A Group Variational Transformation Neural Network for Fractional Interpolation of Video Coding

Sifeng Xia, Wenhan Yang, Yueyu Hu, Siwei Ma and Jiaying Liu*

Peking University, Beijing, 100871, China

{xsfatpku,yangwenhan,huyy,swma,liujiaying}@pku.edu.cn

Abstract

Motion compensation is an important technology in video coding to remove the temporal redundancy between coded video frames. In motion compensation, fractional interpolation is used to obtain more reference blocks at sub-pixel level. Existing video coding standards commonly use fixed interpolation filters for fractional interpolation, which are not efficient enough to handle diverse video signals well. In this paper, we design a group variational transformation convolutional neural network (GVTCNN) to improve the fractional interpolation performance of the luma component in motion compensation. GVTCNN infers samples at different sub-pixel positions from the input integer-position sample. It first extracts a shared feature map from the integer-position sample to infer various sub-pixel position samples. Then a group variational transformation technique is used to transform a group of copied shared feature maps to samples at different sub-pixel positions. Experimental results have identified the interpolation efficiency of our GVTCNN. Compared with the interpolation method of High Efficiency Video Coding, our method achieves 1.9% bit saving on average and up to 5.6% bit saving under low-delay P configuration.

1 Introduction

Motion compensation is a significant technology in video coding for the temporal redundancy removal between video frames. Specifically, during inter-prediction, there is at least one reference block to be searched from the previously coded frames for each block to be coded. With the reference block, only the motion vector that indicates the position of the reference block and the residual between the blocks need to be coded, which can bring about bit saving in many cases.

However, due to the spatial sampling of digital video, adjacent pixels in a video frame are not continuous, which means that reference blocks at the integer position may not be similar enough to the block to be coded. In order to search for better reference blocks, video coding standards like High Efficiency Video Coding (HEVC) generate reference samples at sub-pixel positions by performing fractional interpolation over the retrieved integer-position sample.

Interpolation methods adopted by the coding standards usually use fixed interpolation filters. For example, MPEG-4 AVC/H.264 [1] uses a 6-tap filter for half-pixel interpolation and a simple average filter for quarter-pixel interpolation for luma

*Corresponding author

This work was supported by National Natural Science Foundation of China under contract No. 61772043. We also gratefully acknowledge the support of NVIDIA Corporation with the GPU for this research.

component. HEVC uses a DCT-based interpolation filter (DCTIF) [2] for fractional interpolation. It is efficient to adopt simple fixed interpolation filters for motion compensation of video coding in real applications. However, the quality of the interpolation results generated by fixed filters may be limited, since fixed filters can not fit for natural and artificial video signals with various kinds of structures and content.

Recently, many deep learning based methods have been proposed for low-level image processing problems, *e.g.* image interpolation [3], denoising [4, 5], super-resolution [6–9] and these methods have shown impressive results. In [3], Yang *et al.* proposed a variational learning network that effectively exploits the structural similarities for image representation. The deep learning based denoising method [4] utilizes a deep convolutional neural network (CNN) to infer a noise map from the noisy image for denoising. Dong *et al.* proposed a super-resolution method called SRCNN [6], which is the first method that uses CNN for super-resolution and has obtained significant gain over traditional super-resolution methods. In [7], a deeper CNN network with residual learning is built to further improve the super-resolution performance. Besides, edge information is additionally used to guide the inference of high-resolution images in [8]. Hu *et al.* [9] proposed a global context aggregation and local queue jumping network for image super-resolution considering both reconstruction quality and time consumption.

Considering the great performance brought by the deep learning based methods in low-level image processing problems and the high implementation efficiency of deep learning based methods brought by GPU acceleration, it is a new opportunity to utilize deep based interpolation methods in motion compensation for video coding. Yan *et al.* [10] first proposed a CNN-based interpolation filter to replace the half-pixel interpolation part of HEVC. Their method has obtained obvious gain over HEVC, which has also demonstrated the superiority of deep learning based interpolation in video coding. However, in their method, only the half-pixel interpolation is replaced and the quarter-pixel interpolation is still the one of HEVC. Furthermore, they chose to train one model for each half-pixel position, which leads to a high storage cost and sets a barrier to the applications in some extreme conditions.

In this paper, we propose a deep learning based fractional interpolation method to infer reference samples of all sub-pixel positions for motion compensation in HEVC. A group variational transformation convolutional neural network (GVTCNN) is designed to infer samples of various sub-pixel positions with one single network. The network firstly extracts a shared feature map from the integer-position sample. Then a group variational transformation method is used to infer different samples from a group of shared feature maps. Experimental results show the superiority of our GVTCNN in fractional interpolation which further benefits video coding.

The rest of the paper is organized as follows. Sec. 2 introduces the proposed group variational transformation convolutional neural network based fractional interpolation. Details of fractional interpolation in HEVC are first introduced and analyzed. Then the architecture of the network is illustrated and the group variational transformation is described. Process of the generation of training data is also presented. Experimental results are shown in Sec. 3 and concluding remarks are given in Sec. 4.

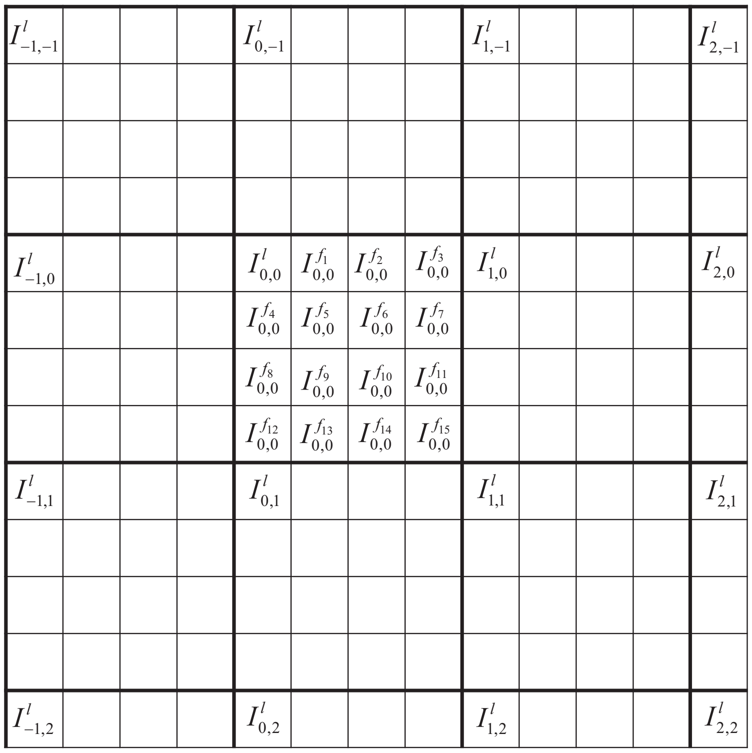

Figure 1: Half-pixel and quarter-pixel sample positions in luma component fractional interpolation.

2 Group Variational Transformation Neural Network Based Fractional Interpolation

2.1 Fractional Interpolation in HEVC

During motion compensation of the luma component in HEVC, at least one integer-position sample in the previously coded frame is first searched for each query block which is to be coded. As shown in Fig. 1, half-pixel position samples I^{f_2}, I^{f_8} and $I^{f_{10}}$ are first interpolated based on the integer-position sample I^l. And quarter-pixel position samples are then interpolated with the values of integer and half-pixel position samples. The most appropriate reference sample is finally selected among the integer, half-pixel position and quarter-pixel position samples to facilitate coding the query block.

HEVC adopts a uniform 8-tap filter for half-pixel interpolation and 7-tap filters for quarter-pixel interpolation. The fixed interpolation filters may not be flexible enough to accomplish all the interpolation tasks of various kinds of video scenes well. Moreover, the interpolation of each sub-pixel only covers a small area of the integer-position sample, which means that only limited reference information is utilized for sub-pixels generation. As a result, the interpolation results of HEVC may not be good enough in some hard cases like the scenes with complex structures.

Deep convolutional neural network based methods have obtained much success in such kinds of low-level image processing problems. With the help of training data,

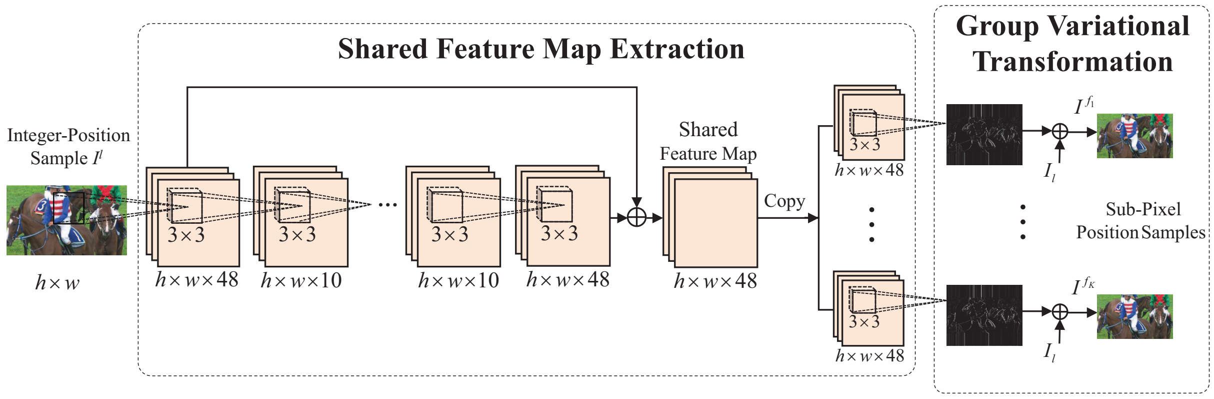
Shared Feature Map Extraction
Group Variational Transformation
Integer-Position Sample I^l
$h \times w$
3×3
$h \times w \times 48$
$h \times w \times 10$
Shared Feature Map
Copy
I^{f_1}
I^{f_K}
I_l
Sub-Pixel Position Samples

Figure 2: Framework of the proposed group variational transformation convolutional neural network (GVTCNN).

deep CNN based methods learn a mapping from the input signal x to the target result y' by:

$$y' = f(x, \Theta), \tag{1}$$

where Θ represents the set of learnt parameters of the convolutional neural network, which are learnt based on the training data with the back-propagation algorithm.

2.2 Group Variational Transformation Neural Network

The proposed GVTCNN consists of two components: the shared feature map extraction part and the group variational transformation part. The shared feature map is first extracted by GVTCNN from the integer-position sample I^l and group variational transformation infers the residual maps of the samples at different sub-pixel positions based on the shared feature map.

Fig. 2 shows the architecture of GVTCNN. The integer-position sample I^l is the input of the network. $h \times w \times c$ represents the size of each convolutional layer, where h and w are respectively the height and the width of the feature map, and c is the channel number of the feature map. 3×3 indicates the size of the convolutional kernal. In this paper, we use 3×3 kernel size for all the convolutional layers. The parametric rectified linear units (PReLU) [11] are utilized for nonlinearity between the convolutional layers.

Specifically, we define f_k^{out} to be the output of the k-th convolutional layer. f_k^{out} is obtained by:

$$f_k^{out} = P_k\left(W_k * f_{k-1}^{out} + B_k\right), \tag{2}$$

where f_{k-1}^{out} is the output of the previous layer, W_k is the convolutional filter kernel of the k-th layer and B_k is the bias of the k-th layer. f_0^{out} is the input integer-position sample. The function $P_k(\cdot)$ is the PReLU function of the k-th layer:

$$P_k(x) = \begin{cases} x, & x > 0, \\ a_k * x, & x \le 0. \end{cases} \tag{3}$$

x is the input signal and a_k is the parameter to be learned for the k-th layer. a_k is initially set as 0.25 and all channels of the k-th layer share the same parameter a_k.

In the shared feature map extraction component, a feature map with 48 channels is initially generated from the integer-position sample, followed by 8 convolution layers with 10 channels which are lightweight and cost less to save the learnt parameters. The 10-th layer later derives a 48 channel feature map. The residual learning technique is utilized in shared feature map extraction for accelerating the convergency of the network. So that we add the 1-st layer to the 10-th layer and then activate the sum with PReLU function to obtain the shared feature map. After 9 convolutional layers with 3×3 kernel size, the receptive field of each point in the shared feature map is 19×19, which means that a large nearby area in the integer-pixel position sample has been considered for the feature extraction of each pixel.

Considering the spatial correlation and continuity of the sub-pixels, we argue that there is no need to separately extract a feature map for the generation of each sub-pixel position sample. In other words, we do not need to train a network for each sub-pixel position sample, which is inconvenient for real applications. As a result, after the shared feature map extraction, the shared feature map is used to infer the sub-pixel samples at different locations. The group variational transformation is further performed over the shared feature maps with a specific convolutional layer for each sub-pixel sample. Different residual maps are then generated and we obtain the final inferred sub-pixel position samples by adding the residual maps to the integer-position sample.

During training process, mean square error is used as the loss function. Let $F(\cdot)$ represent the learnt network that infers sub-pixel position samples from the integer-position sample and Θ denote the set of all the learnt parameters including the convolutional filter kernels, bias and a_k of the PReLU function in each layer. The loss function can be formulated as follows:

$$L(\Theta) = \frac{1}{n} \sum_{i=1}^{n} \left\| F(x_i, \Theta) - y_i \right\|^2, \tag{4}$$

where pairs $\{x_i, y_i\}_{i=1}^{n}$ are the generated ground-truth pairs of integer-position and sub-pixel position samples and n is the total number of the pairs.

2.3 Training Data Generation for Motion Compensation

In the deep CNN based interpolation and super-resolution methods, the ground truth high-resolution images are directly used as the label for the loss function and the down-sampled images are used as the input. However, there are big differences between image resolution recovery problems and fractional interpolation in video coding.

Firstly, methods of interpolation and super-resolution recover a high-resolution image from the low-resolution image, where ground truth high-resolution images exist. The resolution recovery quality can be measured by simply calculating the d-ifferences between the recovered images and the ground truth images. Fractional interpolation in video coding differently aims to generate more sub-pixel position

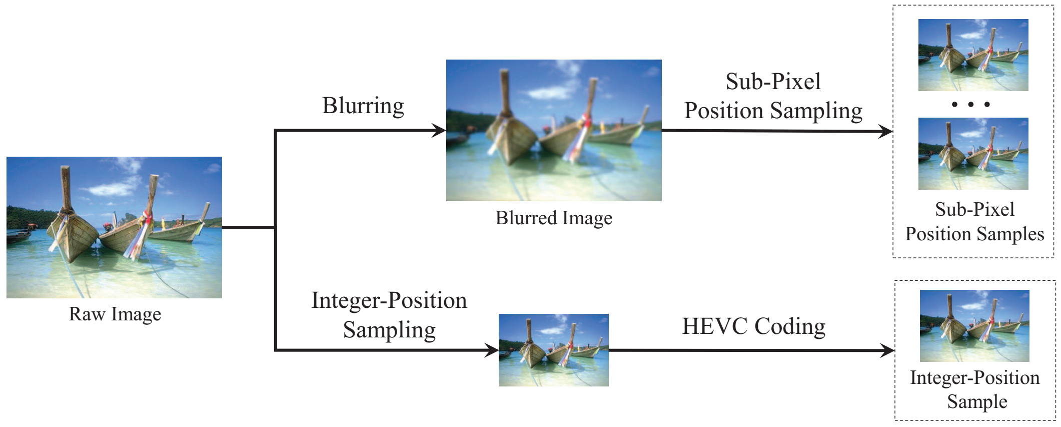
Blurring
Sub-Pixel Position Sampling
Blurred Image
Sub-Pixel Position Samples
Raw Image
Integer-Position Sampling
HEVC Coding
Integer-Position Sample

Figure 3: Flow chart of the training data generation for GVTCNN.

samples for motion compensation. And the efficiency of sub-pixel position samples generation is measured by final coding performance. Secondly, fractional interpolation is performed over the reconstructed previously coded reference frame, therefore the additional information loss brought by the reconstruction in video coding should also be considered in training data generation. Besides, in many video coding s-tandards, half-pixel position interpolation and quarter-pixel position interpolation are performed separately, which is reasonable since these two kinds of interpolation provide reference samples at different sub-pixel levels.

As a result, we correspondingly take several measures in training data generation to make the trained network applied to fractional interpolation in video coding. The overall flow chart of the process of training data generation has been shown in Fig. 3. Referring to the method mentioned in [10], the training data is first blurred with a gaussian filter to simulate the correlations between the integer-position sample and sub-pixel position samples. Sub-pixel position samples are later sampled from the blurred image. As for the input integer-position sample generation, an intermediate integer sample is previously down-sampled from the raw image. Then, the intermediate down-sampled version is coded by HEVC video coding to simulate the information loss of the reconstructed reference sample.

Moreover, we train two networks separately for 3 half-pixel position samples and 12 quarter-pixel position samples to better generate the samples at different sub-pixel levels. And there are some differences between the training data generation of the network that infers half-pixel position samples and the network that infers quarter-pixel position samples (respectively called GVTCNN-H and GVTCNN-Q). In the process of training data generation for GVTCNN-H, 200 training images and 200 testing images in the set *BSDS500* [12] at size 481×321 and 321×481 are used for training. 3×3 Gaussian kernels with random standard deviations in the range $[0.5, 0.6]$ are used for blurring. By dividing the images into 2×2 patches without overlapping, pixels at the top-left of the patches in the raw images are sampled to obtain the integer-position sample. And pixels at other three positions of the

Table 1: BD-rate reduction of the proposed method compared to HEVC.

Class	Sequence	BD-rate		
		Y	U	V
Class B	Kimono	-3.7%	0.9%	1.3%
	BQTerrace	-5.6%	-4.2%	-5.5%
	BasketballDrive	-3.3%	-0.9%	-1.1%
	ParkScene	-1.1%	0.0%	-0.7%
	Cactus	-2.2%	-0.6%	-0.9%
	Average	-3.2%	-1.0%	-1.4%
Class C	BasketballDrill	-2.3%	-1.4%	-0.7%
	BQMall	-2.8%	-1.3%	-1.2%
	PartyScene	-0.6%	0.0%	-0.6%
	RaceHorsesC	-1.9%	-0.7%	-1.6%
	Average	-1.9%	-0.8%	-1.0%
Class D	BasketballPass	-3.1%	-1.2%	-1.6%
	BlowingBubbles	-1.6%	-0.2%	-0.5%
	BQSquare	1.6%	1.7%	2.7%
	RaceHorses	-1.8%	-1.7%	-1.6%
	Average	-1.2%	-0.3%	-0.2%
Class E	FourPeople	-2.2%	-0.9%	-0.7%
	Johnny	-2.7%	-0.4%	0.7%
	KristenAndSara	-2.1%	0.5%	0.4%
	Average	-2.3%	-0.3%	0.1%
Class F	BasketballDrillText	-1.7%	-0.6%	-0.3%
	ChinaSpeed	-1.4%	-1.9%	-1.6%
	SlideEditing	0.2%	0.1%	0.0%
	SlideShow	-0.1%	0.2%	-0.6%
	Average	-0.8%	-0.6%	-0.6%
All Sequences	Overall	-1.9%	-0.6%	-0.7%

patches are separately sampled from the blurred image to derive the sub-pixel position samples.

For GVTCNN-Q, the inferred samples are at a smaller sub-pixel level. The sampling will be performed based on 4×4 patches, where 12 samples are extracted from pixels at 1/4 or 3/4 positions vertically or horizontally in the patch. It is not appropriate to continue using images at relatively low resolution for training data generation here, since differences between the integer samples and sub-pixel samples may be a little big with a sampling step length of 4, which may affect the training of GVTCNN-Q. So images at larger size are chosen to generate the training data to

Table 2: BD-rate reduction of CNNIF and the proposed GVTCNN-H for different classes.

Class	Sequence	CNNIF[10]			GVTCNN-H		
		Y	U	V	Y	U	V
Class C	BasketballDrill	-1.2%	-0.6%	0.2%	-1.9%	-1.3%	-0.4%
	BQMall	-0.9%	0.3%	0.7%	-2.0%	-0.8%	-0.9%
	PartyScene	0.2%	0.5%	0.3%	-0.3%	-0.1%	-0.1%
	RaceHorsesC	-1.5%	-0.5%	-0.1%	-1.6%	-1.0%	-0.2%
	Average	-0.9%	-0.1%	0.3%	-1.4%	-0.8%	-0.4%
Class D	BasketballPass	-1.3%	-0.4%	0.3%	-2.4%	-1.2%	-0.7%
	BlowingBubbles	-0.3%	0.4%	0.8%	-0.9%	0.9%	-0.5%
	BQSquare	1.2%	2.9%	3.1%	1.9%	2.0%	3.7%
	RaceHorses	-0.8%	-0.9%	0.0%	-1.1%	-0.9%	-0.2%
	Average	-0.3%	0.5%	1.0%	-0.6%	0.2%	0.6%
Class E	FourPeople	-1.3%	-0.4%	0.1%	-2.1%	-0.5%	-0.3%
	Johnny	-1.2%	-0.4%	-0.7%	-2.7%	-1.1%	-0.6%
	KristenAndSara	-1.0%	0.3%	0.2%	-2.3%	0.1%	0.1%
	Average	-1.2%	-0.2%	-0.1%	-2.4%	-0.5%	-0.3%

keep the continuity and similarity between the input integer samples and the target sub-pixel position samples. 10 YUV sequences at size 1024×768 and 1920×1080 are used to extract 89 high resolution frames to synthesize training data[1]. Standard deviations of Gaussian kernels ranges from $[0.7, 0.8]$.

3 Experimental Results

3.1 Experimental Settings

During the training process, the training images are decomposed into 32×32 sub-images with a stride of 16. The GVTCNN is trained on Caffe platform [13] via Adam [14] with standard back-propagation. The learning rate is initially set as a fixed value 0.0001 and dropped after $30,000$ iterations by a factor of 10. The batch size is set as 128. Models after $50,000$ iterations are used for testing. The network is trained on one Titan X GPU.

The proposed method is tested on HEVC reference software HM 16.15 under the low delay P (LDP) configuration. BD-rate is used to measure the rate-distortion. The quantization parameter (QP) values are set to be 22, 27, 32 and 37. We also compare with the CNN based half-pixel interpolation method proposed in [10]. During the training data generation, each intermediate integer sample is coded by four QPs: 22, 27, 32 and 37. And we train a GVTCNN-H and GVTCNN-Q for each QP based on the corresponding training data. For other QPs, models belong to their nearest QP are

[1]`http://media.xiph.org/video/derf/`

chosen. A CU level rate-distortion optimization is also integrated to decide whether to replace our deep based interpolation method with the interpolation method of HEVC.

Table 3: Average BD-rate reduction achieved by training the networks separately and uniformly.

Class	GVTCNN-Separate			GVTCNN-Uniform		
	Y	U	V	Y	U	V
Class C	-1.9%	-1.0%	-1.2%	-1.9%	-0.8%	-1.0%
Class D	-1.3%	-0.1%	-0.2%	-1.2%	-0.3%	-0.2%
Class E	-2.4%	-0.1%	-0.4%	-2.3%	-0.3%	0.1%

3.2 Experimental Results and Analysis

Performance of the proposed deep learning based interpolation method in video coding for classes B, C, D, E and F is shown in Table 1. Our method has obtained on average a 1.9% BD-rate saving and up to 5.6% BD-rate saving for the test sequence *BQTerrace*.

We also compare our method with the CNN based half-pixel interpolation method proposed in [10] (called CNNIF). CNNIF only replaces the half-pixel interpolation without rate-distortion optimization and is tested on HM 16.7. For fair comparison, we also test our method on HM 16.7 and only replace the half-pixel interpolation with our GVTCNN-H without rate-distortion optimization. The BD-rate reduction of the two methods for several testing classes are shown in Table 2. Our method still has gain over CNNIF. And the gain will be larger after integrating the GVTCNN-Q and rate-distortion optimization.

In order to further identify the effectiveness of our group variational transformation method, we additionally train the networks for each sub-pixel position at each QP separately. Average BD-rate reduction for some testing classes are shown in Table 3. As can be observed, results of training the networks separately and uniformly are comparable.

4 Conclusion

In this paper, we propose a group variational transformation deep convolutional neural network for fractional interpolation in motion compensation of video coding. The network first uniformly extracts a shared feature map from the input integer-position sample and then a group of copied shared feature maps are transformed to samples at various sub-pixel positions. Training data generation of the proposed network is also carefully analyzed and designed. Experimental results show that our method has obtained on average a 1.9% BD-rate saving on the test sequences compared with HEVC. Effectiveness of the group variational transformation method adopted in our network is also well identified by experiments.

References

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H. 264/AVC video coding standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[2] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, “Overview of the high efficiency video coding (HEVC) standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[3] W. Yang, J. Liu, S. Xia, and Z. Guo, “Variation learning guided convolutional network for image interpolation,” in *Proc. IEEE Int’l Conf. Image Processing*, 2017.

[4] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, “Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising,” *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[5] K. Zhang, W. Zuo, S. Gu, and L. Zhang, “Learning deep CNN denoiser prior for image restoration,” in *Proc. IEEE Int’l Conf. Computer Vision and Pattern Recognition*, 2017.

[6] C. Dong, C. C. Loy, K. He, and X. Tang, “Learning a deep convolutional network for image super-resolution,” in *Proc. European Conf. Computer Vision*, 2014.

[7] J. Kim, J. Kwon Lee, and K. Mu Lee, “Accurate image super-resolution using very deep convolutional networks,” in *Proc. IEEE Int’l Conf. Computer Vision and Pattern Recognition*, 2016.

[8] W. Yang, J. Feng, J. Yang, F. Zhao, J. Liu, Z. Guo, and S. Yan, “Deep edge guided recurrent residual learning for image super-resolution,” *IEEE Transactions on Image Processing*, vol. 26, no. 12, pp. 5895–5907, 2017.

[9] Y. Hu, J. Liu, W. Yang, S. Deng, L. Zhang, and Z. Guo, “Real-time deep image super-resolution via global context aggregation and local queue jumping,” in *Proc. IEEE Visual Communication and Image Processing*, 2017.

[10] N. Yan, D. Liu, H. Li, and F. Wu, “A convolutional neural network approach for half-pel interpolation in video coding,” in *Proc. IEEE Int’l Symposium on Circuits and Systems*, 2017.

[11] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proc. IEEE Int’l Conf. Computer Vision*, 2015.

[12] D. Martin, C. Fowlkes, D. Tal, and J. Malik, “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics,” in *Proc. IEEE Int’l Conf. Computer Vision*, 2001.

[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in *Proc. ACM Int’l Conf. Multimedia*, 2014.

[14] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. Int’l Conf. Learning Representations*, 2015.