VimicroABCnet: An Image Coder Combining A Better Color Space Conversion Algorithm and A Post Enhancing Network

Ming Li*[2], Changsheng Xia*[2], Jianhua Hu*[2], Zhangming Huang*[2], Yundong Zhang[1], Dekai Chen[2], Jinwen Zan[1], Guoxin Li[1], Jing Nie[2]

[1]Vimicro AI Chip Technology Corporation
The National Key Laboratory of Digital Multimedia Chip Technology
Room607,6/F, Shining Tower, No.35 Xueyuan Road, Haidian District, Beijing 100191 China
[2]Guangdong Vimicro Microelectronics Corporation
Building 16,Hengqin Financial District, Hengqin New Area, Zhuhai City, Guangdong Province, P.R.C
li.ming@zxelec.com

Abstract

*The framework of combining a better color space conversion (ABC) algorithm,and a post enhancing network for image coding, called VimicroABCnet[1] , is described in this paper. The ABC algorithm employs the principle component analysis[9] method, to find a new primary base axis offering the highest variance for each individual image. The RGB values of each pixel are pre-processed by a 64x64 template filtering. The pixels are then converted by the proposed ABC algorithm, before being encoded by an open source coder[2]. During decoding, the least square method (LSM) has been introduced to estimate the optimal inverse conversion, instead of using a matrix inversion directly. Another feature of the VimicroABCnet is the enhancing network, which adopts the architecture of a classic ResNet[3], and post-processes the decoded RGB image after ABC. Experiments on the CLIC2019 valid dataset have shown significant RGB-PSNR boost of **0.26db** or **7.4%** bits save@**0.145bpp**, and **1.2db/22.5%@1.0bpp**, making use of the ABC algorithm; and a RGB-PSNR boost of **0.30db@0.15bpp**, making use of the enhancing network, respectively. Combining both techniques, an improvement of **0.56db** or **12%** bits save@**0.15bpp**; and a decrease in the compressed file size of about 17.8% are achieved in the transparent track. It is noted that each of the two techniques contributes equally. Methods to speed up the decoder model are also discussed.*

1. Introduction

In this paper, we combine the Vimicro's novel ABC method with a post enhancing CNN network to improve the backbone H266 image codec's RGB-PSNR performance. The ABC(A Better Color Space Conversion For Image Compression[1]) method not only significantly reduce the codec's BD-rate, but also keeping the decoder's efficiency as a traditional decoder. The post enhancing network has been documented in many previously work[5],[6],[7], and proved to be very useful to improve the PSNR of a ready decoded image.

The ABC method is to find a new primary new base axis to form a new RGB to YCbCr conversion matrix as eq.1, we define T$_{enc}$ as the matrix and Offset$_{enc}$ as the offset vector in this eq. The original picture is of RGB 3-dimension. We assume each channel's entropy is proportional to their variances. To find a new axis that present the most entropy, the PCA is natural choice since it's simple and effective. Data samples are pixels' RGB minus corresponding averages. The PCA also produce the second/third base axes as well, which can be used as directions of base axes for Cb/Cr directly.

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} x1 & x2 & x3 \\ y1 & y2 & y3 \\ z1 & z2 & z3 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} Y\_offset \\ Cb\_offset \\ Cr\_offset \end{bmatrix} \quad (1)$$

In this paper, we replace the 16x16 grid's average sample scheme with a common 64x64 convolution kernel. Each pixel's RGB value is subtracted by the average of its 64x64 neighbours and used as the samples of PCA[9]. This new scheme improve the performance about 0.013db@0.15bpp. ABC later use the least square method to estimate a new YCbCr to RGB conversion matrix and offsets to further improve the performance.

We also use a classic ResNet architecture as the post enhancing network. ResNet is very fast to converge when training. We test this architecture with different depths and widths. The result shows there is a limit for this type of post network, which is with even more parameters the resulting performance boost is still kept under a boundary.

*These authors share first-authorship

2. Details of our approach

2.1. Brief description of Vimicro's ABC method

Since the detailed description for ABC is documented in [1] and it's only a part of our implementation of VmicroABCnet, we only picture a rough framework here.

2.1.1 Find the primary axis using PCA

The primary axis in the new space should produce the most variance, and the converted data is as the input to Y channel pipeline. PCA data samples are as eq.2:

$$\begin{aligned} R_{sample} &= each\_pixel'R - average\_R \\ G_{sample} &= each\_pixel'G - average\_G \\ B_{sample} &= each\_pixel'B - average\_B \end{aligned} \tag{2}$$

Dimension of the sample set is [h*w, 3], h and w is height and width of the image. The normalization is dropped here because it would make R/G/B's average variance equal to 1. In this paper, we replace the original 16x16 grid's average method with a common 64x64 convolutional filter, the center of the filter is 1-1/64*64, the other coefficients of the filter is -1/64*64. This filter actually subtract the center pixel(R/G/B respectively) by its 64x64 neighbors. The 64x64 size is determined on fact that most max CTU's size in the H266 codec is of 64x64(although 128x128 is permitted but it's rare in practices). The new convolution method increases the RGB-PSNR about 0.013dB for CLIC valid dataset @ 0.15bpp.

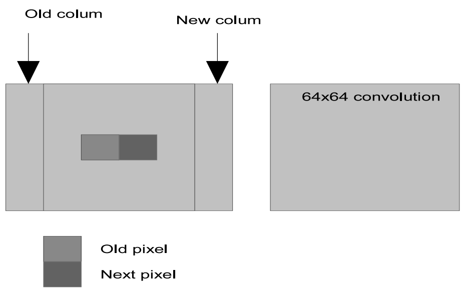

Figure 1: Accelerating the convolution by subtracting old column and adding the new column

The 64x64 convolution is very time consuming. We exploit the fact that the convolution is processed pixel by pixel, and the change of average of the 64x64 area is only affected by the old and new column of the filter , Fig.1, the average neighbours of the new pixel is calculated by subtracting the old column and plusing the new one. The acceleration decrease the 64x64 operations for each pixel to 64x2.

2.1.2 Optimization of the reverse conversion

In decoding, we optimize the T$_{dec}$ and Offset$_{dec}$, we rewrite a equivalent formula as eq.3

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = T_{dec} \cdot \begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} + Offset_{dec} \tag{3}$$

Optimizing the T$_{dec}$ and Offset$_{dec}$ is to minimize the mean square error between eq.3's resulting RGB and the uncompressed RGB. Each row of T$_{dec}$ and Offset$_{dec}$ are estimated within one LSM process. The T$_{dec}$ should be very closed to T$_{enc}^{-1}$, but is more optimal, we can see the improvement in the experiment section of [1].

2.1.3 Architecture of encoder

We use the standard H266 encoder as our baseline codec, wrapping it up with the ABC and post enhancing network. As in Fig.2, the original RGB image is firstly feeded into the PCA-based module to estimate the T$_{enc}$ and Offset$_{enc}$, which are used as the coefficients for the later RGB to YCbCr conversion. Cb/Cr are scaled up and down with normal bi-cubic interpolation. The reconstructed YCbCr 444 data is used with the original images' RGB to estimated the T$_{dec}$ and Offset$_{dec}$. The T$_{enc}$ and Offset$_{enc}$ are output together with the H266 bitstream. Overhead of our method is minor, totally 12 float coefficients. For a typical 2M picture, the overhead is only 12*32/2000000 = 0.000192bpp.

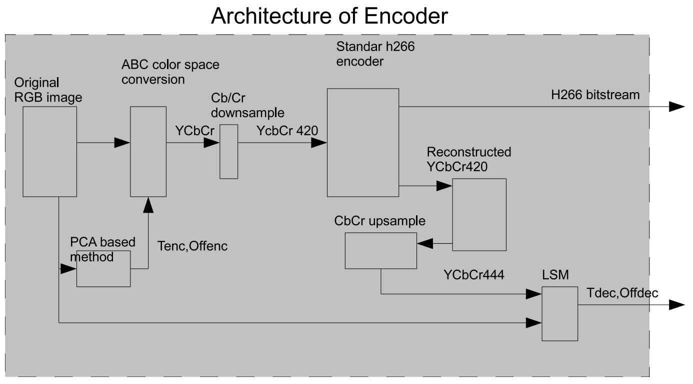

Figure 2: Architecture of our encoder

2.2. Decoder and Post enhancing network

2.2.1 Architecture of decoder

The decoder of our method is showed as 3, the YCbCr 420 is the output of the H266 decoder, and upsampled to 444 format. By using the T$_{dec}$ and Offset$_{dec}$ comes together with the main bitstream, the reconstructed RGB image is converted from the YCbCr 444 data.

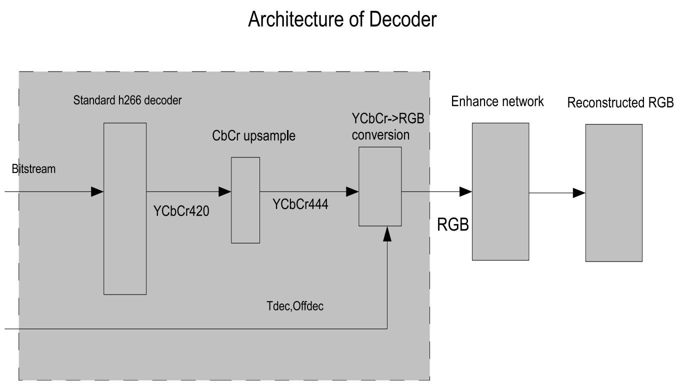

Figure 3: Architecture of our decoder

2.2.2 Post enhancing network

We later use a classic ResNet architecture as the post enhancing network. ResNet is very fast to converge when training. We test different architecture with different depths and widths. The result shows that there is a limit for this type of post network, which is with even more parameters the resulting performance boost is still be kept under a boundary. The architecture of the post net is borrowed from [4], as shown in Fig.4.

The input for the network is the RGB reconstructed image from the H266 decoder(with ABC). The output of the network is the final enhanced RGB image. The main architecture is a ResNet, two 3x3 linear CNN is at the entry and exit of the network, with a begin-end skip route adding into the final node. The middle body of the network is stacks of sub-network of ResNet, the number of stack is d, and the width of each sub-network is k. All the non-linear parts is done with leakyRelu inside this sub-network. The original network in [4] is of d=6 and k=64. We further increase these with d=11 and k=80 and later d=11 and k=160. Depth of 11 is chosen so the perception field of a final output pixel is about 70x70, which is close the normal max CTU size of 64x64 as mentioned in early section. We can see the performance and size of these models in the later sections.

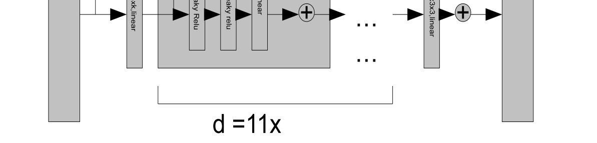

Figure 4: Architecture of post network

2.2.3 Training the network

The train dataset includes the CLIC2018 train and CLIC2018 test set, which is of about 1900+ images. All images are cropped into 64x64/224x224 as sample patches without overlapping. There are totally 960k/88k patches at all. We use Adam optimizer to train the model, with the initial learning rate of 4e-4. Larger lr would cause NaN problem in training in our experience.

For the lowrate track, all input images are firstly encoded by H266 plus ABC, using QP36 which is the closest QP that produce 0.15bpp compression rate for the valid dataset. The reconstructed patches and the corresponding original ones are used as input and output of our network. Patches' size is of 64x64. The loss function is the mean square error distance of the output RGB patches and corresponding original patches.

For the transparent track, all input images are firstly encoded by H266 plus ABC, using QP20 which is the closest QP that produce 40dB and 0.993 MSSIM for the valid dataset. Patches' size is of 224x224 which is bigger than the lowrate track case, because the minimal input for Tensorflow for MSSIM operation is 176x176. The loss function is 1- MSSIM of the output RGB patches and corresponding original patches. Only MSSIM loss is used because the traditional image encoder like H266 always emphasize the PSNR rather than MSSIM performance, so the encoded images always have PSNR margin over 40dB when the MSSIM meets the 0.993 requirement.

We use a special adaptive method for decreasing the lr in training. The average MSE/MSSIM loss in the current epoch are defined as a reference named avgMSE/avgMSSIM loss, they are calculated at the end of the training of each epoch. In the training for of lowrate/transparent track, if the current avgMSE/avgMSSIM loss loss is larger than the last epoch, the learning rate is multiplied by a const c, in our case c=0.9$\sim$0.92, or else keep unchanged. This strategy keep the training converging fast at the beginning when the loss keep dropping fast, and slow down when the loss rebounds. Compared with the traditional predefined lr stepping or exponential decreasing method, our adaptive method significantly accelerate the training.

3. Performance experiment

In the lowrate track experiment, we have the following test groups and corresponding results. The test is done on the CLIC valid data.

1. Normal H266 and normal BT.601 color space conversion as baseline
2. H266 plus ABC conversion
3. H266 plus ABC plus post enhancing(d=11,k=160)

	d=6,k=64	d=11,k=80	d=11,k=160
para num	450k	1.2M	5M
psnr@0.15bpp	32.38	32.44	32.46

Table 1: 0.15bpp RGB performance of network with different size

	size of files
H266,QP20	34.2M
H266+ABC	30.5M
H266+ABC+post net(k=80,d=11)	28.5M
H266+ABC+post net(k=160,d=11)	28.1M

Table 2: Transparent track performance table

We also test different post network's performances as in Table.1. It's obvious the post network has its limit, even the large model's parameters boom to 5M than the moderate 1.2M model, the PSNR boost is tiny 0.02dB.

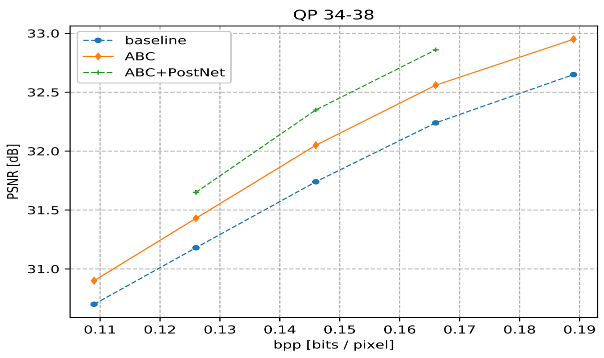

Figure 5: Comparison in lowrate track

Combine H266 codec and ABC and post enhancing network, the performance on transparent track is also significant improved as shown in Table.2

4. Speeding up

Three measures are used to speed up the model, later two are our contributions:

1. The internal H266 decoder use AVX2 instructions to speed up important modules.
2. We parallelized the CPU and GPU using a midway image pool scheme on the server as illustrated in * in Table.3. This help to utilize GPU from 33% to 66%
3. The post network can be disabled to speed up the model even without GPU support. As row 3 compared with row 4 in Table.3.

In our opinion, a qualified modern image decoder in commercial must meet the following requirement:

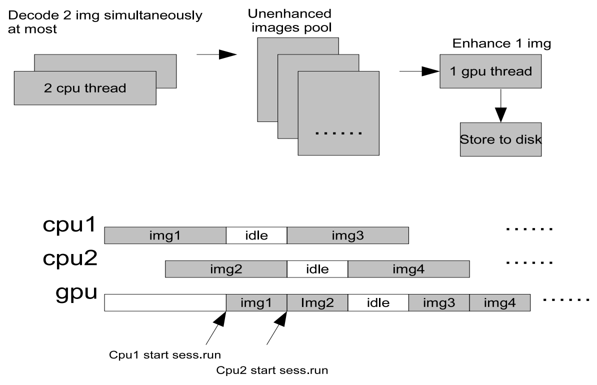

Figure 6: Our solution for decoding with post net

	psnr(dB)	decode time ms(per image)	decoder size
BPG	30.84	800	400k
H266+BT.601	31.87	900	900k
H266+ABC	32.16	900	900k
H266+ABC+post net (k=80,d=11)	32.45	1650/3100*	5M

Table 3: RGB PSNR performance of different decoders, * with/without parallelizing CPU/GPU

1. Its performance must significantly excel the existing codec such as BPG,H266
2. Decoding speed must be fast, a normal 2M∼3M image should be decoded within 1 second.
3. Must meet common available hardware such as low-powered potable PC without GPU support.
4. Decoder's size must be small for convenient download.

As to our knowledge, VimicroSpeed without post network enabled is the only participant of CLIC2019 that meets all this requirements. The relative statics is shown in Table3

References

[1] Y.Zhang, M.Li *"A Better Color Space Conversion Based on Learned Variances For Image Compression"*, regular paper submitted to CLIC2019

[2] H266 (https://de.wikipedia.org/wiki/h.266/), 2018. 4

[3] K.He *"Deep Residual Learning for Image Recognition"*, arXiv:1512.03385

[4] Lei.Zhou *"Variational Autoencoder for Low Bit-rate Image Compression"*, challenge paper of CLIC 2018

[5] Jianhua.Hu *"Combine Traditional Compression Method With Convolutional Neural Networks "* challenge paper of CLIC 2018

[6] Kai Cui *"Decoder Side Image Quality Enhancement exploiting Inter-channel Correlation in a 3-stage CNN: Submission to CLIC 2018"*

[7] C. Dong, Y. Deng *"Compression artifacts reduction by a deep convolutional network"* In Proceedings of the IEEE International Conference on Computer Vision, pages 576–584, 2015. 1

[8] International Telecommunication Union *"BT.601 : Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios"*

[9] Jolliffe, I.T. *"Principal Component Analysis, second edition (Springer)."*,