# Acoustic Modeling with Deep Belief Networks for Russian Speech Recognition

Mikhail Zulkarneev, Ruben Grigoryan, and Nikolay Shamraev

FSSI Research Institute "Spezvuzautomatika", Rostov-on-Don, Russia
zulkarneev@mail.ru,
r.grigoryan@niisva.org,
ncam1977@yahoo.com

**Abstract.** This paper presents continuous Russian speech recognition with deep belief networks in conjunction with HMM. Recognition is performed in two stages. In the first phase deep belief networks are used to calculate the phoneme state probability for feature vectors describing speech. In the second stage, these probabilities are used by Viterbi decoder for generating resulting sequence of words. Two-stage training procedure of deep belief networks is used based on restricted Boltzmann machines. In the first stage neural network is represented as a stack of restricted Boltzmann machines and sequential training is performed, when the previous machine output is the input to the next. After a rough adjustment of the weights second stage is performed using a back-propagation training procedure. The advantage of this method is that it allows usage of unlabeled data for training. It makes the training more robust and effective.

**Keywords:** Speech Recognition, Hidden Markov Models, Deep Neural Networks, Restricted Boltzmann machines.

## 1 Introduction

Nowadays speech recognition systems widely employ Hidden Markov models (HMM) for acoustic modeling and Gaussian mixtures (GM) for state modeling [1]. Popularity of the models results from their simplicity as well as from effective theoretically-based training and decoding algorithms. Along with the simplicity of the models they show high recognition accuracy when they are tested on different speech recognition tasks[1].

Nevertheless these models have some shortcomings. First, they quite roughly model context relation - the connection between states is determined by the a single parameter - state transition probability. Second, GMs are ineffective for modeling data located in non-linear set or near it. For example modeling points located close to the sphere surface requires just a few parameters, if you use the appropriate class of models, however, it requires a great number of parameters, if you use Gaussian mixture with diagonal covariance matrix (this model is often used in speech recognition systems).

Paper [2] suggests using recurrent neural networks for context modeling. The hidden layer of this networks has recurrent transitions, and it enables keeping information on

[1] The work was financially supported by the state contract GK 07.524.11.4023 (Minobrnauka RF).

M. Železný et al. (Eds.): SPECOM 2013, LNAI 8113, pp. 17–24, 2013.

the past. But information on the future is also essential in speech recognition, therefore paper [2] offers a bidirectional recurrent neural networks model. Per se these are two recurrent neural networks - in forward and back directions. The networks have the same output layer receiving output signal from both networks. In addition the authors suggest applying a technology named long short-term memory for enlarging the length of the context being used [3].

Involvement of neural networks allows overcoming the second disadvantage of HMM+GM, as they proved to be universal approximates for non-linear functions. New efficient training algorithms and computing machinery came up and they allow training of neural networks with more hidden layers and larger output layer. As a result many research groups obtained positive results using neural networks for speech recognition [4].

The present paper suggests using DBN combined with HMM for continuous Russian speech recognition. Recognition is performed in two stages. At the first stage we use DBN to calculate phoneme states for feature vectors from sequence $O = o_1, ..., o_T$, representing speech message (melcepstral coefficients or their cosine transformation are used as features). The second stage involves Viterbi decoder for obtaining resulting word sequence.

## 2 Method Description

The paper investigates a possibility to implement hybrid system DBN+HMM for continuous Russian speech recognition. DBN are used for calculation of probabilities of phoneme states $p(s_i|o_t)$ for feature vectors for sequence $O = o_1, ..., o_T$, representing speech message. These probabilities are further applied for speech recognition by token passing algorithm [6] - the regular procedure for HMM and Gaussian mixture based speech recognition systems. The token passing algorithm uses likelihood $p(o_t|s_i)$, and DBN produces a posteriori probabilities of states $p(s_i|o_t)$, therefore Bayes formula is used to recalculate these values:

$$p(o_t|s_i) = \frac{p(s_i|o_t)p(o_t)}{p(s_i)} \tag{1}$$

where $p(s_i)$ - prior probability of state $s_i$. $p(o_t)$ doesn't change for fixed $o_t$ when calculating $p(o_t|s_i)$, therefore it can be omitted and we can use $p(s_i|o_t)/p(s_i)$ as likelihood.

For speech recognition decoding we used HMM and HDecode from HTK tool set [7], and thus we face a problem of transmitting the obtained probabilities into the utility. To solve this problem $p_i^{(t)} = p(s_i|o_t)/p(s_i)$ is transformed by $x_i^{(t)} = f(p_i^{(t)})$ and presented as a file of parameters compatible with HDecode. Furthermore HMM phonemes are prepared for parameter files processing: a tree-states HMM is generated for each phoneme, each state being described by normal distribution with zero means vector and dispersion vector whose elements have huge absolute value ($\approx 10^{30}$), apart from element $i$, where $i$ - number of the given state with dispersion equal 1. When we choose these normal distribution parameters and transformation $f(p_i^{(t)})$ in the form $x_i^{(t)} = \sqrt{log(p_{i\max}^{(t)}/p_i^{(t)})}$

normal distribution value for vector $x^{(t)} = \{x_i\}$ will be proportional to value $p_i^{(t)}$, but at the same time it will not go beyond 1 ($p_{i\max}^{(t)}$ - maximum value of $p_i^{(t)} = p(s_i|o_t)/p(s_i)$, considering that $p_{\max}(s_i|o_t) = 1$, this value equals to $p_{i\max}^{(t)} = 1/p_{\min}(s_i)$).

### 2.1 Neural Networks Used for Acoustic Modeling

The DBN with the layer number not greater than 7 are investigated in the paper. However this limit isn't fundamental and is due to the limit of time required for training. The output for an element is value $y_j = \sigma(x_j)$, where $\sigma(x) = 1/(1 + e^{-x})$ is a sigmoid function, $x_j = b_j + \sum_i y_j w_{ij}$ is activation of neuron, $b_j$ - bias of element $j$, $i$ - index of element of lower layer, $w_{ij}$ - weight of connection between element $j$ and element $i$. Apart from the elements with a logistic function there is an output layer softmax which serves as a transfer function, and produces probabilities each state in each HMM:

$$p_j = \frac{e^{x_j}}{\sum_k e^{x_k}} \tag{2}$$

where $k$ = index of output layer elements. Activation is counted by this formula:

$$x_k = b_k + \sum_i y_i w_{ik} \tag{3}$$

### 2.2 DBN Training

#### Back-propagation Procedure

We used procedure of object function derivative back-propagation. It is assumed that we set a training sample $\{x_i, t_i\}$. Object function represents disparity between the network output value and the object value (correct neuron output value). In case we use function softmax as an output function, it's naturally to choose cross entropy between target probabilities $d$ and output probabilities $p$ as object function:

$$C = -\sum_j d_j \log p_j \tag{4}$$

We used negative object function gradient in parameter space as increment of DBN parameters because we are interested in minimum value of object function:

$$\Delta w_{ij} = -\epsilon \frac{\partial C}{\partial w_{ij}} \tag{5}$$

where $\epsilon$ is learning rate.

In the training set $D$ we exchanged object function $C$ for as mean value by the whole sample

$$\langle C \rangle = \frac{1}{|D|} \sum_{e \in D} \sum_j d_j^{(e)} \log p_j^{(e)} \tag{6}$$

However in case of large training samples such as speech training databases it is more efficient to calculate gradient not by the whole training sample but by it's small random batch. For gradient smoothing we introduced "moment" $0 < \alpha < 1$ which suppresses fluctuation of gradient in minimum object function area and thus encourages convergence:

$$\Delta w_{ij}(t) = \alpha \Delta w_{ij}(t-1) - \epsilon \frac{\partial \langle C \rangle}{\partial w_{ij}(t)} \tag{7}$$

For the purpose of training of displacement $b_j$ they are interpreted as weights of pseudo elements with outputs equal to one.

**Two-Staged DBN Training**

Training of DBN with a large number of layers using back-propagation procedure is quite embarrassing due to vanishing/exploding effect resulting from linearity of expressions for loss function gradient recalculation. This fact hindered training of DBN with a large number of layers. However the situation changed after work [5] had been published. The paper suggested two-staged training for DBN using restricted Boltzmann machines (RBM). The backbone idea is that we should perform rough front end adjusting of models weights by back-propagation procedure. It enables more efficient training as it removes vanishing/exploding effects and decreases impact of the retraining.

On the first stage Deep Belief network (DBN) is created as an RBM stack. To perform it we train RBM whose output is used for next RBM training and we keep on this procedure until we gain the required number of DBN layers. (Fig. 1 shows an example of three-layered DBN training). As a result we design a multi-layered generative model (Fig. 1c).

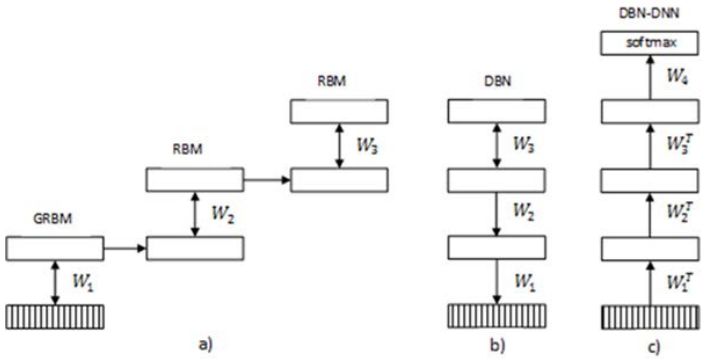


**Fig. 1.** DBN-DNN training

At the second stage we added softmax layer to the obtained model and built the so-called DBN-DNN model, trained by the error back-propagation method. During the training process model parameters undergo only a slight change (with the exception of $W_4$) and model parameters are adjusted.

**RBM Training**

RBM is a bilateral graph (see Fig. 2) where visible elements represent input vector $v$ while $h$ represent feature vector generated on the basis of input vector and weigh matrix $W$. RBM elements are binary and stochastic as their states are determined by the joint probability $p(v, h|\theta)$.

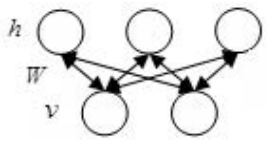


**Fig. 2.** Restricted Boltzmann machine

In RBM joint probability of vectors $v$ and $h$ is determined by energy:

$$p(v, h|\theta) = \frac{e^{E(v,h|\theta)}}{Z} \tag{8}$$

$$E(v, h|\theta) = -\sum_{i=1}^{V}\sum_{j=1}^{H} w_{ij} v_i h_j - \sum_{i=1}^{V} b_i v_i - \sum_{j=1}^{H} a_j h_j \tag{9}$$

where $\theta = (W, b, a)$, $b$ and $a$ - bias vectors for visible and hidden layers, $V$ and $H$ - number of visible and hidden elements, $Z$ - normalization factor which is called static sum:

$$Z = \sum_{v}\sum_{h} e^{-E(v,h|\theta)} \tag{10}$$

As there are no transitions between hidden states, probability $p(h|v, \theta)$ can be represented as product of factors, each factor being dependent on one element of vector $h$. Due to this fact probability $p(h_j = 1|v, \theta)$ can be represented as:

$$p(h_j = 1|v, \theta) = \sigma(a_j + \sum_{i=1}^{V} w_{ij} v_i) \tag{11}$$

The same is true for probability $p(v_i = 1|h, \theta)$:

$$p(v_i = 1|h, \theta) = \sigma(b_i + \sum_{j=1}^{H} w_{ij} h_i) \tag{12}$$

Maximum likelihood training can not be implemented for large RBM because the problem of calculation of log probability derivative for training data $D$ is exponential. However paper [8] offers an efficient training procedure implying increment of parameters as the following:

$$\Delta w_{ij} \propto \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{reconstruction} \tag{13}$$

The first right summand is occurrence frequency, when $v_i = 1$ and $h_j = 1$ simultaneously, when input receives vector $v$ from the training sample and $h$ is calculated by means of random-number generator in correspondence with probability distribution (11). The second summand is frequency of the same occurrence when vector $v$ is reconstruction of input data corresponding to probability (12). It's essential that reconstruction should not depend on input vector $v$, hence procedure of generation $h$ and further reconstruction of $v$ is repeated several times.

When input vector elements are real numbers we eliminate an unlimited growth of elements for vector $v$ through transforming energy equation as the following:

$$E(v, h|\theta) = \sum_{i=1}^{V} \frac{(v_i - b_i)^2}{2} - \sum_{i=1}^{V} \sum_{j=1}^{H} w_{ij} v_i h_j - \sum_{j=1}^{H} a_j h_j \tag{14}$$

And equation $p(v_i|h, \theta)$ like this:

$$p(v_i|h, \theta) = N(b_i + \sum_{j=1}^{H} w_{ij} h_j, 1) \tag{15}$$

where $N(\mu, \Sigma)$ - normal distribution with mean vector $\mu$ and covariance matrix $\Sigma$. The present paper uses normalized input data therefore covariance matrix is identity matrix. RBM with real input vectors is called Gaussian-Bernoulli RBM (GRMB).

### 2.3 Experiment Description and Results

The paper takes up an experimental research into hybrid DBN-HMM systems for continuous Russian speech recognition. Experiments were carried out on a telephone database collected by speech processing lab of FSSI "Research Institute "Spezvuzautomatika", Transcribed part of the database was divided into three parts: 25 hours were used for training, 1 hour was used for validation and 1 hour for test. In addition 17 hours of unlabeled speech were used for pretraining of DBN.

We used Theano [9] library for neural networks, this library is implemented in Python with C extensions. We used HTK library for speech recognition and evaluation of testing results.

26 melcepstral coefficients normalized by global dispersion were used as feature vectors. 15 feature vectors widow was used to take the account of context. Therefore the resulting feature vectors came to 390.

Our research investigated DBN with 2, 3 and 5 layers with 1000 elements for one layer. We used the above-mentioned procedure for DBN training.

Continuous speech recognition involved 650 000 words vocabulary. 2-gram and 3-gram language models trained on the database collected by speech processing lab of FSSI "Research Institute "Spezvuzautomatika" (200 million words), were applied for recognition. Recognition was performed on the basis of probabilities, obtained by DBN.

The process involved 2 stages:

- recognition with 2-gram language model and lattice building
- lattice rescoring with 3-gram language model

**Table 1.** Results of test

| Model | Accuracy |
|---|---|
| triphones | 41% |
| DBN: 2 layers x 1000 el. | 28% |
| DBN: 3 layers x 1000 el. | 33% |
| DBN: 5 layers x 1000 el. | 45% |

As base system we used a context dependent HMM (phones) based system. The results are presented in table 1.

### 2.4 Conclusion

HMMs possess a number of drawbacks which impose a limitation on the further increase of accuracy of speech recognition. Multilayered neural networks are promising substitution for HMMs. The progress of computer technology and appearance of new algorithms of training of multilayered neural networks made it more actual. The research efforts of multilayered neural networks in the field of speech recognition have intensified recently and significant progress is already achieved in this area. In this article deep neural networks were applied for modelling phonemes of Russian with the subsequent recognition of speech. The window of the big size (15 windows in the size 0.01 sec.) was used to include the context, which captured a feature vectors, belonging to adjacent phonemes. Deep neural networks calculated a posteriori probabilities of phonemes states which then were used for recognition of speech. The experiments on recognition of speech from the telephone channel have shown, that deep neural networks can demonstrate better quality of acoustic modelling in comparison with HMMs. The further progress of the described approach is connected with an increase of an amount of data for the pretraining stage of deep neural networks training and investigation of possible applications of deep neural networks with other configurations.

## References

1. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–285 (1989)
2. Graves, A., Fernández, S., Schmidhuber, J.: Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition. In: Duch, W., Kacprzyk, J., Oja, E., Zadrożny, S. (eds.) ICANN 2005. LNCS, vol. 3697, pp. 799–804. Springer, Heidelberg (2005)
3. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation 9(8), 1735–1780 (1997)
4. Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-R., Jaitly, N., Senior, A.W., Vanhoucke, V., Nguyen, P., Sainath, T., Kingsbury, B.: Deep Neural Networks for Acoustic Modeling in Speech Recognition. Signal Processing Magazine (2012)
5. Hinton, G.E., Osindero, S., Teh, Y.: A fast learning algorithm for deep belief nets. Neural Computation 18, 1527–1554 (2006)

6. Young, S.J., Russel, N.H., Thornton, J.H.S.: Token Passing: A Simple Conceptual Model for Connected Speech Recognition Systems, Cambridge University, technical report (1989)
7. Young, S.J.: The HTK Book. Version 3.4 (2006)
8. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Neural Computation 14(8), 1711–1800 (2002)
9. Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., Bengio, Y.: Theano: A CPU and GPU Math Expression Compiler. In: Proceedings of the Python for Scientific Computing Conference (SciPy), Austin, June 30-July 3 (2010)