ISSN 0747-9239, Seismic Instruments, 2018, Vol. 54, No. 1, pp. 8–16. © Allerton Press, Inc., 2018.
Original Russian Text © K.V. Kislov, V.V. Gravirov, 2017, published in Seismicheskie Pribory, 2017, Vol. 53, No. 1, pp. 17–28.

Deep Artificial Neural Networks as a Tool for the Analysis of Seismic Data

K. V. Kislov* and V. V. Gravirov**

Institute of Earthquake Prediction Theory and Mathematical Geophysics, Russian Academy of Sciences, Moscow, Russia

**e-mail: kvkislov@yandex.ru*

***e-mail: gravirov@rambler.ru*

Abstract—The number of seismological studies based on artificial neural networks has been increasing. However, neural networks with one hidden layer have almost reached the limit of their capabilities. In the last few years, there has been a new boom in neuroinformatics associated with the development of third-generation networks, deep neural networks. These networks operate with data at a higher level. Unlabeled data can be used to pretrain the network, i.e., there is no need for an expert to determine in advance the phenomenon to which these data correspond. Final training requires a small amount of labeled data. Deep networks have a higher level of abstraction and produce fewer errors. The same network can be used to solve several tasks at the same time, or it is easy to retrain it from one task to another. The paper discusses the possibility of applying deep networks in seismology. We have described what deep networks are, their advantages, how they are trained, how to adapt them to the features of seismic data, and what prospects are opening up in connection with their use.

Keywords: deep neural networks, deep learning, greedy algorithm, seismic data, multitask learning

DOI: 10.3103/S0747923918010073

INTRODUCTION

Artificial neural networks (NNs) are widely used for the processing of seismic data (Böse et al., 2008; Gravirov et al., 2012; Lin et al., 2012; Kislov and Gravirov, 2017). It is possible to train the NN to solve such complex tasks as pattern recognition, signal detection, nonlinear modeling, classification, and regression using a training sample in which the correct answer is known for each example (supervised learning). However, the expansion of neural network technologies is constrained by a large number of heuristic rules for network design and training. The main thing is that, in this case, it is never known whether the architecture of the constructed network is optimal or whether it has been trained in the best way, i.e., whether the global minimum of the error function is found.

Although an NN with one hidden layer can approximate any function with any accuracy, it can be considered to be a lookup table for the training sample with the more or less correct interpolation of intermediate values and extrapolation at the edges (Cybenko, 1989). The main limitations of the applicability of the NN are also associated with the problems of overfitting, stability-plasticity tradeoff, and the curse of dimensionality (Friedman, 1994). Obviously, different methods are developed to bypass these difficulties, but they are mostly heuristic.

A trained NN works fast, but the learning process requires an indefinite time. In addition, the preparation of the training sample is usually a time-consuming process in itself (Gravirov and Kislov, 2015). Some solutions for this problem have also been found. For example, some NNs can cluster examples with an unknown answer (unsupervised learning), which reduces preparatory work (Köhler et al., 2010).

Preparation of the training sample (study, evaluation, analysis, and preprocessing) is time-consuming, but it largely determines the efficiency of the network. It should be added that this is almost an art, and the result depends heavily on the experience of the researcher.

The choice of the model and the data representation method also depends on their type. Seismic records differ significantly from other types of data. Time windows, which are usually used for data analysis, often contain a lot of noise and are of high dimensionality, while the data is redundant (impulse signals can be located anywhere in the window). The data can be represented by either one or several channels. If we compare two similar signals, it is difficult to determine how close they are in the feature space. Because of this, it is difficult to identify the number of shared areas and, consequently, to apply heuristic rules for calculating the required number of network neurons

and the required number of learning examples. In addition, seismic signal record sometimes occupy a significant time interval in the data. Thus, we should either use a very wide time window, or the model should have a memory of past input vectors. Finally, the data can reflect several seismic events simultaneously.

In order to overcome these difficulties, specific methods are used to form a feature vector, as well as different methods for reducing the dimensionality of the data and the preliminary filtering of noise (Gravirov et al., 2013; Gravirov and Kislov, 2014; Madureira and Ruano, 2009). Nevertheless, there is a danger that, as a result of this preprocessing, useful information can be lost.

Tasks that have high-dimensional time series containing noise at the input cannot be solved qualitatively by the traditional shallow methods. Such networks perform only a small number of nonlinear transformations and cannot accurately simulate complex data.

DEEP NEURAL NETWORKS

In recent years, the theory and practice of NNs has received a new impulse caused by the successful application of deep learning methods. Deep artificial neural networks (DNNs) have won numerous contests for pattern recognition, classification, and regression. Complex tasks are already being solved using DNNs.

DNN is a network with two or more hidden layers. If the layers are smaller, these are small networks and shallow learning methods. If there are more than ten layers, this is so-called very deep learning. Currently, very deep learning is rare. Additional layers make it possible to extract data features from the lower layers, i.e., to extract features from features, which creates the potential to model complex data with fewer neurons than in a shallow network (Schmidhuber, 2015).

Deep learning is a set of algorithms based on learning multiple levels of presentation of the data (abstraction levels). Training sample features are automatically and simultaneously detected on a large number of levels, which makes it possible to use these representations to create an output signal; the deeper the learning, the higher the degree of abstraction. Thus, DNNs have a more compact representation (Le Roux and Bengio, 2008). At the same time, remembering almost all of the training examples does not lead to a loss of generalizing ability, i.e., there is no overfitting (Tat'yankin and Dyubko, 2015).

At least intuitively, it was clear earlier that DNNs have some advantages. What prevented their development? Training a network with several hidden layers using the backpropagation method is associated with the need to overcome the problem of vanishing gradients. The error signal (gradient) and, consequently, the correction of the synaptic weights decreases exponentially with an increasing number of layers. This can lead to a situation where the gradient becomes so small that the training either slows down a lot or ceases to work. Conversely, the gradient in the earlier layers is sometimes too large. In this case, gradients can explode. Thus, when using the backpropagation method in the DNN training, the gradient is unstable and, in the earlier layers, tends to either explode or vanish.

This problem was solved using a greedy algorithm for the layer-by-layer pretraining of the network using unlabeled data, i.e., unsupervised learning. A restricted Boltzmann machine (RBM) is most often used as a greedy algorithm. The RBM network consists of two layers (Fig. 1), i.e., the visible **v** (input) layer and the hidden **h** layer. It is necessary to adjust the model parameters so that the generated vector is as close as possible to the input vector. The generated vector is the vector obtained by probabilistic inference from the hidden layer, the values of which are in turn obtained by the probabilistic inference from the visible layer, i.e., from the original vector.

RBM neurons are treated as stochastic. The output signal of this neuron can be +1 or –1. The probability is determined by the following sigmoid function

$$P(x) = \frac{1}{1 + \exp(-x/T)},$$

where T is the so-called neuron temperature, which determines the inclination of the sigmoid.

By analogy with the Boltzmann distribution known from statistical mechanics (also sometimes called the Gibbs distribution), the joint probability (the probability of a system state for given parameters) is defined as

$$P(\mathbf{x}, \mathbf{h}) = \frac{1}{Z}\exp\left(-E(\mathbf{x}, \mathbf{h})\right),$$

where Z is the normalizing function. The energy of the system (e.g., for a binary-binary RBM) is $E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{n} a_i v_i - \sum_{j=1}^{m} b_j h_j - \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} v_i h_j$, where **v**, **h** are the values of visible and hidden neurons; **a**, **b**, **W** are adjustable parameters, including the bias weights (offsets) of the visible and hidden neurons and the weight matrix; n is the number of visible neurons; and m is the number of hidden neurons.

In order for the model to fully reflect the data set, it is necessary to find such parameters (**a**, **b**, **W**) that the probability of the visible layer (probability of the image generated by the model)

$$P(\mathbf{v}) = \sum_{t}^{M} p(\mathbf{v}, \mathbf{h}_t) = \frac{1}{Z}\sum_{t}^{M} \exp(-E(\mathbf{v}, \mathbf{h}_t))$$

is maximal, where M is the number of all possible states of the hidden layer. The rule for updating

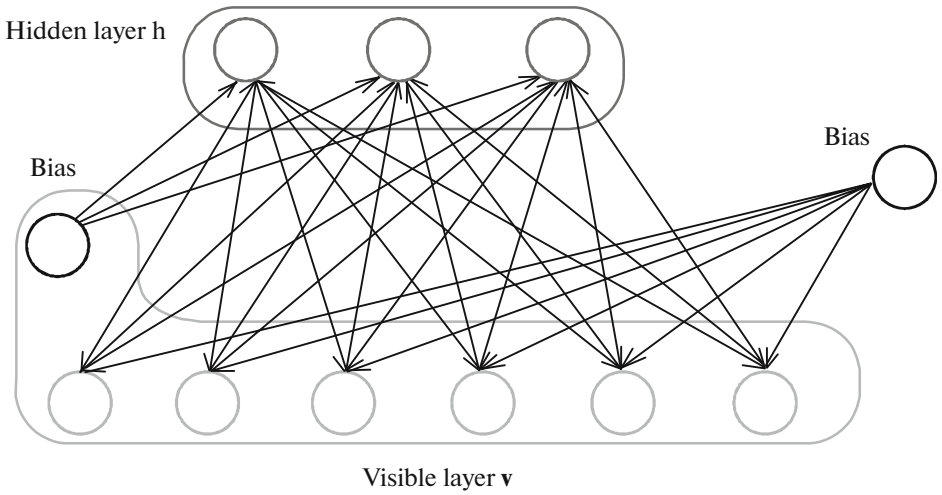

Fig. 1. Restricted Boltzmann machine (RBM).

parameters is calculated based on the Contrastive Divergence Learning Algorithm. For practical calculations, it is convenient to use the log-likelihood function $\ln P$ (Hinton et al., 2006).

In addition to the RBM, the greedy algorithm can be based on the autoencoder. The autoencoder (autocoder, autoassociator) is the NN with one hidden layer (Fig. 2), trained by the backpropagation method to reconstruct its own inputs ($\mathbf{y} = \mathbf{x}$). The input and output layers have an equal number of neurons. The input $\mathbf{x}$ is mapped to the hidden layer $\mathbf{z}$ (so-called latent variables) $\mathbf{z} = f_1(\mathbf{W}\mathbf{x} + \mathbf{a})$, where f is the element-wise activation function (e.g., the sigmoid function). Then, $\mathbf{z}$ is mapped to the output layer $\mathbf{y} = f_2(\mathbf{W}'\mathbf{z} + \mathbf{b})$. Thus, the autoencoder also learns to minimize the recovery error (e.g., the root-mean-square error) $E(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2 = \|\mathbf{x} - f_2(\mathbf{W}'(f_1(\mathbf{W}\mathbf{x} + \mathbf{a})) + \mathbf{b})\|$. The hidden layer should contain fewer neurons than the input layer (dimensionality reduction). Sparse activation is also used. In this case, the hidden layer is larger than the input one. In the case of sparse activation, the number of active neurons is significantly smaller than that of inactive neurons that produce a low signal, e.g., for the hyperbolic tangent activation function, the inactive neuron output should be close to -1. Both the dimensionality reduction and sparse activation help in the learning process to identify useful structures in the input data.

By using the autoencoder or RBM in the greedy algorithm, it is possible to model complex structures in the data. The greedy pretraining algorithm consists of the successive learning of hidden layers starting from the input, as described below.

(1) Conduct unsupervised learning of the autoencoder or RBM on a large training sample.

(2) After learning, keep the responses of the hidden network layer for the entire training sample.

(3) Train another autoencoder or RBM taking these responses (step 2) as a new training sample.

(4) Repeat steps 2, 3 the required number of times.

(5) Assemble the DNN as shown in Fig. 3. Hidden layers of the autoencoder or RBM are simply used to construct an DNN.

(6) Fine-tune the obtained deep architecture. The backpropagation method is most commonly used for a small number of labeled data (supervised learning). During the fine-tuning, only one or two of the last layers of the network are often final trained.

Several other algorithms can be used.

The fact that the greedy layer-by-layer unsupervised learning algorithm can be applied to the DNN optimization has been confirmed experimentally. The weights of the hidden layers neurons are usually in a state that corresponds to the global minimum of the error function and are fine-tuned very quickly. In this case, a large number of parameters are revealed, i.e., the range of tasks that the DNN can solve becomes wider.

The autoencoder minimizes the error according to some performance functional by considering noise to be information, while the RBM is a generative model (as opposed to discriminative), which learns to generate the representation with the greatest probability. For simple tasks, e.g., for classification by a small number of classes, a discriminative model is sufficient. This model will simulate a training sample, even if the data is noisy, while the generative model will assign a low probability to spikes. The generative model is more robust to noise and more capable of generalization.

In addition to the presented algorithm, other DNN training and coding methods are being developed in order to reduce the pretraining time (Schmidhuber, 2015).

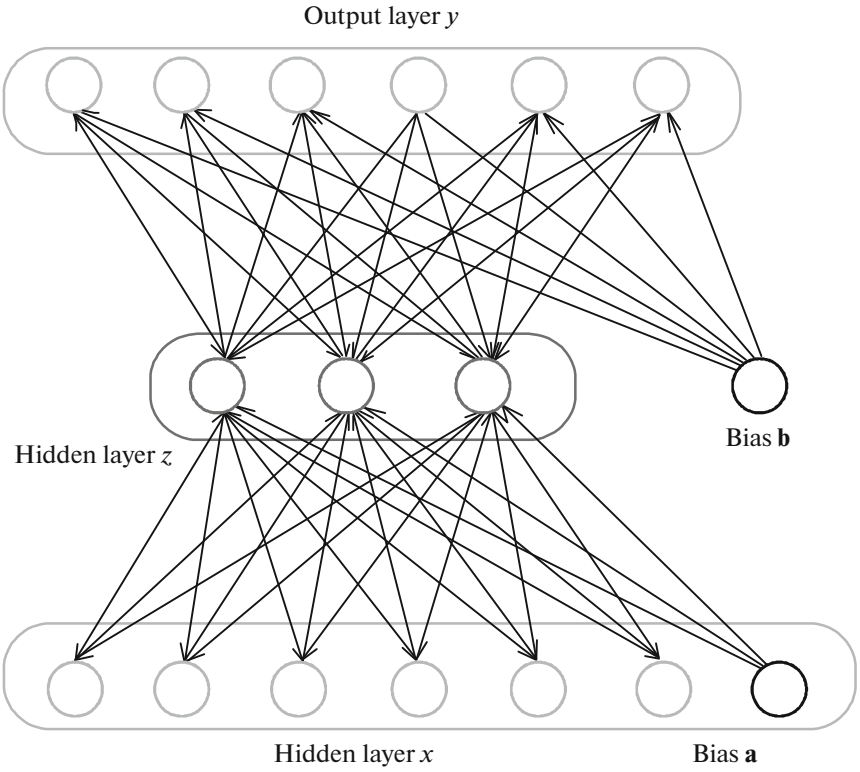

Fig. 2. Autoencoder.

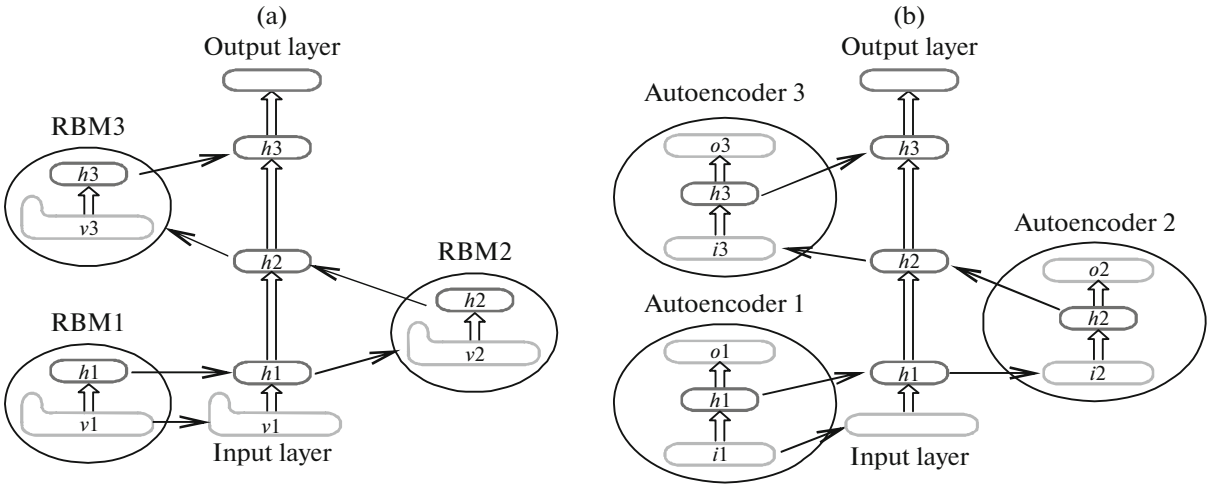

Fig. 3. Deep neural networks with weights pretrained by (a) RBM and (b) autoencoder.

Deep networks achieve a high degree of generalization, i.e., they try to guess the most probable result. When a new layer is added to the model, the lower bound of the log likelihood ($\ln P$) is lifted (Hinton, 2007). Unfortunately, it is unknown how much this improves the result. Therefore, the number of hidden layers is determined by trial and error or using some heuristic rules.

The more layers the NN has, the greater the computational costs it requires for training. Just recently, a supercomputer was required to train the NN. If the input vector size, model parameters, and the training

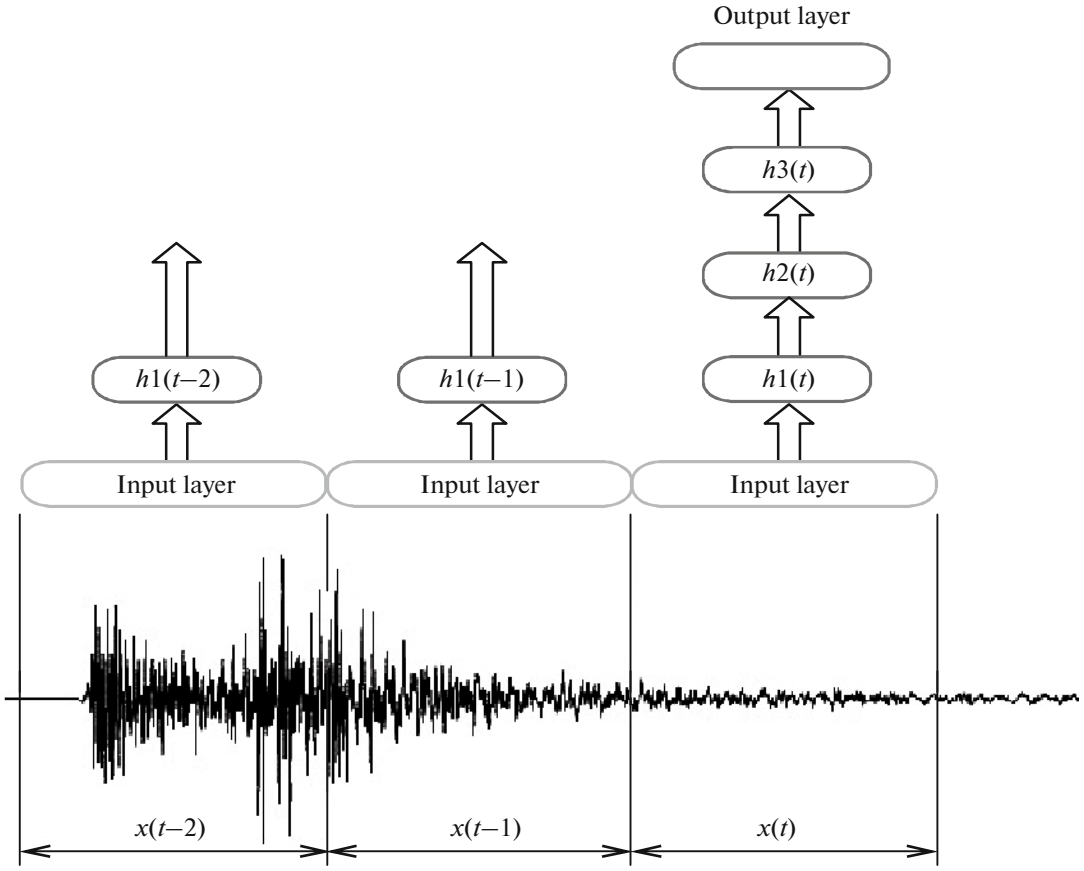

Fig. 4. Recurrent NN. Three time steps are shown. *t* is time, in lengths of the time window. Neurons of the first hidden layer are connected to the input of the current time window and the neurons of the first hidden layer from the previous time window.

sample are large, a graphics processor (GPU) can be used to reduce learning time. It can carry out matrix and vector calculations and has a high throughput, which speeds up the training significantly (by several orders of magnitude). Before choosing the speedy learning algorithm, it makes sense to consider the possibility of the training hardware acceleration.

FEATURES OF SEISMIC DATA ANALYSIS

The DNN unsupervised learning is successfully used in the study of sets of static data, e.g., images. When analyzing seismic data, their temporal nature should be taken into account. The need to study long signals with a specific sequence of features (phases and coda waves of earthquake) requires the use of a huge input vector, which can be impractical. In this case, it is necessary to use a model that remembers its previous states. In different models, memory is implemented differently. A typical example is a recurrent network.

In the recurrent NN, states of neurons in the current time window depend on the input vector and on the state of the neurons in the previous window (see, e.g., Fig. 4). This can create an effect of a wave with a potentially infinite period (Hochreiter et al., 2001; Mikolov et al., 2010). These networks are usually trained iteratively using a procedure known as the backpropagation-through-time algorithm.

Another example is a convolutional neural network (CNN). They are usually used for image processing. Nevertheless, CNNs can be used to analyze seismic data.

Usually, CNNs consist of three types of layers. In the convolutional layer, each neuron of the hidden layer is not connected to all neurons of the previous layer, but receives signals from only a few neurons located in a small neighborhood (Fig. 5). Neurons use the same sets of weights to communicate with the previous layer and are combined into feature maps. While processing time series data, the network performs convolution on overlapping windows. This makes it possible to detect signal features regardless of their position in the input vector.

Another type is pooling layers, which are also called subsampling layers. One neuron of this layer replaces several neurons of the previous layer in one

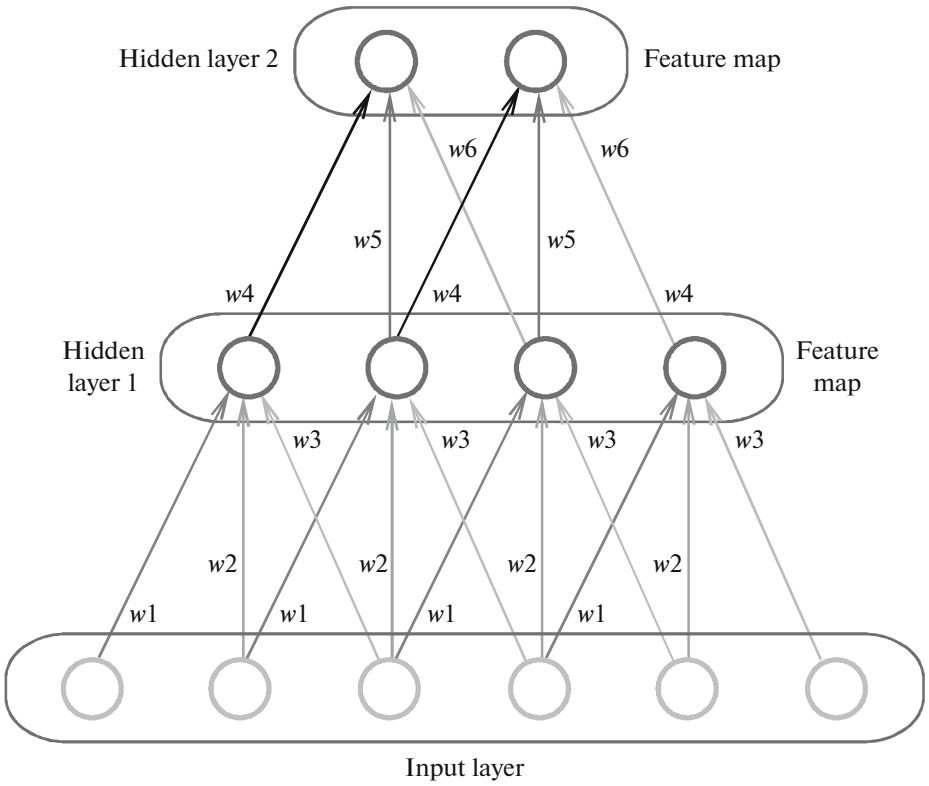

Fig. 5. Two-layer convolutional neural network.

way or another. The goal of the pooling is to achieve the invariance of small local distortions and reduce the dimensionality of the feature space. Thus, adding a new layer increases the memory length.

The third type is a fully connected layer, which, based on the features identified in the previous layers, performs proper classification. The outputs of all neurons of the previous layer are fed to the input of each of the neurons of the fully connected layer.

Layers of different types can alternate in any order to handle large input vectors in which there is a temporal relationship between the individual inputs. The output layer is typically fully connected. A number of different DNN architectures, as well as different algorithms for pretraining and coding their parameters, can be used to process seismic data. Obviously, there is a need for the further development of deep learning algorithms, which can take into account the specifics of temporal relationships in the data (Wiskott and Sejnowski, 2002).

MULTITASK LEARNING

Seismic record analysis can be carried out for different purposes and tasks. For example, at the initial stage, any seismologist learns to work with seismic data and read seismograms. Something similar happens in the case of unsupervised DNN learning. The ability to adapt the model to several tasks (multitask learning) is very important. Deep architecture reveals intermediate representations that can be used to solve different tasks. Good representations can be applied to a variety of tasks, since each requires a subset of data features. The trunk of the network tree (Fig. 6) learns to work with a certain type of data, while the branches are designed to perform certain tasks. DNN transfer learning can be carried out easily and quickly while maintaining the overall configuration of the trunk.

If an object, process, or phenomenon that interests us can be investigated by different methods involving different types of data, it would be highly desirable to combine these data in a single analysis system. For example, seismic, gravimetric, electrometric, magnetometric, radiometric, and other data are used in geophysical methods for the exploration of hydrocarbon deposits. Multitask learning with different input parameters creates a model that is fully capable of investigating a process, object, or phenomenon and can solve different problems that use all of the available information (Fig. 7). The deep architecture can be used both as a discriminative model for analyzing the input data (using a bottom-up processing) and as a generative model for producing new data samples (using a top-down processing).

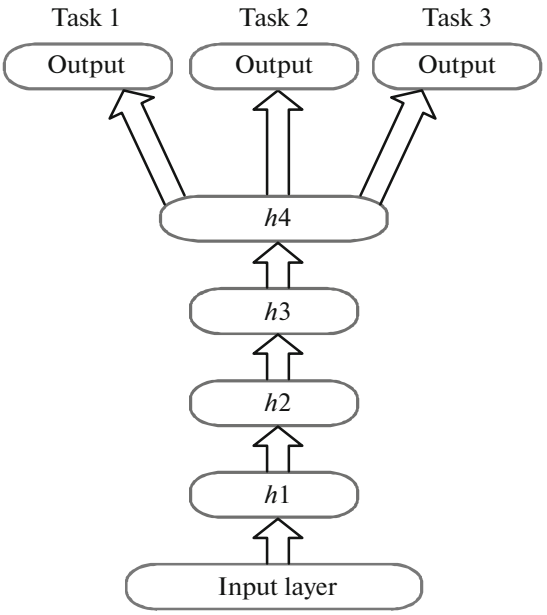

Fig. 6. Multitask learning.

RESULTS AND DISCUSSION

Deep learning makes it possible to identify higher levels of abstraction, which improves the network's generalization ability. The following advantages of DNNs can be distinguished:

unlabeled data can be used to pretrain the DNN (unsupervised learning);

learning how to solve a particular problem requires a small amount of labeled data (supervised learning);

higher learning speed and fewer errors occur during operation;

they are simple to retrain to solve other Tasks.

Nevertheless, deep learning methods often look like a black box, and most of the rules for building an DNN are empirical rather than theoretical. Two main questions related to the development and training of DNN are (1) how many neurons are needed on each hidden layer and (2) how many hidden layers are necessary and sufficient for the successful operation. Until now, these problems have been solved by trial and error. On the other hand, there are already some algorithms and methods that can be used. On the Internet, there is software (including free software) that can be used to build and train DNNs (Perervenko, 2014).

The process of extracting features of the source data has already been well demonstrated for static problems. The use of the DNN for tasks focused on the time series analysis requires some caution. Tested models (see above) are well suited for the processing of raw (unprocessed) seismic data (or data that has been slightly preprocessed). In general, in the case of the DNN, less attention should be paid to the problem of preliminary data processing and it is necessary to concentrate more on identifying their features (Längkvist et al., 2014).

As soon as the size of the training sample exceeds a certain limit, the DNN qualitatively surpasses other neural network architectures. In particular, a lot of data for DNN pretraining is required if it is supposed to be retrained from one task to another. For unsupervised learning, it will be correct to use not thousands, but millions or even billions of examples. Moreover,

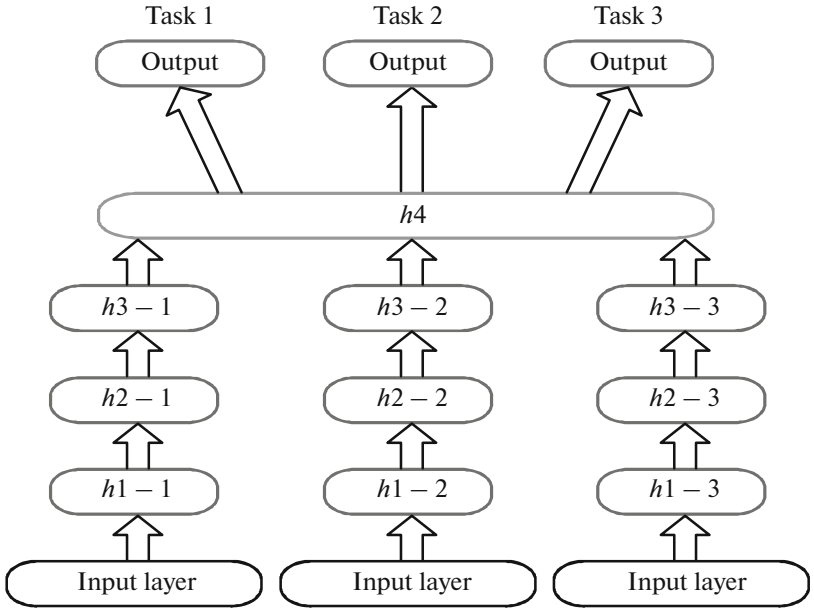

Fig. 7. Multitask learning with different input parameters.

the size of the training sample is often more important than the network architecture. In this case, it is impossible to advance from the simple to the complex. For example, pretraining on 5000 versus 50000 examples can be equally inefficient. The point at which a qualitative jump will occur is unknown in advance.

This imposes some limitations on the range of tasks that can be solved using the DNN. For example, studies of large earthquakes should be conducted with caution, since there are relatively few records of these events. On the other hand, deep learning becomes one of the most powerful methods, and DNNs can become a very convenient tool for investigating low-magnitude events and noise, as well as for mineral exploration.

This interesting apparatus for data analysis has not yet received significant attention from geophysicists. In (Holdaway, 2015), it is proposed to use an DNN with five hidden layers to remove noise from seismic data. We have not seen any other works yet.

CONCLUSIONS

Deep artifical neural networks offer a better selection of features in the time series data compared to shallow NNs and take into account more features. Obviously, not all existing DNN architectures and their training options are presented in the paper. For example, a promising direction is the development of models that change their architecture in the learning process.

Practical research work is now far ahead of mathematicians' ability to prove anything about the DNN. There are many versions of deep architecture and, in most cases, there is no mathematical proof why they are good or better than others. Deep training is a rapidly developing field, and new architectures, versions, and algorithms appear almost daily.

The DNN formula for success is as follows:

(1) Pretraining algorithms make it possible to find a good starting point for fine-tuning conducted by the backpropagation method from which the gradient descent method makes it possible to achieve a good local minimum (and most often a global minimum).

(2) Processing huge data sets makes the training procedure reliable, but the DNN becomes less subject to overfitting. .

(3) GPU makes it possible to build very deep networks. When a powerful GPU is used, the DNN can contain tens of millions of neurons.

The overall result is that the DNN is a promising tool for analyzing seismic data.

REFERENCES

Böse, M., Wenzel, F., and Erdik, M., PreSEIS: A Neural Network-Based Approach to Earthquake Early Warning for Finite Faults, *Bull. Seismol. Soc. Am.*, 2008, vol. 98, no. 1, pp. 366–382.

Cybenko, G., Approximation by superpositions of a sigmoidal function, *Math. Control Signals Syst.*, 1989, no. 2, pp. 303–314.

Friedman, J.H., An overview of predictive learning and function approximation, in *From Statistics to Neural Networks*, vol. 136 of *NATO ASI Series*, Cherkassky, V., Friedman, J.H., and Wechsler, H., Eds., Berlin: Springer, 1994, pp. 1–61.

Gravirov, V.V. and Kislov, K.V., The seismic data preparation program for training of an artificial neural network for ultrashort earthquake warning system, in *Book of Abstracts, 10th International Conference "Problems of Geocosmos"*, St. Petersburg, 2014, pp. 86–87. http://geo.phys.spbu.ru/geocosmos/book_of_abstracts.pdf. Accessed December 31, 2014.

Gravirov, V.V. and Kislov, K.V., *DataCollector: Software for Compiling the Output Data Volumes to Learn Neural Networks in the Matlab System*, State Software Registration Certificate no. 2015611797, 2015.

Gravirov, V.V., Kislov, K.V., and Vinberg, F.E., The informative signal separation from the high noise level non-stationary seismic data with the use of the neural network classifiers, *Ind. Autom. Control Syst. Controllers*, 2012, no. 12, pp. 55–59.

Gravirov, V.V., Kislov, K.V., Gravirova, L., and Vinberg, F., The use of wavelet transformation techniques in structure of an artificial neural network for recognition of early arrival of earthquakes on strongly noisy seismic records, in *Book of Abstracts CTBT: Science and Technology 2013*, Vienna, Austria, 2013, p. 139. http://www.ctbto.org/fileadmin/user_upload/ SnT2013/bookofabstracts.pdf. Accessed December 31, 2013.

Hinton, G.E., Learning multiple layers of representation, *Trends Cognit. Sci.*, 2007, no. 10, pp. 428–434.

Hinton, G.E., Osindero, S., and The, Y.-W., A fast learning algorithm for deep belief nets, *Neural Comput.*, 2006, vol. 18, pp. 1527–1554.

Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J., Gradient flow in recurrent nets: The difficulty of learning long-term dependencies, in *A Field Guide to Dynamical Recurrent Networks*, Kolen, J.F. and Kremer, S., Eds., Los Alamitos, Calif.: IEEE Press, 2001.

Holdaway, K.R., Advanced seismic attribute analysis: Deep learning, *Finding Petroleum forum "Transforming Subsurface Interpretation"*, London, 2015. http://80f54e3872b7907eeda8-4de6a2a3e09efd2caa4972e2c52e8eac.r36.cf1.rackcdn.com/keith%20holdaway.pdf. Accessed December 31, 2015.

Kislov, K.V. and Gravirov, V.V., Use of artificial neural networks for classification of noisy seismic signals, *Seism. Instrum.*, 2017, vol. 53, no. 1, pp. 87–101.

Köhler, A., Ohrnberger, M., and Scherbaum, F., Unsupervised pattern recognition in continuous seismic wavefield records using Self-Organizing Maps, *Geophys. J. Int.*, 2010, vol. 182, no. 3, pp. 1619–1630.

Längkvist, M., Karlsson, L., and Loutfi, A., A review of unsupervised feature learning and deep learning for time-series modeling, *Pattern Recognit. Lett.*, 2014, vol. 42, no. 1, pp. 11–24.

Le Roux, N. and Bengio, Y., Representational power of restricted Boltzmann machines and deep belief networks, *Neural Comput.*, 2008, vol. 20, pp. 1631–1649.

Lin, Chu-Chieh J., Lin, Pei-Yang, Chang, Tao-Ming, Lin, Tzu-Kun, Weng, Yuan-Tao, Chang, Kuo-Chen, and Tsai, Keh-Chyuan, Development of on-site earthquake early warning system for Taiwan, in *Earthquake Research and Analysis – New Frontiers in Seismology*, D'Amico, S., Ed., InTech, 2012, Ch. 13, pp. 329–358.

Madureira, G. and Ruano, A.E., A neural network seismic detector, *Acta Tech. Jaurinensis*, 2009, vol. 2, no. 2, pp. 159–170.

Mikolov, T., Karafiat, M., Burget, L., Cernocky, J., and Khudanpur, S., Recurrent neural network based language model, *11th Annual Conference of the International Speech Communication Association*, Makuhari, Chiba, Japan, 2010, pp. 1045–1048.

Perervenko, V., The third generation of neural networks: Deep networks, 2014. https://www.mql5.com/ru/articles/1103#1_4. Accessed December 31, 2014.

Schmidhuber, J., Deep learning in neural networks: An overview, *Neural Networks*, 2015, vol. 61, pp. 85–117.

Tat'yankin, V.M. and Dyubko, I.S., Deep confidence neural networks compared to multilayer perceptron, *Vestn. Yugorsk. Gos. Univ.*, 2015, no. 2, pp. 87–89.

Wiskott, L. and Sejnowski, T.J., Slow feature analysis: Unsupervised learning of invariances, *Neural Comput.*, 2002, vol. 14, pp. 715–770.

Translated by O. Pismenov