An Implementation of Picture Compression with A CNN-based Auto-encoder

Ming Li
VimicroAI
Building 16,Hengqin Financial District,
Zhuhai city, Guangdong Provice, P.R.C
li.ming@zxelec.com

Jianhua Hu,Changsheng Xia,Yundong Zhang
VimicroAI
Building 16,Hengqin Financial District,
Zhuhai city, Guangdong Provice, P.R.C

Abstract

We mainly use the importance-map CNN method introduced by Mu.Li[1] to compress the CLIC2018 validation and test pictures. The framework is an autoencoder, with the bottleneck containing a 4-bit importance map and a 1/8 scale-down feature maps(FMs) of 64-channel and 1-bit contents. We re-implemented this model in the Tensorflow/python enviroment. Different from the original work, we modify the network a little to ge better performance and creatively replace the entropy-coding scheme with a much simpler reorder and run-length coding method. We also share some techniques and experiences for model training and fine tuning the encoder for the CLIC2018 test pictures. Method of controlling the final bit rate is also mentioned.

1. Introduction

The recent great successes achieved by deep natural net work are mainly in fields of image classification, semantic segmentation, object detection. But it hasn't been deeply researched for the image/video compression application. Mu.Li[1] introduced the importance-map method, using stacked CNN layers with strides to encode the input image into the bottleneck. The bottleneck contains a 4-bit importance map indicating the importance level of the corresponding reception field. The areas which are important are allocated with more coding bits. The bottleneck also including a 64 channels, 1-bit feature maps(as FMs). FMs contains the main compressed information of the input map. The importance map is generated from the last second layer in encoder, forming the 1bit masking layer to elementwise multiply(mux out) the last layer of encoder. The importance map and the 1-bit feature maps are all of 1/8 size of the original image. Convolution filters with strides is used for scaling down instead of the normal usage of pooling.

The generative network is also using a deep CNNs. The depth to space technique[2] is mainly used to scale-up the FMs. Mu.Li used a complex CNTX model derived from CABAC[3] for the entropy coding, achieved compress rate of about 80%. We alternatively use a much simpler coding scheme also achieve the similar compress rate. We implement Mus model in Tensorflow/ python environment, including the training, hard-example mining, training, entropy coding and evaluation.

2. Architecture of our approach

The structure of the codec is shown in Fig1. It is quoted from Mus paper[1]. It has a main path which generates the 1-bit binary FMs and branch to form the importance map. As can be seen, the importance map actually indicates those most detailed areas like objects edges, people's hair, grasses , etc. It helps to allocate more bits to preserve those details. This is done by mapping the 4bit importance map into a 64 channel 1-bit mask layer. The quantization is done by using the sigmoid activation followed by a round function.

2.1. Encoder

The detailed structure of encoder is as Table.1. N×N×C means the filter size and output channel number, s means stride, relu or sigmoid is the activation function for each layer. Note this structure is different from Mus original paper, a 3×3×256 convolution layer(in red) is inserted between 2 residual blocks. All paddings use SAME mode. This helps to get better performance.

2.2. Decoder

The decoder consists of depth to space layers to up-scale FMs. Compared with deconvolution, it is much cheaper and keeps good performance. The detailed structure of decoder is also different from the original paper , and is listed in Table2. All padding use SAME mode.

2.3. Residual block

The residual block we use is descripted in Fig2. It is the same as the original paper. Note that bs mean block_size for the depth to space module.

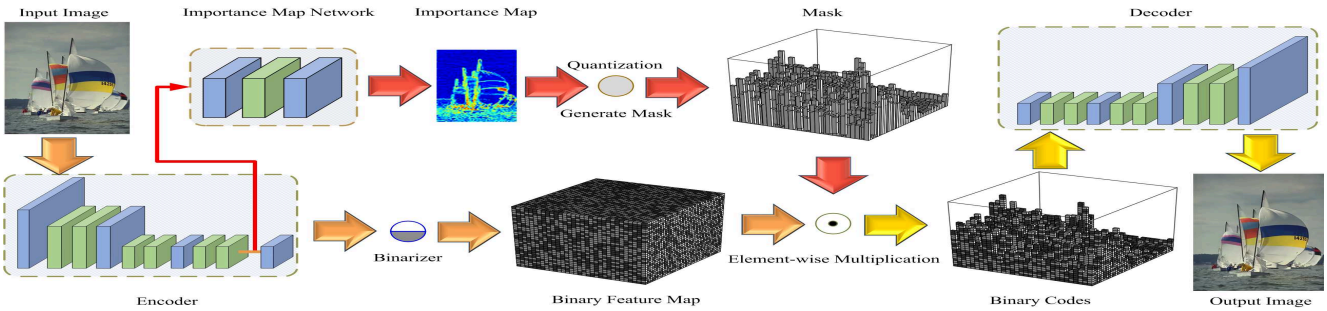

Figure 1. Architecture of Mus papaer[1]

main path RGB 3ch image	importance map path
$8 \times 8 \times 128, s = 4, relu$	
$ResBlk, num = 128$	
$4 \times 4 \times 256, s = 2, relu$	
$ResBlk, num = 256$	
$3 \times 3 \times 256, s = 1, relu$	
$ResBlk, num = 256$	↘ copy from left
$1 \times 1 \times 64, s = 1,$ $sigmoid$	$3 \times 3 \times 128, s = 1, relu$
↓	$1 \times 1 \times 1, s = 1, sigmoid$
1bit compressed 64 channel FM,M/8 x N/8 x 64 x 1bit	4bit importance map, M/8 x N/8 x 4bit

Table 1. Our detailed encoder structure

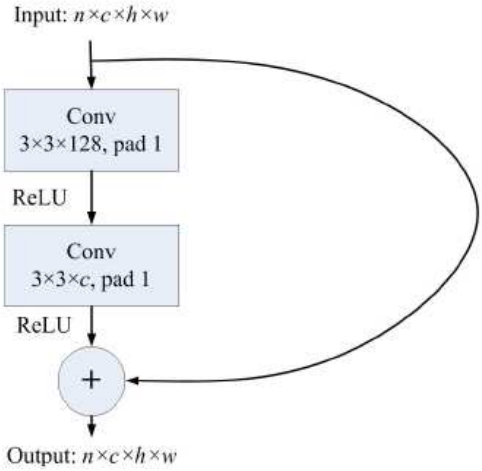

Figure 2. Residual block structure[1]

2.4. Importance map generation

The importance maps path is from the last second layer in the main path. The original paper did not mention the details but we assume that the importance map can be easily obtained just like edge extraction using easy Soble filter. So we use a simple 2-layer network to achieve this. As list in Table.3. The importance map is within [0,1] after the sigmoid activation, we next multiply it with 16 and floor the results into integers within [0,15]. We define it as P_{xy} to indicate importance level at location x,y. The result above is then mapped into a 1-bit mask with the same size(M/8 ×N/8×64) of the 1-bit compressed FMs in the main path. Let M_{xyk} be the bit at location x,y in channel k(k is within [0,63]). And the mapping function is:

$$M_{xyk} = \begin{cases} 1, & \text{if } P_{xy} \geq k \\ 0, & \text{otherwise} \end{cases}$$

3. Entropy coding

Mu used very complex CNTX model to predict each bit when encoding the bottleneck's entropy. Hes paper showed the compression rate is about 80%. We use a much simpler coding scheme also achieving the similar compression rate. We found when scanning the 1-bit FMs line by line, then channel by channel is tending to get consecutive 1 or 0 bits. This gives possibility to map this sequence in to run-length form and squeeze more when using a range coder(we actually used the range-coder the organizer had suggested). So before entropy coding, we reshape the 1-bit FMs from H×W×C to C×H×W, and we further invert each even line to make the scan sequence more consecutive. Note that we skip bits that are not available(indicated by the mask layer). The scan sequence is as Fig3. The run-length mapping is as Fig4. The very first bit is the same as input, but the bits after are set to 1 only when the current bit is different from the previous one. The 4-bit importance map is simply encoded with range coder without reorder and run-length mapping as the 1-bit FMs do.

4. Training

We use Adam optimizer to train the model, with the learning rate at 1e-5. We do not follow the step-down L-R method mentioned in [1] because Adam optimizer adapts LRs for different trainable variables individually and automatically. We first trained the model with the importance

layer description
$input M/8 \times M/8 \times 64(0 or 1)$
$3 \times 3 \times 512, s = 1, relu$
$ResBlk, num = 512$
$3 \times 3 \times 512, s = 1, relu$
$residualblk, num = 512$
$depth_to_space, bs = 2$
$3 \times 3 \times 256, s = 1, relu$
$residualblk, num = 256$
$depth_to_space, bs = 4$
$3 \times 3 \times 32, s = 1, relu$
$3 \times 3 \times 3, s = 1, None$

Table 2. Our detailed decoder structure

importance extraction network
$3 \times 3 \times 128, s = 1, relu$
$1 \times 1 \times 1, s = 1, sigmoid$

Table 3. Importance map branch

map fixed to all 1 output, letting all 64 channels FMs pass the bottleneck. When the result converged, we then unrestricted the importance map's parameters and started to shrink the channels.

4.1. Gradient

The key loss gradient we need to focus is for the function that maps 4-bit importance map into 1-bit mask. And also the rounding function following the sigmoid activation of the last layer in encoder. For the rounding function, we simply use 1 for the gradient backward propagation. For the backward propagation for the 4bit importance map to mask transform, we use the same formula in the original paper, please refer to sec 3.1.3 in [1]

4.2. Mining hard examples

We used all 1633 clic2018 train-set pictures, plus about 10000 hard images randomly select from ImageNet 2012 valid-set. The hard images were selected by the ratio of file size and the image size. Only the bit/pixel rate above 0.8 is selected. We believe the images of higher compress rate, contain more difficult cases for reconstruction and hence accelerate the training. All images are cropped into 128x128 as sample patches without overlapping.

4.3. Loss function

The loss function is aimed to balance the distortion and the final bit length. The first part Loss1 is MSE of all pixels channel(RGB) between the reconstruction and original patches, the second part Loss2 is the mean of importance map(result after sigmoid activation)

Loss = Loss1 + λ * Loss2

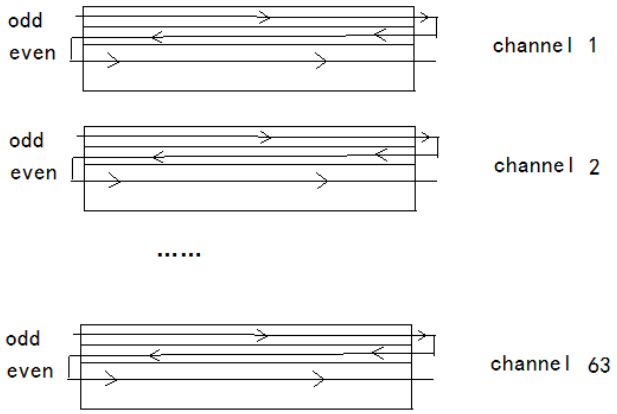

Figure 3. 1-bit FMs scan sequence

we chose a λ around 2300 that could achieve the 0.15bpp target compression-rate for the valid data.

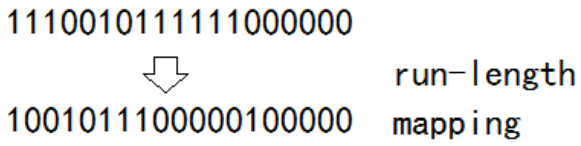

Figure 4. run-length mapping

5. Fine tuning model for test data

When the clic2018 test data was released, we found two major problems when encoding them with the model we had trained on the train pictures. Firstly the encoded bits is 13.8MB slightly exceeded the 13.3MB limitation. The second problem was the reconstructions of a few pictures distorted dramatically, the MSE is above 3000 which was obviously unusual. We addresses these problems on the encoder side, keeping decoder unchanged.

5.1. More restriction on bit rate

To address the first issue, at the beginning, we multiplied the importance map(after sigmoid) with a scale factor of 0.95 and resulted in shrinking the channels in the mask. But more pictures distorted unusually after this modification. After carefully inspected, we found compression rate of all these pictures were under 0.15bpp. We concluded that distortions of these heavily compressed images is very sensitive to bit rate reduction. Finally, we only scaled the importance map of images of compression rate above 0.15bpp with a factor of 0.93.

5.2. Fine training for encoder

After constricting the size within 13.3M, there were still few images distorted too much, with MSEs above 3000, severely degrading the final total average PSNR. It is because of lack of sufficient training data and training time. We addressed this problem by freezing the decoder and fine training the encoder with only these bad pictures. After fine training, all these pictures were encoded again with the new encoder, and reconstructed with the old decoder, resulting all MSEs below 350.

References

[1] Mu Li, Wangmeng Zuo, Shuhang Gu, Debin Zhao *Learning Convolutional Networks for Content-weighted Image Compression*, arXiv:1703.10553v2

[2] G.Toderici, D. Vincent, N. Johnston, S.J.Hwang, D.minnen, J.shor, and M.covell *Full resolution image compression with recurrent netural networks*, arXiv preprint arXiv:1608.05148, 2016. 1,2,3,

[3] 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):620636, 2003. 2, 5