Noname manuscript No.
(will be inserted by the editor)

Restricted Boltzmann Machine as an Aggregation Technique for Binary Descriptors

Szymon Sobczak · Rafal Kapela · Kevin McGuinness · Aleksandra Swietlicka · Dariusz Pazderski · Noel E. O'Connor

Received: date / Accepted: date

Abstract The article presents a novel approach to the challenge of real-time image classification with deep neural networks. The proposed architecture of the neural network exploits computationally efficient local binary descriptors and uses a Restricted Boltzmann Machine (RBM) as a feature space projection step so that the resulting depth of the deep neural network can be reduced. A Contrastive Divergence procedure is used both for RBM training and for feature projection. The resulting neural networks exhibit performance close to the current state of the art but are characterized by a small model memory footprint (i.e., number of parameters) and extremely efficient computational complexity (i.e, response time). The low number of parameters makes these architectures applicable in embedded systems with limited memory or reduced computational capabilities.

Keywords Restricted Boltzmann Machine · image local binary descriptors · aggregation techniques of feature vectors

1 Introduction

Binary image descriptors, such as BRIEF [5], BRISK [21], FREAK [3], and ORB [27], are a popular alternative to the more common dense real-valued local descriptors like SIFT [22], SURF [4], HOG [8], and even features extracted from convolutional layers of pretrained deep neural networks [28] (CNN features). These descriptors, which represent local image patches using compact binary strings, offer several advantages over their real-valued counterparts. First, they are substantially more compact meaning more such descriptors can be held uncompressed in memory, allowing for large datasets to be processed efficiently. Second, binary descriptors usually require far less disk space. Third, they are typically orders of magnitude faster to extract from images than other approaches, which is particularly advantageous on hardware constrained devices like mobile phones and tablets. Finally, pairs of such descriptors are usually compared using Hamming distances, which are simpler and faster to calculate than Euclidean or similar distances defined on real-valued vectors.

Local image descriptors, be them binary or real-valued, can be extracted at automatically identified key points (such as Harris or Hessian-affine [23] interest points) or extracted densely on a regular grid. In either case, to describe an entire image or image region, it is necessary to somehow combine these local descriptors. In the case of local features extracted at key points, or features from arbitrarily shaped image regions, the number of such descriptors is variable and not known in advance. Ideally, we would like to aggregate, or encode, variable numbers of descriptors into a single fixed-length descriptor that describes the complete image or region. Fixed-length descriptors provide several advantages in subsequent applications. For example, the most popular models used in classification, like support vector machines, logistic regression, and many deep networks, assume fixed-length inputs. Fixed-length representations allow computing similarities using cosine distances, his-

Szymon Sobczak, Rafal Kapela, Aleksandra Swietlicka, Dariusz Pazderski
Poznan University of Technology
Tel.: +48-61-665-21084
E-mail: *name.surname*@put.poznan.pl

Kevin McGuinness, Noel E. O'Connor
Insight Centre for Data Analytics
Dublin City University
Tel.: +353-1-700-5078 E-mail: *name.surname*@insight-centre.org

togram intersections, and other popular kernels used in classification and retrieval applications.

There have been a whole range of feature aggregation/encoding mechanisms proposed for real-valued dense descriptors like SIFT. The most popular of these is the bag of words (BoW) encoding [29] (also known as bag of features and histogram of words). Bag of words models first require that a sample of data be used to fit a codebook using the k-means clustering algorithm. Typically, for retrieval applications, k is very large (≈10K-1M elements) requiring approximate nearest neighbour methods be used to fit the codebook [26]. Once an appropriate codebook is found, an image can be represented as a histogram of the codewords that appear in the image. This histogram is typically very sparse and has the same dimension as the codebook (k). Many methods for retrieval and classification are based on this approach. Moreover, many extensions have been proposed, which make use of first and second-order differences with the codewords (centroids) fit using k-means and similar mixture-density models. Examples include VLAD encoding [16] (first-order) and Fisher vector encoding [25] (second-order). Bag of words models have been successfully applied not only to engineered features like SIFT and HOG, but also to learned features from convolutional neural networks [24].

Although bag of words based models work well for aggregating real-valued feature vectors into fixed-sized descriptors, they are less appropriate for encoding binary descriptors. The reason is that these approaches rely on the k-means and similar mixture density models for generating a codebook. Such models are inappropriate for binary vectors because they use Euclidean distances to compare vectors, and compute centroids by averaging vectors; Hamming distances are more appropriate for binary vectors, and averaging multiple binary vectors produces a non-binary vector for which the Hamming distance cannot be used. Therefore, in order to use binary features in standard classification and retrieval pipelines, new aggregation methods are necessary.

This article proposes and investigates two methods for aggregating binary descriptors. All of them are based on restricted Boltzmann machines [9] (RBMs) to model the distribution of binary descriptors in an image, and then represent the image using the parameters of the model. We propose to use RBMs with stochastic binary units, which are especially well-suited to modeling the distributions of binary vectors, and whose parameters can be estimated relatively efficiently using contrastive divergence [14]. Unlike popular methods for encoding real-valued local descriptors like bag of words and Fisher vectors, none of the proposed methods require fitting a density model (codebook or mixture model, for example) on a large sample of descriptors, which makes the proposed methods appropriate for cross-dataset use where the underlying feature distributions may differ significantly.

We evaluate and compare our two new methods in the context of a multiclass classification task on a popular classification dataset and consider several different binary descriptors. The results demonstrate that the proposed approaches outperform standard bag of words encoding, the most widely used aggregation technique for real valued descriptors. We find that the RBM-based encoding technique is very effective in terms of accuracy, yet significantly less computationally costly than the other deep neural architectures. We also investigate the best hyperparameter settings for the RBM encoding technique, and make some recommendations based on our findings.

2 Motivation

Technological development in last years was especially important in context of neural networks and image recognition. Nowadays the computing power of personal computers is sufficient to run very complex neural networks consisting of hundreds of layers, like for example ResNets[13] or GoogleNets[30]. Adding parameters to the network may lead to improvement of the efficiency but also increases the time of response. The problem has been raised in [15], where authors created a small and efficient neural network that doesn't need a lot of resources to run. Our approach to this issue is by using RBM as an aggregator for binary descriptors. We think that thanks to this we can obtain relatively high-level image features faster by using less computational resources than other approaches.

3 Restricted Boltzmann machines

The Restricted Boltzmann Machine (RBM) is a generative model representing a probability distribution. It can be understood as a stochastic neural network that can learn a probability distribution based on the input data. Training an RBM is performed as fitting the distribution represented by RBM to the training data in the best possible way [10]

RBMs are a variation of general Boltzmann machines with the restriction that their neurons form a biograph. The structure of RBMs is built of two types of neurons – visible units $\mathbf{v}$, which correspond to the inputs, and hidden units $\mathbf{h}$, which are trained. Visible and hidden units form two layers of the RBM (Figure 1). In comparison with general Boltzmann machines (which are recurrent)

in RBMs there are only connections between visible and hidden units and no connections in-between the layers. The units of a standard RBM are binary-valued and are numbers generated from a Bernoulli distribution.

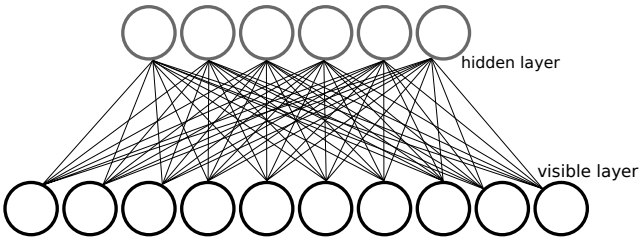

Fig. 1 The architecture of the RBM

It is possible to define the energy function of a restricted Boltzmann machine (which is analogous to the Hopfield recurrent neural network) as:

$$E(\mathbf{v},\mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j h_j w_{ij} v_i \quad (1)$$

or in the vector form:

$$E(\mathbf{v},\mathbf{h}) = -\mathbf{a}^T\mathbf{v} - \mathbf{b}^T\mathbf{h} - \mathbf{h}^T\mathbf{W}\mathbf{v} \quad (2)$$

where $\mathbf{W}$ (w_{ij}) is a weight matrix describing the connections between visible v_i and hidden h_j units. Additionally a_i and b_j are the biases for visible and hidden units, respectively. For each pair of visible and hidden units there is also a probability defined as:

$$P(\mathbf{v},\mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})} \quad (3)$$

where:

$$Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \quad (4)$$

is the so called "partition function" and it is given by summing over all possible pairs of visible and hidden units.

The learning procedure of the RBM can be composed in three steps:

(1) update of the hidden units,
(2) update of the visible units,
(3) update of weights and biases.

The first step is performed with use of conditional probability P defined by [14]:

$$P(h_j = 1|\mathbf{v}) = \sigma\left(b_j + \sum_{i=1}^{m} w_{ij} v_i\right), \quad (5)$$

where $\sigma(\cdot)$ is the following sigmoid function:

$$\sigma(x) = \frac{1}{1+e^{-x}}. \quad (6)$$

When this probability reaches some random number taken from a uniform distribution, then the corresponding hidden unit is activated.

Update of the visible units proceeds similarly, where the conditional probability is of the following form [10]:

$$P(v_i = 1|\mathbf{h}) = \sigma\left(a_i + \sum_{j=1}^{n} w_{ij} h_j\right). \quad (7)$$

The update of weights and biases is performed using the maximum-log-likelihood method, which can be described by the following formula [10]:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \eta \frac{\partial L(\mathbf{v},\mathbf{h})}{\partial w_{ij}}\bigg|_{w_{ij}^{(t)}} - \lambda w_{ij}^{(t)}, \quad (8)$$

where η is a learning rate, λ is a weight decay, and

$$L(\mathbf{v},\mathbf{h}) = \log P(\mathbf{v}) = \log\left(\frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}\right). \quad (9)$$

In order to compute the gradient we take advantage of the Contrastive Divergence (CD-k, where k refers to number of repetitions) method [10] and consider the following approximation

$$\frac{\partial L(\mathbf{v},\mathbf{h})}{\partial w_{ij}} \cong \langle v_i h_j\rangle_{data} - \langle v_i h_j\rangle_{model}, \quad (10)$$

where brackets $\langle\cdot\rangle_P$ express the expected value over the probability distribution given in the subscript and [10]:

$$P(\mathbf{h}|\mathbf{v}) = \prod_{j=1}^{n} P(h_j|\mathbf{v}). \quad (11)$$

Correspondingly, biases a_i and b_j are updated with following rules [10]:

$$a_i^{(t+1)} = a_i^{(t)} + \eta\left(v_i^{(0)} - v_i^{(k)}\right), \quad (12)$$

$$b_j^{(t+1)} = b_j^{(t)} + \eta\left(P\left(h_j = 1|\mathbf{v}^{(0)}\right) - P\left(h_j = 1|\mathbf{v}^{(k)}\right)\right). \quad (13)$$

4 Method

In the previous section we recalled a well known formulas that describe the way the RBM machines work and are trained. In this section we will present the two ideas of use of the RBMs as a technique of aggregation of binary descriptors. The first one is a simpler technique that can be used in real-time systems since it requires relatively small amount of operations to calculate. Second is based on the first one and employs it in more complex environment for more accurate image description.

4.1 Contrastive Divergence as an expansion of a feature space

Consider equation (10) which describes the update of the RBM weights during the training in order to fit the model into the data. After the successful training, the same equation can be used to provide the information about the distance between the new feature vector and the trained model. Thus, we can treat it intuitively as a sort of Fisher vector encoding technique where the derivative of the Gaussian kernels in a feature space serves as the descriptor transformation. The difference is that techniques like Fisher kernels and VLAD are not designed for binary descriptors. In addition, they tend to compress the features based on the assumption that the information provided by them is sufficiently rich to make it possible. However, for low dimensional binary descriptors, such an assumption is not justified due to the fact that they do not carry enough information to be compressed. For this reason, only the high dimensional binary features are applicable for techniques such as Locality Similarity Hashing (LSH) [20].

Our approach proposed in this article can be viewed as a non orthogonal transformation of binary data that leads to the dimensionality expansion. The resulting $V \times H$ matrix can be serialized to form the Contrastive Divergence Feature (CDV). The idea is presented in figure 2.

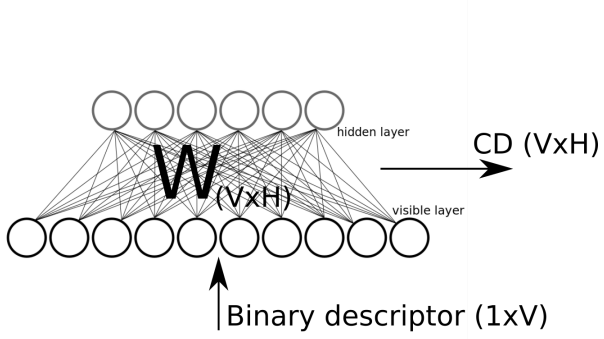

Fig. 2 Contrastive Divergence as a feature transformation technique

As an input we use the following, well known, local image binary features:

- Fast REtinA Keypoint (FREAK);
- Binary Robust Invariant Scalable Keypoints (BRISK);
- Local Binary Patterns with eight neighbors (LBP8);
- Local Binary Patterns with sixteen neighbors (LBP16).

In order to aggregate the input features a simple mean of the CDVs can be computed. Here, we can aggregate features from the region of consistent texture and color distribution to improve the efficiency of the description. For this reason we employ Selective Search (SS) [31] algorithm which gives image region candidates that are featured with mentioned characteristics. The result of image description is the set of mean CDVSS kernels. The overall procedure can be written as follows:

$$CDVSS = \begin{bmatrix} CDVSS_1 \\ CDVSS_2 \\ \cdots \\ CDVSS_N \end{bmatrix}, \tag{14}$$

where

$$CDVSS_i = \frac{1}{M} \sum_{i=1}^{N} (CDV_i) \tag{15}$$

is a mean of the CDVs inside the SS region.

4.2 RBM CD as a feature extraction layer in Deep Neural Networks

Deep Neural Networks usually consist of many convolutional layers, where each detects features from a previous layer. The general rule is that the deeper the layer the more complex features processed [19]. Thus adding an RBM to a convolutional neural network makes it possible to preprocess the input image to get relatively complex features in earlier layers. As a result less convolutional layers are needed overall. Figure 3 presents a simplified data workflow in the network.

Input layer is a single channel image which is directly processed by the binary descriptor layer, so every single pixel is transformed to a binary string. The next layer is an RBM which works as described in the previous section, so the output size of this layer is $W \times H \times (v \cdot h)$, where W and H denote width and height that are the same the dimensions of input image, v denotes number of visible units in the RBM and must be the same as number of bits in a single binary descriptor, h denotes number of hidden units in RBM and this is a customizable parameter.

For tests we used a network composed of three convolutional layers followed by max-pooling and two fully-connected layers with dropout between them.

As a binary descriptor we propose to apply the following descriptors:

- LBP8,
- LBP16,
- LBP8_rgb - 24bits descriptor composed of three 8-bits descriptor, each for one color channel,
- Reduced BRISK - BRISK reduced to 16 bits.

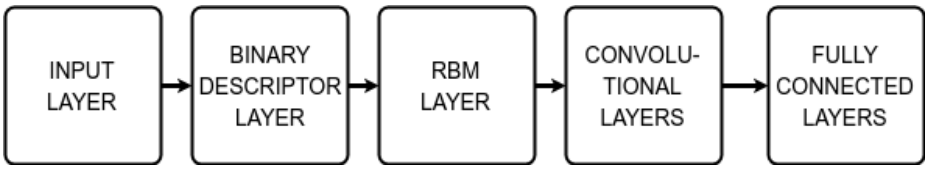

Fig. 3 Deep Neural Network with RBM as a feature extractor.

Input images were resized to 256×256 pixels. The RBM in the network has two hidden units, so the input dimensions of data for first convolutional layer is 256x256x(2·N), where N denotes the number of bits of the descriptor.

5 Experimental results

In order to check the correctness of the proposed image classification approach and compare it with the state of the art solutions, multiple test scenarios were conducted. For this purpose we used a rather simple image dataset – *Caltech-101* – as given the popularity of the database the results for comparison from other sources [2] can be easily obtained. The major features of the dataset are as follows:

- 101 categories;
- about 40 to 800 images per category;
- 20 images per category used for training.

On the other hand, in real-time processing scenarios an investigation of processing speed of the proposed algorithm should be conducted. For this reason we compared our solution to the most common deep learning algorithms. Again, nowadays it is relatively easy to find the results for the best performing solutions in the literature [11], [1]. The only problem is that they are sometimes given for a different hardware setup than used in our experiments. To accommodate this we took the closest, in terms of architecture and computational resources, architecture. Then, based on the benchmark results given by the vendor of the hardware we computed some penalty factor just like it is usually done [1].

For tests, the measurements have been performed on the following machine:

- CPU: Intel i5-6400, 2.70GHz
- RAM: 16GB, DDR4, 2133 MHz
- GPU: Nvidia GTX1070

Our software implementation is a two-fold system that can be run on CPU only machines or the ones equipped with CUDA technology. It implements CD_K method described by equation (10) for training. Due to high computational requirements that come together with high demand for a RAM memory the latter implementation is recommended for training. The testing environment was Ubuntu Linux with CUDA, cuDNN and GPU drivers installed along with the libraries necessary to run our application.

5.1 CD feature space tests

First conducted experiment was an evaluation of the simplified algorithm described in section 4.1. It is based on the feature space expansion for binary descriptors and then selective search based grouping of descriptors. The CDVSS kernels are matched basing on the k nearest neighbors procedure (KNN). The image class with the highest number of votes is the one chosen as the result. We have chosen four binary descriptors for our evaluation: BRISK, FREAK, LBP8 and LBP16. For simplicity we checked two RBM architectures which consist of 100 and 200 hidden neurons. Clearly, the number of visible units depends on the size of the descriptor. We noticed, that the considered cases are the most representative since the results do not vary too much for other setups. Next, we compared the number of images assigned for training the RBM model and for the KNN nodes that described given classes. Table 1 shows **accuracy/mAP** results.

As can be seen from the results the best performing binary descriptor is BRISK with close second result given for the FREAK descriptor. This is not a surprise because BRISK (mark in bold font) and FREAK hold the best results among the key-point based local image description techniques [7]. As mentioned, any architecture change of the RBM model above the base value that allows to learn the descriptors (about 90 for all the descriptors) does not introduce significant change in accuracy. Another result worth noting is that LBP features, which are not as descriptive as key-point based descriptors, are not that far in terms of accuracy and mAP results which makes them a potentially interesting choice for real-time processing applications since they are relatively easier to calculate.

5.2 CD feature space + deep neural network tests

Since the results presented above seem to be promising, the next conducted experiment was the evaluation of

Table 1 The evaluation results of CD-based KNN matching procedure

	100/30	100/50	200/30	200/50
BRISK	0.598;0.603	**0.605;0.613**	0.604;0.603	0.604;0.612
FREAK	0.613;0.575	0.618;0.580	0.614;0.571	0.618;0.574
LBP8	0.583;0.560	0.606;0.594	0.591;0.583	0.592;0.574
LBP16	0.581;0.550	0.581;0.551	0.582;0.554	0.584;0.558

the technique presented in section 4.2. It was a natural choice to take into account the initial usage of RBMs in deep belief networks and the capabilities of deep architectures. Moreover, it is also natural to expect more from the CD binary feature projections when more complex inference than KNN is implemented as a feature classification.

Accuracy results achieved by Deep Neural Network with RBM as features extractor are presented in the table 2.

As can be seen we have tested three LBP-based features (8/16 neighbors and 8 neighbors) formed and concatenated for each, separate RGB channel. In addition, in order to have a representative of the local binary feature descriptors we have chosen the BRISK descriptor for our experiments. Note that feeding the entire descriptor (512 binary values) to the RBM with n hidden units would result in $512 \times n$ number of channels inside the deep architecture of the neural network. That is why we have reduced the dimensionality of the BRISK descriptor to 16.

In table 3 we present how our method improves accuracy of the network. We compared accuracy of the network used in previous tests with some architectures that do not contain RBM inside, as can be seen our approach achieved the best result.

We also performed tests on networks consisting of different number of convolutional layers in order to check how this parameter affects accuracy and time of forward step. During those changes we kept the size of input to feedforward neural network the same by adjusting strides. The results are relative to network consisting of 3 convolutional layers and are presented in the table 4.

For all the binary descriptors used the training accuracy achieved for a validation data set was on a very good level. However, only LBP8 and LBP16 binary features have shown accuracy above the KNN-based matching technique. This is mainly due to the fact that most of the features in the presented descriptors are redundant, thus, causing a worsening of the generalization of the deep neural network.

The same convolutional neural network that works without RBM achieves accuracy close to 64%, thus we confirmed that the processing with LBP and RBM increases the accuracy. Other networks like ResNet [13] or Inception [30] achieves better results, but they are also much bigger and need much more resources. The MobileNet[15] network achieves around 70% depending on their implementation[17].

5.3 Time performance

Clearly, the processing time depends on the hardware setup. All the results in this section are presented for the setup given above. Especially, since the core of this system is implemented in CUDA for parallel computations on the GPU side the graphics card unit parameters are very important. In order to compare our results with other approaches calculated on different GPUs, we introduced some penalty factor based on the performance parameters just like in [1].

The table 5 presents computing time of each step in the processing pipeline. For clarity, we do not present any preprocessing times related to image forming or noise removal.

The time differences in the descriptor computing procedure depends mostly on the size of the descriptor. A reduced BRISK descriptor is used for comparison just to check if the local binary descriptors carry any additional information to the one presented in the LBP descriptors. Since the recognition accuracy does not show that this may be true, this descriptor was not implemented in CUDA technology, thus, the time presented in the table is for its CPU implementation.

Parallel implementation of RBM and LBP allows to have same RBM processing time for different dimensions of the input, for example in LBP8 and LBP_rgb. Note, that LBP_rgb is treated in the CUDA implementation as three separate channels of LBP8, thus, the calculation time does not differ from the LBP8. LBP16 however changes the dimmensionality of the RBM input and this results in extended time needed for calculating th RBM response (RBM has twice as much parameters in this case).

The CNN column of the Table 5 presents calculation times for the convolutional and fully connected parts of the deep architecture presented here. They are all

Table 2 The evaluation of the accuracy of CD-RBM as an entry for DNN

descriptor	validation accuracy	validation top5	training accuracy
LBP8	**0.68**	0.82	0.99
LBP16	0.65	0.87	0.98
LBP8_rgb	0.42	0.70	0.81
reduced BRISK	0.51	0.72	0.98

Table 3 The evaluation of RBM accuracy comparing to other approaches

method	LBP + RBM + CNN	LBP + CNN	CNN (raw grayscale images)	CNN (raw rgb images)
accuracy	0.68	0.63	0.54	0.62

Table 4 The evaluation of system accuracy and processing time comparing to the architecture with 3 convolutional layers.

number of convolutional layers	accuracy	time
2	-16%	-16%
4	-3%	+2%
5	-3%	+4%

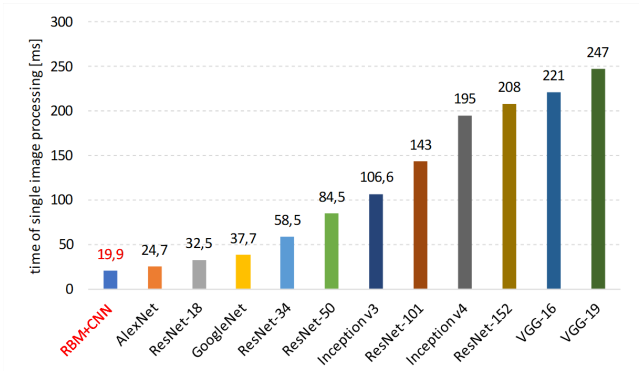

Fig. 4 Time of processing single image for common used neural network architectures (only forward step).

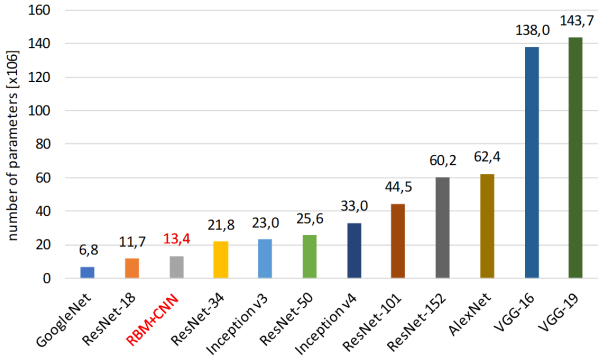

Fig. 5 Number of parameters for common used neural network architectures [6].

similar across the proposed architectures since the number of parameters in these layers is also similar. The only exception is LBP_rgb which has three channels of RBM responses so that more operations are needed to calculate the response from the last parts of the network. This is reflected in the increased value in this row. The last column presents the cumulative time for particular architecture.

The entire network is trained in two steps, first in which we train the RBM, and second where we train the following layers. For each of them we are training to the point where the accuracy and error values reach plateau point (i.e., the gradient of the error function does not change significantly). This usually takes from 2 to 3 hours.

As was shown in Table 5 the overall times of the network response are relatively low. Figure 4 shows the comparison of our solution to the most common state-of-the-art algorithms. Again. for clarity we present only the forward times since they represent the image recognition procedure. The closest competitors in terms of computational time seems to be the AlexNet (24,7ms) [18], GoogleNet/Inception (37,7ms) [30] and ResNet (32,5ms) [13] models. Obviously behind these times stand the three factors:

- number of parameters;
- the architecture – i.e., number of data dependencies in order to calculate the final output;
- implementation and hardware setup.

Figure 5 presents number of parameters in the considered architectures. As it can be seen the number of parameters in the AlexNet exceeds significantly the number of parameters in the other two and our solutions. Since the processing time is shorter than the GoogleNet

Table 5 The evaluation of times between following stages of the CD-RBM pipeline

descriptor name	descriptor computing time	RBM processing time	CNN	$\sum$
LBP8	3.7ms	13ms	3.2ms	19.9 ms
LBP8_rgb	5.4ms	13ms	4ms	22.4 ms
LBP16	6.6ms	21ms	3.4ms	31 ms
reduced BRISK	1500ms	13ms	3.4ms	1513.4 ms

and ResNet it means that the network is relatively shallow comparing to the other solutions (i.e., they have more internal data dependencies). Its implementation is also relatively efficient since the caffe model uses grouped convolutions [1]. Relatively high number of parameters in this network excludes this network to be used in the higher batch modes. Our solution features both a low number of parameters and low processing time which facilitates its use in real-time applications that run on embedded platforms.

6 Conclusion

This article illustrates the use of Restricted Boltzman Machines and specifically the Contrastive Divergence training procedure in the image classification task. Two approaches have been taken:

- CD calculated in the image region for KNN-based class matching;
- CD as a layer in the deep neural network.

Both approaches are characterized with relatively good accuracy that is comparable to the best results achieved so far [12]. The high top-5 accuracy shows that the proposed models are very close to the correct answer even if the maximum response indicates another image class. In terms of model memory footprint (i.e., number of parameters) and computational complexity (i.e, response time) it was shown that our approach with CD-DNN is the fastest available now. Also, the relatively low number of parameters makes these architectures applicable in embedded systems with limited memory or reduced computational capabilities.

For future work, we will investigate how to reduce the dimensionality of RBM output to be able to use a smaller neural network as the final classifier. Reducing the dimensionality should remove some redundant data which will possibly give higher accuracy and smaller processing time,

Right now the size of the first layers in the network highly depends on the size of the descriptor and number of hidden units in the RBM, we want to investigate a method that will allow to have these parameters flexible but without big influence on the processing time.

Since the memory footprint of our model is relatively low, one future direction will be to implement a batch mode for parallel input processing.

References

1. Github, cnn-benchmarks. https://github.com/jcjohnson/cnn-benchmarks. Accessed: 2018-11-22
2. Hao Wooi Lim's blog, friday, august 21, 2009, table of results for caltech 101 dataset. http://zybler.blogspot.com/2009/08/table-of-results-for-famous-public.html. Accessed: 2018-11-22
3. Alahi, A., Ortiz, R., Vandergheynst, P.: Freak: Fast retina keypoint. In: Computer vision and pattern recognition (CVPR), pp. 510–517 (2012)
4. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded up robust features. In: European conference on computer vision (ECCV), pp. 404–417 (2006)
5. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: BRIEF: Binary robust independent elementary features. In: European conference on computer vision (ECCV), pp. 778–792 (2010)
6. Canziani A. Culurciello E, P.A.: An analysis of deep neural network models for practical applications (2016). URL https://arxiv.org/abs/1605.07678
7. Chatoux, H., Lecellier, F., Fernandez-Maloigne, C.: Comparative study of descriptors with dense key points. 2016 23rd International Conference on Pattern Recognition (ICPR) pp. 1988–1993 (2016)
8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. 886–893 (2005)
9. Fischer, A., Igel, C.: An introduction to restricted boltzmann machines. Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications pp. 14–36 (2012)
10. Fischer, A., Igel, C.: An introduction to restricted boltzmann machines. In: L. Alvarez, M. Mejail, L. Gomez, J. Jacobo (eds.) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, pp. 14–36. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
11. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. CoRR **abs/1406.4729** (2014). URL http://arxiv.org/abs/1406.4729
12. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. CoRR **abs/1406.4729** (2014). URL http://arxiv.org/abs/1406.4729
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016). DOI 10.1109/CVPR.2016.90

14. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Neural computation **14**(8), 1771–1800 (2002)
15. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Efficient convolutional neural networks for mobile vision applications (2017). URL https://arxiv.org/pdf/1704.04861.pdf
16. Jégou, H., Douze, M., Schmid, C., Pérez, P.: Aggregating local descriptors into a compact image representation. In: Computer Vision and Pattern Recognition (CVPR), pp. 3304–3311 (2010)
17. Kornblith, S., Shlens, J., Le, Q.V.: Do better imagenet models transfer better? (2018). URL https://arxiv.org/pdf/1805.08974.pdf
18. Krizhevsky, A., Sutskever, I., E. Hinton, G.: Imagenet classification with deep convolutional neural networks. Neural Information Processing Systems **25** (2012). DOI 10.1145/3065386
19. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012). URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
20. Kulis, B., Grauman, K.: Kernelized locality-sensitive hashing for scalable image search. In: 2009 IEEE 12th International Conference on Computer Vision, pp. 2130–2137 (2009). DOI 10.1109/ICCV.2009.5459466
21. Leutenegger, S., Chli, M., Siegwart, R.Y.: Brisk: Binary robust invariant scalable keypoints. In: International conference on computer vision (ICCV), pp. 2548–2555 (2011)
22. Lowe, D.G.: Object recognition from local scale-invariant features. In: International conference on computer vision (ICCV), vol. 2, pp. 1150–1157 (1999)
23. Mikolajczyk, K., Schmid, C.: An affine invariant interest point detector. In: European conference on computer vision (ECCV), pp. 128–142 (2002)
24. Mohedano, E., McGuinness, K., O'Connor, N.E., Salvador, A., Marques, F., Giro-i Nieto, X.: Bags of local convolutional features for scalable instance search. In: International Conference on Multimedia Retrieval (ICMR), pp. 327–331 (2016)
25. Perronnin, F., Dance, C.: Fisher kernels on visual vocabularies for image categorization. In: Computer Vision and Pattern Recognition (CVPR), pp. 1–8 (2007)
26. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: Computer Vision and Pattern Recognition (CVPR), pp. 1–8 (2007)
27. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: An efficient alternative to sift or surf. In: International conference on computer vision (ICCV), pp. 2564–2571 (2011)
28. Sharif Razavian, A., Azizpour, H., Sullivan, J., Carlsson, S.: CNN features off-the-shelf: an astounding baseline for recognition. In: Computer Vision and Pattern Recognition (CVPR) Workshops, pp. 806–813 (2014)
29. Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: International Conference on Computer Vision (ICCV), vol. 2, pp. 1470–1477 (2003)
30. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Computer Vision and Pattern Recognition (CVPR) (2015). URL http://arxiv.org/abs/1409.4842
31. Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T., Smeulders, A.W.M.: Selective search for object recognition. International Journal of Computer Vision **104**(2), 154–171 (2013). DOI 10.1007/s11263-013-0620-5. URL https://doi.org/10.1007/s11263-013-0620-5