AUTOENCODER BASED IMAGE COMPRESSION: CAN THE LEARNING BE QUANTIZATION INDEPENDENT?

Thierry Dumas, Aline, Roumy and Christine Guillemot

INRIA Rennes Bretagne-Atlantique
thierry.dumas@inria.fr, aline.roumy@inria.fr, christine.guillemot@inria.fr

ABSTRACT

This paper explores the problem of learning transforms for image compression via autoencoders. Usually, the rate-distortion performances of image compression are tuned by varying the quantization step size. In the case of autoencoders, this in principle would require learning one transform per rate-distortion point at a given quantization step size. Here, we show that comparable performances can be obtained with a unique learned transform. The different rate-distortion points are then reached by varying the quantization step size at test time. This approach saves a lot of training time.

***Index Terms*—** Image compression, deep autoencoders, quantization.

1. INTRODUCTION

Image coding standards all use linear and invertible transforms to convert an image into coefficients with low statistical dependencies, i.e suited for scalar quantization. Notably, the discrete cosine transform (DCT) is the most commonly used for two reasons: (i) it is image-independent, implying that the DCT does not need to be transmitted, (ii) it approaches the optimal orthogonal transform in terms of rate-distortion, assuming that natural images can be modeled by zero-mean Gaussian-Markov processes with high correlation [1]. Deep autoencoders have been shown as promising tools for finding alternative transforms [2, 3, 4]. Autoencoders learn the encoder-decoder non-linear transform from natural images.

In the best image compression algorithms based on autoencoders [5, 6, 7], one transform is learned per rate-distortion point at a given quantization step size. Then, the quantization step size remains unchanged at test time so that the training and test conditions are identical. By contrast, image coding standards implement adaptive quantizations [8, 9]. Should the quantization be imposed during the training? To answer this, we propose an approach where the transform and the quantization are learned jointly. Then, we investigate whether, at test time, the compression falls apart when the coefficients obtained with the learned transform are quantized using quantization step sizes which differ from those in the training stage. The code to reproduce our numerical results and train the autoencoders is available online[1].

Matrices and tensors are denoted by bold letters. $\|\mathbf{X}\|_F$ is the Frobenius norm of $\mathbf{X}$. $\mathbf{X} \odot \mathbf{Z}$ is the elementwise multiplication between $\mathbf{X}$ and $\mathbf{Z}$.

2. JOINT LEARNING OF THE TRANSFORM AND THE QUANTIZATION

Section 2 introduces an efficient autoencoder for image compression. Then, it details our proposal for learning jointly this autoencoder transform and the quantization.

2.1. Autoencoder for image compression

An autoencoder is a neural network with an encoder g_e, parametrized by $\boldsymbol{\theta}$, that computes a representation $\mathbf{Y}$ from the data $\mathbf{X}$, and a decoder g_d, parametrized by $\boldsymbol{\phi}$, that gives a reconstruction $\hat{\mathbf{X}}$ of $\mathbf{X}$, see Figure 1. Autoencoders can be used for denoising or dimensionality reduction. When it is used for compression, the representation is also quantized, leading to the new quantized representation $\hat{\mathbf{Y}} = \mathcal{Q}(\mathbf{Y})$. If an autoencoder has fully-connected layers [10, 11, 12], the number of parameters depends on the image size. This implies that one autoencoder has to be trained per image size. To avoid this, an architecture without fully-connected layer is chosen. It exclusively comprises convolutional layers and non-linear operators. In this case, $\mathbf{Y} \in \mathbb{R}^{h \times w \times m}$ is a set of m feature maps of size $n = h \times w$, see Figure 1.

The basic autoencoder training minimizes the image reconstruction error [13]. In order to create a rate-distortion optimization, the authors in [6] add the minimization of the entropy of the quantized representation. Moreover, a bit allocation is performed by learning a normalization for each feature map of $\mathbf{Y}$. The encoder followed by the normalizations at the encoder side, parametrized by $\boldsymbol{\varphi}_e$, are denoted $\overline{g}_e(\,.\,;\boldsymbol{\theta},\boldsymbol{\varphi}_e)$. Similarly, the normalizations at the decoder side, parametrized by $\boldsymbol{\varphi}_d$, followed by the decoder are denoted $\overline{g}_d(\,.\,;\boldsymbol{\varphi}_d,\boldsymbol{\phi})$. Finally, this leads to (1).

This work has been supported by the French Defense Procurement Agency (DGA).

[1] www.irisa.fr/temics/demos/visualization_ae/visualizationAE.htm

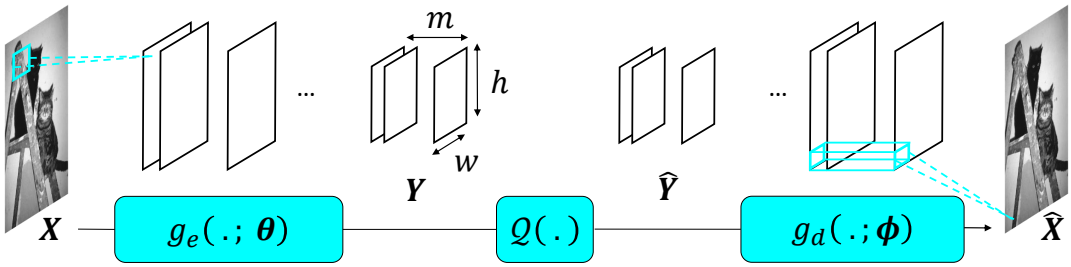


Fig. 1: Illustration of an autoencoder for image compression.

$$\min_{\substack{\boldsymbol{\theta},\boldsymbol{\varphi}_e,\\ \boldsymbol{\varphi}_d,\boldsymbol{\phi}}} \mathbb{E}\left[\left\|\mathbf{X}-\overline{g}_d\left(\mathcal{Q}\left(\overline{g}_e\left(\mathbf{X};\boldsymbol{\theta},\boldsymbol{\varphi}_e\right)\right);\boldsymbol{\varphi}_d,\boldsymbol{\phi}\right)\right\|_F^2+\gamma\sum_{i=1}^{m}H_i\right]$$

$$H_i=-\frac{1}{n}\sum_{j=1}^{n}\log_2\left(\hat{p}_i\left(\hat{y}_{ij}\right)\right),\gamma\in\mathbb{R}_+^* \tag{1}$$

$\hat{p}_i$ is the probability mass function of the i^{th} quantized feature map coefficients $\{\hat{y}_{ij}\}_{j=1\ldots n}$. The expectation $\mathbb{E}[.]$ is approximated by averaging over a training set of images. Unfortunately, $\mathcal{Q}$ makes minimization (1) unusable. Indeed, the derivative of any quantization with respect to its input is 0 at any point. Consequently, $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}_e$ cannot be learned via gradient-based methods [14]. To get around this issue, [6] fixes the quantization step size to 1 and approximates the uniform scalar quantization with the addition of a uniform noise of support [-0.5, 0.5]. Note that, even though the quantization step size is fixed, the bit allocation varies over the different feature maps via the normalizations. In the next section, we consider instead to remove the normalizations and learn explicitly the quantization step size for each feature map of $\mathbf{Y}$.

2.2. Learning the quantization step sizes

We address the problem of optimizing the quantization step size for each feature map of $\mathbf{Y}$. Because of the quantization, the function to be minimized is an implicit function of the quantization step sizes $\{\delta_i\}_{i=1\ldots m}$. The target is to make it an explicit function of $\{\delta_i\}_{i=1\ldots m}$. For $q\in\{\ldots,-\delta_i,0,\delta_i,\ldots\}$,

$$\hat{p}_i\left(q\right)=\int_{q-0.5\delta_i}^{q+0.5\delta_i}p_i\left(t\right)dt=\delta_i\tilde{p}_i\left(q\right) \tag{2}$$

$\tilde{p}_i=p_i*l_i$ where p_i is the probability density function of the i^{th} feature map coefficients $\{y_{ij}\}_{j=1\ldots n}$ and l_i denotes the probability density function of the continuous uniform distribution of support $[-0.5\delta_i,0.5\delta_i]$. The normalizations are removed from (1) and, using (2), (1) becomes (3).

$$\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathbb{E}\left[\left\|\mathbf{X}-g_d\left(g_e\left(\mathbf{X};\boldsymbol{\theta}\right)+\boldsymbol{\mathcal{E}};\boldsymbol{\phi}\right)\right\|_F^2+\gamma\sum_{i=1}^{m}\tilde{h}_i\right] \tag{3}$$

$$\tilde{h}_i=-\log_2\left(\delta_i\right)-\frac{1}{n}\sum_{j=1}^{n}\log_2\left(\tilde{p}_i\left(y_{ij}+\varepsilon_{ij}\right)\right)$$

The i^{th} matrix of $\boldsymbol{\mathcal{E}}\in\mathbb{R}^{h\times w\times m}$ contains n realizations $\{\varepsilon_{ij}\}_{j=1\ldots n}$ of $\mathcal{E}_i$, $\mathcal{E}_i$ being a continuous random variable of probability density function l_i. In (3), the function to be minimized is differentiable with respect to $\boldsymbol{\theta}$. $\boldsymbol{\theta}$ can thus be learned via gradient-based methods. However, $\{\delta_i\}_{i=1\ldots m}$ cannot yet be learned as the function to be minimized in (3) is not differentiable with respect to $\{\delta_i\}_{i=1\ldots m}$. This is resolved using the change of variable $\mathcal{E}_i=\delta_i\mathcal{T}$ where $\mathcal{T}$ is a random variable following the continuous uniform distribution of support $[-0.5,0.5]$. Now, the minimization over $\{\delta_i\}_{i=1\ldots m}$ is feasible, see (4).

$$\min_{\substack{\boldsymbol{\theta},\boldsymbol{\phi},\\ \delta_1,\ldots,\delta_m}} \mathbb{E}\left[\left\|\mathbf{X}-g_d\left(g_e\left(\mathbf{X};\boldsymbol{\theta}\right)+\boldsymbol{\Delta}\odot\mathbf{T};\boldsymbol{\phi}\right)\right\|_F^2+\gamma\sum_{i=1}^{m}\tilde{h}_i\right]$$

$$\tilde{h}_i=-\log_2\left(\delta_i\right)-\frac{1}{n}\sum_{j=1}^{n}\log_2\left(\tilde{p}_i\left(y_{ij}+\delta_i\tau_{ij}\right)\right) \tag{4}$$

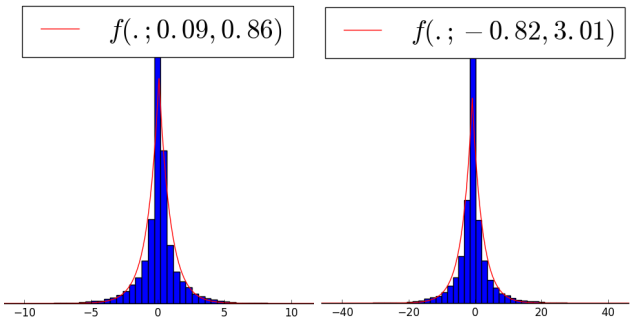
The i^{th} matrix of $\mathbf{T}\in\mathbb{R}^{h\times w\times m}$ contains n realizations $\{\tau_{ij}\}_{j=1\ldots n}$ of $\mathcal{T}$. All the coefficients in the i^{th} matrix of $\boldsymbol{\Delta}\in\mathbb{R}^{h\times w\times m}$ are equal to δ_i. A detail has been left out so far: $\tilde{p}_i$ is unknown. In a similar manner to [5, 6], $\tilde{p}_i$ can be replaced by a function $\tilde{f}_i$, parametrized by $\boldsymbol{\psi}^{(i)}$, and $\boldsymbol{\psi}^{(i)}$ is learned such that $\tilde{f}_i$ fits $\tilde{p}_i$.

In the end, we end up with three groups of parameters: $\{\boldsymbol{\theta},\boldsymbol{\phi}\}$, $\{\delta_i\}_{i=1\ldots m}$ and $\{\boldsymbol{\psi}^{(i)}\}_{i=1\ldots m}$. These three groups are learned by alternating three different stochastic gradient descents. All the training heuristics are detailed in the code[1].

Section 2 has developped an approach for learning explictly the transform and a quantization step size for each feature map of $\mathbf{Y}$. Before evaluating this approach in Section 4, Section 3 studies what would happen if, at test time, the coefficients in $\mathbf{Y}$ are quantized using quantization step sizes that differ from those in the training stage. This first requires understanding the internal structure of $\mathbf{Y}$ after the training.

3. INSIDE THE LEARNED REPRESENTATION

This section studies the different feature maps of $\mathbf{Y}$ after the training. To this end, a deep convolutional autoencoder must first be built and trained. g_e is the composition of a convolutional layer, a generalized divisive normalization (GDN) [15], a convolutional layer, a GDN and a convolutional layer.



(a) $i = 50$ (b) $i = 125$

Fig. 2: Normed histogram of the i^{th} feature map of $\mathbf{Y}$.

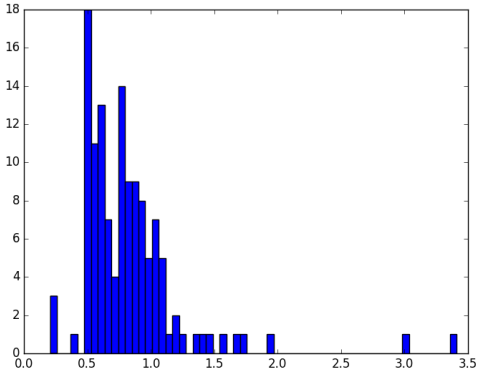


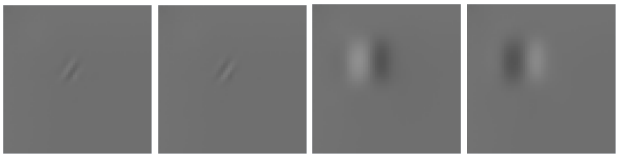
Fig. 3: Histogram of the $m-1$ scales provided by the fitting.

g_d is the reverse composition, replacing each GDN with an inverse generalized divisive normalization (IGDN) [15] and each convolutional layer with a transpose convolutional layer [16]. It is important to stress that $m = 128$, $\mathbf{X}$ has one channel and the convolutional strides and paddings are chosen such that h and w are 16 times smaller than respectively the height and the width of $\mathbf{X}$. Therefore, the number of pixels in $\mathbf{X}$ is twice the number of coefficients in $\mathbf{Y}$. The training set contains 24000 luminance images of size 256×256 that are extracted from ImageNet [17]. The minimization is (4), $\gamma = 10000.0$. Note that, if a GDN was placed immediately after g_e, a IGDN was placed immediately before g_d and, $\forall i \in [|1, m|], \delta_i = 1.0$ was not learned, the autoencoder architecture and the training would correspond to [6].

3.1. Distribution of the learned representation

After the training, a test set of 24 luminance images of size 512×768 is created from the Kodak suite[2]. Here, $\mathbf{X}$ refers to a test luminance image. Figure 2 shows the normed histogram of the 50^{th} feature map of $\mathbf{Y} = g_e(\mathbf{X}; \boldsymbol{\theta})$ and that of its 125^{th} feature map, averaged over the test set. Every feature map of $\mathbf{Y}$, except the 90^{th}, has a normed histogram similar to those displayed. To be more precise, let's write the probability density function of the Laplace distribution with mean $\mu \in \mathbb{R}$ and scale $\lambda \in \mathbb{R}_+^*$, denoted $f(\,.\,;\mu,\lambda)$.

[2] *r0k.us/graphics/kodak/*



(a) $\alpha = 8.0$ (b) $\alpha = -8.0$ (c) $\alpha = 20.0$ (d) $\alpha = -20.0$

Fig. 4: 64×64 crop at the top-left of $\hat{\mathbf{X}}$. $j = 50$ in (a) and (b). $j = 125$ in (c) and (d).

$$f(x;\mu,\lambda) = \frac{1}{2\lambda}\exp\left(-\frac{|x-\mu|}{\lambda}\right)$$

$\forall i \in [|1, m|]$, $i \neq 90$, there exists $\mu_i \in \mathbb{R}$ and $\lambda_i \in \mathbb{R}_+^*$ such that $f(\,.\,;\mu_i,\lambda_i)$ fits well the normed histogram of the i^{th} feature map of $\mathbf{Y}$. Note that most of the $m-1$ scales belong to $[0.5, 2.0]$, see Figure 3. For transformed coefficients having a zero-mean Laplace distribution, [18] proves that a uniform reconstruction quantizer (URQ) with constant decision offsets approaches the optimal scalar quantizer in terms of squared-error distortion for any quantization step size. Yet, in our case, (i) the $m-1$ Laplace probability density functions are not zero-mean, (ii) uniform scalar quantizers are used instead of this URQ. The point (i) is not problematic as an extra set of luminance images is used to compute an approximation $\overline{\mu}_i \in \mathbb{R}$ of the mean of the i^{th} feature map of $\mathbf{Y}$, then, at test time, the i^{th} feature map of $\mathbf{Y}$ is centered via $\overline{\mu}_i$ before being quantized. Note that $\{\overline{\mu_i}\}_{i=1\ldots m}$ does not depend on the test luminance images, thus incurring no transmission cost. Regarding the point (ii), it must be noted that the decoder mapping of the URQ is exactly the decoder mapping of the uniform scalar quantization with same quantization step size. Since our case comes close to the requirements of the proof in [18], at test time, the rate-distortion trade-off should not collapse as the quantization step sizes deviate from the learned values. This will be verified in Section 4.

3.2. Internal structure of the learned representation

The shortcoming of the previous fitting is that it does not reveal what information each matrix of $\mathbf{Y}$ encodes. To discover it, further visualizations are needed. The most common way of exploring a deep convolutional neural network (CNN) trained for image recognition is to look at the image, at the CNN input, resulting from the maximization over its pixels of a given neural activation in the CNN [19, 20, 21]. Precisely, [19, 20] maximize over the image pixels a given neural activation at the CNN output, i.e a class probability. This shows what image features characterize this class according to the CNN. In our case, the maximization over the image pixels of a given coefficient in $\mathbf{Y}$ does not yield interpretable images. Indeed, the coefficients in $\mathbf{Y}$ are not bounded. This may explain why the maximization often returns saturated images.

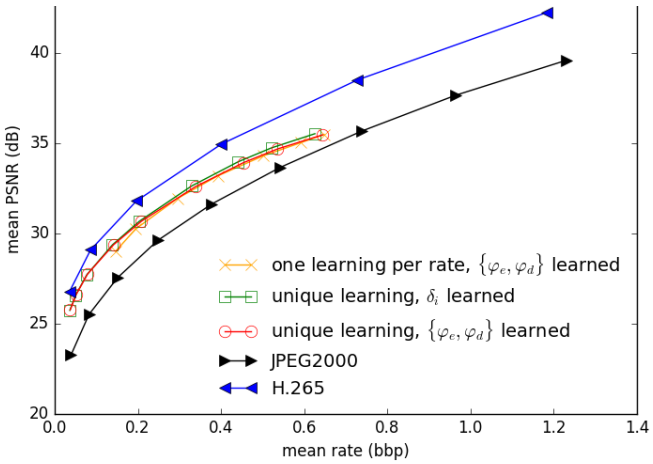


Fig. 5: Rate-distortion curves averaged over the 24 luminance images from the Kodak suite.

Alternatively, the information the j^{th} feature map of $\mathbf{Y}$ encodes, $j \in [|1, m|]$, can be seen as follows. $\forall i \in [|1, m|]$, all the coefficients in the i^{th} feature map of $\mathbf{Y}$ are set to $\overline{\mu}_i$. This way, the feature maps of $\mathbf{Y}$ contains no significant information. Then, a single coefficient in the j^{th} feature map of $\mathbf{Y}$ is set to $\alpha \in \mathbb{R}$ and $\hat{\mathbf{X}} = g_d\left(\mathcal{Q}\left(\mathbf{Y}\right); \boldsymbol{\phi}\right)$ is displayed. α is selected such that it is near one of the two tails of the Laplace distribution of the j^{th} feature map of $\mathbf{Y}$. Figure 4 shows the 64×64 crop at the top-left of $\hat{\mathbf{X}}$ when the single coefficient is located at the top-left corner of the j^{th} feature map of $\mathbf{Y}$, $j \in \{50, 125\}$. We see that the 50^{th} feature map of $\mathbf{Y}$ encodes a spatially localized image feature whereas its 250^{th} feature map encodes a spatially extended image feature. Moreover, the image feature is turned into its symmetrical feature, with respect to the mean pixel intensity, by moving α from the right tail of the Laplace distribution of the j^{th} feature map of $\mathbf{Y}$ to the left tail. This linear behaviour is observed for each feature map of $\mathbf{Y}$.

It is interesting to see that, given the fitting in Section 3.1, $\mathbf{Y}$ is similar to the DCT coefficients for blocks of prediction error samples in H.265 [9] in terms of distribution. However, when looking at the information each feature map of $\mathbf{Y}$ encodes, $\mathbf{Y}$ has nothing to do with these DCT coefficients.

4. EXPERIMENTS

We now evaluate in terms of rate-distortion performances: (i) whether the way of learning the quantization matters, (ii) whether, at test time, it is efficient to quantize the coefficients obtained with the learned transform using quantization step sizes which differ from those in the training stage. This is done by comparing three cases.

The 1^{st} case follows the approach in [6]. One transform is learned per rate-distortion point, the bit allocation being learned via the normalizations. In details, an autoencoder is trained for each $\gamma \in S = \{10000.0, 12000.0, 16000.0, 24000.0, 40000.0, 72000.0, 96000.0\}$. During the training and at test time, the quantization step size is fixed to 1.0.

In the 2^{nd} case, a unique transform is learned, the bit allocation being done by learning a quantization step size per feature map. More precisely, a single autoencoder is trained for $\gamma = 10000.0$ and $\{\delta_i\}_{i=1\ldots m}$ is learned, see Section 2. At test time, the rate varies as the quantization step sizes are equal to the learned quantization step sizes multiplied by $\beta \in \mathcal{B} = \{1.0, 1.25, 1.5, 2.0, 3.0, 4.0, 6.0, 8.0, 10.0\}$.

In the 3^{rd} case, a unique transform is learned, the bit allocation being learned via the normalizations. In details, a single autoencoder is trained for $\gamma = 10000.0$ and, during the training, the quantization step size is 1.0. At test time, the rate varies as the quantization step size spans $\mathcal{B}$.

In the 2^{nd} case, the autoencoder has the architecture described at the beginning of Section 3. In the 1^{st} and 3^{rd} case, a GDN is also placed after g_e and a IGDN is placed before g_d. The autoencoders are trained on 24000 luminance images of size 256×256 that are extracted from ImageNet. Then, at test time, the 24 luminance images from the Kodak suite are inserted into the autoencoders. The rate is estimated via the empirical entropy of the quantized coefficients, assuming that the quantized coefficients are i.i.d. Note that, for the 2^{nd} and the 3^{rd} case, we have also implemented a binarizer and a binary arithmetic coder to compress the quantized coefficients losslessly, see the code[1]. The difference between the estimated rate and the exact rate via the lossless coding is always smaller than 0.04 bbp. Figure 5 shows the rate-distortion curves averaged over the 24 luminance images. The JPEG2000 curve is obtained using ImageMagick. The H.265 [22] curve is computed via the version HM-16.15. There is hardly any difference between the 2^{nd} and the 3^{rd} case. This means that the explicit learning of the transform and the quantization step sizes is equivalent to learning the transform and the normalizations while the quantization step size is imposed. Note that, in the 2^{nd} case, the learning of $\{\delta_i\}_{i=1\ldots m}$ involves 128 parameters whereas, in the 3^{rd} case, that of $\{\boldsymbol{\varphi}_e, \boldsymbol{\varphi}_d\}$ involves 33024 parameters. The 2^{nd} and the 3^{rd} case perform as well as the 1^{st} case. The minimization (4) and the training in [6] provide learned transforms which can be used with various quantization step sizes at test time. It is convenient not to train one autoencoder per compression rate as a single training takes 4 days on a NVIDIA GTX 1080. Finally, we see that the learned transforms yield better rate-distortion performances than JPEG2000. The quality of image reconstruction for the experiment in Figure 5 and another experiment on luminance images created from the BSDS300 [23] can be seen online[1].

5. CONCLUSION

Using a unique transform learned via autoencoders and various quantization step sizes at test time, it is possible to compress as well as when learning one transform per rate-distortion point at a given quantization step size. Moreover, the learned transformed outperform other image compression algorithms based on transforms.

6. REFERENCES

[1] Thomas Wiegand and Heiko Schwarz, "Source coding: part I of fundamentals of source and video coding," *Foundations and Trends in Signal Processing*, vol. 4 (1-2), pp. 1–222, January 2011.

[2] George Toderici, Sean M. O'Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar, "Variable rate image compression with recurrent neural networks," in *ICLR*, 2016.

[3] Karol Gregor, Frederic Besse, Danilo J. Rezende, Ivo Danihelka, and Daan Wierstra, "Towards conceptual compression," *arXiv preprint arXiv:1604.08772*, April 2016.

[4] Thierry Dumas, Aline Roumy, and Christine Guillemot, "Image compression with stochastic winner-take-all auto-encoder," in *ICASSP*, 2017.

[5] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár, "Lossy image compression with compressive autoencoders," in *ICLR*, 2017.

[6] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli, "End-to-end optimized image compression," in *ICLR*, 2017.

[7] Oren Rippel and Lubomir Bourdev, "Real-time adaptive image compression," in *ICML*, 2017.

[8] Michael W. Marcelli, Margaret A. Lepley, Ali Bilgin, Thomas J. Flohr, Troy T. Chinen, and James H. Kasner, "An overview of quantization in JPEG 2000," *Image Communication*, vol. 17 (1), pp. 73–84, January 2002.

[9] Thomas Wiegand and Heiko Schwarz, "Video coding : part II of fundamentals of source and video coding," *Foundations and Trends in Signal Processing*, vol. 10 (1-3), pp. 1–346, December 2016.

[10] Ruslan R. Salakhutdinov and Geoffrey E. Hinton, "Semantic hashing," in *SIGIR Workshop on Information Retrieval and Applications of Graphical Models*, 2007.

[11] L. Deng, M. Seltzer, D. Yu, A. Acero, A. Mohamed, and G. Hinton, "Binary coding of speech spectrograms using a deep auto-encoder," in *INTERSPEECH*, 2010.

[12] Alex Krizhevsky and Geoffrey E. Hinton, "Using very deep autoencoders for content-based image retrieval," in *ESANN*, 2011.

[13] Yoshua Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2 (1), pp. 1–127, 2009.

[14] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, October 1986.

[15] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli, "Density modeling of images using a generalized normalization transform," in *ICLR*, 2016.

[16] Vincent Dumoulin and Francesco Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, March 2016.

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li, "ImageNet: a large-scale hierarchical image database," in *CVPR*, 2009.

[18] Gary J. Sullivan, "Efficient scalar quantization of exponential and laplacian random variables," *IEEE Transactions on Information Theory*, vol. 42, pp. 1365–1374, September 1996.

[19] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, "Deep inside convolutional networks: visualizing image classification models and saliency maps," in *ICLR*, 2014.

[20] Anh Nguyen, Jason Yosinki, and Jeff Clune, "Deep neural networks are easily fooled: high confidence predictions for unrecognizable images," in *CVPR*, 2015.

[21] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, "Understanding neural networks through deep visualization," in *ICML*, 2015.

[22] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22 (12), pp. 1649–1667, December 2012.

[23] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. Int'l Conf. Computer Vision*, 2001.