YASO

Dong Wei
Northeastern University
China
wei295205@gamil.com

Mei Yang
Northeastern University
China
yyangm1104@163.com

Abstract

This paper presents a lossy image compression method based on neural network. Our architecture just consists of a recurrent neural network (RNN)-based encoder and decoder, a binarizer. This paper makes contributions in the following two aspects: 1) Preprocess the input images so that the encoder could work on images with arbitrary size; 2) Optimize the number of output channels of the binarizer, and our method ensures that the compressed image uses less than 0.15 bpp; 3) Our network is suitable for high-resolution images thanks to a sub-pixel architecture. As a consequence, we find that the optimized method generally exhibits better rate-distortion performance than standard JPEG compression methods on the Kodak dataset.

1. Introduction

In recent years, image compression attracts increasing interest in image processing and computer vision due to its potential applications in many vision systems. The aim of image compression is to reduce irrelevance and redundancy of an image in order to store or transmit the image at low bit rates. In traditional codecs such as JPEG [8], this is achieved via a pipeline which roughly breaks down into 3 modules: transformation, quantization and encoding. However, despite the careful construction and assembly of these pipelines, there still remains significant room for improvement of compression efficiency. For example, the transformation is fixed in place irrespective of the distribution of the inputs, and is not adapted to their statistics in any way. In addition, hard-coded approaches often compartmentalize the loss of information within the quantization step. As such, the transformation module is chosen to be bijective: however, this limits the ability to reduce redundancy prior to coding. Moreover, the encode-decode pipeline cannot be optimized for a particular metric beyond manual tweaking: even if we have the perfect metric for image quality assessment, traditional approaches cannot directly optimize their reconstructions for it.

In approaches based on machine learning, structure is automatically discovered, rather than manually engineered. one natural approach to implement the encoder-decoder image compression pipeline is to use an autoencoder to map the target through a bitrate bottleneck, and train the model to minimize a loss function penalizing it from its reconstruction. This requires carefully constructing a feature extractor and synthesizer for the encoder and decoder, selecting an appropriate objective, and possibly introducing a coding scheme to further compress the fixed-size representation to attain variable-length codes.Finally, existing compression algorithms may be far from optimal for new media formats. While the development of a new codec can take years, a more general compression framework based on neural networks may be able to adapt much quicker to these changing tasks and environments.

2. Related work

Deep learning has been used for lossy image compression and achieved competitive performance. Toderici et al.[6] proposed a general framework for variable-rate images compression and a novel architecture based on convolutional and deconvolutional LSTM recurrent networks. Further, Toderici et al.[7] proposed a neural network which is competitive across compression rates on images of arbitrary size. Theis et al.[5] proposed compressive autoencoders, which use a smooth approximation of the discrete of the rounding function and upper-bound the discrete entropy rate loss for continuous relaxation.

In this paper, the method we described builds on the work of Toderici et al.[7]. In fact, the implementation of Toderici et al.[7] can not compress arbitrary size images, but requires the image's height and width in pixels by a multiple of 32, which means that this method does not work for some images. What's more, it results in that each iteration adds 1/8 bit per pixel (bpp). It means that if we adopt iteration = 1, the bpp will be 0.125, but this value has a bit of a gap with 0.15 which is the target in comptition. These above problems will lead to competitiveness of the approach Toderici proposed.

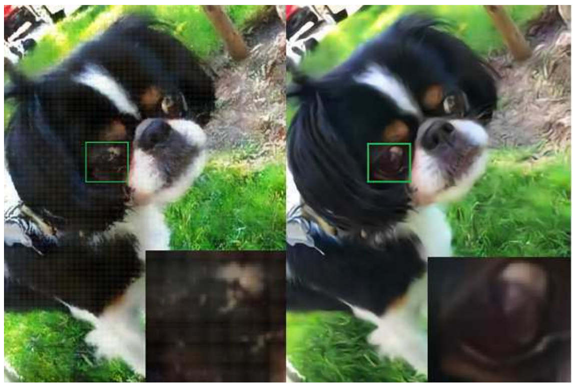

Figure 1. Left: the decoded image (0.125 bpp) by Residual GRU model of Toderici et al.[7] with L_1 loss, where we could see blocking artifacts, ring effects and blurring on the eyes, abrupt intensity changes on the image. Right: the decoded image by compression framework with L_2 loss at the same bit rate with left, where the compression artifacts vanished and generate more details.

3. Methods

This section describes our proposed framework for compression and the associated preprocessing methodlogy. Afterwards, this section presents the sub-pixel architecture we used in the Depth-to-Space unit.

Figure 2 shows the architecture used in our encoder and decoder networks. The compression networks are comprised of an encoding network, a binarizer and a decoding network, where decoding network and encoding network contain recurrent network components. The input images are encoded firstly, and then be transformed into binary codes that can be stored or transmitted to the decoder. The decoder network creates an estimate of the original input image based on the binary code. The difference from the network of Toderici et al.[7] is we only use single iteration in our paper.

3.1. Architecture

We can compactly represent our networks as follows:

$$c = B(E(x)), \quad \hat{x} = D(b), \quad e = x - \hat{x} \tag{1}$$

where E , B and D represent the encoder, binarizer, and decoder. c is the binary representation. $\hat{x}$ is the reconstruction of the original image x. e is the residual between x and the reconstruction $\hat{x}$. To ensure accurate bit-rate counts, the binarizer quantizes its input to be $c \in (-1, 1)^m$, and this will give us our nominal (pre-entropy coding) bit rate, where m is the number of bits produced after encoding using the approach reported in Toderici et al. [1]. And m is a linear function of input size. in the work of Toderici et al. [3], for image patches of 32×32, $m = 128$, while in our work $m = k$.

In Toderici et al.[7] , each $32 \times 32 \times 3$ input image is reduced to a binarized representation. This results in each iteration adds $1/8$ bit per pixel (bpp). The bpp given in the challenge is 0.15, so we must change the output channels of binarizer in Figure 2 to make sure that our approach can achieve pretty results.

Firstly, we take account of the multiple relationship between the bpp in Toderici et al.[7] and it in challenge at present. Changing 32 to 38, although 38 is not an integer power of 2, it still works. $k = 152$.

In addition, we modifie the loss function. We think that model trained with L_2 loss is much smoother than which trained with L_1 loss, especially in the case of low bits. In the method of Toderici et al.[7], there is a clear block effect in the pictures which reconstructed with the first iteration though the block is due to the patch-based encoder in most condition. However, when we choose L_2 loss function, the image will be much smooth as shown in Figure 1. So our total loss of the network is:

$$|e|^2 \tag{2}$$

Here we choose to use LSTM recurrent units [1], with the formulation proposed by Zaremba et al.[10]respectively at iteration t. Given the current input x_t, previous cell state c_{t-1}, and previous hidden state h_{t-1}, the new cell state c_t and the new hidden state h_t are computed as:

$$[f, i, o, j]^T = [\sigma, \sigma, \sigma, \tanh]^T((Wx_t + Uh_{t-1}) + b), \tag{3}$$
$$c_t = f \odot c_{t-1} + i \odot j, \tag{4}$$
$$h_t = o \odot \tanh(c_t), \tag{5}$$

where $\odot$ denotes element-wise multiplication, and b is the bias. The activation function σ is the sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$. The W and U are convolutional linear transformations.

3.2. Sub-pixel architecture

Figure 2 shows the spatial extent of the input-vector convolutional kernel along with the output depth. All convolutional kernels allow full mixing across depth. For example, the unit D-RNN#3 has 256 convolutional kernels that operate on the input vector, each with 3×3 spatial extent and full input-depth extent (128 in this example, since the depth of D-RNN#2 is reduced by a factor of four as it goes through the Depth-to-Space unit).Inspired by the core idea from Shi et al. [4], who demonstrated that super-resolution can be achieved much more efficiently by operating in the low resolution space, that is, by convolving images and then upsampling instead of upsampling first and then convolving an image. Convolutions of the Depth-to-Space unit are performed in a downsampled space to speed up computation, and upsampling is performed using sub-pixel convolutions (convolutions followed by reshaping/reshuffling of the coefficients).

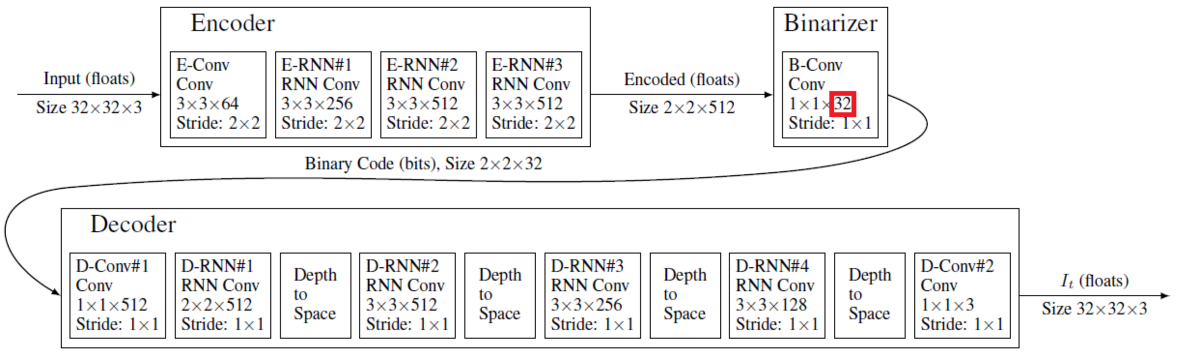
Encoder
Input (floats)
Size 32×32×3
E-Conv Conv 3×3×64 Stride: 2×2
E-RNN#1 RNN Conv 3×3×256 Stride: 2×2
E-RNN#2 RNN Conv 3×3×512 Stride: 2×2
E-RNN#3 RNN Conv 3×3×512 Stride: 2×2
Encoded (floats)
Size 2×2×512
Binarizer
B-Conv Conv 1×1×32 Stride: 1×1
Binary Code (bits), Size 2×2×32
Decoder
D-Conv#1 Conv 1×1×512 Stride: 1×1
D-RNN#1 RNN Conv 2×2×512 Stride: 1×1
Depth to Space
D-RNN#2 RNN Conv 3×3×512 Stride: 1×1
Depth to Space
D-RNN#3 RNN Conv 3×3×256 Stride: 1×1
Depth to Space
D-RNN#4 RNN Conv 3×3×128 Stride: 1×1
Depth to Space
D-Conv#2 Conv 1×1×3 Stride: 1×1
I_t (floats)
Size 32×32×3

Figure 2. A single iteration of our shared RNN architecture in Toderici et al.[7], the number of channels represented by the position of the red box is where we need to change. This is the most direct way to control bpp through the network structure.

Upsampling is achieved by convolution followed by a reorganization of the coefficients. This reorganization turns a tensor with many channels into a tensor of the same dimensionality but with fewer channels and larger spatial extent (for details, see Shi et al [4]).

3.3. Preprocessing

Due to the special coding network structure, this model can only compress the image of height and width in pixels by a multiple of 32. In order to overcome this difficulty, we tried two different image preprocessing methods. One is to resize the input image to an integer multiple of 32 directly, and then perform an encoding and decoding operation followed by an operation to resize the decoded image back to its original size; Another method is to pad the input image firstly, and then use the model to compress the image. The image output by the decoder is cropped so that the image remains in its original size. We think that the interpolation of resizing will cause the corresponding position of the pixel to change although comparing to the method which pads the edge it may not bring an increase in the number of bits. But when we use metrics such as Peak Signal to Noise Ratio (PSNR) to evaluate, it does not perform so good as we thought. On the contrary, when we perform zero-fill operation on image pixels, since the content of this part of redundant information is relatively simple, it is easy to compress. And although it has a slight impact on bpp, it will benefit when calculating the evaluation metrics.

4. Result

In this subsection, we introduce the training dataset, data augmentation, and other implementation details. Then we analyze the effectiveness of two kinds of image preprocessing methods. Finally, we collect these metrics on the widely used Kodak Photo CD dataset [Kodak]. The dataset consists of 24 PNG images (landscape/portrait) which are 768×512 and have never been compressed with a lossy algorithm.

4.1. Training

Training dataset and data augmentation. Our model is trained on 240000 images on the web with the following data augmentation strategies:

Color distort. Applying some photo-metric distortions similar to[2].

Random crop. Each minibatch uses 32×32 patches randomly sampled from these images.

Other implementation details. We trained two models. The preprocessing method used by"resized model" is to resize the input image to an integer multiple of 32 , and the "paded model" preprocess images by padding. Both of these models were trained using Adam[3] with a learning rates of $\{0.1; 0.3; 0.5; 0.8; 1\}$ and a batch size of 32, for a total of 150000 steps. The L_2 loss was normalized by the number of pixels in the patch. During training we used the unmodified L_2 error measure. The "our model" is the same as the "paded model" but change D-RNN#1 from 2×2 to 3×3. And the model used in the challenge submission is the "our model" just trained with a total of 28000 steps.

4.2. Experiments

In order to assess the performance of our models, we use a perceptual, full reference image metric for comparing original, uncompressed images to compressed, degraded ones. It is important to note that there is no consensus in the field for which metric best represents human perception so the best we can do is sampling from the available choices while acknowledging that each metric has its own strengths and weaknesses. We use Multi-Scale Structural Similarity (MS-SSIM)[9], a well-established metric for comparing lossy image compression algorithms. We apply MS-SSIM to each of the RGB channels independently and average the results, MS-SSIM giving a score between 0 and 1, higher

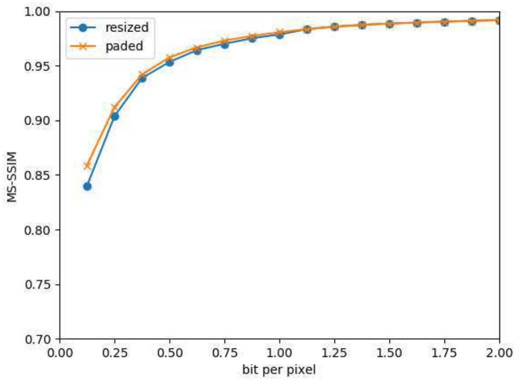

Figure 3. results of two image preprocessing methods

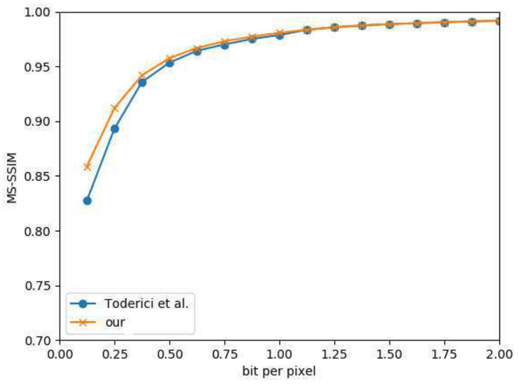

Figure 4. Comparison of different compression algorithms with MS-SSIM on the Kodak Photo CD image dataset. We note that the blue line refers to the results of Toderici et al.[7] achieved without entropy encoding.

values implying a closer match between the test and reference images.

As we analyzed in section3.2, the pre-processing method of padding images on the Kodak dataset achieves a higher MS-SSIM score than the method of resizing as shown in Figure 3 . In addition, we show the comparison between the quality of the decoded image of the model trained by our method and the model provided by Toderici et al.[7]in Figure 4 . Finally we compared with JPEG, as shown in Figure 5 . It can be seen that our framework significantly outperforms Toderici et al.[7] and JPEG on all test bit-rates of all test images in terms of MS-SSIM.

5. Discussion

In this paper, we presents a lossy image compression method based on neural network. Our method ensures that encoder could work on images of arbitrary size and the total

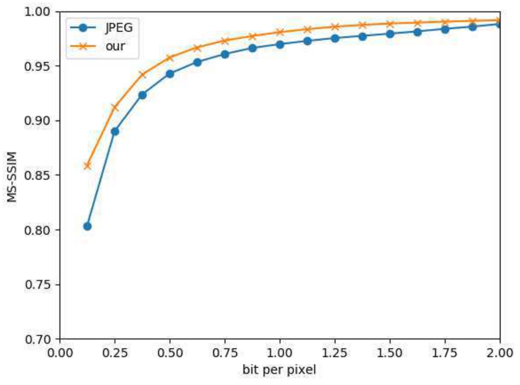

Figure 5. Rate-distortion curves of the Kodak Photo CD image dataset.

size of all compressed images don't exceed the given total maximum size. There is still much room to improvement in our work. In the future work, we would like to explore the optimization of compressive autoencoders to further improve compression performance.

References

[1] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[2] A. G. Howard. Some improvements on deep convolutional neural network based image classification. *Computer Science*, 2013.

[3] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *Computer Science*, 2014.

[4] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. pages 1874–1883, 2016.

[5] L. Theis, W. Shi, A. Cunningham, and F. Huszr. Lossy image compression with compressive autoencoders. 2017.

[6] 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. *Computer Science*, 2016.

[7] 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 *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5435–5443, 2017.

[8] G. K. Wallace. The jpeg still picture compression standard. *Communications of the Acm*, 38(1):xviii–xxxiv, 1991.

[9] 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*, pages 1398–1402 Vol.2, 2004.

[10] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *Eprint Arxiv*, 2014.