JAM CM Journal of Applied Mathematics and Computational Mechanics 2015, 14(3), 49-58
www.amcm.pcz.pl p-ISSN 2299-9965
DOI: 10.17512/jamcm.2015.3.05 e-ISSN 2353-0588

VIDEO KEY FRAME DETECTION BASED ON THE RESTRICTED BOLTZMANN MACHINE

Michał Knop, Tomasz Kapuściński, Wojciech K. Mleczko

Institute of Computational Intelligence
Czestochowa University of Technology
Częstochowa, Poland
michal.knop@iisi.pcz.pl, tomasz.kapuscinski@iisi.pcz.pl, wojciech.mleczko@iisi.pcz.pl

Abstract. In this paper we present a new method for key frame detection. Our approach is based on a well-known algorithm of the Restricted Boltzmann Machine (RBM), which is a pivotal step in our method. The frames are compared to the RBM matcher, which allows one to search for key frame in the video sequence. The Restricted Boltzmann Machine is one of sophisticated types of neural networks, which can process the probability distribution, and is applied to filtering image recognition and modelling. The learning procedure is based on the matrix description of RBM, where the learning samples are grouped into packages, and represented as matrices. Our research confirms a potential usefulness for video key frame detection. The proposed method provides better results for professional and high-resolution videos. The simulations we conducted proved the effectiveness of our approach. The algorithm requires only one input parameter.

Keywords: *Restricted Boltzmann Machine, key frame detection, video compression*

1. Introduction

The popularity of high-definition digital movies has increased in recent years. The time needed to process individual frames is increased significantly when high resolution frames are analyzed. For this reason, the need arises to find more efficient methods of content analysis. A widely investigated scheme is to treat a video as a stream of individual images with the same dimensions [1, 2]. However, such methods don't take into account similarities of adjacent frames. More efficient methods divide video into scenes with similar frames and are commonly based on key frame detection algorithms [3-5]. Key frame detection is a challenging issue that is solved in various ways, such as histogram-based methods, entropy analysis, and correlation of images [6-8]. Unfortunately, most methods are sensitive to movement of objects in the scene and noise. Some researchers attempt to overcome these problems with SURF and SIFT algorithms to find and compare keypoints of frames. In our work, we implemented a method that uses an artificial neural network called Restricted Boltzmann Machine (RBM). In the proposed approach, we used a mapping error to compare frames and retrieve key frames.

2. Related works

2.1. Entropy method

The method proposed by Knop an Dobosz in [9] is based on entropy from theory of information which is used to determine the complexity of information in each frame. For grayscale video material, entropy can be calculated using the formula below:

$$E_j = -\sum_{i=0}^{N} p_i log(p_i), \tag{1}$$

where E_j is a characteristic value calculated for current frame, p_i is the next pixel from this frame, and N is a number of all pixels in the image. For an *HSV* color model we need to calculate entropy value for each coefficient in the palette. Components of the *HSV* model are not equally significant. The hue *H* is the most important one because even the smallest changes of this value should lead to scene changes. The saturation of color *S* has relatively little importance. Small to medium changes in saturation are fully acceptable. Color value *V* has the lowest importance. The final value of entropy for given frame in an *HSV* color space can be calculated using the following formula:

$$F_n(E) = a \cdot E_j(H) + b \cdot E_j(S) + c \cdot E_j(V). \tag{2}$$

Parameters a, b and c correspond to weighting factors when calculating final entropy and their values are set to 0.9, 0.3 and 0.1 respectively [10]. To detect a new key frame we compare values of entropy between current key frame and subsequent frames. For comparison, the following formula was used:

$$E_{j,k} = \frac{| E_j - E_k |}{E_k}, \tag{3}$$

where j is the number of the current frame and *k* is the number of the current key frame. The first key frame is the first image taken from the video. The next key frames are determined based on result of comparison in the formula (see Eq. (3)). If the calculated value of $E_{j,k}$ is lower than the assumed threshold, the current frame is set as the new key frame.

2.2. SURF key frame detection method

SKFD is a method of detecting key frames using SURF. SURF is an algorithm used to determine keypoints. We can use keypoints from two images and attempt to match them to determine similarities. The SURF algorithm is based on SIFT and works much faster because it uses Integral Images rather than Difference of Gaussians [11-14].

SURF has four major steps:

1. Computing Integral Images,
2. Fast-Hessian Detector,
3. Interest Point Descriptor,
4. Generating of keypoints.

A single key point is represented by two vectors. The first one contains point position, scale, response/strength of detected feature, orientation, and laplacian value. The second vector contains intensity distribution of the pixels around point of interest and is made of 64 or 128 values.

The output of this step is *factor*, percentage relation of the matched keypoints to all keypoints. *factor* is in range $0 \leq factor \leq 1$, where 0 means no similarity and 1 means complete similarity. Next, we compare *factor* with input parameter t. If *factor* is lower than t, then frame *next* is tagged as a new key frame. *next* is then set to another video frame. This stage is repeated until we process all frames. Correctness of SURF algorithm is vital because it determines *factor* values. If the values are incorrect, the method might miss or find too many key frames [8, 15, 16]. The method can be represented as the following block diagram (see Fig. 1)

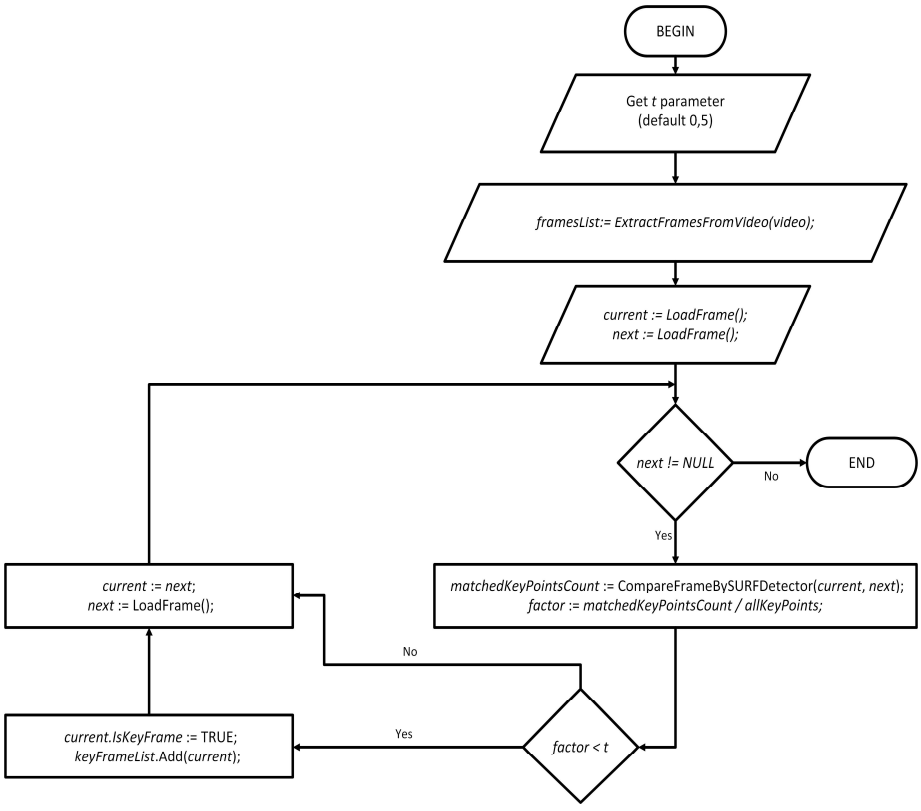

Fig.1. SURF key frames detection

3. Restricted Boltzmann Machine learning

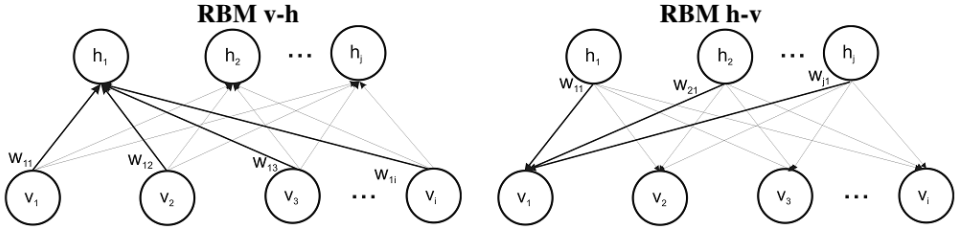

Fig. 2. Schematic representation of a Restricted Boltzmann Machine (RBM) - bidirectional

The Restricted Boltzmann Machine is a two-layer recurrent neural network. The input values are presented on the layer called "visible" as vector $\mathbf{v}_0(t) = [v_{10}(t),\ldots, v_{i0}(t),\ldots,v_{M0}(t)]$. M is the number of inputs, t indicates the specific sample. The data are transmitted to the layer called "hidden", as vector $\mathbf{h}_0(t) = [h_{10}(t),\ldots,h_{j0}(t),\ldots,h_{N0}(t)]$, where N is the number of outputs, as depicted in Figure 2. Given an observed state, the energy of the joint configuration of the visible and hidden units (v, h) is given by:

$$E(\mathbf{v},\mathbf{h}) = \sum_{i\in visible} b_{vi}v_i - \sum_{j\in hidden} b_{hj}h_j - \sum_{i,j} v_i h_j w_{ij}, \tag{4}$$

where v_i, h_j are the binary states of visible unit i and hidden unit j, b_{vi}, b_{hj} are their biases and w_{ij} is the weight between them. The network assigns a probability to every possible pair of a visible and a hidden vector via this energy function: where v_i, h_j are the binary states of visible unit i and hidden unit j, b_{vi}, b_{hj} are their biases and w_{ij} is the weight between them. The network assigns a probability to every possible pair of a visible and a hidden vector via this energy function:

$$p(\mathbf{v},\mathbf{h}) = \frac{1}{Z}e^{-E(\mathbf{v},\mathbf{h})}, \tag{5}$$

where the partition function Z, is given by summing over all possible pairs of visible and hidden vectors:

$$Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}. \tag{6}$$

The probability that the network assigns to a visible vector, $\mathbf{v}$, is given by summing over all possible hidden vectors:

$$p(\mathbf{v}) = \frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}. \tag{7}$$

Given a random input configuration $\mathbf{v}$, the state of the hidden unit j is set to 1 with probability:

$$P\left(h_j = 1 \mid \mathbf{v}\right) = \sigma\left(b_{hj} + \sum_i v_i w_{ij}\right), \tag{8}$$

where $\sigma(x)$ is the logistic sigmoid function $\sigma(x) = \frac{1}{1 + exp(-x)}$. Similarly, given a random hidden vector, the state of the visible unit i can be set to 1 with probability:

$$P\left(v_i = 1 \mid \mathbf{h}\right) = \sigma\left(b_{vi} + \sum_i h_j w_{ij}\right). \tag{9}$$

The probability that the network assigns to a training image can be raised by adjusting the weights and biases to lower the energy of that image and to raise the energy of other images, especially those that have low energies and therefore make a big contribution to the partition function. The derivative of the log probability of a training vector with respect to a weight is surprisingly simple.

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_\infty, \tag{10}$$

where $\langle \cdot \rangle_0$ denotes the expectations for the data distribution (p_0) and $\langle \cdot \rangle_\infty$ denotes the expectations for the model distribution (p_∞) [17]. It can be done by starting at any random state of the visible units and performing alternating Gibbs sampling for a very long time. One iteration of alternating Gibbs sampling consists of updating all of the hidden units in parallel using Equation (8) followed by updating all of the visible units in parallel using Equation (9).

To solve this problem, Hinton proposed a much faster learning procedure - the Contrastive Divergence algorithm [18, 19]. The use of this procedure can be applied in order to correct the weights and bias of the network:

$$\Delta w_{ij} = \eta\left(\langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_\infty\right), \tag{11}$$

$$\Delta b_{\mathrm{v}i} = \eta(v_{i0} - v_{i\infty}), \tag{12}$$

$$\Delta b_{\mathrm{h}j} = \eta(h_{j0} - h_{j\infty}). \tag{13}$$

4. Proposed method

The proposed method for the key frame detection is based on the RBM algorithm. Our method consists of several steps. The first step divides input video into individual frames. Next, we create an input matrix for RBM network based on a key frame. Another stage is an attempt to reconstruct the pattern of key frame based on subsequent frames delivered to the input of the network. This stage is crucial. As a result, of the comparison between two frames (key frame and next frame) we obtain the error mapping, which is in the range ($0 \leq error \leq 1$), where 1 means no similarity and 0 – complete similarity. Next we compare the obtained error with the assumed threshold. If the effect of this comparison is higher than the expected threshold, the algorithm labels current frame as a key frame. Then the network is trained according to the new key frame. Otherwise, the network is given next frame. Diagram below (Fig. 3) shows the proposed algorithm**.**

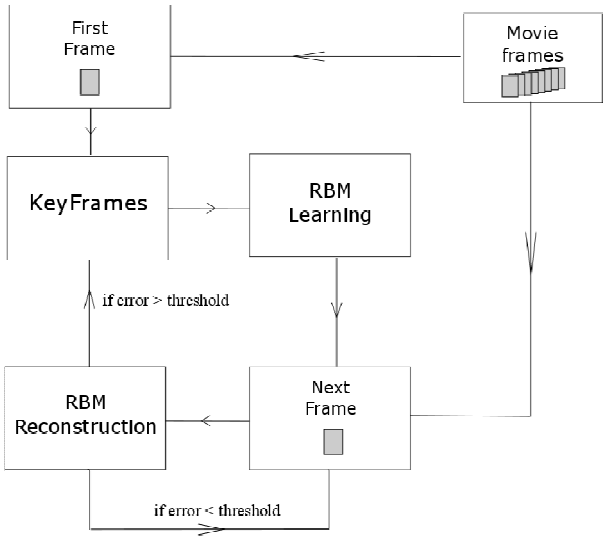

Fig. 3. Diagram of the proposed method

5. Experimental result

Algorithm stages were implemented in the Java programming language. The RBM implementation is based on an algorithm by A. Karpathy [20], adapted to the proposed method. Other steps were implemented by the authors. The process of training the network for each key frame consisted of 50 epochs.

During the first step, a sequence of numbered frames is obtained. Among these frames, some are detected as key frames. Whether a frame is a key frame or not is determined by calculating an error using equation (10) and comparing it with the threshold. If the error is greater than the threshold, the given frame is marked as

a key frame. In our first experiment, we set the threshold to 0.06 (see Fig. 5). This value was determined empirically.

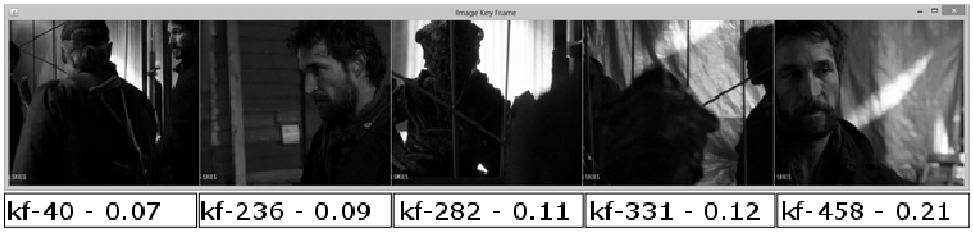

Fig. 4. Experiment 1. Threshold value was set to 0.06

Experimental results presented in Figure 4 show similar efficiency to the method proposed in the article. Detected key frames match the key frames detected in the mentioned work. The difference in frame numbers is a result of different numbering scheme. In our work frame numbers start with 0 while in article [8] frame numbers start with 1.

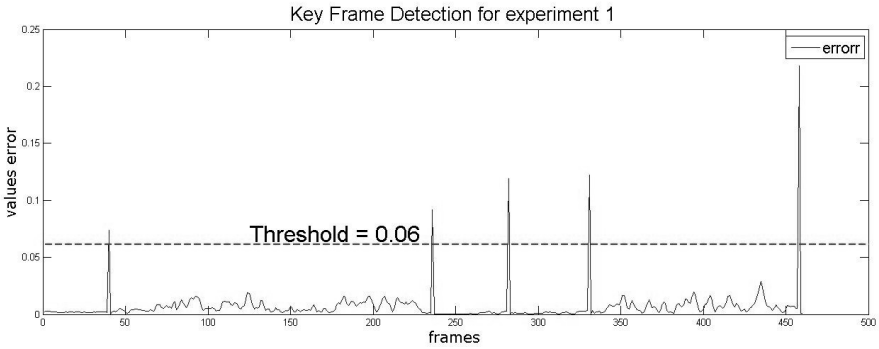

Fig. 5. Key frame detection

Similar results were obtained in the second experiment where input video had a resolution of 624x256 (Fig. 6). We had to increase the threshold to 0.12 because action in the movie was more dynamic than in the previous experiment, which resulted in generally higher values of error mapping Figure 7.

Fig. 6. Experiment 2. Threshold value was set to 0.12

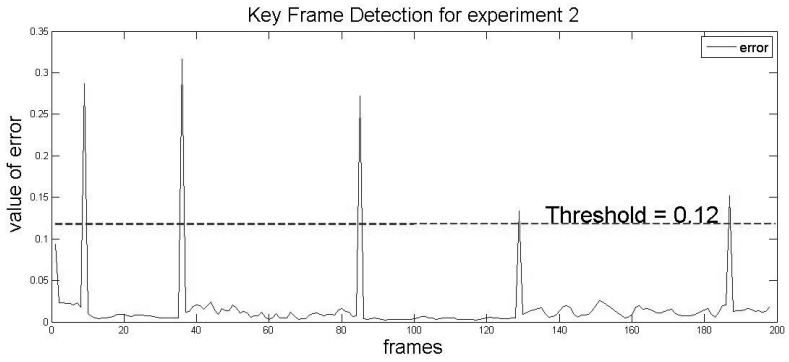

Fig. 7. Key frame detection for experiment 2

In experiment 3, the video input was obtained from a USB camera with a resolution of 640x480. Movement of the camera during recording and changes of lighting caused significant fluctuations which resulted in problems with comparing adjacent frames (Fig. 8). The threshold for this experiment had to be set to 0.17 in order to reduce a number of detected false-positive key frames. This value didn't eliminate all unnecessary key frames, but higher values would only cause more problems with key frame detection (Fig. 9).

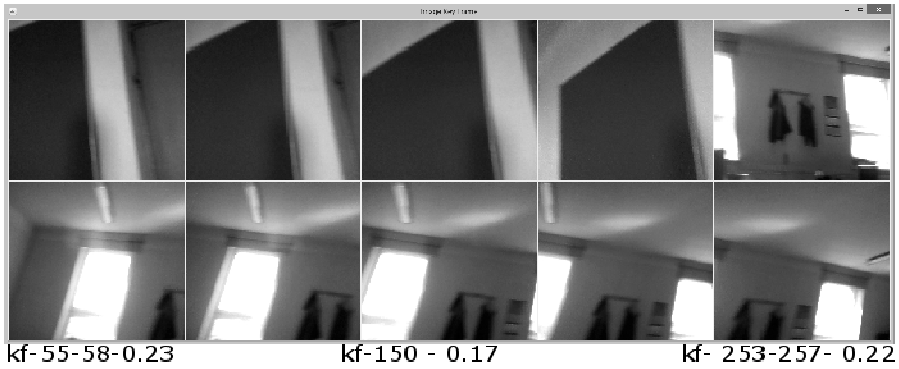

Fig. 8. Experiment 3. Threshold value was set to 0.17

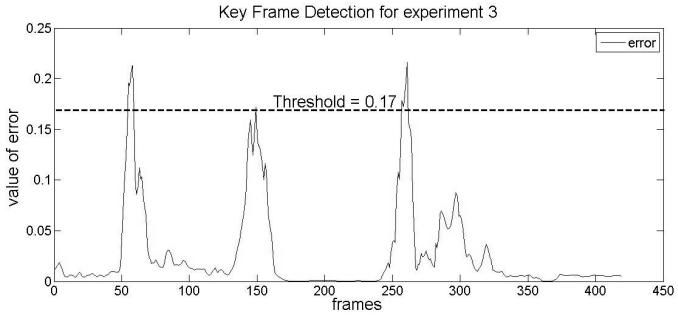

Fig. 9. Key frame detection for experiment 3

6. Conclusions

The presented method is a novel approach for video compression. Simulations verified the correctness of the algorithm. The presented approach requires only one input parameter - threshold. During the experiments we discovered that the proposed method has better results in professional videos because this type of video has better lighting and the transition of objects is slower. Amateur videos are often recorded without a tripod, thus the movement of the camera is more rapid. The algorithm is more effective in high resolution videos, but it requires more execution time (bigger input matrix). The method has been tested on various videos, but due to lack of space we had to narrow the presented simulations. In addition, our algorithm is resistant to any logo located on the video. In future research we will attempt to implement a method which automatically adapts threshold value.

Acknowledgment

The work presented in this paper was supported by a grant BS/MN 1-109-302/14/P "New video compression method using neural image compression algorithm".

References

[1] Rutkowski L., Jaworski M., Pietruczuk L., Duda P., Decision trees for mining data streams based on the Gaussian approximation, IEEE Transactions on Knowledge and Data Engineering 2014, 26(1), 108-119.

[2] Rutkowski L., Pietruczuk L., Duda P., Jaworski M., Decision trees for mining data streams based on the Mcdiarmid's bound, IEEE Transactions on Knowledge and Data Engineering 2013, 25(6), 1272-1279.

[3] Seeling P., Scene change detection for uncompressed video, [in:] Technological Developments in Education and Automation, Springer Netherlands, 2010, 11-14.

[4] Xinying Wang, Zhengke Weng, Scene abrupt change detection, [in:] Canadian Conference on Electrical and Computer Engineering 2000, 2, 880-883.

[5] Gentao Liu, Xiangming Wen, Wei Zheng, Peizhou He, Shot boundary detection and keyframe extraction based on scale invariant feature transform, [in:] Eighth IEEE/ACIS International Conference on Computer and Information Science, ICIS 2009, 1126-1130.

[6] Cierniak R., Knop M., Video compression algorithm based on neural networks, [in:] Artificial Intelligence and Soft Computing, volume 7894 of Lecture Notes in Computer Science, Springer, Berlin - Heidelberg 2013, 524-531.

[7] Knop M., Cierniak R., Shah N., Video compression algorithm based on neural network structures, [in:] Artificial Intelligence and Soft Computing, volume 8467 of Lecture Notes in Computer Science, Springer International Publishing, 2014, 715-724.

[8] Grycuk R., Knop M., Mandal S., Video key frame detection based on surf algorithm, [in:] Artificial Intelligence and Soft Computing, Lecture Notes in Computer Science, Springer, Berlin - Heidelberg 2015, 572-583.

[9] Knop M., Dobosz P., Neural video compression algorithm, [in:] Image Processing and Communications Challenges 6, volume 313 of Advances in Intelligent Systems and Computing, Springer International Publishing, 2015, 59-66.

[10] Zhong Qu, Lidan Lin, Tengfei Gao, Yongkun Wang, An improved keyframe extraction method based on HSV colour space, Journal of Software 2013, 8(7).

[11] Bay H., Tuytelaars T., Van Gool L., Surf: Speeded up robust features, [in:] Computer Vision-ECCV 2006, Springer 2006, 404-417.

[12] Bay H., Ess A., Tuytelaars T., Van Gool L., Speeded-up robust features (SURF), Computer Vision and Image Understanding 2008, 110(3), 346-359.

[13] Lowe D.G., Object recognition from local scale-invariant features, [in:] The Proceedings of the Seventh IEEE International Conference on Computer Vision 1999, 2, 1150-1157.

[14] Lowe D.G., Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 2004, 60(2), 91-110.

[15] Grycuk R., Gabryel M., Korytkowski M., Scherer R., Voloshynovskiy S., From single image to list of objects based on edge and blob detection, [in:] Artificial Intelligence and Soft Computing, volume 8468 of Lecture Notes in Computer Science, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L.A. Zadeh, J.M. Zurada eds., Springer International Publishing, 2014, 605-615.

[16] Grycuk R., Gabryel M., Korytkowski M., Scherer R., Content-based image indexing by data clustering and inverse document frequency, [in:] Beyond Databases, Architectures, and Structures, volume 424 of Communications in Computer and Information Science, S. Kozielski, D. Mrozek, P. Kasprowski, B. Małysiak-Mrozek, D. Kostrzewa eds., Springer International Publishing, 2014, 374-383.

[17] Le Roux N., Bengio Y., Representational power of restricted boltzmann machines and deep belief networks, Neural Computation 2008, 20(6),1631-1649.

[18] G. Hinton, Training products of experts by minimizing contrastive divergence, Neural Computation 2002, 14(8), 1771-1800.

[19] Hinton G., A practical guide to training restricted Boltzmann machines, Momentum 2010, 9(1), 926.

[20] Karpathy A., Cpsc 540 project: Restricted Boltzmann machines.