Learning Convolutional Networks for Content-weighted Image Compression

Mu Li[1]   Wangmeng Zuo[2]   Shuhang Gu[1]   Debin Zhao[2]   David Zhang[1]

[1]Department of Computing, The Hong Kong Polytechnic University

[2]School of Computer Science and Technology, Harbin Institute of Technology

csmuli@comp.polyu.edu.hk, cswmzuo@gmail.com, shuhanggu@gmail.com, dbzhao@hit.edu.cn

csdzhang@comp.polyu.edu.hk

Abstract

Lossy image compression is generally formulated as a joint rate-distortion optimization problem to learn encoder, quantizer, and decoder. Due to the non-differentiable quantizer and discrete entropy estimation, it is very challenging to develop a convolutional network (CNN)-based image compression system. In this paper, motivated by that the local information content is spatially variant in an image, we suggest that: (i) the bit rate of the different parts of the image is adapted to local content, and (ii) the content-aware bit rate is allocated under the guidance of a content-weighted importance map. The sum of the importance map can thus serve as a continuous alternative of discrete entropy estimation to control compression rate. The binarizer is adopted to quantize the output of encoder and a proxy function is introduced for approximating binary operation in backward propagation to make it differentiable. The encoder, decoder, binarizer and importance map can be jointly optimized in an end-to-end manner. And a convolutional entropy encoder is further presented for lossless compression of importance map and binary codes. In low bit rate image compression, experiments show that our system significantly outperforms JPEG and JPEG 2000 by structural similarity (SSIM) index, and can produce the much better visual result with sharp edges, rich textures, and fewer artifacts.

1. Introduction

Image compression is a fundamental problem in computer vision and image processing. With the development and popularity of high-quality multimedia content, lossy image compression has been becoming more and more essential in saving transmission bandwidth and hardware storage. An image compression system usually includes three

---

components, *i.e.* encoder, quantizer, and decoder, to form the codec. The typical image encoding standards, *e.g.*, JPEG [27] and JPEG 2000 [21], generally rely on hand-crafted image transformation and separate optimization on codecs, and thus are suboptimal for image compression. Moreover, JPEG and JPEG 2000 perform poor for low rate image compression, and may introduce visible artifacts such as blurring, ringing, and blocking [27, 21].

Recently, deep convolutional networks (CNNs) have achieved unprecedented success in versatile vision tasks [12, 33, 16, 8, 31, 7, 13, 2]. As to image compression, CNN is also expected to be more powerful than JPEG and JPEG 2000 by considering the following reasons. First, for image encoding and decoding, flexible nonlinear analysis and synthesis transformations can be easily deployed by stacking multiple convolution layers. Second, it allows to jointly optimize the nonlinear encoder and decoder in an end-to-end manner. Several recent advances also validate the effectiveness of deep learning in image compression [25, 26, 3, 23]. However, there are still several issues to be addressed in CNN-based image compression. In general, lossy image compression can be formulated as a joint rate-distortion optimization to learn encoder, quantizer, and decoder. Even the encoder and decoder can be represented as CNNs and optimized via back-propagation, the learning with non-differentiable quantizer is still a challenging issue. Moreover, the whole compression system aims to jointly minimize both the compression rate and distortion, where entropy rate should also be estimated and minimized in learning. As a result of quantization, the entropy rate defined on discrete codes is also a discrete function and requires continuous approximation.

In this paper, we present a novel CNN-based image compression framework to address the issues raised by quantization and entropy rate estimation. The existing deep learning based compression models [25, 26, 3] allocate the same number of codes for each spatial position, and the discrete code used for decoder has the same length with the encoder output. That is, the length of the discrete code is
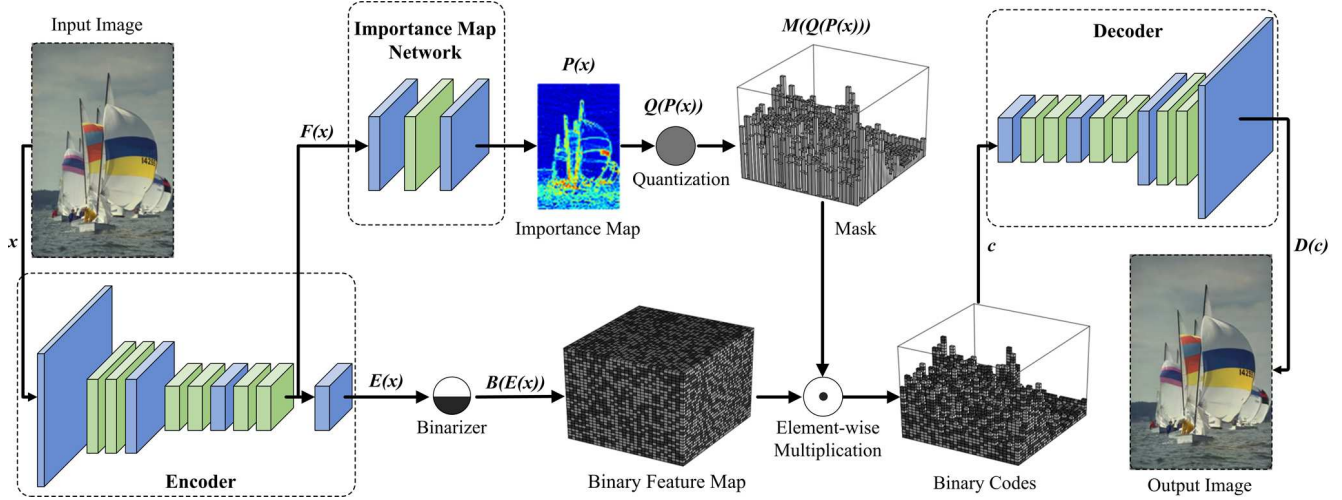
Figure 1. Illustration of the CNN architecture for content-weighted image compression.

spatially invariant. However, it is generally known that the local informative content is spatially variant in an image or video [32]. Thus, the bit rate should also be spatially variant to adapt to local informative content. To this end, we introduce a content-weighted importance map to guide the allocation of local bit rate. Given an input image $\mathbf{x}$, let $\mathbf{e} = E(\mathbf{x}) \in \mathbb{R}^{n \times h \times w}$ be the output of encoder network, which includes n feature maps with size of $h \times w$. $\mathbf{p} = P(\mathbf{x})$ denotes the $h \times w$ non-negative importance map. Specifically, when $\frac{l-1}{L} \leq \mathbf{p}_{i,j} < \frac{l}{L}$, we will only encode and save the first $\frac{nl}{L}$-th bits $\{\mathbf{e}_{1ij}, ..., \mathbf{e}_{\frac{nl}{L}ij}\}$ at spatial location (i, j). Here, L is the number of the importance level, and $\frac{n}{L}$ is the number of bits for each importance level. The other bits $\{\mathbf{e}_{(\frac{nl}{L}+1)ij}, ..., \mathbf{e}_{nij}\}$ at (i, j) are automatically set to 0 and need not to be saved into the codes. By this way, we can allocate more bits to the region with rich content, which is very helpful in preserving texture details with less sacrifice of bit rate. Moreover, the sum of the importance map $\sum_{i,j} \mathbf{p}_{i,j}$ naturally serves as a continuous estimation of compression rate, and can be directly adopted as a compression rate controller.

Benefited from importance map, we do not require to use any entropy rate estimation in training the encoder and decoder, and can adopt a simple binarizer for quantization. The binarizer sets those features with the sigmoid outputs which are higher than 0.5 to 1 and the others to 0. Inspired by the binarized CNN [34, 18, 4], we introduce a proxy function to approximate the binary operation in backward propagation. As shown in Figure 1, our compression framework consists of four major components: convolutional encoder, importance map network, binarizer, and convolutional decoder. With the introduction of continuous importance map and proxy function, all the components can be jointly optimized in an end-to-end manner.

Note that we do not include any entropy rate estimate in the training of the compression system. And the local spatial context of the codes is not utilized. Therefore, we design a convolutional entropy coder to predict the current code from its context, and apply it to the context-adaptive binary arithmetic coding (CABAC) framework [14] to further compress the binary codes and importance map.

Our whole framework is trained on a subset of the ImageNet database [5] and tested on the Kodak dataset. In low bit rate image compression, our system achieves much better rate-distortion performance than JPEG and JPEG 2000 in terms of both SSIM metric and visual quality. More remarkably, the compressed images by our system are visually more pleasing with sharp edges, rich textures, and fewer artifacts. Compared with other CNN-based systems [3], ours performs favorably in retaining texture details while suppressing visual artifacts.

To sum up, the main contribution of this paper is to introduce the content-weighted importance map and binary quantization into the image compression system. The importance map not only can be used to substitute entropy rate estimate in joint rate-distortion optimization, but also can be adopted to guide the local bit rate allocation. With binary quantization and the proxy function, our compression system can be end-to-end trained, and obtain notable improvement on visual quality over JPEG and JPEG 2000.

2. Related Work

For the existing image standards, *e.g.*, JPEG and JPEG 2000, the codecs are separately optimized. In the encoding stage, they first perform a linear image transformation. Quantization and lossless entropy coding are then utilized to minimize the compression rate. For example, JPEG [27] applies discrete cosine transform (DCT) on 8×8 image

patches, quantizes the frequency components and compresses the quantized codes with the Huffman encoding. JPEG 2000 [21] uses a multi-scale orthogonal wavelet decomposition to transform an image, and encodes the quantized codes with the Embedded Block Coding with Optimal Truncation. In the decoding stage, decoding algorithm and inverse transformation are designed to minimize distortion. However, the traditional image compression methods often suffer from compression artifacts especially in low compression rate. Several traditional methods [20] and deep CNN models [6] have been proposed to tackle this issue. Jiang *et al*. [10] further present a ComCNN to pre-process the image before encoding it with an existing codec (*e.g*., JPEG, JPEG2000 and BPG), together with a RecCNN for post-processing the decoding results. Instead of compression artifact removal, we propose a deep full convolutional model for image compression which can greatly eliminate the compression artifacts especially in low bit rate.

Recently, deep learning based image compression models have been investigated. For lossless image compression, deep learning models have achieved state-of-the-art performance [22, 15, 19]. For the lossy image compression, Toderici *et al*. [25] present a recurrent neural network (RNN) to compress 32×32 images. Toderici *et al*. [26] further introduce a set of full-resolution compression methods for progressive encoding and decoding of images. The most related works are those of [3, 23, 1] based on convolutional autoencoders. Ballé *et al*. [3] use generalized divisive normalization (GDN) for joint nonlinearity, and replace rounding quantization with additive uniform noise for continuous relaxation of distortion and entropy rate loss. Theis *et al*. [23] adopt a smooth approximation of the derivative of the rounding function, and upper-bound the discrete entropy rate loss for continuous relaxation. Agustsson *et al*. [1] introduce a way to process the quantization in a soft-to-hard way which can ease the training of compression networks. Rippel *et al*. [19] propose a deep auto-encoder with the featuring pyramidal analysis and generative adversarial training which can run in real-time. Our content-weighted image compression system is different with [3, 23, 1, 19] in rate loss, quantization, and continuous relaxation. Instead of rounding and entropy, we define our rate loss on importance map and adopt a simple binarizer for quantization. Moreover, the code length after quantization is spatially invariant in [3, 23, 1, 19, 10]. By contrast, the local code length in our model is content-aware and is very useful in improving visual quality.

Another related topic of work is semantic perceptual image processing. Timofte *et al*. [24] exploit segmentation information in image super-resolution, and saliency was introduced to the image compression system by Prakash *et al*. [17]. Compared with these methods, the importance map in our compression system is learnt from the image directly for compression task, and can be jointly optimized with the encoder and decoder during training.

Our work is also related to binarized neural network (BNN) [4], where both weights and activations are binarized to $+1$ or -1 to save memory storage and running time. Courbariaux *et al*. [4] adopt a straight-through estimator to compute the gradient of the binarizer. In our compression system, the encoder outputs are binarized to 1 or 0, and a similar proxy function is used in backward propagation.

3. Content-weighted Image Compression

As illustrated in Figure 1, our content-weighted image compression framework is composed of four components, *i.e.* convolutional encoder, binarizer, importance map network, and convolutional decoder. Given an input image $\mathbf{x}$, the convolutional encoder defines a nonlinear analysis transformation by stacking convolution layers, and outputs $E(\mathbf{x})$. The binarizer $B(E(\mathbf{x}))$ assigns 1 to the encoder outputs which are higher than 0.5, and 0 to the others. The importance map network takes the intermediate feature maps of the encoder as input, and yields the content-weighted importance map $P(\mathbf{x})$. The rounding function is adopted to quantize $P(\mathbf{x})$ and then a mask $M(P(\mathbf{x}))$ which has the same size of $B(E(\mathbf{x}))$ is generated with the guidance of the quantized $P(\mathbf{x})$. The binary code is then trimmed based on $M(P(\mathbf{x}))$. Finally, the decoder defines a nonlinear synthesis transformation to produce decoding result $\hat{\mathbf{x}}$. In the following, we first introduce the four components, and then present the formulation and learning of our model.

3.1. Components and Gradient Computation

3.1.1 Convolutional encoder and decoder

Both the encoder and decoder in our framework are fully convolutional networks and can be trained by back-propagation. The encoder network consists of three convolution layers and three residual blocks. Following [9], each residual block has two convolution layers. Analogous to [13] in single image super-resolution, we remove the batch normalization operations from the residual blocks, and empirically find that it is helpful in suppressing visual compression artifacts in smooth areas. The input image $\mathbf{x}$ is first convolved with 128 filters with size 8×8 and stride 4 and followed by one residual block. The feature maps are then convolved with 256 filters with size 4×4 and stride 2 and followed by two residual blocks to output the intermediate feature maps $F(\mathbf{x})$. Finally, $F(\mathbf{x})$ is convolved with m filters with size 1×1 to yield the encoder output $E(\mathbf{x})$. It should be noted that we set $n = 64$ for low compression rate models with less than 0.5 bpp, and $n = 128$ otherwise.

The network architecture of decoder $D(\mathbf{c})$ is symmetric to that of the encoder, where $\mathbf{c}$ is the code of an image $\mathbf{x}$. To upsample the feature maps, we adopt the depth to space op-

eration mentioned in [26]. Please refer to our project web-page[1] for more details on the network architecture of the encoder and decoder.

3.1.2 Binarizer

Since sigmoid nonlinearity is adopted in the last convolution layer of the encoder, the encoder output $\mathbf{e} = E(\mathbf{x})$ should be in the range of $[0, 1]$. e_{ijk} denotes an element in $\mathbf{e}$. The binarizer is defined as

$$B(e_{ijk}) = \begin{cases} 1, & \text{if } e_{ijk} > 0.5, \\ 0, & \text{if } e_{ijk} \leq 0.5. \end{cases} \quad (1)$$

However, the gradient of the binarizer function $B(e_{ijk})$ is zero almost everywhere except that it is infinite when $e_{ijk} = 0.5$. In the back-propagation algorithm, the gradient is computed layer by layer with the chain rule. Thus, such setting makes any layers before the binarizer (*i.e.*, the whole encoder) never be updated during training.

Fortunately, some recent works on binarized neural networks (BNN) [34, 18, 4] have studied the issue of propagating gradient through binarization. Based on the straight-through estimator on the gradient [4], we introduce a proxy function $\tilde{B}(e_{ijk})$ to approximate $B(e_{ijk})$. Here, $B(e_{ijk})$ is still used in forward propagation calculation, while $\tilde{B}(e_{ijk})$ is used in back-propagation. Inspired by BNN, we adopt a piecewise linear function $\tilde{B}(e_{ijk})$ as the proxy of $B(e_{ijk})$,

$$\tilde{B}(e_{ijk}) = \begin{cases} 1, & \text{if } e_{ijk} > 1, \\ e_{ijk}, & \text{if } 1 \leq e_{ijk} \leq 0, \\ 0, & \text{if } e_{ijk} < 0. \end{cases} \quad (2)$$

Then, the gradient of $\tilde{B}(e_{ijk})$ can be easily obtained by,

$$\tilde{B}'(e_{ijk}) = \begin{cases} 1, & \text{if } 1 \leq e_{ijk} \leq 0, \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

3.1.3 Importance map

In [3, 23], the code length after quantization is spatially invariant, and entropy coding is then used to further compression the code. Actually, the difficulty in compressing different parts of an image should be different. The smooth regions in an image is easier to be compressed than those with salient objects or rich textures. Thus, fewer bits should be allocated to the smooth regions while more bits should be allocated to the regions with complex structures and details. For example, given an image with an eagle flying in the blue sky in Figure 2, it is reasonable to allocate more bits to the eagle and fewer bits to blue sky. Moreover, when the whole code length for an image is limited, such allocation scheme can also be used for rate control.
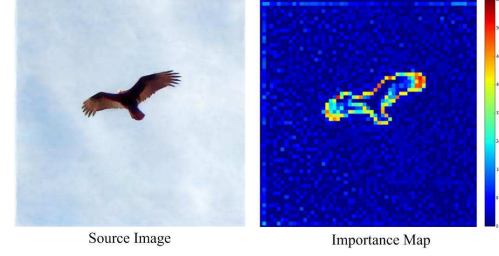
Source Image   Importance Map

Figure 2. Illustration of importance map. The regions with sharp edges or rich textures generally have higher values and should be allocated more bits.

We introduce a content-weighted importance map for bit allocation and compression rate control. It is a feature map with only one channel, and its size should be the same with that of the encoder output. The value of importance map is in the range of $(0, 1)$. An importance map network is deployed to learn the importance map from an input image $\mathbf{x}$. It takes the intermediate feature maps $F(\mathbf{x})$ from the last residual block of the encoder as input, and uses a network of three convolution layers to produce the importance map $\mathbf{p} = P(\mathbf{x})$.

Denote by $h \times w$ the size of the importance map $\mathbf{p}$, and n the number of feature maps of the encoder output. In order to guide the bit allocation, we should first quantize each element in $\mathbf{p}$ to an integer no more than n, and then generate an importance mask $\mathbf{m}$ with the size of $n \times h \times w$. Given an element p_{ij} in $\mathbf{p}$, the quantizer to importance map is defined as,

$$Q(p_{ij}) = l - 1, \text{if } \frac{l-1}{L} \leq p_{ij} < \frac{l}{L}, l = 1, \ldots, L. \quad (4)$$

where L is the importance levels and $(n \bmod L) = 0$. Each important level is corresponding to $\frac{n}{L}$ bits. As mentioned above, $p_{ij} \in (0, 1)$. Thus, $Q(p_{ij})$ has only L types of different quantity values i.e., $0, \ldots, L - 1$. It should be noted that, $Q(p_{ij}) = 0$ indicates that zero bit will be allocated to this location, and all its information can be reconstructed based on its context in the decoding stage. In this way, the importance map can not only be treated as an alternative of entropy rate estimation but also naturally take the context into account.

With $Q(\mathbf{p})$, the importance mask $\mathbf{m} = M(\mathbf{p})$ can then be obtained by,

$$\mathbf{m}_{kij} = \begin{cases} 1, & \text{if } k \leq \frac{n}{L} Q(p_{ij}), \\ 0, & \text{else.} \end{cases} \quad (5)$$

The final coding result of the image $\mathbf{x}$ can then be represented as $\mathbf{c} = M(\mathbf{p}) \circ B(\mathbf{e})$, where $\circ$ denotes the element-wise multiplication operation. Note that the quantized importance map $Q(\mathbf{p})$ should also be considered in the code.

Thus all the bits with $\mathbf{m}_{kij} = 0$ can be safely excluded from $B(\mathbf{e})$. Therefore, instead of n, only $\frac{n}{L}Q(p_{ij})$ bits are needed for each location (i, j). Besides, in video coding, just noticeable distortion (JND) models [32] have also been suggested for spatially variant bit allocation and rate control. Different from [32], the importance map is learned from training data via joint rate-distortion optimization.

Finally, in back-propagation, the gradient $\mathbf{m}$ with respect to p_{ij} should be computed. Unfortunately, due to the quantization operation and mask function, the gradient is zero almost everywhere. To address this issue, we rewrite the importance map m as a function of p,

$$\mathbf{m}_{kij} = \begin{cases} 1, & \text{if } \lceil \frac{kL}{n} \rceil < Lp_{ij}, \\ 0, & \text{else} \end{cases} \tag{6}$$

where $\lceil . \rceil$ is the ceiling function. Analogous to binarizer, we also adopt a straight-through estimator of the gradient,

$$\frac{\partial \mathbf{m}_{kij}}{\partial p_{ij}} = \begin{cases} L, & \text{if } Lp_{ij} - 1 \le \lceil \frac{kL}{n} \rceil < Lp_{ij} + 1, \\ 0, & \text{else.} \end{cases} \tag{7}$$

3.2. Model formulation and learning

3.2.1 Model formulation

In general, the proposed content-weighted image compression system can be formulated as a rate-distortion optimization problem. Our objective is to minimize the combination of the distortion loss and rate loss. A tradeoff parameter γ is introduced for balancing compression rate and distortion. Let $\mathcal{X}$ be a set of training data, and $\mathbf{x} \in \mathcal{X}$ be an image from the set. Therefore, the objective function our model is defined as

$$\mathcal{L} = \sum_{\mathbf{x} \in \mathcal{X}} \{ \mathcal{L}_D(\mathbf{c}, \mathbf{x}) + \gamma \mathcal{L}_R(\mathbf{x}) \} \tag{8}$$

where $\mathbf{c}$ is the code of the input image $\mathbf{x}$. $\mathcal{L}_D(\mathbf{c}, \mathbf{x})$ denotes the distortion loss and $\mathcal{L}_R(\mathbf{x})$ denotes the rate loss, which will be further explained as follows.

Distortion loss. Distortion loss is used to evaluate the distortion between the original image and the decoding result. Although better results may be obtained by assessing the distortion in the perceptual space, we simply use the squared ℓ_2 error to define the distortion loss,

$$\mathcal{L}_D(\mathbf{c}, \mathbf{x}) = \|D(\mathbf{c}) - \mathbf{x}\|_2^2. \tag{9}$$

Rate loss. Instead of entropy rate, we define the rate loss directly on the continuous approximation of the code length. Suppose the size of encoder output $E(\mathbf{x})$ is $n \times h \times w$. The code by our model includes two parts: (i) the quantized importance map $Q(\mathbf{p})$ with the fixed size $h \times w$; (ii) the

trimmed binary code with the size $\frac{n}{L} \sum_{i,j} Q(p_{ij})$. Note that the size of $Q(\mathbf{p})$ is constant to the encoder and importance map network. Thus $\frac{n}{L} \sum_{i,j} Q(p_{ij})$ can be used as rate loss.

Due to the effect of quantization $Q(p_{ij})$, the function $\frac{n}{L} \sum_{i,j} Q(p_{ij})$ cannot be optimized by back-propagation. Thus, we relax $Q(p_{ij})$ to its continuous form, and use the sum of the importance map $\mathbf{p} = P(\mathbf{x})$ as rate loss,

$$\mathcal{L}_R^0(\mathbf{x}) = \sum_{i,j} (P(\mathbf{x}))_{ij}. \tag{10}$$

For better rate control, we can select a threshold r, and penalize the rate loss in Eqn. (10) only when it is higher than r. Then we define the rate loss in our model as,

$$\mathcal{L}_R(\mathbf{x}) = \begin{cases} \sum_{i,j} (P(\mathbf{x}))_{ij} - r, & \text{if } \sum_{i,j} (P(\mathbf{x}))_{ij} > r \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

The threshold r can be set based on the code length for a given compression rate. By this way, our rate loss will penalize the code length higher than r, and makes the learned compression system achieve the comparable compression rate around the given one.

3.2.2 Learning

Benefited from the relaxed rate loss and the straight-through estimator of the gradient, the whole compression system can be trained in an end-to-end manner with an ADAM solver [11]. We initialize the model with the parameters pre-trained on the the training set $\mathcal{X}$ without the importance map. The model is further trained with the learning rate of $1e^{-4}$, $1e^{-5}$ and $1e^{-6}$. In each learning rate, the model is trained until the objective function does not decrease.

4. Convolutional entropy encoder

Due to no entropy constraint is included, the entropy of the code generated by the compression system in Sec. 3 is not maximal. This provides some leeway to further compress the code with lossless entropy coding. Generally, there are two kinds of entropy compression methods, *i.e.* Huffman tree and arithmetic coding [30]. Among them, arithmetic coding can exhibit better compression rate with a well-defined context, and is adopted in this work.

4.1. Encoding binary code

The binary arithmetic coding is applied according to the CABAC [14] framework. Note that CABAC is originally proposed for video compression. Let $\mathbf{c}$ be the code of n binary bitmaps, and $\mathbf{m}$ be the corresponding importance mask. To encode $\mathbf{c}$, we modify the coding schedule, redefine the context, and use CNN for probability prediction. As to coding schedule, we simply code each binary bit map
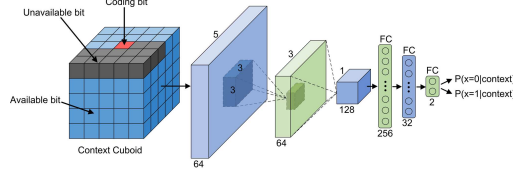
Figure 3. The CNN for convolutional entropy encoder. The red block represents the bit to predict; dark blocks mean unavailable bits; blue blocks represent available bits.

from left to right and row by row, and skip those bits with the corresponding important mask value of 0.

Context modeling. Denote by c_{kij} a binary bit of the code **c**. We define the context of c_{kij} as $CNTX(c_{kij})$ by considering the binary bits both from its neighbourhood and from the neighboring binary code maps. Specifically, $CNTX(c_{kij})$ is a $5 \times 5 \times 4$ cuboid. We further divide the bits in $CNTX(c_{kij})$ into two groups: the available and unavailable ones. The available ones represent those can be used to predict c_{kij}. While the unavailable ones include: (i) the bit to be predicted c_{kij}, (ii) the bits with the importance map value 0, (iii) the bits out of boundary and (iv) the bits currently not coded due to the coding order. Here we redefine $CNTX(c_{kij})$ by: (1) assigning 0 to the unavailable bits, (2) assigning 1 to the available bits with value 0, and (3) assigning 2 to the available bits with value 1.

Probability prediction. One usual method for probability prediction is to build and maintain a frequency table. As to our task, the size of the cuboid is too large to build the frequency table. Instead, we introduce a CNN model for probability prediction. As shown in Figure 3, the convolutional entropy encoder $En(CNTX(c_{kij}))$ takes the cuboid as input, and outputs the probability that the bit c_{kij} is 1. Thus, the loss for learning the entropy encoder can be written as,

$$\mathcal{L}_E = \sum_{i,j,k} m_{kij} \{ c_{kij} \log_2(En(CNTX(c_{kij}))) $$
$$+ (1 - c_{kij}) \log_2(1 - En(CNTX(c_{kij}))) \}. \quad (12)$$

where **m** is the importance mask. The convolutional entropy encoder is trained using the ADAM solver on the contexts of binary codes extracted from the binary feature maps generated by the trained encoder. The learning rate decreases from $1e^{-4}$ to $1e^{-6}$ as we do in Sec. 3.

4.2. Encoding quantized importance map

We also extend the convolutional entropy encoder to the quantized importance map. To utilize binary arithmetic coding, a number of binary code maps are adopted to represent the quantized importance map. The convolutional entropy encoder is then trained to compress the binary code maps.
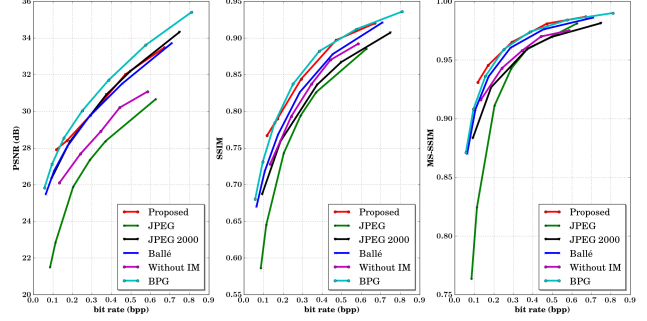

Figure 4. Comparison of the rate-distortion curves by different methods: (a) PSNR, (b) SSIM, and (c) MS-SSIM. "Without IM" represents the proposed method without importance map.

5. Experiments

Our content-weighted image compression models are trained on a subset of ImageNet [5] with about $10,000$ high quality images. We crop these images into 128×128 patches and take use of these patches to train the network. After training, we test our model on the Kodak PhotoCD image dataset with the metrics for lossy image compression. The compression rate of our model is evaluated by the metric bits per pixel (bpp), which is calculated as the total amount of bits used to code the image divided by the number of pixels. The image distortion is evaluated with Multi-Scale Structure Similarity (MS-SSIM), Peak Signal-to-Noise Ratio (PSNR), and the structural similarity (SSIM) index. For the time complexity, it takes about 0.48 second to compress a image in Kodak dataset.

In the following, we first introduce the parameter setting of our compression system. Then both quantitative metrics and visual quality evaluation are provided. Finally, we further analyze the effect of importance map and convolutional entropy encoder on the compression system.

5.1. Parameter setting

In our experiments, we set the number of binary feature maps n according to the compression rate, *i.e.* 64, when the compression rate is less than 0.5 bpp and 128 otherwise. Then, the number of importance level is chosen based on m. For $n = 64$ and $n = 128$, we set the number of importance level L to be 16 and 32, respectively. Moreover, different values of the tradeoff parameter γ in the range $[0.0001, 0.2]$ are chosen to get different compression rates. For the choice of the threshold value r, we just set it as $r_0 hw$ for $n = 64$ and $0.5 r_0 hw$ for $n = 128$. r_0 is the wanted compression rate represented with bit per pixel (bpp).

5.2. Quantitative evaluation

For quantitative evaluation, we compare our model with JPEG [27], JPEG 2000 [21], BPG and the CNN-based method by Ballé *et al*. [3]. Among the different variants of JPEG, the optimized JPEG with 4:2:0 chroma sub-
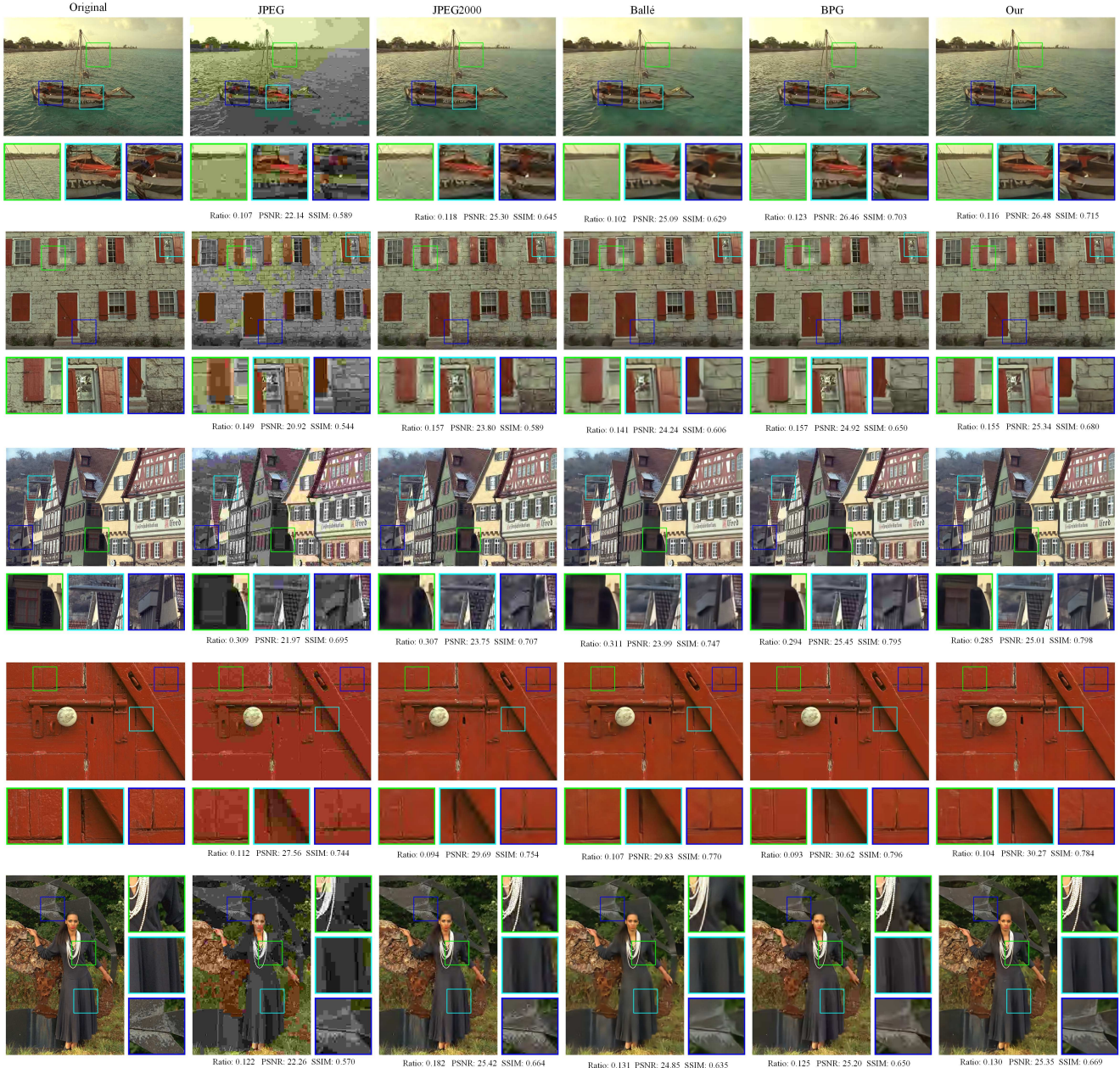
Figure 5. Images produced by different compression systems at different compression rates. From the left to right: groundtruth, JPEG, JPEG 2000, Ballé [3], BPG and ours. Our model achieves the best visual quality at each rate, demonstrating the superiority of our model in preserving both sharp edges and detailed textures. (Best viewed on screen in color)

sampling is adopted. For a fair comparison, all the results by Ballé [3], JPEG, and JPEG2000 on the Kodak dataset are downloaded from http://www.cns.nyu.edu/~lcv/iclr2017/.

Using MS-SSIM [29], SSIM [28] and PSNR as performance metrics, Figure 4 gives the rate-distortion curves of these five methods. In terms of PSNR, BPG has the best performance. And the results by JPEG 2000, Ballé [3] and ours are very similar, but are much higher than that by JPEG. In terms of SSIM and MS-SSIM, our system has

similar performance with BPG and outperforms all the other three competing methods, including JPEG, JPEG 2000, and Ballé [3]. Due to SSIM and MS-SSIM is more consistent with human visual perception than PSNR, these results indicate that our system performs favorably in terms of visual quality.

5.3. Visual quality evaluation

We further compare the visual quality of the results by JPEG, JPEG 2000, Ballé [3], BPG and our system in low
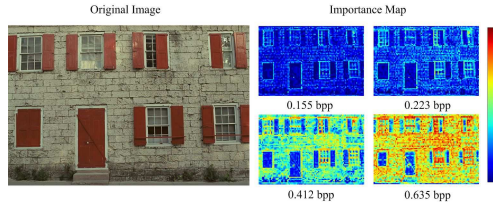
Figure 6. The important maps obtained at different compression rates. The right color bar shows the palette on the number of bits.

Figure 7. Performance of convolutional entropy encoder: (a) for encoding binary codes and importance map, and (b) by comparing with tradition CABAC.

compression rate setting. Figure 5 shows the original images and the results produced by the five compression systems. Visual artifacts, *e.g.*, blurring, ringing, and blocking, usually are inevitable in the compressed images by traditional image compression standards such as JPEG and JPEG 2000, while blurring and ringing effect can still be observed from the results by BPG. And these artifacts can also be perceived in the second and third columns of Figure 5. Even Ballé [3] is effective in suppressing these visual artifacts. In Figure 5, from the results produced by Ballé [3], we can observe the blurring artifacts in row 1, 2, 3, and 5, the color distortion in row 2 and 5, and the ringing artifacts in row 2. By contrast, the results produced by our system exhibit much less noticeable artifacts and are visually much more pleasing.

From Figure 5, Ballé [3] usually produces the results by blurring the strong edges or over-smoothing the small-scale textures. Specifically, in row 5 most details of the necklace have been removed by Ballé [3]. One possible explanation may be that before entropy encoding it adopts a spatially invariant bit allocation scheme. Actually, it is natural to see that more bits should be allocated to the regions with strong edges or detailed textures while less to the smooth regions. By contrast, an importance map is introduced in our system to guide spatially variant bit allocation. Moreover, instead of handcrafted engineering, the importance map is end-to-end learned to minimize the rate-distortion loss. As a result, our model is very promising in keeping perceptual structures, such as sharp edges and detailed textures.

5.4. Experimental analyses on important map

To assess the role of importance map, we train a baseline model by removing the importance map network from our framework. Both entropy and importance map based rate loss are not included in the baseline model. Thus, the compression rate is controlled by modifying the number of binary feature maps. Figure 4 also provides the ratio-distortion curves of the baseline model. We can see that, the baseline model is inferior to JPEG 2000 and Ballé [3] in terms of MS-SSIM, PSNR, and SSIM, thereby validating the necessity of importance map for our model. Using the image in row 5 of Figure 5, the compressed images by our model with and without importance map are also shown in our project webpage. Obviously more detailed textures and better visual quality can be obtained by using the impor-
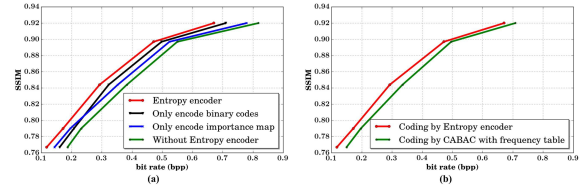
tance map.

Figure 6 shows the importance map obtained at different compression rates. We can see that, when the compression rate is low, due to the overall bit length is very limited, the importance map only allocates more bits to salient edges. With the increasing of compression rate, more bits will be allocated to weak edges and mid-scale textures. Finally, when the compression rate is high, small-scale textures will also be allocated with more bits. Thus, the importance map learned in our system is consistent with human visual perception, which may also explain the superiority of our model in preserving the structure, edges and textures.

5.5. Entropy encoder evaluation

The model in Sec. 3 does not consider entropy of the codes, allowing us to further compress the code with convolutional entropy encoder. Here, two groups of experiments are conducted. First, we compare four variants of our model: (i) the full model, (ii) the model without entropy coding, (iii) the model by only encoding binary codes, and (iv) the model by only encoding importance map. From Figure 7(a), both the binary codes and importance map can be further compressed by using our convolutional entropy encoder. And our full model can achieve the best performance among the four variants. Second, we compare our convolutional entropy encoder with the traditional content based arithmetic coding (CABAC) with small context (*i.e.* the 5 bits near the bit to encode). As shown in Figure 7(b), our entropy encoder can take larger context into account and performs better than CABAC. Besides, we also note that our method with either CABAC or convolutional encoder can outperform JPEG 2000 in terms of SSIM.

6. Conclusion

A CNN-based system is developed for content weighted image compression. With the importance map, we suggest a non-entropy based loss for rate control. Spatially variant bit allocation is also allowed to emphasize the salient regions. Using the straight-through binary estimator, our model can be trained in an end-to-end manner. A convolutional entropy encoder is introduced to further compress the binary codes and the importance map. Experiments clearly show the superiority of our model in retaining structures and removing artifacts, leading to favorably visual quality.

References

[1] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. Van Gool. Soft-to-hard vector quantization for end-to-end learned compression of images and neural networks. *arXiv preprint arXiv:1704.00648*, 2017. 3

[2] E. Agustsson and R. Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, volume 3, 2017. 1

[3] J. Ballé, V. Laparra, and E. P. Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016. 1, 2, 3, 4, 6, 7, 8

[4] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016. 2, 3, 4

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 2, 6

[6] C. Dong, Y. Deng, C. Change Loy, and X. Tang. Compression artifacts reduction by a deep convolutional network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 576–584, 2015. 3

[7] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, pages 184–199. Springer, 2014. 1

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014. 1

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 3

[10] F. Jiang, W. Tao, S. Liu, J. Ren, X. Guo, and D. Zhao. An end-to-end compression framework based on convolutional neural networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017. 3

[11] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 1

[13] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017. 1, 3

[14] D. Marpe, H. Schwarz, and T. Wiegand. Context-based adaptive binary arithmetic coding in the h. 264/avc video compression standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):620–636, 2003. 2, 5

[15] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016. 3

[16] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015. 1

[17] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. Storer. Semantic perceptual image compression using deep convolution networks. In *Data Compression Conference (DCC), 2017*, pages 250–259. IEEE, 2017. 3

[18] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016. 2, 4

[19] O. Rippel and L. Bourdev. Real-time adaptive image compression. In *International Conference on Machine Learning*, pages 2922–2930, 2017. 3

[20] R. Rothe, R. Timofte, and L. Van Gool. Efficient regression priors for reducing image compression artifacts. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 1543–1547. IEEE, 2015. 3

[21] A. Skodras, C. Christopoulos, and T. Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5):36–58, 2001. 1, 3, 6

[22] L. Theis and M. Bethge. Generative image modeling using spatial lstms. In *Advances in Neural Information Processing Systems*, pages 1927–1935, 2015. 3

[23] L. Theis, W. Shi, A. Cunningham, and F. Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017. 1, 3, 4

[24] R. Timofte, V. De Smet, and L. Van Gool. Semantic super-resolution: When and where is it useful? *Computer Vision and Image Understanding*, 142:1–12, 2016. 3

[25] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar. Variable rate image compression with recurrent neural networks. *arXiv preprint arXiv:1511.06085*, 2015. 1, 3

[26] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell. Full resolution image compression with recurrent neural networks. *arXiv preprint arXiv:1608.05148*, 2016. 1, 3, 4

[27] G. K. Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992. 1, 2, 6

[28] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 7

[29] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*, volume 2, pages 1398–1402. Ieee, 2003. 7

[30] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987. 5

[31] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *NIPS*, pages 341–349, 2012. 1

[32] X. Yang, W. Ling, Z. Lu, E. P. Ong, and S. Yao. Just notice-able distortion model and its applications in video coding. *Signal Processing: Image Communication*, 20(7):662–680, 2005. 2, 5

[33] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 2017. 1

[34] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016. 2, 4