Improve Image Codec's Performance By Variating Post Enhancing Neural Network:Submission of zxw for CLIC2020

Ming Li[*], Yundong Zhang[*], Changsheng Xia[*], Jinwen Zan[*], Zhangming Huang,
Dekai Chen, Guoxin Li, Jing Nie
VimicroAI
Building 16,Hengqin Financial District,
Zhuhai city, Guangdong Provice, P.R.C
li.ming@zxelec.com

Abstract

*Adding post enhancing filter after traditional image decoder to improve reconstruction quality is nowadays a very common method [1],[2],[3],[5]. Researchers use a large network filter or repeatedly stack single/multiple types of relative simple filters together. They all achieved better results. On the other hand the training materials and training time increases exponentially for a larger network scale, and the performance improvement becomes less and little. We learn this experience from the CLIC2019 low-rate track, where we proposed the VimicroABCnet and VimicroSpeed[4], with 2 post filters of different scale. The later one(**5 time** larger than the small one) achieved the final test's PSNR by improvement of only **0.02db@0.15bpp**. In this paper, we propose a method to variate an existing post network filter(base filter). The base filter is altered into different ones, alternation only happens to weights. The key of the method is to divide the training data into different groups. Based on the pre-trained base filter, different altered filters are individually fine-trained with different group of training data. There are different ways to divide the training data, and we use a relative simple one. Sort by compression rate(with traditional codec) and bin the training images in to 4/8 group of training data subsets. With the new filters plus the base one, we now have 5/9 filters candidates in encoding phase and choose the best. The CLIC2019 test data show that PSNR increases **0.04db@0.15bpp** and **0.06db@0.15bpp** than the one filter VimicroSpeed method. This method requires the same training data and perfectly suitable for multi-GPU training scheme, and retraining the altered filters is much easier and consuming less time than training a relative large network filter. Also the result is better(5 filters scheme@0.04db vs VimicroABCnet@0.02db)*

[*]These authors share first-authorship

1. Introduction

Adding post enhancing filter after traditional image decoder to improve reconstructed quality is nowadays a very common method. Our last year's submission VimicroABCnet[4] and VimicroSpeed used this architecture plus the optimal PCA-based color space conversion. Shown as 1

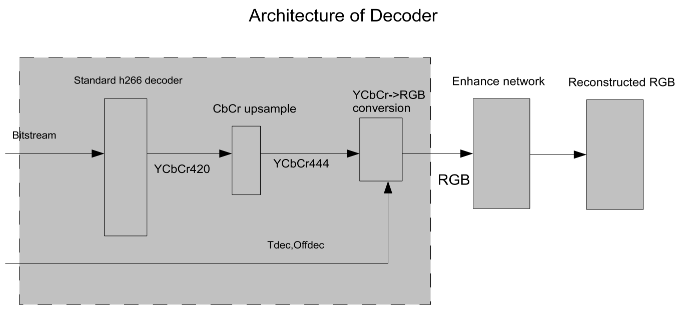

Figure 1: Architecture of our CLIC2019 decoder

The VimicroABCnet and VimicroSpeed used the same prefix H.266 decoder but with post enhancing network of different scale. The larger one has 6M parameters versus 1.2M, about 5 times large. And final CLIC2019 test data shows that even with 5 times larger structure the PSNR improvement is only 0.02db@0.15bpp. Training the larger network requires exponentially time and data. It implies that by only increasing the enhancing network's scale is not the solution to improve the post network.

We use the same H.266 codec and enhancing network architecture for CLIC2020, but the enhancing network is altered. The base filter(the one used by VimicroSpeed with 1.2m paras) is altered into multiple ones(here we test 4/8), the modification only happens to the weights. With the new filters plus the base one, we now have 5/9 filters candidates. The 5 filter scheme is same sized of the old 6M large model, but the PSNR boost is much better(0.04db vs 0.02db), and the training is much easier and less time consuming. With the 9 filter scheme, the improvement can further reach 0.06db. The decoder we used in CLIC2020 is shown as 2

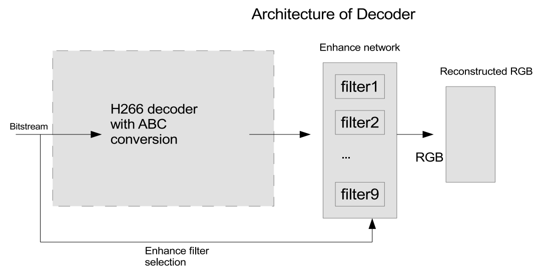

Figure 2: Architecture of zxw decoder

2. Details of our approach

2.1. Base filter and its larger version

The base filter we use(used in VimicroSpeed) comes from classic ResNet architecture. The architecture of the post net is borrowed from [2], as shown in Fig.3. 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 leaky-Relu inside this sub-network.

Our base filter is configured with d=11 and k=80, the number of parameters is about 1.2M. Its bigger version(used in VimicroABCnet) is configured with d=11 and k=160, with about 6M parameters. Depth of 11 is chosen so the perception field of a final output pixel is about 70x70, which is close to the normal max CTU size of 64x64 as H.266 encoder normally uses.

2.2. Variate the base filter into multiple

We try to variate a existing post network filter(1.2M model). The base filter is altered into different ones, alter-

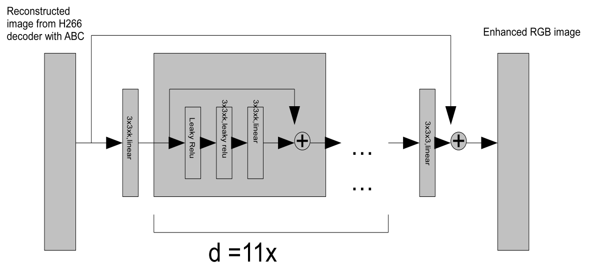

Figure 3: Architecture of post network

nation only happens to weights. We then only need to fine train the same model with different training data. Within each training, data must has some kind of similarity to help the current filter to fit in. So the problem becomes how to divide the training data into different groups. We here refer each training image's compression rate(CR), calculated as

$$compressed\_file\_size/(image\_height * image\_width)$$

We assume that image with different CR are somehow similar and group them. All training images are firstly compressed with H.266 encoder, the reconstructed images are as input and original images are as output for the enhance network. Encoder with QP=36 achieves about average 0.15bpp compression rate. Then all the CR(compression rate) are sorted from high to low and all images are reordered by its CR. Divide the images evenly in to N bins along their new order, and make sure each bin have total_num/N images. Even division helps to balance training data for each altered filter. N here we select 4 and 8 as our configuration. In the encoder phase, we test all the filter candidates, and select the best one, adding the selection information into bitstream, the bit increase is minor.

2.3. Training the filters

The total train dataset includes the CLIC2018 train and CLIC2018 test set, which is of about 1900+ images. All images are cropped into 140x140 as sample patches, the center 70x70 areas of all patches are non-overlapping. There are about 800k 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. The total data is used to train the base filter, and divided into bins to train the altered ones.

Based on the pre-trained base filter, different altered filters are individually fine-trained with different group of data. There are 2 different ways to train the new altered filters:

1. Train each new filter with its own data from ground up.

2. Train the base filter with all data first(as we did for VimicroSpeed), use this pretrained weights and fine-train each alterative with its own data.

We test both methods, finding that method 1 would train new filter always(almost) worse than the base filter. It's because each alterative only fit for it's data population and lacks the generality. The Later is much better, because each filter is firstly trained for the generality before biased to its own bin. Meanwile, the later training scheme is much time saving, generality training for each filter is saved.

Preposessing the training data and designing loss function are also problems. Normally each image must be divided into patches[3], and batches them into the training models to reduce the MSE between the reconstructed and original patches. Many former researches ignore the padding and simply minimize the MSE of the whole patch. As the enhance network becomes larger and deeper, the corresponding reception field for each pixel in the output become larger too. In our model it's about 70x70. If we simply use the 70x70 patches for training, only the center pixel would be correctly trained, while others are not fully trained because some of their reception filed is outside the patches and replaced with padding artifact. We address this by using larger patches, and only the MSE of center part counts. As illustrated as Fig.4

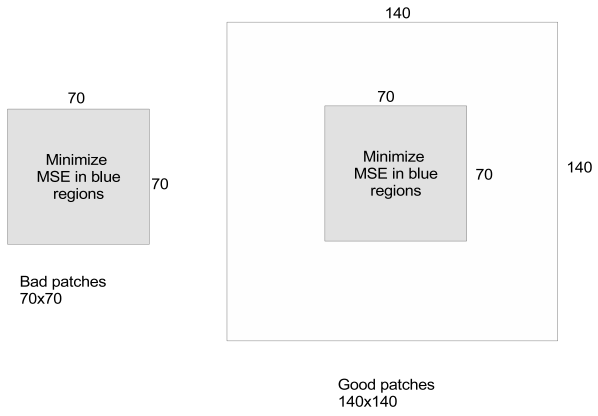

Figure 4: Example of good and bad patches

3. Result and future works

By training the extra 4/8 altered filters plus the base one, we finally have the 5/9 filters decoding schemes, we test them with the CLIC2019 test data at 0.15pp. The PSNR results shown as Table1

Apparently the current training images grouping scheme is coarse and not optimal, although it achieves much better PSNR boost than VimicroABCnet(0.04dB vs 0.02dB). Our future work will focus on how to optimally grouping the training data. One of the potential method is using a light network to predict the grouping category, and the predicting network should be jointly trained with all enhancing filters. The architecture of such task will look like Fig.5

	psnr	parameters
1.2M base filter	31.07	1.2M
6M larger filter	31.09	6M
1.2M X5 filter	31.11	6M
1.2M X9 filter	31.13	10.8M

Table 1: PSNR perf for CLIC2019 lowrate test

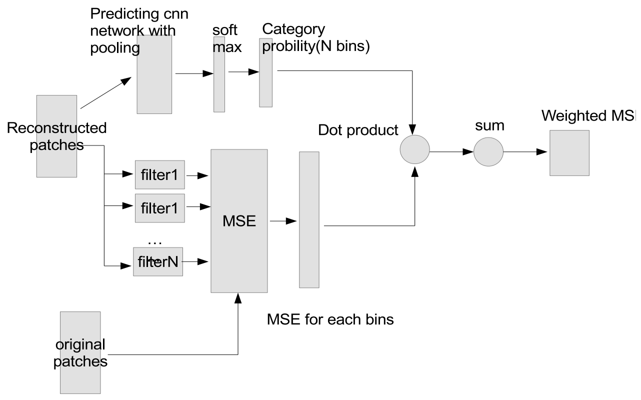

Figure 5: Architecture of our future work

The input reconstructed patches is firstly put into a predicting CNN network, with pooling to extract embedded features, then fully connect into the softmax block to predict each bin's probability. Patches is also fed into all candidate filters and estimate each's MSE loss with the original patch. Each MSE estimated from a filter corresponds to each bin. At last, dot product the bins' probabilities with their MSE to get the weighted MSE. The loss function is to reduce the weighted MSE. The training will train the enhance filters and the bin classifier all at once. Major challange about this task is how to assure each bin is trained to different biases, the initial parameter setting may be crucial.

References

[1] Seunghyun Cho *"Low Bit-rate Image Compression based on Post-processing with Grouped Residual Dense Network"*, challenge paper of CLIC2019

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

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

[4] Ming.Li *"VimicroABCnet: An Image Coder Combining A Better Color Space Conversion Algorithm and A Post Enhancing Network"* challenge paper of CLIC 2019

[5] Wei.Jia *"Residue guided loop filter for HEVC post processing"* IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY