Compression artifact removal using multi-scale reshuffling convolutional network

Zhimin Tang
Department of Automation, Xiamen University
Xiamen 361005, China
tangzhimin@stu.xmu.edu.cn

Linkai Luo*
Department of Automation, Xiamen University
Xiamen 361005, China
*Corresponding author: luolk@xmu.edu.cn

Abstract

In this work, we aim to build a high efficient deep network to remove artifact of compressed image. The degeneracy and small receptive field problem might be caused by reducing the computational cost of convolutional network via general approaches, such as pooling features to low resolution space, reducing the width of network and using small size convolutional kernel. For the reasons, we propose a multi-scale reshuffling network to efficiently reduce the compression artifact of compressed images without degeneracy. We firstly present a reshuffling network which includes a downscaling and a upscaling reshuffling operation. The downscaling reshuffling periodically rearranges high resolution to low resolution space without any information loss. The upscaling reshuffling is the reverse transformation of downscaling reshuffling, which allows us reconstructing high resolution image from the low resolution features. A densely connected structure is applied to efficiently extract features without degeneracy. The low resolution representations is gradually recovered to the higher resolution spaces which leads to a multi-scale structure. Results show the effectiveness of the proposed method.

1. Degeneracy problem and computational cost of convolutional network

In image restoration scenario, including compression artifact removal, we need to predict the latent high quality image from a degraded input image. We find that general methods for computational cost reduction of convolutional neural network (CNN) might lead network to suffer from the degeneracy. Degeneracy of network might force it learning a lot of redundancy, which results in an inefficient network, to obtain an accurate prediction.

1.1. Degeneracy problem

Orhan [8] explained that the difficulty of training deep network is mainly due to the degeneracy problem. We ascribe degeneracy to the singularity of linear transformation, rectified linear unit (ReLU) and pooling layer, as these might cause too much input information to be lost and therefore decrease the performance of deep network. Zhang *et al*. [11] shown that most convolutional or pooling layers would cause information loss through an information theoretic view. Singularity would lead consistent deactivation of nodes and decrease the effective representation dimension of deep features. ReLU ignores the negative activations. Downscaling pooling layers directly reduce the spatial size of feature maps. The shallow layers should be wide enough to keep sufficient effective information, which leads to a redundant structure. He *et al*. [5] successfully trained very deep network using skip connections. Orhan *et al*. [9] explained that skip connections can eliminate singularities. However, skip connection can not prevent pooling layer from information loss and is not efficient enough in image restoration tasks.

1.2. Computational cost of convolutional layer

We assume that convolutional layer operates with stride 1 and pads the input to produce output feature maps with the same spatial size of input. The convolutional layer is parameterized by a convolution kernel $\boldsymbol{F}$ of shape $K \times L \times M \times N$, where K and L is the width and height of convolution kernel, M is the number of input channels and N is the number of output channels. A Convolutional layer takes as input $\boldsymbol{X}$ with shape of $I \times J \times M$ and outputs $\boldsymbol{A}$ with shape of $I \times J \times N$, where I and J is the width and height of feature maps. The output feature maps of convolutional layer are computed as:

$$\boldsymbol{A}_{i,j,n} = \sum_{k,l,m} \boldsymbol{X}_{i+k,j+l,m} \cdot \boldsymbol{F}_{k,l,m,n} \qquad (1)$$

Since the computational cost of a CNN is focused on

the convolution operation, the complexity of convolutional layer can be described in term of the number of multiply-add operations (mult-adds). The number of mult-adds in a convolutional layer is:

$$(I \cdot J) \cdot (K \cdot L) \cdot M \cdot N \tag{2}$$

2. Multi-scale reshuffling network

In this section, we will analyze the reason why the degeneracy and small receptive field problem would be caused when reducing the computational cost of convolutional network via general approaches. We then propose a multi-scale reshuffling network to efficiently reduce the compression artifact of compressed images without degeneracy.

2.1. General approaches for computational cost reduction and their drawbacks

In accordance with Eq. (2), the computational cost of a convolutional layer can be reduced in three aspects:

1. Reduce the spatial size, $I \times J$, of feature maps using spatial pooling.
2. Reduce the width, M and N, of network.
3. Reduce the spatial size, $K \times L$, of convolutional kernel size.

However, these approaches might cause some problems:

1. We can downscale the feature maps using pooling layer. However, the pooling layer is mainly for the invariance of features in object recognition tasks. In image restoration tasks, the layers before pooling should learn a lot of unnecessary redundancy to avoid degeneracy, since pooling layer directly reduce the resolution of features and some spatial information will lost.
2. Insufficient number of features will limit the representative performance of network.
3. If the convolutional kernel size reduced to 1×1, the complexity can be decreased by $K \times L$ times, but receptive field will reduced to be 1 and the convolutional layer will learn no context information. The small receptive field problem is caused.

2.2. Reduce spatial size of feature maps using reshuffling network

In order to efficiently extract low resolution (LR) features without degeneracy, we propose a reshuffling network 1 which includes a downscaling operation and a reshuffling upscaling operation.

As illustrated in Fig. 1(a), T features with shape of $I \times J$ is periodically rearranged to $T \cdot s^2$ features with shape of $I/s \times J/s$ by the downscaling reshuffling operation. In Fig. 1, T is equal to 1. The spatial height and width of input feature maps must be divisible by the reshuffling scale s in the downscaling operation. Mathematically, the downscaling reshuffling operation can be described as:

$$\mathcal{R}_{\downarrow}(\boldsymbol{X})_{i,j,t} = \boldsymbol{X}_{i \cdot s + \lfloor mod(t,s^2)/s \rfloor, j \cdot s + mod(mod(t,s^2),s), \lfloor t/s^2 \rfloor} \tag{3}$$

where i, j is the coordinates in high resolution (HR) space, t is the index of feature map. Therefore, the HR features is downsized to LR space with no information loss and allows us extracting features with low computational complexity.

The upscaling reshuffling operation, illustrated in Fig. 1(b), is the reverse transformation of downscaling reshuffling, which have been used into image super-resolution [10]. It is mathematically formulated as:

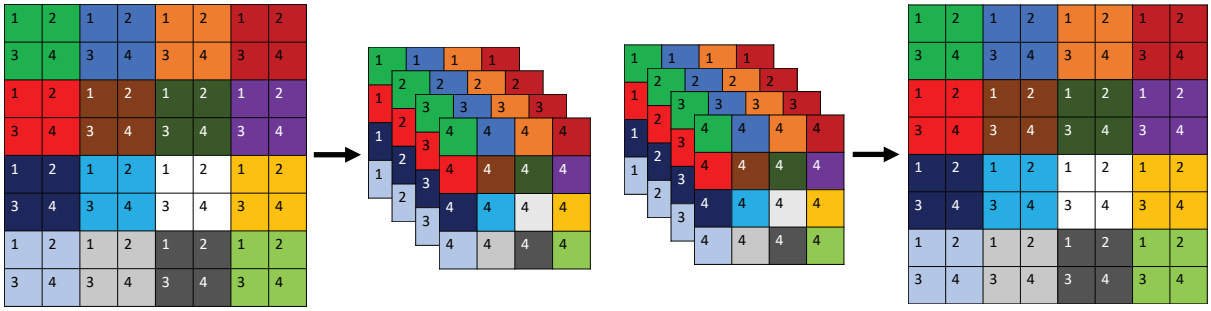
$$\mathcal{R}_{\uparrow}(\boldsymbol{X})_{i,j,t} = \boldsymbol{X}_{\lfloor i/s \rfloor, \lfloor j/s \rfloor, t \cdot s^2 + s \cdot mod(i,s) + mod(j,s)} \tag{4}$$

The number of input features must be divisible by s^2. The upscaling reshuffling operation can reconstructing HR image from the LR representations.

Since the downscaling operation keeps all the activations of the input layer, it would not encounter the trouble of degeneracy like downscaling pooling and the latent redundancy in shallower layers might be reduced. On the other hand, the receptive field of neurons will increased from $K \times L$ to $K \cdot s \times L \cdot s$ in the following layer benefit from extracting features in LR space.

2.3. Reduce the width of network and the convolutional kernel size

In order to compress the number of feature maps and shrink the size of convolution kernel. In the l^{th} block of network, we firstly convolve $\boldsymbol{A}^{(l-1)}$, the output of the $(l-1)^{th}$ block, of shape $I^{(l)} \times J^{(l)} \times M^{(l)}$ with kernel $\boldsymbol{F}^{(l)}$ of shape $1 \times 1 \times M^{(l)} \times k \cdot N^{(l)}$ and produce a $I^{(l)} \times J^{(l)} \times k \cdot N^{(l)}$ tensor $\boldsymbol{B}^{(l)}$. For the convenience of reconstructing HR image, $N^{(l)}$ should be a multiple of s^2. For a compressed representation, $k \cdot N^{(l)} \ll M^{(l)}$. Then, $\boldsymbol{B}^{(l)}$ is activated by exponential linear unit (ELU) [4] for non-linearity. Unlike ReLU, ELU have no dead gradient section and it is reversible. The negative activations is also activated by ELU. Although a downscaling and a 1×1 convolution have a receptive field of s^2, it is equal to a block-wise operation. For instance, a downscaling of scale 8 followed a 1×1 convolution have an equal structure of discrete cosine transform in JPEG. In order to learn the connection between spatial blocks, a 3×3 convolutional layer is followed and produces a $I^{(l)} \times J^{(l)} \times N^{(l)}$ tensor $\boldsymbol{C}^{(l)}$. Because of the extreme compactness of $\boldsymbol{C}^{(l)}$, the representative performance will be restricted and the degeneracy will be caused. Inspired by the densely connected networks [6], we therefore



(a) Downscaling reshuffling with scale 2

(b) Upscaling reshuffling with scale 2

Figure 1. Downscaling reshuffling and upscaling reshuffling with scale 2. The width and height of the high resolution features must be divisible by the reshuffling scale s and the number of low resolution features must be divisible by s^2

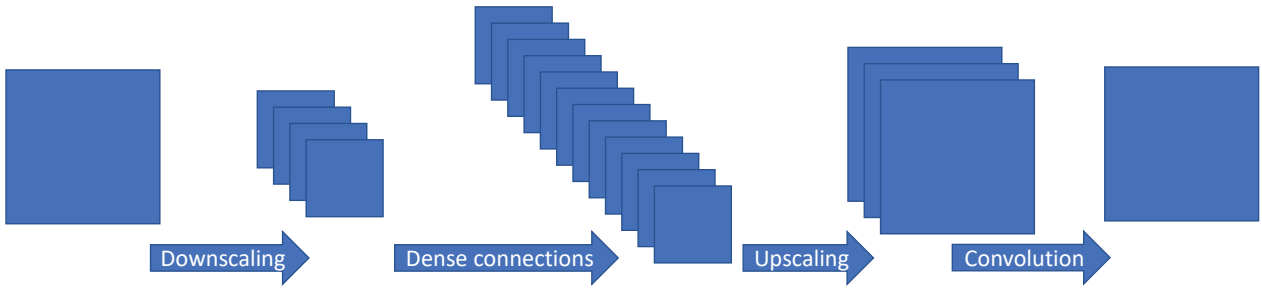


Figure 2. Reshuffling network.

Input	÷2	÷2	÷2	dense block	×2	dense block	×2	dense block	×2	3 × 3 convolution
Numble of features	12	48	192	1280	320	640	160	320	80	3

Table 1. Architecture of our multi-scale network. Symbol '÷2' indicates a downscaling reshuffling with scale 2, symbol '×2' indicates a upscaling reshuffling with scale 2.

PSNR (dB) / ours	PSNR (dB) / BPG	Decoder size (Bytes)	Decoding time (ms)	Bit rate (bpp)
30.2363	29.7928	4,984,420	30,311,917	0.14978

Table 2. Evaluations on the test data of CLIC 2018

concatenate the input $A^{(l-1)}$ to output of this block

$$\boldsymbol{A}^{(l)} = [\boldsymbol{A}^{(l-1)}, \boldsymbol{C}^{(l)}] \tag{5}$$

Fig. 2 shows an example of the reshuffling network.

Because of the extremely compressed representation and small size convolutional kernel, this architecture has a much higher computational efficiency. Besides, it is worth noting that there is no downscaling and upscaling operation if the reshuffling scale equal to 1.

2.4. Multi-scale feature extraction

In order to learn different scales of features, we propose a multi-scale reshuffling architecture. We decompose a scale of 2^t reshuffling network into t reshuffling block of scale 2. The LR features is gradually recoverd to HR features. We concatenate lager scale features to the smaller scale features to make full use of all features. Table 1 shows the layer out of the multi-scale reshuffling network.

2.5. Learning

Given a training dataset $\{\mathbf{X}_i^C, \mathbf{X}_i^G\}_{i=1}^N$, where N is the amount of training patch pairs and $\{\mathbf{X}_i^C, \mathbf{X}_i^G\}$ are the i^{th} compressed and ground truth patch pair. The objective function of our method can be expressed as:

$$\mathcal{L}(\Theta) = \frac{1}{N}\sum_{i=1}^{N} \parallel \mathbf{X}_i^C - \mathbf{X}_i^G \parallel_1 \tag{6}$$

Both the entire network and objective function are differentiable, so we can simply use the back-propagation algorithm to train our models. Adam [7] is chosen as the optimizer to update parameters.

3. Experiments

In addition to the training data provided by the organizers of CLIC 2018, the training set of NTIRE 2017 [3] is used.

Up-down flipping and left-right flipping are used for data augmentation. To generate compressed and ground truth image pair, we use the Better Portable Graphics [1] algorithm to compress the ground truth image to a compressed file and then decompress it to the corresponding compressed image. We sample images in YCbCr color space by 4:2:0 to reduce the number of values to be compressed. The jctvc encoder with depth of 10 bits is chosen to encode files. For a variable bit rate model, we compress images using different quantization parameters of 37, 38, 39 and 40. All the models are trained on a GTX 1080Ti GPU with machine learning framework Tensorflow [2]. Table 2 shows the evaluations on the test data, we compute a single mean squared error value by averaging across all RGB channels of all pixels of the whole dataset, and from that calculate a peak signal-to-noise ratio (PSNR) value. Besides the performance of our method, the PSNR of BPG is also evaluated. The decoding time of our method is evaluated on the challenge server in which a single CPU and 8 G RAM is provided.

References

[1] Bpg image format. https://bellard.org/bpg/.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] 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*, July 2017.

[4] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[6] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017.

[7] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[8] A. E. Orhan. Why is it hard to train deep neural networks? degeneracy, not vanishing gradients, is the key. https://severelytheoretical.wordpress.com/2018/01/01/why-is-it-hard-to-train-deep-neural-networks-degeneracy-not-vanishing-gradients-is-the-key/, 2018.

[9] A. E. Orhan and X. Pitkow. Skip connections eliminate singularities. In *ICLR*, 2018.

[10] 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. In *CVPR*, 2016.

[11] J. Zhang, T. Liu, and D. Tao. An information-theoretic view for deep learning. *arXiv preprint arXiv:1804.09060*, 2018.