Software technology for deep learning of belief neural networks

1st Victor V. Krasnoproshin
Belarusian State University
Minsk, Belarus
Krasnoproshin@bsu.by

2nd Vadim V. Matskevich
Belarusian State University
Minsk, Belarus
Matskevich1997@gmail.com

***Abstract*—The paper provides the framework structure and contents description for solving applied problems using deep belief networks. Original network architecture, focused on parallel data processing, set of algorithms implementing training processes based on the annealing method and solving problems are proposed.**

The effectiveness of the described framework is demonstrated by the example of solving the problem of compressing color images.

***Index Terms*—framework, annealing method, deep belief network, parallel computations, training, dataset**

I. Introduction

Currently, a wide range of applied problems is being solved using neural network technologies implemented in the form of frameworks. A framework is a software package that implements a specific architecture of a neural network to solve a specific range of tasks.

The most difficult stage of neural network data processing is the network training process [1] [2]. The existing today frameworks use mainly gradient training methods [3]. With visible popularity, gradient methods have certain disadvantages. Therefore, the problem of training is still relevant [4].

The paper offers a description of the original framework implementing the architecture of deep belief networks, for the training of which the annealing method is used. This method is lack of the main disadvantages of gradient methods, but it works much slower [5]. An approach to solving this problem is proposed, and the efficiency of the proposed framework is demonstrated by example of solving the problem of compressing color images.

II. Problem Analysis

Deep belief networks are used to solve a number of applied problems such as: medical diagnostics, pattern recognition, image processing, selection of semantically significant features, etc. [6].

In order to describe the architecture of a deep belief network, it is necessary first to describe the architecture of the restricted Boltzmann machine. It is known that any deep belief network always contains layers of this type of machine.

At the heart of the machine is the concept of a stochastic neuron.

Formally, a restricted Boltzmann machine can be represented as a fully connected bipartite graph $G = (X, U)$,

$$\begin{cases} X = X_1 \cup X_2, X_1 \cap X_2 = \emptyset \\ U = \{u = (x_1, x_2) | \forall x_1 \in X_1, \forall x_2 \in X_2\}, \end{cases} \tag{1}$$

where X – vertex set – stochastic neurons, U – edges set – synaptic connections, while vertices of subset X_1 -– set the neurons of the input layer, X_2 – output layer neurons.

The number of neurons in the input layer is determined by the size of the input image, and the number of neurons in the output layer is determined based on the requirements for data compress ratio.

The output signals of layers of a restricted Boltzmann machine implement some laws of the probability distribution. Different types of machines are built depending on the laws of distribution used. In this paper, we will talk about machines types of Gauss-Bernoulli and Bernoulli-Bernoulli, because they are the most common.

For restricted Boltzmann machine of Gauss-Bernoulli type to each vertex of the input layer we assign a set of parameters $VB = \{b\}$ – vertex offsets and $\sigma = \{\sigma\}$ – vertex variances, and to the vertices of the output layer – set of parameters $HB = \{g\}$ – vertex offsets. The sizes of the sets are equal respectively

$$|VB| = |\sigma| = |X_1|, |HB| = |X_2| \tag{2}$$

Each edge connecting a pair of vertices of the input and output layers will be assigned a set of parameters $W = \{w\}$ – the weights of the edges.

The size of the set is equal to the following value

$$|W| = |X_1||X_2| \tag{3}$$

Thus, the described family of neural networks can be defined by four types of parameters:

$$RBM = (W, VB, \sigma, HB) \tag{4}$$

Note. A restricted Boltzmann machine of Bernoulli-Bernoulli type does not have set of parameters σ.

A deep belief network contains several layers consisting of restricted Boltzmann machines and, in addition, for generating the output signal may contain a multilayer perceptron (depending on the problem being solved).

A deep belief network in layers consisting of restricted Boltzmann machines solves the problem of data compression, which can be formally described as follows.

Let X be the space of input images of some fixed dimension, Y – the space of compressed images of much smaller dimension than the space X. I.e:

$$\begin{cases} dimX = fix \\ dimY \ll dimX \end{cases} \tag{5}$$

Then the task of data compression is to build compression functions f and recovery g, such that:

$$\begin{cases} f : X \to Y, g : Y \to X \\ d : X \times X \to \Re \\ d(x, g(f(x))) \to min, \forall x \in X, \end{cases} \tag{6}$$

where d is a function that evaluates the differences between two given vectors.

Note. In practice, data compression is carried out for a specific subject area. This, in turn, imposes certain restrictions on the input data and, therefore, reduces the dimension of the space X.

As noted, the most time-consuming step in the use of neural networks is the training process. Since a deep belief network always contains layers of restricted Boltzmann machines, the effectiveness of training the entire network as a whole depends on the effectiveness of solving the problem. Network training can be written as an optimization problem for each of the layers.

Let a training dataset x and a functional for evaluating the quality of data compression d (6) be given. It is needed to find the values of the parameters $(w^*, b^*, g^*, \sigma^*)$, giving a minimum of functional F, i.e.

$$F(x, d, w^*, b^*, g^*, \sigma^*) = \min_{w,b,g,\sigma} F(x, d, w, b, g, \sigma) \tag{7}$$

Note. A restricted Boltzmann machine of Bernoulli-Bernoulli type does not contain the parameter σ and the quality functional F, therefore, does not depend on σ.

To solve optimization problems, you can use either the gradient descent method or random search.

The gradient descent method has fast convergence, but at the same time has several disadvantages:

1) converges to local minimum points [5], which significantly reduces the quality of the solution;
2) requires differentiability of the objective function, which significantly reduces the class of problems to be solved.

The random search method is not widespread [7], however, it has some advantages:

1) does not require objective function differentiability, which significantly expands the class of applied problems;
2) under certain conditions [8] and from any initial approximation [9] it has convergence to the global minimum.

Given the above, we obtain the following training task.

Let a training dataset of N input images of dimension $dimX$ be given and requirements for data compression be fixed, i.e. $dimY = fix$.

It is necessary to develop a deep belief network architecture and a training algorithm (based on the annealing method) so that the following conditions are met:

1) training time should be acceptable (no more than a day);
2) the quality of training should be as high as possible, while the algorithm should require as little data as possible for training.

III. Framework Description

To solve this problem, software was developed in the form of a framework that includes all the necessary algorithms that implement the functioning of deep belief networks.

The framework proposed in the work consists of five main modules: trainDeepNN, compressImages, decompressImages, loadFromFileDeepNN, buildDeepNN.

The compressImages module compresses color images. The decompressImages module - restoring original images from their compressed representation. The loadFromFileDeepNN module - loading the network from the hard drive.

The buildDeepNN module builds a deep belief network with the following architecture.

Since the input data are usually images, the first layer of a deep belief network is formed as an ensemble of M_1 restricted Boltzmann machines of Gauss-Bernoulli type. This made it possible to "cover" the entire numerical range of input data. All machines forming one layer have the same architecture within the layer, so dimX must be a multiple of M_1. All subsequent ones are represented by ensembles of restricted Boltzmann machines of Bernoulli-Bernoulli type. Therefore, for each network layer, the following restriction must be satisfied. The product of the number of machines in the layer by the size of the input layer of each should be equal to the product of the number of machines in the previous layer and the size of the hidden layer of each. The output layer of the network is represented by an ensemble of M_s restricted Boltzmann machines of Bernoulli-Bernoulli type. To complete the data compression requirement, the total number of neurons in the hidden layers of machines must be strictly equal to dimY. The number of adjustable parameters in each of the machines must be strictly less than N. This is necessary to ensure the generalizing ability of the network.

Let's show architecture with an example with four layers (see Fig. 1).

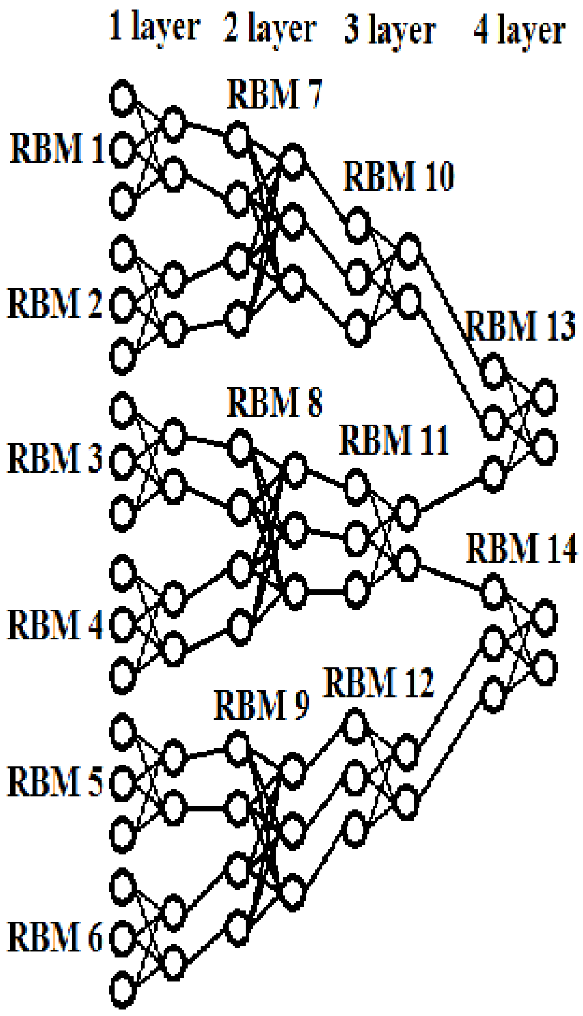

Figure 1. Original deep belief network architecture.

The first layer consists of an ensemble of six restricted Boltzmann machines. The input layer of each machine consists of three neurons, the hidden - of two. The second layer of the network consists of an ensemble of three machines. The size of machine input layer is four neurons, the hidden one is three. The third layer of the network consists of an ensemble of three restricted Boltzmann machines, each of which has three neurons in the input layer and two in the hidden one. The last layer of the deep belief network consists of an ensemble of two machines, each of which has three neurons in the input layer and two in the hidden one.

The proposed architecture has several advantages:

1) decomposition of network layers ensures complete independence of the trained machines-components within the network layer, which allows to parallelize the training process;
2) the architecture can significantly reduce the number of configurable network parameters, which reduces the training dataset size and significantly reduces the computational complexity of the training process;
3) the architecture fully meets the constraints of the problem for an effective solution using heterogeneous computing devices [10].

The trainDeepNN module implements the main function of the framework. He provides training of received deep belief network. The internal composition of this block is presented in the form of the following scheme (see Fig. 2). At the beginning of the module execution, OpenCL is configured on computing devices. Then, a deep belief network training cycle by layers begins. When moving to the next layer, data is preliminarily converted to the architecture of the current layer. After this, cyclic training of the restricted Boltzmann machines that form the current layer is started. The cycle includes initialization of the initial state of the machines, data transfer to computing devices and a training cycle using the original annealing method algorithm.

The following algorithm is proposed that implements the ideology of this method.

At the preliminary stage, initialization (setting the initial values) of the parameters (W, VB, HB, σ), initial temperature T_0 is performed.

The main stage of the training algorithm implements a procedure for sequentially updating the values of the specified parameters using a certain quality functional.

Describe the process to update settings in more detail. For simplicity, consider it on the example of the set of para-meters W. For other sets, this procedure is identical.

To the set of parameters W, we associate a segment L_w of length l. After that, each element of the set W is sequentially placed in the center of the given segment. To determine the direction of change of parameter values, we generate a random variable from 0 to 1. If it is more than 0.5, then the value of the parameter increases, otherwise it decreases.

New parameter values are defined as follows. A random permutation is generated, the number of elements of which is equal to the number of elements of the set W. We order the elements of the set W in accordance with the permutation and change the values of the first W_p elements of the set. The new value of the parameter is determined as a result of the implementation of a uniformly distributed random variable on the segment, the ends of which are the current value of the parameter, and the end of the segment towards which the change is made.

Similarly, actions are sequentially performed for the sets VB, HB, σ.

For newly obtained parameter values, the quality functional is calculated.

As the latter, it is proposed to use the following function:

$$F(W, VB, HB, \sigma) = \frac{1}{NR} \sum_{i=1,N} \sum_{j=1,R} |x_{ij} - f^{-1}(y_{ij})|, \tag{8}$$

where y_{ij} – reconstituted input signal of restricted Boltzmann machine, f^{-1} - inverse function of the preliminary transformation of input data.

Then a decision is made to move to a new state with probability:

$$P(y|x) = \min\{1, \exp(-(F(y) - F(x))/T_i)\}, \tag{9}$$

where x – current state, y – state selected for transition, F – minimized objective function, T_i - temperature of i-th iteration.

— in case of change of state cooling takes place by the rule:

$$T_{i+1} = T_0 / \ln(k+1), \tag{10}$$

where k is the number of completed transitions to a new state.

— otherwise the temperature does not change.

After cooling, the received solution is checked for optimality:

— the solution is optimal if the time allocated for training has expired.

If the received solution is optimal then:

— algorithm stop,

— otherwise move to the next iteration.

We will check the efficiency of using the neural network of the proposed architecture using the example of the problem of compressing color images.

IV. Experiments and Results

The «STL-10» data from the Stanford University repository was used as baseline [11]. The dataset contains one hundred thousand unmarked color images measuring 96x96 pixels. Each image is described by 27648 integer numbers (in the range from 0 to 255) specifying the content of red, green and blue colors [12]. Based on the characteristics obtained (the sample is given approximately $2,8 * 10^9$ numbers, contains descriptions of arbitrary, unmarked objects), we can conclude that the process of compressing images of a given sample with low losses is a rather difficult problem.

For data processing, a standard computer with an 4-core processor and a video card was used: video card: nvidia 1060 3gb; processor: intel i7 4770k 3.5 GHz; RAM: 2x8 Gb 1600 MHz; hard disk: samsung 850 pro 256 Gb; operating system: Lubuntu 16.04.

The compiler gcc was used as software (libraries OpenMP and CUDA version 9.1 [13]) with options: «gcc superOpenCLFramework.cpp -lstdc++ -D_FORCE_INLINES -O2 -l OpenCL -lgomp -lm -fopenmp». Measurement of operations time was performed using function «gettimeofday».

The following deep belief network architecture was used in the experiments.

The first layer consisted of a combination of 432 restricted Boltzmann machines of Gauss-Bernoulli type. The number of neurons in the input layer and hidden was 64 and 16, respectively, for all machines in the layer. The second layer consisted of a combination of 108 restricted Boltzmann machines of Bernoulli-Bernoulli type. The number of neurons in the input layer and hidden was 64 and 16, respectively, for all machines in the layer. The third layer consisted of a combination of 27 restricted Boltzmann machines of Bernoulli-Bernoulli type. The number of neurons in the input layer and hidden was 64 and 16, respectively, for all machines in the layer.

Images compression ratio was tuned by disabling the last layers of the deep belief network. So 3 layers of the net-work provided 512-fold compression, 2 layers – 128-fold, and the first layer of the network – 32-fold. For the training of the first layer of the network, 2000 images were used, for the second and third – 4000.

In experiments, we will use the number of bits to encode one pixel of the image as a compression ratio. For encoding without compression of one pixel of a color image, 24 bits are required. Reducing the number of bits for encoding a pixel leads to image compression, for example, with 24-fold compression, the number of bits per image pixel will be equal to one.

There are many different functionals for evaluating the quality of image compression [14], however, many of them are not widely used due to the fact that for their calculation it is necessary to fix the parameter values in the functional itself for the correct comparison of the results or require a large amount of calculations.

As a quality functional, to estimate losses during image compression, we will use the most common PSNR (peak signal-to-noise ratio) quality functional that does not require fixing parameters and a large amount of computations. This quality function calculates the ratio of the maximum possible signal to the mean square error for color raster images as follows.

$$PSNR(X, Y) = 10 \log_{10} \frac{3N * 255^2}{\sum_i (x_i - y_i)^2}, \tag{11}$$

where X, Y – compared images, N – the number of pixels in the image X.

The higher the value of the functional, the less compression loss and higher quality. For lossless compression, this quality function is equal to plus infinity. For a high compression ratio (approximately 0.05 bits per

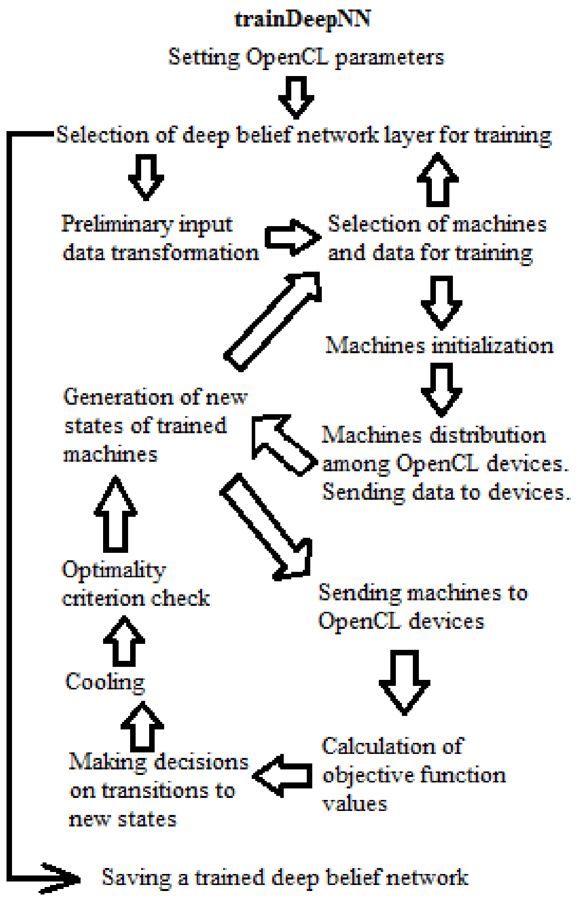
trainDeepNN
Setting OpenCL parameters
Selection of deep belief network layer for training
Preliminary input data transformation
Selection of machines and data for training
Machines initialization
Generation of new states of trained machines
Machines distribution among OpenCL devices. Sending data to devices.
Optimality criterion check
Sending machines to OpenCL devices
Cooling
Calculation of objective function values
Making decisions on transitions to new states
Saving a trained deep belief network

Figure 2. Function trainDeepNN.

pixel or less), values in the range of 8 and higher are considered adequate. For medium compression (1 bit per pixel), normal values are considered to be from 20 above.

Based on the results of the experiments, the following results were obtained (see Table 1):

Table I
DEEP BELIEF NETWORK TRAINING

Framework Efficiency	**Compress Ratio (bit/pixel)**		
	0,75	***0,1875***	***0,046875***
quality function (PSNR)	19	16,9	14,72
training time (h)	6	10	11

The results show the high efficiency of the deep belief neural network architecture and the training algorithm based on the annealing method. To configure a separate machine consisting of 1168 parameters in the first layer of the net-work, only 2000 images were needed, while for machines of subsequent layers consisting of 1104 parameters, only 4000 images were needed, which indicates a very high efficiency of the annealing method in training neural networks.

The time spent on training the network shows that, with the proper selection of the training algorithm parameters, the annealing method can have a high convergence rate.

The obtained results confirm the assumption that the annealing method can be used to train deep belief networks [15]. Considering that the STL-10 dataset is rather complicated, so in comparison with other results [16] [17], it can be argued that the developed original annealing method algorithm is quite efficient.

V. Conclusion

The paper provides a description of the framework for solving applied problems using a deep belief network. An original network architecture and a set of algorithms implementing training processes and problem solving are proposed.

The effectiveness of the framework functioning is demonstrated on the example of the problem of compression of colored images. The problem was solved on a computer with a multi-core processor and video card.

It is shown that the developed deep belief networks training algorithm lacks many of the disadvantages of gradient methods. Based on the obtained experimental results, we can conclude that the developed framework with appropriate revision (for a wider class of applied problems) has a certain potential.

The proposed framework technology focused on deep belief networks is easily integrated into the methodology of open semantic networks and can be used for further development of this technological area.

References

[1] V.V. Krasnoproshin, V.V. Matskevich Statistical approach to image compression based on a restricted Boltzmann machine // Proceedings of the 12-th International Conference “Computer Data Analysis and Modeling”- CDAM’2019, Minsk, 2019, -p.p. 207-213.
[2] Vadim Matskevich, Victor Krasnoproshin Annealing method in training restricted Boltzmann machine // Proceedings of the 14-th International Conference - PRIP’2019, Minsk, 2019, -p.p. 264-268.
[3] Aicher C., Foti N.J., Fox E.B Adaptively truncating backpropagation through time to control gradient bias [electronic resource]. – link: arxiv.org/abs/1905.07473 – Access date: 04.01.2020.
[4] Xavier Glorot, Yoshua Bengio Understanding the difficulty of training deep feedforward neural networks // Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, PMLR Vol.9, 2010, -p.p 249-256.
[5] Treadgold N.K., Gedeon T.D. Simulated annealing and weight decay in adaptive learning: the SARPROP algorithm // IEEE Transactions on Neural Networks vol. 9, Issue: 4 , Jul 1998.
[6] V.A. Golovko, A.A. Kroshenko Using deep belief neural networks to highlight semantically significant attributes // Proceedings of the 5-th International Conference - OSTIS’2015, Minsk, 2019, -p.p. 481-486.
[7] Locatelli M. Convergence properties of simulated annealing for continuous global optimization // Jornal of Applied Probability Vol. 33, Issue 4 December 1996, pp. 1127-1140.
[8] Hajek B. Cooling schedules for optimal annealing // mathematics of operations research vol.13, No 2, May 1988.
[9] Sanguthevar Rajasekaran On the Convergence Time of Simulated Annealing [electronic resource]. – link: repository.upenn.edu/cis_reports/356/ – Access date: 04.01.2020

[10] V.V. Krasnoproshin, V.V. Matskevich Effective Data Processing on Heterogeneous Computing Devices // Bulletin of Brest state technical university. Series - physics, math, computer science. 2018. Vol. 5 (113), -p.p 15-18.

[11] STL-10 dataset [electronic resource]. – link: academictorrents.com/ details/a799a2845ac29a66c07cf74e2a2838b6c5698a6a – Access date: 25.02.2018.

[12] STL-10 dataset description [electronic resource]. – link: stanford.edu/ acoates//stl10/ – Access date: 24.02.2018.

[13] CUDA toolkit [electronic resource]: – link: developer.nvidia.com/cuda-downloads – Access date: 23.02.2018.

[14] Dogancan Temel. Ghassan AlRegib Perceptual Image Quality Assessment through Spectral Analysis of Error Representations // Signal Processing: Image Communication, Vol. 70, 2019, -p.p 37-46, ISSN 0923-5965.

[15] L.M. Rasdi Rere, Mohamad Ivan Fanany, Aniati Murni Arymurthy Simulated Annealing Algorithm for Deep Learning // Procedia Computer Science Vol. 72, 2015 -p.p 137–144.

[16] Toderici G., Vincent D., Johnston N., Sung Jin Hwang, Minnen D., Shor J., Covell M. Full Resolution Image Compression with Recurrent Neural Networks [electronic resource]: – link: arxiv.org/abs/1608.05148 – Access date: 04.01.2020.

[17] Zhou X., Xu L., Liu S., Lin Y., Zhang L., Zhuo C. An Efficient Compressive Convolutional Network for Unified Object Detection and Image Compression // Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33, 2019. -p.p 5949-5956.

Программная технология глубокого обучения доверительных нейронных сетей

Краснопрошин В.В., Мацкевич В.В.

В докладе предлагается описание структуры и состава фреймворка для решения прикладных задач с использованием глубоких доверительных сетей. Предложены оригинальная архитектура сети, ориентированная на параллельную обработку данных, набор алгоритмов реализующих процессы обучения на основе метода отжига и решения задач. Эффективность работы описанного фреймворка демонстрируется на примере решения задачи сжатия цветных изображений.

Ключевые слова: Фреймворк, метод отжига, глубокая доверительная сеть, параллельные вычисления, обучение, выборка.

Received 12.02.2020