RESTRICTED BOLTZMANN MACHINE IMAGE COMPRESSION

Markus Küchhold*, Maik Simon* and Thomas Sikora
Communication Systems Group, Technische Universität Berlin
Einsteinufer 17, 10587 Berlin
{kuechhold, simon, sikora}@nue.tu-berlin.de

***Abstract*—We propose a novel lossy block-based image compression approach. Our approach builds on non-linear autoencoders that can, when properly trained, explore non-linear statistical dependencies in the image blocks for redundancy reduction. In contrast the DCT employed in JPEG is inherently restricted to exploration of linear dependencies using a second-order statistics framework. The coder is based on pre-trained class-specific Restricted Boltzmann Machines (RBM). These machines are statistical variants of neural network autoencoders that directly map pixel values in image blocks into coded bits. Decoders can be implemented with low computational complexity in a codebook design. Experimental results show that our RBM-codec outperforms JPEG at high compression rates, both in terms of PSNR, SSIM and subjective results.**

***Index Terms*—Restricted Boltzmann Machines, Autoencoder, Image Compression**

I. Introduction

Our challenge is the development of image compression approaches that combine excellent efficiency with very low encoder and decoder complexity. Low complexity decoders are demanded i.e. in browser implementations and in mobile devices, sensor networks or drones. Here the speed of en-/decoding and/or the power consumption is of prime concern.

The most widely employed lossy compression algorithm for still images is the JPEG image coding standard [19][21] and maintains reasonable compression efficiency. The computational complexity of the decoder is dominated by the DCT implementation. In general, image coders based on codebooks, such as vector quantization approaches [2][9][14], can provide very low complexity at the decoder. An index is transmitted to the decoder and pixels stored in a decoder codebook are directly used for reconstructing pixels in blocks. No further signal processing operations are required. In sparse coding approaches [3] signals are often reconstructed using a superposition of codebook entries. Recent machine learning methods and approaches led to significant improvements in image compression [4][5][20].

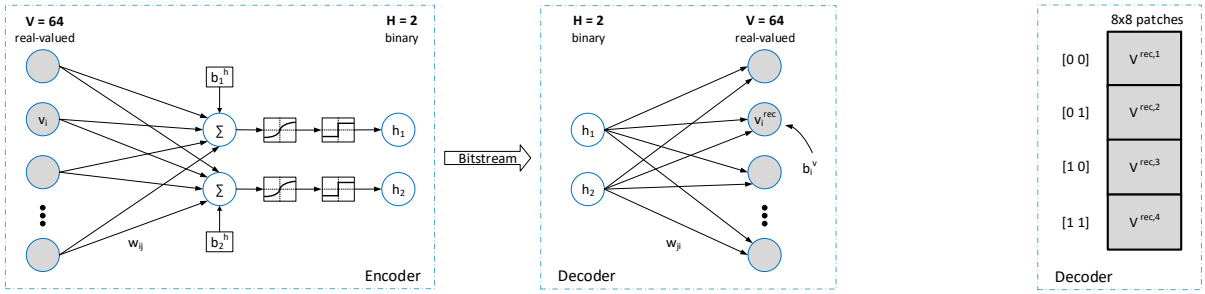
In this paper we provide first results for an image compression approach based on Restricted Boltzmann Machines. RBMs are stochastic variants of neural network autoencoders which became very popular stochastic feature learning models in recent years. RBMs are frequently applied in fields of classification [16], feature learning [6] and collaborative filtering [18]. In addition, they are applied in autoencoder architectures to reduce the dimensionality of data [13]. These non-linear autoencoders can, when properly trained, explore non-linear redundancies in the image blocks for redundancy reduction. In contrast the DCT employed in JPEG is inherently restricted to exploration of linear redundancies using a second-order statistics framework. Compared to "classical" neural networks RBMs can generate the statistical distribution of the input data and present important deep learning building blocks. An important aspect for our interest in RBMs is their fast training capability. For training the Contrastive Divergence algorithm proposed by Hinton [10] allows to derive highly efficient RBMs with much fewer training data compared to classical neural networks [12][17]. To our knowledge RBMs have as of today not been used for image compression.

Our approach to image compression follows the block-based technique of JPEG. However, the DCT in the encoder and decoder is replaced by a set of class-specific RBM autoencoders. RBM decoders can be implemented in a codebook design to trade off signal processing operations with codebook memory for fast decoding. The pre-categorization classifies each block in a specific RBM class according to its main edge direction. The mean value of a block is quantized and DPCM coded as with JPEG. The remaining texture details in the difference blocks are compressed choosing different binary hidden layer sizes of the RBMs.

II. Restricted Boltzmann Machines

A Restricted Boltzmann Machine is a stochastic variant of a general Boltzmann Machine with the restriction that an RBM has only connections between a stochastic hidden and a stochastic visible layer. This constraint makes training of the network parameters more feasible but does not impose too many restrictions on the capability of the network for efficient coding. Our proposed codec uses indiviually trained Gaussian-Bernoulli RBMs (GB-RBM) to map the real-valued pixels of an image patch into binary codewords. A GB-RBM has V real-valued visible units and H binary hidden units [13]. An example for a GB-RBM with a visible layer size $V = 64$, respectively for an 8x8 input block, and a hidden layer size $H = 2$ is shown in Fig. 1a. The GB-RBM assigns a probability, governed by an energy function [22], to each

* Equal contribution



(a) GB-RBM autoencoder

(b) codebook

Fig. 1: (a) An example of the class-specific RBM for coding the difference block D_k. (b) A learnt codebook can replace the decoder part.

image patch:

$$E(\mathbf{v},\mathbf{h}) = \sum_{i}^{V} \frac{(v_i - b_i^v)^2}{2\sigma_i^2} - \left(\sum_{j=1}^{H} b_j^h h_j + \sum_{i=1}^{V} \sum_{j=1}^{H} \frac{v_i}{\sigma_i} h_j w_{ij} \right). \quad (1)$$

The energy function represents the joint configuration of the stochastic visible and hidden units, where v_i is the i-th state of the visible state vector $\mathbf{v}$ and h_j is the j-th state of the hidden state vector $\mathbf{h}$. The real-valued weight between visible unit i and hidden unit j is represented by w_{ij}. The real-valued bias into hidden unit j is given by b_j^h and the real-valued bias into visible unit i is given by b_i^v. The Gaussian visible units v_i are controlled by the standard deviation σ_i. Weight matrix and biases of the RBM need to be "trained" based on a set of training data from real images.

A. RBM encoder, decoder and learning

We employ the Contrastive Divergence learning algorithm [10][11][15] for learning. In the following we discuss briefly the main training steps for the optimization of the weight matrix. The updates for the biases can be derived in a similar way. Given a set of training data from real images, the real-valued pixels of an 8x8 training patch are assigned to visible units. To obtain the binary hidden state vector $\mathbf{h}$ for a given training input $\mathbf{v} = \mathbf{v}^c$, a conditional probability for each hidden unit is computed in Eq. 2. For each block in the training set the computed conditional probability is equivalent to the logistic function.

$$P(h_j = 1|\mathbf{v}^c) = \frac{1}{1 + e^{-(\sum_{i=1}^{V} \frac{v_i^c}{\sigma_i} w_{ij} + b_j^h)}} \quad (2)$$

To get the binary state of each hidden layer, the following case distinction is used:

$$h_j = \begin{cases} 0 \text{ if } P(h_j|\mathbf{v}) < 0.5 \\ 1 \text{ if } P(h_j|\mathbf{v}) \geq 0.5. \end{cases} \quad (3)$$

The above Eq. 3 and Eq. 2 are the RBM encoder equations and the derived binary hidden vectors represent the coded bits for an 8x8 block. The decoder reconstruction of the visible layer pixels in each 8x8 block is computed by using the coded bits in the hidden layer:

$$v_i^{rec} = \sigma_i h_j w_{ji} + b_i^v. \quad (4)$$

The above forms a stochastic RBM autoencoder, the "decoder" depicted in Fig. 1a. To update the parameters during training, the probability for the activation of the hidden units under the condition of the reconstructed training input v_i^{rec} is given by Eq. 5:

$$P(h_j = 1|\mathbf{v}^{rec}) = \frac{1}{1 + e^{-(\sum_{i=1}^{V} \frac{v_i^{rec}}{\sigma_i} w_{ij} + b_j^h)}}. \quad (5)$$

With learning rate α_w and C the number of blocks in the training set the final update rule for the weight matrix is:

$$\Delta w_{ij} = \frac{\alpha_w}{\sigma_i} \sum_{c=1}^{C} \left[v_i^c p\left(h_j = 1|\mathbf{v}^c\right) - v_i^{rec} p\left(h_j = 1|\mathbf{v}^{rec}\right) \right]. \quad (6)$$

III. System Architecture

The architecture of the proposed RBM-codec is depicted in Fig. 2. A MxN gray-scaled image is divided in K 8x8 blocks. The mean is removed from each block and coded separately as with JPEG. Since we employ a stochastic framework removal of mean is mandatory [11].

After normalization the value range of D lies between $[-1, 1]$. We employ the Histogram of Oriented Gradients (HOG) algorithm [7] to classify the 8x8 difference blocks D_k of I according to distinctive edge orientation. 8, 16 or 32 classes are used depending on the chosen configuration. For encoding of each D_k each class uses one class-specific trained GB-RBM.

A. Encoding

The components of D_k are encoded using class-specific GB-RBMs. For our block sizes of 8x8 the visible layer size is fixed to $V = 64$ units. Adaptive number of bits per blocks can be generated by the coder to account for varying complexity of coded textures - by choosing different hidden layer sizes with $H = [2, 4, 6, 8, 12, 16]$. A GB-RBM with H hidden units has 2^H binary states, which represent possible codewords. Those binary codewords are obtained by computing the conditional probability, as denoted in Eq. 2. The standard deviation σ_i is derived from the training data step and fixed for coding. To get the final binary state of a hidden layer Eq. 3 is used. Notice that from 2^H states only H states are linear independent

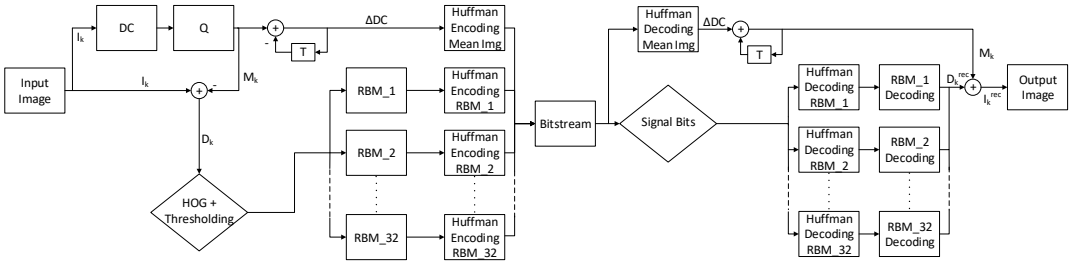


Fig. 2: Architecture of the proposed new RBM-codec (*setup3_32RBMs*).

TABLE I: General Setup Configurations

configuration	setup1_8RBMs	setup2_16RBMs	setup3_32RBMs
signal bits	$S = 3$	$S = 4$	$S = 5$
number of RBM classes (het/hom)	8 (7/1)	16 (14/2)	32 (28/4)
possible hidden layer sizes	[2,4,6,8,12,16] H_{hom} =H	[2,4,6,8,12,16] H_{hom} = H	[6,8,12,16] $H_{\text{hom}} = 6$
threshold$_{\text{hom}}$	$t_{\text{hom}} = 0.001$	$t_{\text{hom}} = 0.001$	$t_{\text{hom}} = 0.001$
thresholding parameters	–	$t_{\text{he}} = 3 \cdot 10^{-3}$ $t_{\text{ho}} = 5 \cdot 10^{-4}$	$t_{\text{he\_1}} = 0.033$ $t_{\text{he\_2}} = 0.013$ $t_{\text{he\_3}} = 0.005$ $t_{\text{ho\_1}} = 5 \cdot 10^{-4}$ $t_{\text{ho\_2}} = 2 \cdot 10^{-4}$ $t_{\text{ho\_3}} = 5 \cdot 10^{-5}$

base images. To take this into account the proposed codec was designed in three configurations. In a first step of encoding, the blocks D_k are divided into seven classes according to their main edge direction using HOG algorithm. The homogeneous blocks are assigned to the 8th class by a variance threshold t_{hom}. All eight classes form the base model (*setup1_8RBMs*) for two other configurations. For the second configuration (*setup2_16RBMs*) two additional variance thresholds were used to divide the existing eight classes into further subclasses. As a result, the codec has in *setup2_16RBMs* 16 GB-RBMs, which are separately trained. To achieve higher quality the third configuration (*setup3_32RBMs*) with 32 GB-RBMs is implemented using six additional thresholds. In the first and second configuration all class-specific GB-RBMs have the same number of hidden units (H). In the third configuration the four homogeneous classes were constrained to a hidden layer size of $H_{\text{hom}} = 6$. The possible sizes of each configuration and the used classifying thresholds are depicted in Table I. The thresholding parameters were derived from experimentation. To achieve a better bit saving each RBM codeword is be finally Huffman encoded. The stored/transmitted bits of each block include the signal-bits S to indicate the used GB-RBM, the bits of the Huffman codeword r_{D_k}, which represents the difference block, and the bits of the mean image r_{M_k}. The resulting bits per pixel for an image accounts to :

$$\text{r}_{\text{RBM-Codec}} = \frac{\sum_{k=1}^{K}(\frac{S}{64} + \text{r}_{D_k} + \text{r}_{M_k})}{K}[\text{bpp}]. \qquad (7)$$

B. Decoding

The decoder reconstructs the mean blocks M_k and the difference blocks D_k separately. The implemented DC-decoder delivers the reconstructions of DC values of blocks I_k, as shown in Fig. 2. The signal-bits S indicate which GB-RBM and its corresponding Huffman dictionary was used for encoding of D_k. The selected Huffman dictionary returns the respective binary hidden layer state of the GB-RBM. The Eq. 4 is used to obtain the reconstructed block $D_k^{rec} = \mathbf{v}^{rec}$. To receive the final reconstructed image I^{rec}, the mean image M is added to the reconstruction of D^{rec}.

C. Complexity of the decoder

Our desired property of the proposed RBM-codec is the extremely fast decoding of D^{rec}. An RBM with hidden layer size H can generate $2^H - H$ output 8x8 image blocks, which are linear combinations of only H base images. I.e. for $H = 2$ there exist two base images for the binary states $[1\ 0]$ and $[0\ 1]$ and state $[1\ 1]$ is their sum:

$$\mathbf{v}^{rec} = \sigma[1\ \ 1]w' = \sigma\left([0\ \ 1] + [1\ \ 0]\right)w', \qquad (8)$$

where w' is $(2\text{x}64)$. The decoding can be implemented in different ways. Each encoded 8x8 block can be reconstructed using Eq. 4, where the binary state vector **h** is multiplied with the trained weight matrix (our biases are all zeros). The reconstruction can also be implemented using a codebook design as illustrated in Fig. 1b. The decoded Huffman codewords are then codebook indices. 8x8 blocks are reconstructed by the patches $\mathbf{v}^{rec}$ stored in the codebooks without any arithmetical operation (except the mean needs to be added). Alternatively reconstruction can also be done as in Eq. 8 where only the base images are store in a smaller codebook and for reconstruction of the pixels the base images are superimposed according to the bit pattern. No multiplications are required as with the JPEG inverse DCT.

IV. EXPERIMENTS AND RESULTS

All three presented RBM-codec configurations were trained separately and for each single GB-RBM a class-specific training set was used. The complete dataset for our training contained 2.6 million randomly chosen 8x8 samples from the ImageNet dataset [8]. Each configuration was trained with the same training set. The class-specific training samples were

permuted after each epoch. The used learning rates α and other parameters for the training of each RBM are shown in Table II. The weight matrix was randomly initialized to small values, the biases for the visible and hidden units were set to zero, based on the recommendations of Hinton [11]. We observed that learning the visible biases resulted in lower visual quality, therefore we set the learning rate α_{b_v} to zero. Optimization is done using Contrastive Divergence [10]. The proposed RBM-codec was implemented in MATLAB.

All three RBM-codec configurations were tested with gray-scaled standard images from the Miscellaneous dataset [1]. Based on the implemented configurations, the proposed RBM-codec is currently limited to lower bpp ratios. The results of our experiments on images with low resolutions are depicted in Table III. This allows comparison between the lowest possible quality level of JPEG and the RBM-codec at a similar bit rate. Also same comparison at the highest possible quality level of the RBM-codec is shown. For all six test images the RBM-codec outperforms JPEG drastically at lowest possible JPEG bpp values. For test image *Lena_crop* at the lowest quality of JPEG, the RBM-codec achieves a gain of 1.55dB (with 10% less bits). A visual performance comparison between JPEG and the RBM-codec is shown in Fig. 3. It is evident that images are reconstructed with drastically better quality and more texture details compared to JPEG. This observation can be verified also for higher bit rates of the RBM-codec in Fig. 3c. Also in homogeneous regions a much more visually pleasing result is obtained, resulting in less blocking artifacts and improved texture details. This can be explained inspecting the distribution of the classes for coding this image. 14%, 18%, 6%, 4%, 3%, 4%, 8% and 44% were assigned to the classes 1-8. Class 8 caters for the background with homogenous texture. Class 1 contains vertical and classes 2 and 3 mainly diagonal structures. PSNR and SSIM values of these images are easily cross-referenced with Table III. Not surprisingly, at these low rate SSIM is by far the better quality measure compared to PSNR.

The rate-distortion curves for *Lena_crop* are shown in Fig. 4. This result is typical for the performance of the RBM-codec and illustrates that our performance is somewhere between JPEG and JPEG2000. Based on SSIM measure a bit reduction of up to 50% is achieved.

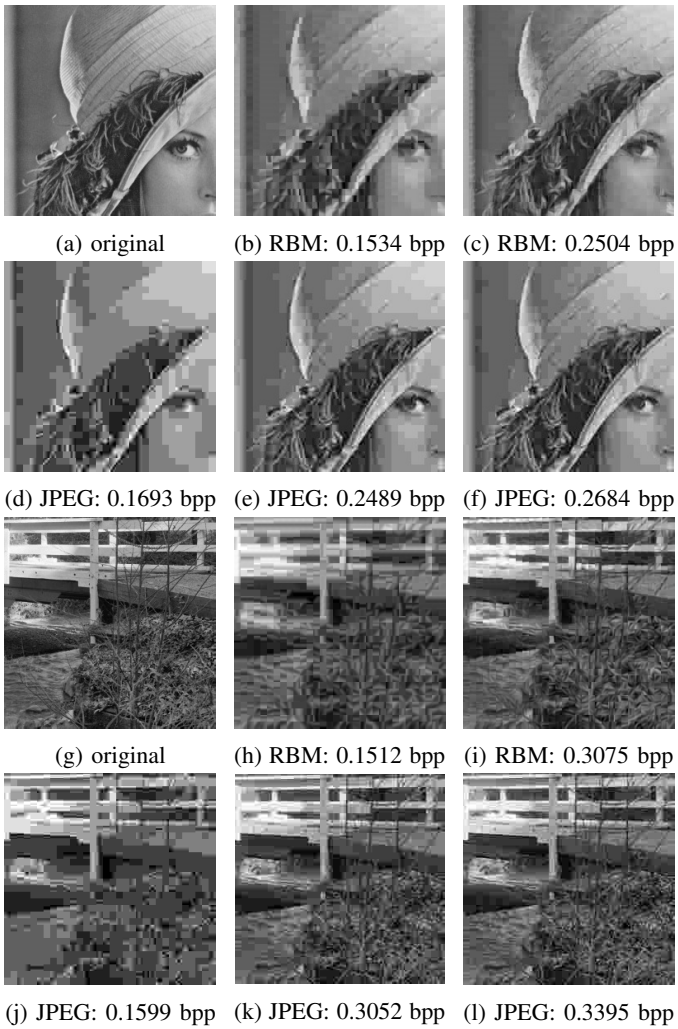
TABLE II: Training Parameters

Parameter	Value	Parameter	Value
α_w	0.00002	batchsize	20
α_{b_h}	0.0001	epochs	1000
α_{b_v}	0.0	σ	0.1

V. CONCLUSION AND FUTURE WORK

Our novel RBM-codec for image compression outperforms the standard JPEG compression scheme at low bit rates significantly. In addition the design at the decoder admits a very low complex "fast" and "power efficient" implementation. The approach builds on a set of stochastic autoencoders that can, when properly trained, explore non-linear dependencies in the image blocks for redundancy reduction. In contrast the DCT employed in JPEG is inherently restricted to exploration of linear redundancies using a second-order statistics framework. We believe that the combination of pre-classification and non-linear mapping is a promising research direction.

The presented codec is currently limited to low numbers of bits per block. We plan in our future work to extend the RBM-codec to higher qualities using more RBMs and to exploit their dependencies in a more efficient way using more advanced



(a) original (b) RBM: 0.1534 bpp (c) RBM: 0.2504 bpp

(d) JPEG: 0.1693 bpp (e) JPEG: 0.2489 bpp (f) JPEG: 0.2684 bpp

(g) original (h) RBM: 0.1512 bpp (i) RBM: 0.3075 bpp

(j) JPEG: 0.1599 bpp (k) JPEG: 0.3052 bpp (l) JPEG: 0.3395 bpp

Fig. 3: Visual comparison between the RBM-codec (b-c), (h-i) and JPEG (d-f), (j-l) for original test image *Lena_crop* (a) and an image detail of *Stream and Bridge* (g).

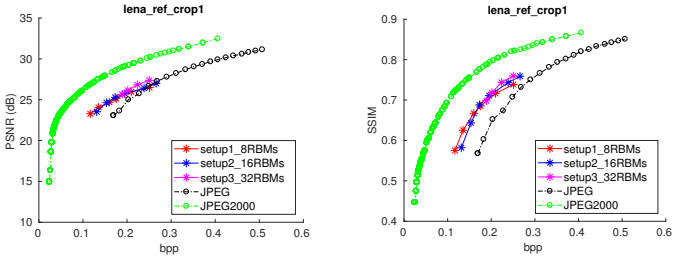


Fig. 4: Rate-distortion curves for test image *Lena_crop*.

training and using more levels of pre-processing. It is also possible to stack RBMs into a deep RBM network to envision deep learning coding strategies.

VI. Acknowledgements

This work was supported by a Google Faculty Research Award 2016 in Machine Perception.

References

[1] University of South California, The USC-SIPI Image Database, 1977. Signal and Image Processing Institute, [Online] Available: http://sipi.usc.edu/database/.

[2] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1141–1151. Curran Associates, Inc., 2017.

[3] M. Aharon, M. Elad, and A. Bruckstein. rmk-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, Nov 2006.

[4] J. Ballé, V. Laparra, and E. P. Simoncelli. End-to-end optimization of nonlinear transform codes for perceptual quality. In *2016 Picture Coding Symposium (PCS)*, pages 1–5, Dec 2016.

[5] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*, 2018.

[6] A. Coates, H. Lee, and A. Y. Ng. An analysis of single-layer networks in unsupervised feature learning. *AISTATS*, 2011.

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, June 2005.

[8] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.

[9] A. Gersho and R. Gray. *Vector Quantization and Signal Compression*. The Springer International Series in Engineering and Computer Science. Springer US, 2012.

[10] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, Aug. 2002.

[11] G. E. Hinton. A Practical Guide to Training Restricted Boltzmann Machines. In *Neural Networks: Tricks of the Trade - Second Edition*, pages 599–619. 2012.

[12] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006.

[13] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

[14] C. Karri and U. Jena. Fast vector quantization using a bat algorithm for image compression. *Engineering Science and Technology, an International Journal*, 19(2):769 – 781, 2016.

[15] A. Krizhevsky and G. Hinton. Learning Multiple Layers of Features from Tiny Images. 2009. Masters thesis, Department of Computer Science, University of Toronto.

[16] H. Larochelle and Y. Bengio. Classification using discriminative restricted boltzmann machines. In W. W. Cohen, A. McCallum, and S. T. Roweis, editors, *ICML*, volume 307 of *ACM International Conference Proceeding Series*, pages 536–543. ACM, 2008.

[17] R. Salakhutdinov and G. Hinton. Deep Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 5, pages 448–455, 2009.

[18] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 791–798, New York, NY, USA, 2007. ACM.

[19] T. Sikora. Trends and perspectives in image and video coding. *Proceedings of the IEEE*, 93(1):6–17, Jan 2005.

[20] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell. Full resolution image compression with recurrent neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5435–5443, July 2017.

[21] G. K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, Feb 1992.

[22] T. Yamashita, M. Tanaka, E. Yoshida, Y. Yamauchi, and H. Fujiyoshii. To be bernoulli or to be gaussian, for a restricted boltzmann machine. In *2014 22nd International Conference on Pattern Recognition*, pages 1520–1525, Aug 2014.

TABLE III: Bit Rates and Quality Measures for Small Resolution

test name	Aerial (256x256)						Clock (256x256)					
quality	JPEG low		medium		RBM high		JPEG low		medium		RBM high	
codec	JPEG	RBM	JPEG	RBM	JPEG	RBM	JPEG	RBM	JPEG	RBM	JPEG	RBM
bit rate [bpp]	0.2162	0.1963	0.2449	0.2444	0.3462	0.3155	0.1801	0.1774	0.2078	0.1948	0.2249	0.2221
PSNR [dB]	19.755	20.532	20.383	21.240	22.324	22.370	23.556	24.665	25.265	25.824	26.101	26.436
SSIM	0.4999	0.5687	0.5500	0.6262	0.6713	0.6947	0.7613	0.8216	0.7958	0.8376	0.8152	0.8552
test name	Elaine (256x256)						Lena_crop (256x256)					
quality	JPEG low		medium		RBM high		JPEG low		medium		RBM high	
codec	JPEG	RBM	JPEG	RBM	JPEG	RBM	JPEG	RBM	JPEG	RBM	JPEG	RBM
bit rate [bpp]	0.1720	0.1595	0.2295	0.2285	0.2731	0.2538	0.1693	0.1534	0.2033	0.2015	0.2684	0.2504
PSNR [dB]	23.429	25.061	26.503	27.580	28.121	28.165	23.080	24.638	25.015	26.159	27.275	27.379
SSIM	0.5965	0.7016	0.7144	0.7912	0.7710	0.8038	0.5680	0.6433	0.6522	0.7184	0.7319	0.7597
test name	Couple (256x256)						Peppers (256x256)					
quality	JPEG low		medium		RBM high		JPEG low		medium		RBM high	
codec	JPEG	RBM	JPEG	RBM	JPEG	RBM	JPEG	RBM	JPEG	RBM	JPEG	RBM
bit rate [bpp]	0.1758	0.1603	0.2196	0.2186	0.2810	0.2772	0.1847	0.1739	0.1975	0.1909	0.2470	0.2482
PSNR [dB]	22.035	23.002	23.575	24.321	25.073	25.426	22.857	24.483	23.518	25.044	26.019	26.806
SSIM	0.5112	0.5847	0.6029	0.6795	0.6844	0.7362	0.6333	0.7399	0.6609	0.7572	0.7416	0.8097

TABLE IV: Bit Rates and Quality Measures for Higher Resolution

test name	Boat (512x512)						Man (1024x1024)					
quality	JPEG low		medium		RBM high		JPEG low		medium		RBM high	
codec	JPEG	RBM	JPEG	RBM	JPEG	RBM	JPEG	RBM	JPEG	RBM	JPEG	RBM
bit rate [bpp]	0.1491	0.1382	0.2091	0.2035	0.2510	0.2607	0.1238	0.1330	0.1912	0.1937	0.2482	0.2476
PSNR [dB]	23.438	23.447	26.254	25.567	27.317	26.711	23.819	24.254	27.081	26.198	28.661	27.731
SSIM	0.5717	0.5992	0.6847	0.6950	0.7268	0.7395	0.5296	0.5946	0.6757	0.6978	0.7345	0.7440
test name	Mandrill (512x512)						Stream and Bridge (512x512)					
quality	JPEG low		medium		RBM high		JPEG low		medium		RBM high	
codec	JPEG	RBM	JPEG	RBM	JPEG	RBM	JPEG	RBM	JPEG	RBM	JPEG	RBM
bit rate [bpp]	0.1531	0.1477	0.2557	0.2594	0.3004	0.3134	0.1599	0.1512	0.2689	0.2626	0.3052	0.3075
PSNR [dB]	19.921	20.292	21.518	21.411	22.045	23.042	21.466	21.354	23.622	23.448	24.086	23.904
SSIM	0.3815	0.4147	0.5403	0.5749	0.5828	0.6084	0.4512	0.4749	0.6122	0.6250	0.6439	0.6582