Journal of Information Hiding and Multimedia Signal Processing ©2017 ISSN 2073-4212
Ubiquitous International Volume **8**, Number **1**, January **2017**

Quality Enhancement of Compressed Video via CNNs

Jingxuan Hou[1,2], Yao Zhao[1,2], Chunyu Lin[1,2], Huihui Bai[1,2] and Meiqin Liu[1,2]
[1]Institute of Information Science
Beijing Jiaotong University, Beijing 100044, China
[2]Beijing Key Laboratory of Advanced Information Science and Network Technology
Beijing 100044, China
14120316@ bjtu.edu.cn, *yzhao@bjtu.edu.cn, cylin@bjtu.edu.cn

Received May, 2016; revised September, 2016

Abstract. *This work presents a compressed video enhancement algorithm based on convolutional neural networks (CNNs), which aims to establish an end-to-end mapping function between the compressed and the original video frames. Finally, the data is stored or transferred in the form of video plus network parameters. We can get a quality improved frame when taking the compressed frame as the input of CNN. Unlike some traditional filtering methods, our method can utilize more prior information, and thus can recover more details. Our algorithm relies on the training data, and independent of the compression method itself. In this work, we adopt H.264/AVC as a typical encoder to verify the effectiveness of our algorithm. Experimental results demonstrate that the proposed algorithm provides better reconstructed quality than that of classical approaches.*
Keywords: CNNs; Enhancement; H.264/AVC; End- to-end mapping.

1. Introduction.

Transform-based coding has been widely adopted in many current image and video compression standards, such as JPEG, JPEG2000, MPEG-X, and H.26X. The major goal of these compression techniques is to efficiently compress a large amount of visual information to fit the bandwidth limits of communication channels while preserving acceptable quality. The idea of all these techniques is to quantize the transform domain coefficients to reduce the volume of data. This approach will bring ringing artifacts, especially at low bitrate compression. In addition, block artifacts are widely existed in block based coding. As a result, the compressed images or videos suffer from noticeable visual distortion.

Different from [1], the focus of our work is not the performance improvement of the compression algorithm itself, but the post-processing of compressed video/image. Many post-processing methods have been proposed in order to enhance the image quality under limited bitrate situation. They can be broadly divided into two categories: image enhancement approach and image restoration approach.

The goal of the image enhancement approach is to subjectively improve the perceived image/video quality. A typical solution is to perform post-filtering to reduce the distortion, such as [2-6]. In [7], a no-reference quality metric for evaluating the blocking artifacts was presented to reduce the impact of filtering on high frequency information. In general, image enhancement approach aiming at smoothing visible artifacts, instead of restoring original pixel value. The main advantage of this kind of approach is usually its relatively low computational complexity.

The way of image restoration approach to solve the artifacts removal problem based on some prior knowledge and observed data at the decoder. Such methods include MAP (Maximum a posteriori)-based method [8], POCS (Projection onto convex sets)-based method [9] and SR (sparse representation)-based method [10], [11]. The method we proposed also belongs to this category. Generally speaking, the image restoration approach achieves better performance at the expense of higher computational complexity.

The rest of this paper is organized as follows. In Section 2, we present the proposed video post-processing framework. In Section 3 experimental results and comparison are demonstrated. Finally, Section 4 concludes this paper.

2. **CNNs for compressed video enhancement.** The purpose of our work is to establish an end-to-end mapping between the compressed frames and the original frames. It can be expressed as:

$$Y = \mathcal{F}(y, \Theta) \tag{1}$$

where $\Theta = \{W_1, W_2..., W_n; B_1, B_2, ..., B_n\}$ is the network parameters, n is the number of layers and y is the compressed sequence.

CNN is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex. CNNS have wide applications in image and video recognition [12] and natural language processing [13] etc. One major benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units. We choose it because the number of its parameters is few enough so that we can ignore the influence of data volumes grow caused by the network parameters to the bitrate.

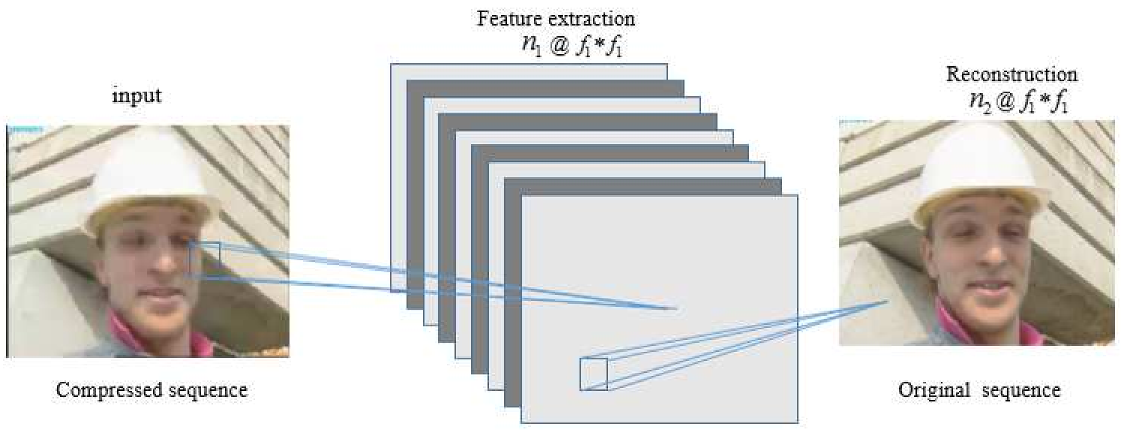

Figure 1. Structure of the network in our work

2.1. **Structure of the Network in our work.** As shown in Figure.1, the network adopted in this paper is a two-layer CNN. The reasons why we choose the two-layer model will be explained in the section 3. Each layer is conceptually represented as an operator: patch extraction, representation operator and reconstruction operator. Then we will introduce them in detail.

The first layer can be considered as a feature extraction layer, it is equivalent to convolving the input image by a set of filters. Partial feature maps of the first layer are shown in Figure.2. Unlike some pre-defined operations, such as Laplacian operation, the operations here are defined by solving the optimization of the network. The relationship between the input and output of the first layer can be expressed as:

$$F_1(y) = max(0, W_1 * y + B_1) \tag{2}$$

where W_1 and B_1 represent the filters and biases respectively, the size of them are $n_1 * f_1 * f_1$, n_1 respectively. Here n_1 is the number of filters, f_1 is the size of every filter. The symbol '$*$' denotes the convolution operation, and the activation function adopted here is ReLu (Rectified Linear Unit) due to these advantages to accelerate convergence. The output $F_1(y)$ is composed of n_1 feature maps.

In the second layer, we reconstruct the final output frame by convolving the feature maps in the first layer with another filter. It can be expressed as:

$$F_1(y) = W_2 * F_1(y) + B_2 \tag{3}$$

Similar to the first layer, W_2 and B_2 represent the filters and biases respectively. The size of them are $n_1 * f_2 * f_2 * n_2$ and n_2 respectively, and $n_2 = 1$ if we only consider the luminance channel. So far, we have established the end-to-end mapping between the input frames and the final output, Y is the desired result.

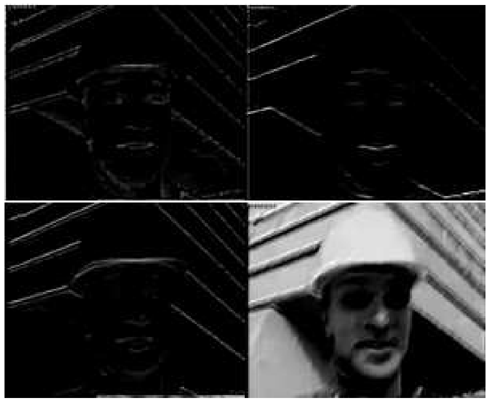

FIGURE 2. Partial feature maps of the first layer

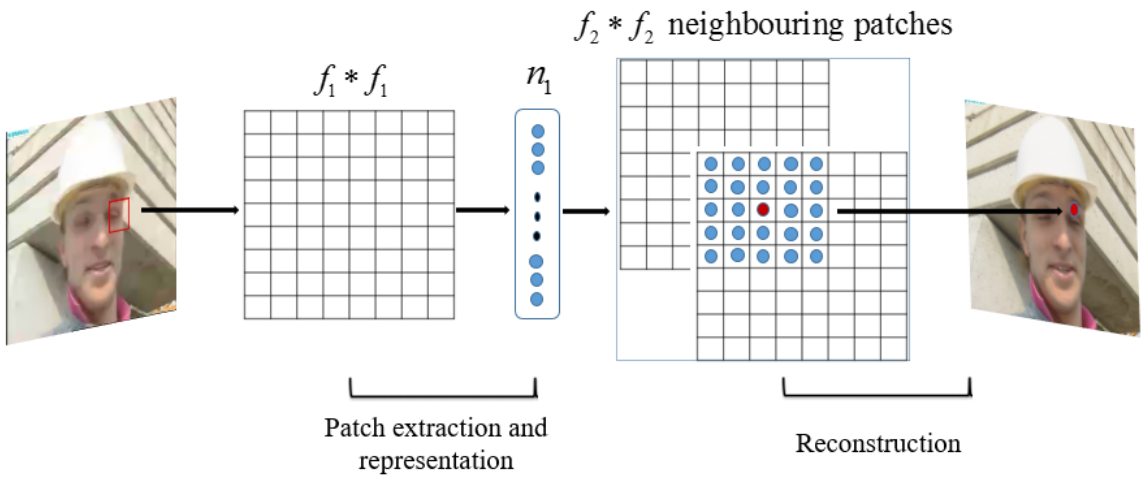
$f_2 * f_2$ neighbouring patches
$f_1 * f_1$
n_1
Patch extraction and representation
Reconstruction

FIGURE 3. An illustration of SR-based methods in the view of a convolutional neural network

The two-layer network is similar to traditional transform-based image restoration methods, such as DCT transform, wavelet transform and the SR-based method. Take SR-based method as an example, as shown in Figure.3, we can regard SR-based method as a kind of CNN. If the dictionary size is n_1 , sparse coding is equivalent to applying linear filters $(f_1 * f_1)$ on the input image. For a patch $(f_1 * f_1)$, its sparse coefficient (n_1) in SR-based method can be regarded as the results of the first layer in CNN. The atoms of the low-resolution dictionary are equivalent to the filters in the first layer. This is illustrated as the left part of Figure.3. The above n_1 coefficients are then projected onto another (high-resolution) dictionary to produce a high-resolution patch. The overlapping high-resolution patches are then averaged, which is equivalent to linear convolutions on the n1 feature maps, as illustrated in the right part of Figure.3. However, the SR solver is an iterative algorithm, not feed-forward. On the contrary, our method is fully feed-forward and can be computed efficiently.

2.2. **Network training.** The premise of network training is to generate training samples. The general approach of deep learning is to take a special image set as the training samples. But experiments suggest that the network trained with a given set does not have good generalization performance because given compression parameters have different effects on different sequences due to the special structure of artifacts. Therefore, we train a set of parameters for each pair of original and compressed videos in order to improve the performance. The data is stored or transferred in the form of video plus network parameters. However, training a network for each pair of sequences greatly increases the workload. So we adopt following methods to improve the training speed when sampling training set:

a) Sampling one in three frames to accelerate the training because of the similarity between adjacent frames. Finally, train the network using all frames to enhance the performance when the network converges.
b) Post-processing is just applied to the luminance channel because the human eye is not sensitive to the color channel.
c) Randomly divide the video frames into patches.
d) Exclude the smooth patches because we find that the network is hard to converge if the samples are smooth.

Finally, the training samples are composed of set $\{Y_i, y_i\}$. Here Y_i and y_i respectively represents the patch sampled from luminance channel of original and compressed frames. To avoid border effects during training, all the convolutional layers have no padding, so the size of output $Y_i((f_y - f_1 - f_2 + 2)^2)$ is smaller than input $y_i(f_y^2)$.

The next step is network training, and it is achieved through minimizing the loss between the reconstructed frames $\mathcal{F}(y_i, \Theta)$ and the corresponding ground truth original frames Y_i. The loss function we choose is MSE (Mean Squared Error), because using MSE as the loss function favours a high PSNR. It can be expressed as:

$$\min_{\Theta} L(y, \Theta) = \frac{1}{n} \sum_{i=1}^{n} ||\mathcal{F}(y_i, \Theta) - Y_i|| \tag{4}$$

where n is the number of training samples. The loss is minimized by stochastic gradient descent with the standard back-propagation. Detailed mathematical derivation can be referred from the literature [14].

We implement our model using the Caffe package. And we use the parameters that have been trained with the same sequence but compressed by different quality factor to initialize the network in order to accelerate training.

3. **Experiments and comparison.** In our experiments, the testing platform is the desktop computer with Intel Core i3-2120 CPU 3.30GHz and 4.00G RAM.We set $f_1 = 9$,

Table 1. Performance for H.264/AVC compressed sequences (coded in Quality-based mode)

sequences	Size	Bitrate (Kbit/s)	H264 (dB)	Processed (dB)	PSNR Gain (dB)
Foreman	352*288	106.51	32.27	33.66	1.06
Mobile	352*288	283.06	27.09	28.32	1.09
Highway	352*288	39.16	34.75	34.96	0.52
Stefan	352*288	224.18	29.04	29.94	0.84
BasketballDrill	832*480	413.86	32.16	32.74	0.59
SlideEditing	1280*720	1181.83	29.39	30.96	1.57
Johnny	1280*720	198.55	35.83	36.20	0.37
KristenAndSara	1280*720	254.38	35.35	36.07	0.72
Average	-	-	31.86	32.59	0.84

Table 2. Performance for H264/AVC compressed sequences (coded by different QP)

sequences	QP	Bitrate (Kbit/s)	H264 (dB)	Processed (dB)	PSNR Gain (dB)
Foreman (352*288)	36	168.35	34.44	35.51	0.92
	38	132.46	33.28	34.57	1.04
	40	106.51	32.27	33.66	1.06
Stefan (352*288)	40	224.18	29.03	29.94	0.84
	42	173.05	27.61	28.49	0.75
	45	116.38	25.65	26.38	0.57

$f_2 = 5$, $n_1 = 64$ and $n_2 = 1$ because we only consider the luminance channel. The size of input (f_y^2) is set to 33*33, so the size of label ($(f_y - f_1 - f_2 + 2)^2$) is 21*21. The filter weights of each layer are initialized by drawing randomly from a Gaussian distribution with zero mean and standard deviation 0.001 (and 0 for biases). According to the denoising [15] and super-resolution [16] case, the learning rate is 10^{-4} for the first layer, and 10^{-5} for the second one. The number of encoded frames of every sequence is 300. GOP size is set to 16 with one I-frame, followed by 15 P-frames, and all the sequences are coded by 30 fps.

The peak signal-to-noise ratio (PSNR) is used to measure the objective quality in all cases. We tested some compressed sequences (include various resolution, such as 352*288, 832*480, 1280*720) that coded in Quality-based mode, the objective and visual quality results are shown in Table.1 and Figure.4 respectively. As can be seen in Figure.4, we can find that our method can effectively remove the ringing artifacts. We also present the experimental results of two sequences that encoded by different quantization parameters in Table.2. Two video post-processing methods [5], [7] as well as the built-in H.264/AVC in-loop filter [17] were used for the comparison, and here GOP size is set to 20 with one I-frame, followed by 19 P-frames in order to keep in line with the comparison methods. The test sequences for comparison are Foreman and Stefan with the CIF format and coded in Bitrate-based mode. Experimental results show that our proposed algorithm has great improvement compared with other methods as shown in Table 3. Moreover, we can find that the performance of our method in Quality-based mode is better than that of the Bitrate-based mode.

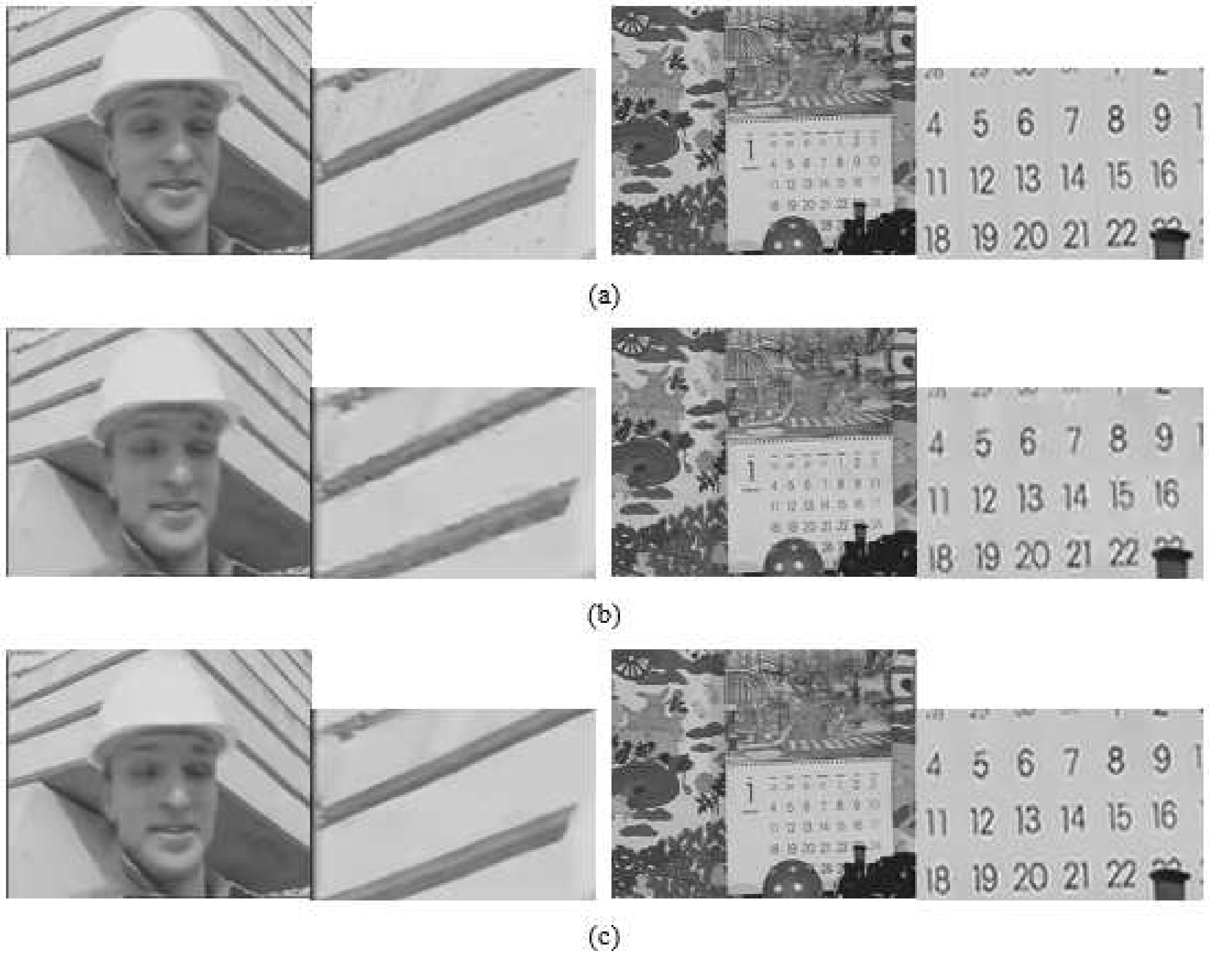

FIGURE 4. Visual comparison of the proposed method. From top to bottom: (a) Original; (b) Compressed (QP=40); (c) Processed frame using our method

TABLE 3. Comparison (PSNR gain) of proposed method for H264/AVC compressed sequences (coded in bitrate-based mode)

Methods	Foreman		Stefan	
	64 Kbit/s	96 Kbit/s	96 Kbit/s	128 Kbit/s
Tais [5] 8*8 filter	0.11	0.05	-0.06	-0.14
Tais [5] 4*4 filter	0.08	-0.02	-0.19	-0.34
Loop filter [17]	0.29	0.40	0.06	0.06
Yehs filter [7]	0.14	0.13	0.04	0.03
Proposed method	0.69	0.68	0.32	0.41

It should be noted that our method is also better than some other methods. But we cannot accurately compare the results because of the codes is unavailable. Such as SR-based method [11], in the case of similar image quality, the PSNR gain of their method for above two sequences are about 0.7dB and 0.4 dB respectively, but the results of our method is about 1.0 dB and 0.7 dB respectively.

In addition, we compare the processing time and performance of two-layer and three-layer model with other parameters is kept consistent in Table.4 and Figure.5 respectively. The three-layer model is generated by adding one hidden layer between the first and second layer of the two-layer model. The filter number and size of the added layer are set to 32 and 1*1 respectively. It is suggests that a reasonably more filters, layers and larger filter size could lead to better results, but the speed will decrease at same time. As the representative, we adopt two-layer networks considering that video processing has high requirement of real time in most cases.

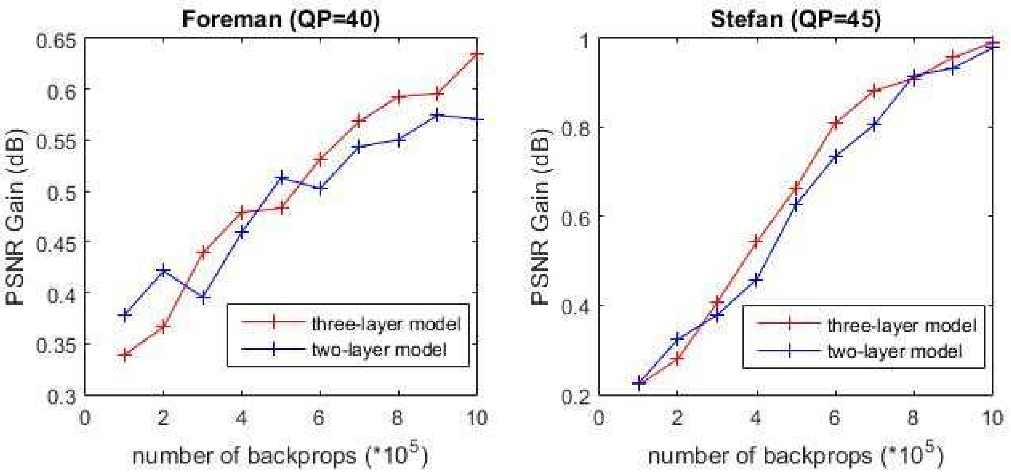

Figure 5. Comparisons (PSNR gain) between the two-layer and three-layer models

Table 4. Time comparision between the two-layer and three-layer models

Sequences	Size	Time(s)	
		Three-layer	Two-layer
Foreman	352*288	3.74	0.37
BasketballDrill	832*480	17.75	1.35
Jonny	1280*720	40.40	2.96

4. **Conclusions.** We have presented a novel deep learning approach for compressed video post-processing. The proposed approach learns an end-to-end mapping between the compressed frames and the original frames. We compare the processing time and performance of two- and three-layer model, and adopt two-layer model because of its substantial time savings. We show the relationship between our method and traditional transform-based method or sparse representation method. Experiments demonstrate that our method has achieved superior performance than some classical methods. We conjecture that additional performance can be further gained by exploring more filters and different training strategies. In addition, due to the limitation of time, the results of some sequences are just temporary and we will improve the results by further iterations.

Acknowledgment. This work was supported in part by Natianal Key R&D Program of China (No. 2016YFB0800404), National Natural Science Foundation of China (No. 61402034, 61210006, 61501379), by Beijing Natural Science Foundation (4154082) and the Fundamental Research Funds for the Central Universities under Grant (2015JBM032).

REFERENCES

[1] R. T. Wang, Y. Zhao, C. Y. Lin, H. H. Bai, and M. Q. Liu, Image Set Compression Based on Undirected Weighted Graph., *Journal of Information Hiding and Multimedia Signal Processing*, vol.6, pp.1053-1061, 2015.

[2] P. List, A. Joch, J. Lainema., G. Bjontegaard, and M. Karczewicz. Adaptive deblocking filter, *IEEE Transactions on Circuits and Systems for Video Technology*, vol.13, no.7, pp.614-619, 2003.

[3] S. C. Tai, Y. Y. Chen, and S. F. sheu, Deblocking filter for low bit rate MPEG-4 video, *IEEE Transactions on Circuits and Systems for Video Technology*, vol.15, no.6, pp.733-741.2005.

[4] J. Kim, C. B. Sim. Compression artifacts removal by signal adaptive weighted sum technique, *IEEE Transactions on Consumer Electronics*, vol.57, no.4, pp.1944-1952, 2011.

[5] S. C. Tai, Y. R. Chen, C. Y. Chen, and Y. H. Chen. Low complexity deblocking method for DCT coded video signals, *Iee Proceedings Vision Image and Signal Processing*, vol.153, no.1, pp.46-56, 2006.

[6] W. Zhang, L. Li, H. Zhu, D. Cheng, S. C. Chu, and J. F. Roddick. No-reference quality metric of blocking artifacts based on orthogonal moments, *Journal of Information Hiding and Multimedia Signal Processing*, vol.5, no.4, pp.701-708, 2014.

[7] C. H. Yeh, S, J. F. Jiang, T. F. Ku, M. J. chen, and J. A. Jhu. Post-processing deblocking filter algorithm for various video decoders, *Image Processing, IET*, vol.6, no.5, pp.534-547, 2012.

[8] T. zcelik, J. C. Brailean, A. K. Katsaggelos. Image and video compression algorithms based on recovery techniques using mean field annealing, in *Proceedings of the IEEE*, vol.83, no.2, pp.304-316, 1995.

[9] H . Paek, R. C. Kim, S. U. Lee. On the POCS-based postprocessing technique to reduce the blocking artifacts in transform coded images, *Circuits and Systems for Video Technology, IEEE Transactions on*, vol.8, no.3, pp.358-367, 1998.

[10] C. Jung, L. Jiao, H. Qi, T. Sun. Image deblocking via sparse representation, *Signal Processing: Image Communication*, vol.27, no.6, pp.663-677, 2012.

[11] C. H. Yeh, L. W. Kang, Y. W. Chiou, C. W. Lin, and S. J. F. Jiang. Self-learning-based post-processing for image/video deblocking via sparse representation, *Journal of Visual Communication and Image Representation*, vol.25, no.5, pp.891-903, 2015.

[12] Y. Wei, W. Xia, J. Huang, B .Ni, J. Dong, Y. Zhao, and S. Yan. CNN: Single-label to Multi-label, *arXiv preprint arXiv*, 1406.5726, 2014.

[13] A. Van Den Oord, S. Dieleman, and B. Schrauwen. (2013). Deep content-based music recommendation, *Neural Information Processing Systems Conference*, vol.26, pp.2643-2651, 2013.

[14] Y. Lcun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, vol.86, no.11, pp.2278-2324, 1998.

[15] V. Jain, S. Seung. Natural image denoising with convolutional networks, *Advances in Neural Information Processing Systems*, pp.769-776, 2009.

[16] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.38, no.2, 2015.

[17] Y. C. Chu, M. J. Chen. High Performance Adaptive Deblocking Filter for H.264, *IEICE - Transactions on Information and Systems*, vol.89, no.1, pp.367-371, 2006.