. M. (2001) Binding and Normalization of Binary Sparse Distributed Representations by Context-Dependent Thinning. Neural Computation 13(2), pp. 411-452 (http://cogprints.soton.ac.uk/documents/disk0/00/00/12/40/index.html).
18. Stubbs, D. (1988) Neurocomputers. M.D. Computing, 5, N3, pp. 14-24.