One-for-All: Grouped Variation Network-Based Fractional Interpolation in Video Coding

Jiaying Liu, *Senior Member, IEEE*, Sifeng Xia, Wenhan Yang, *Member, IEEE*, Mading Li, and Dong Liu, *Member, IEEE*

***Abstract*—Fractional interpolation is used to provide sub-pixel level references for motion compensation in the interprediction of video coding, which attempts to remove temporal redundancy in video sequences. Traditional handcrafted fractional interpolation filters face the challenge of modeling discontinuous regions in videos, while existing deep learning-based methods are either designed for a single quantization parameter (QP), only generating half-pixel samples, or need to train a model for each sub-pixel position. In this paper, we present a one-for-all fractional interpolation method based on a grouped variation convolutional neural network (GVCNN). Our method can deal with video frames coded using different QPs and is capable of generating all sub-pixel positions at one sub-pixel level. Also, by predicting variations between integer-position pixels and sub-pixels, our network offers more expressive power. Moreover, we perform specific measurements in training data generation to simulate practical situations in video coding, including blurring the down-sampled sub-pixel samples to avoid aliasing effects and coding integer pixels to simulate reconstruction errors. In addition, we analyze the impact of the size of blur kernels theoretically. Experimental results verify the efficiency of GVCNN. Compared with HEVC, our method achieves 2.2% in bit saving on average and up to 5.2% under low-delay P configuration.**

***Index Terms*—High efficient video coding (HEVC), fractional interpolation, convolutional neural network (CNN), grouped variation network.**

I. Introduction

ONE of the most important features in video coding is to remove temporal redundancy in video sequences, which can be achieved by motion compensation. Specifically, during the inter prediction, reference blocks are searched from preceding coded frames. With these blocks, only corresponding motion vectors and residuals between reference blocks and the current block need to be coded. Generally, encoding motion vectors and residuals are more efficient than encoding the original block, which leads to distinct bit saving in video coding.

However, due to the spatial sampling of digital video, the locations of adjacent pixels in a video frame are not continuous. The reference blocks at integer positions may have sub-pixel motion shifts to the current block. In order to produce better reference blocks, video coding standards, *e.g.* High Efficiency Video Coding (HEVC), generate sub-pixel reference blocks by fractional interpolation over the retrieved integer-position blocks.

Video coding standards adopt fixed interpolation filters to perform fractional interpolations. For example, MPEG-4/H.264 AVC [1] uses a 6-tap filter for half-pixel interpolation and a simple average filter for quarter-pixel interpolation of luma component. HEVC adopts a DCT-based interpolation filter (DCTIF) [2] for fractional interpolation. These simple fixed interpolation filters are effective for motion compensation. However, the quality of 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 contents.

In the last decade, researchers have been dedicated on improving the design of traditional sub-pixel interpolation filters [1]–[5]. However, these handcrafted filters have limited expressiveness, facing the challenge of various and abundant local contexts in large amount of videos being generated everyday. Moreover, these filters always assume videos to be locally smooth and perform linear interpolations. In fact, videos contain many discontinuous contents such as sharp edges and local details, which are difficult to model using handcrafted filters.

Recently, many deep learning based methods have been proposed for image processing problems, *e.g.* denoising [6], [7], inpainting [8], and super-resolution [9], [10], demonstrating promising abilities and generating impressive results. Considering the great performance brought by the deep learning based methods in 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 learning based interpolation methods in motion compensation for video coding. Yan *et al.* [11] first proposed a CNN-based interpolation filter to replace the half-pixel interpolation part of HEVC. Following that work, Zhang *et al.* [12] adopted a successful network VDSR [10] in image super-resolution problem to improve the half-pixel interpolation in HEVC. Although performance gain can be observed in both methods, they have some drawbacks. First of all, these methods are QP-dependent, which means they have to train one model

Manuscript received April 10, 2018; revised September 2, 2018; accepted November 15, 2018. Date of publication November 22, 2018; date of current version January 16, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61772043 and Grant 61390512, in part by the Peking University C Tencent Rhino Bird Innovation Fund, and in part by the NVIDIA Corporation with the GPU. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Xin Li. *(Corresponding author: Wenhan Yang.)*

J. Liu, S. Xia, W. Yang, and M. Li are with the Institute of Computer Science and Technology, Peking University, Beijing 100871, China (e-mail: liujiaying@pku.edu.cn; xsfatpku@pku.edu.cn; yangwenhan@pku.edu.cn; martinli0822@pku.edu.cn).

D. Liu is with the CAS Key Laboratory of Technology in Geo-Spatial Information Processing and Application Systems, University of Science and Technology of China, Hefei 230027, China (e-mail: dongeliu@ustc.edu.cn).

Digital Object Identifier 10.1109/TIP.2018.2882923

for each QP. Moreover, the method in [11] need to train one model for each half-pixel position (*i.e.* three models for half-pixel positions). Furthermore, they do not support fractional interpolation for quarter-pixel positions. These factors lead to higher storage costs and limit the practicability of the methods.

In this paper, we propose a deep learning based fractional interpolation method to infer sub-pixels for motion compensation in HEVC. A grouped variation convolutional neural network (GVCNN) is designed to predict sub-pixels in different sub-pixel positions under different QPs at once. The network first extracts feature maps from integer-position samples, and then infers variations for different sub-pixels using the same feature maps. In order to make our network more adaptive to practical video coding, we elaborate several measurements to generate training data simulating practical situations. Moreover, we provide theoretical analysis to generate the most appropriate and effective training data. Experimental results show the superiority of our GVCNN in fractional interpolation which further benefits video coding. Our contributions are summarized as follows:

- We propose a grouped variation based neural network to predict variations between half-/quarter-pixels and integer-position pixels. Comparing to learning the mapping between pixel values, learning the variations makes the network easier to train and more expressive.
- Instead of training one model for each QP/sub-pixel position, we present a one-for-all model to support different QPs and generate all sub-pixels at one sub-pixel level (three half-pixel positions or twelve quarter-pixel positions) using a single model.
- In order to simulate practical situations in video coding, we present two operations to generate training data for our network. Specifically, we apply blurring operations to sub-pixels to prevent aliasing effect and encode integer pixels by HEVC to simulate reconstruction error. Furthermore, a theoretical analysis is provided to determine the best size of blur kernels.

The rest of the paper is organized as follows. Sec. II introduces the proposed GVCNN based fractional interpolation. In Sec. III, details of fractional interpolation in HEVC are first introduced and analyzed, and then the architecture of the network is illustrated and the grouped variation is described. Process of the generation of training data is also presented in Sec. IV. Experimental results are shown in Sec. V and concluding remarks are given in Sec. VI.

II. Related Works

A. Traditional Sub-Pixel Interpolation

Sub-pixel interpolations are performed by fixed filters in most video coding standards. MPEG-4 AVC/H.264 [1] adopts a 6-tap filter to interpolate pixels at half-pixel positions, and simply averages interpolated pixels at integer and half-pixel positions to calculate samples at quarter-pixel positions. Audio and Video Coding Standard (AVS) [3] employs a Two-Step Four-Tap (TSFT) interpolation to perform quarter-pixel interpolations, which utilizes a 4-tap cubic convolution filter and a 4-tap cubic B-spline filter. In HEVC [2], half-pixels and quarter-pixels are calculated by an 8-tap filter and a 7-tap filter respectively, with coefficients derived from DCT basis function equations.

These fixed interpolation filters are easy to be implemented and based on the assumptions that video signal is spatially and temporally continuous. However, the assumption does not always hold true in practice. Therefore, further researches have been conducted to perform better sub-pixel interpolations. Lakshman *et al.* [4] proposed a generalized interpolation by combining Infinite Impulse Response (IIR) and Finite Impulse Response (FIR) filters, which provides a greater degree of freedom for selecting basis functions. Kokhazadeh *et al.* [5] attempted to interpolate considering the edges of video objects. To interpolate pixels around edges, they first extrapolated pixels in the dissident side based on pixels in the accordant side and then interpolated half-pixel samples using extrapolated pixels and pixels in the accordant side. Still, these hand-crafted filters cannot provide sufficient capability when facing various video contents.

B. Deep Learning Based Video Coding

Deep learning based methods have shown significant performance in both computer vision applications and image/video processing tasks. Deep learning is one of the efficient ways to solve the problems encountered in developing cutting-edge video coding standards. For instance, deep learning models are able to learn complex patterns in various videos automatically, without the need of manually elaborate designs; multiple layers and activation functions are capable of modeling non-linear transformations, which are very common in video coding.

In recent years, many deep learning based methods emerge from video coding community. Intuitively, CNN can be used in post-processing of decoded video sequences. Lin *et al.* [13] presented an end-to-end mapping from decompressed frames to enhanced versions, attempting to approximate the reverse function of video compression. Dai *et al.* [14] proposed a Variable-filter-size Residue learning CNN (VRCNN) that learns the residue between the decoded frame and the original one. Wang *et al.* [15] further presented a deep CNN-based auto decoder (DCAD) to automatically remove artifacts and enhance the details of HEVC-compressed videos.

Besides utilizing CNN as a post-processing procedure, some researchers investigate the practicability of applying CNN during the video coding process. Park and Kim [16] replaced the Sample Adaptive Offset (SAO) filter in HEVC by their CNN architecture. In [17], Laude and Ostermann proposed to replace the conventional Rate Distortion Optimization (RDO) for the intra prediction mode with CNN. Coding unit (CU) partition mode decision can also be predicted by CNN [18] and Long and Short-Term Memory (LSTM) network [19]. As mentioned in the introduction, Yan *et al.* [11] proposed to replace the half-pixel interpolation of HEVC by the network architecture of SRCNN [9], achieving better results compared with the original HEVC. And similarly Zhang *et al.* [12] replace the half-pixel interpolation by the network architecture of VDSR [10]. In our work, we also focus on improving the accuracy of sub-pixel interpolation using CNN. We propose a

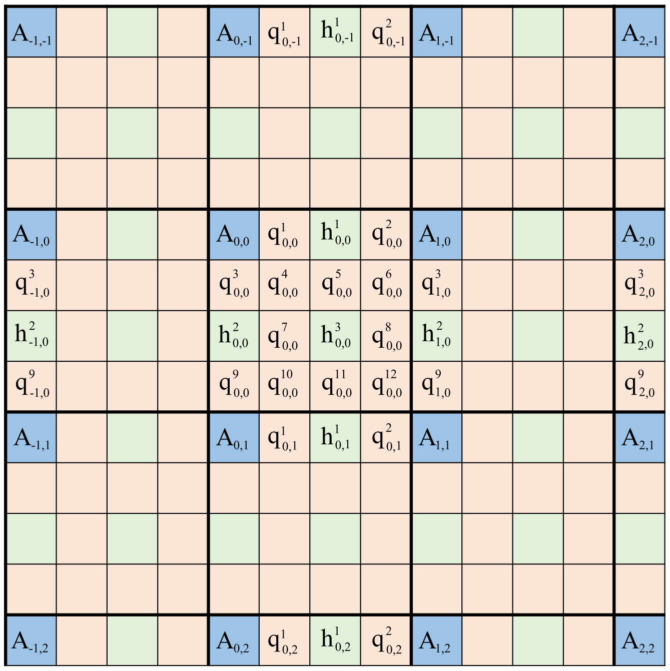

$A_{-1,-1}$
$A_{0,-1}$ $q^1_{0,-1}$ $h^1_{0,-1}$ $q^2_{0,-1}$ $A_{1,-1}$
$A_{2,-1}$
$A_{-1,0}$ $A_{0,0}$ $q^1_{0,0}$ $h^1_{0,0}$ $q^2_{0,0}$ $A_{1,0}$ $A_{2,0}$
$q^3_{-1,0}$ $q^3_{0,0}$ $q^4_{0,0}$ $q^5_{0,0}$ $q^6_{0,0}$ $q^3_{1,0}$ $q^3_{2,0}$
$h^2_{-1,0}$ $h^2_{0,0}$ $q^7_{0,0}$ $h^3_{0,0}$ $q^8_{0,0}$ $h^2_{1,0}$ $h^2_{2,0}$
$q^9_{-1,0}$ $q^9_{0,0}$ $q^{10}_{0,0}$ $q^{11}_{0,0}$ $q^{12}_{0,0}$ $q^9_{1,0}$ $q^9_{2,0}$
$A_{-1,1}$ $A_{0,1}$ $q^1_{0,1}$ $h^1_{0,1}$ $q^2_{0,1}$ $A_{1,1}$ $A_{2,1}$
$A_{-1,2}$ $A_{0,2}$ $q^1_{0,2}$ $h^1_{0,2}$ $q^2_{0,2}$ $A_{1,2}$ $A_{2,2}$

Fig. 1. Positions of different fractional pixels. Blue, green and pink blocks indicate respectively the integer- ($\mathrm{A_{i,j}}$), half- ($\mathrm{h^1_{i,j}}, \mathrm{h^2_{i,j}}, \ldots, \mathrm{h^3_{i,j}}$) and quarter- ($\mathrm{q^1_{i,j}}, \mathrm{q^2_{i,j}}, \ldots, \mathrm{q^{12}_{i,j}}$) pixel positions for luma interpolation.

one-for-all fractional interpolation method based on grouped variation convolutional neural network, which generates interpolations for all half-pixel (or quarter-pixel) positions in a single inference pass and only needs to be trained for once for each sub-level.

III. Grouped Variation Neural Network Based Fractional Interpolation

A. Fractional Interpolation in HEVC

During the motion compensation in HEVC, blocks from preceding coded frames will be used as reference blocks. Fractional interpolation aims to generate sub-pixel position samples based on these reference blocks. Fig. 1 illustrates positions of integer pixels and sub-pixels. To be more specific, positions indicated by $\mathrm{A_{i,j}}$ represent integer samples; $\mathrm{h^k_{i,j}}(\mathrm{k} \in \{1, 2, 3\})$ and $\mathrm{q^k_{i,j}}(\mathrm{k} \in \{1, 2, \cdots, 12\})$ denote half-pixel positions and quarter-pixel positions, respectively. Given a reference block I^{A}, whose pixels are regarded as integer samples ($\mathrm{A_{i,j}}$), the half-pixel blocks $I^{\mathrm{h^k}}$ and quarter-pixel blocks $I^{\mathrm{q^k}}$ are interpolated from I^{A}. With these sub-pixels interpolated, the most similar reference sample is finally selected among integer and sub-pixel position samples to facilitate coding the current block.

HEVC adopts a uniform 8-tap filter for half-pixel interpolation and a 7-tap filter for quarter-pixel interpolation. These fixed interpolation filters may not be flexible enough to accomplish the interpolation of videos that contain diversified scenes and contents. Moreover, these interpolations only consider the information from a very small neighborhood, and cannot make an accurate prediction when facing signals with complex structures.

B. Learning Grouped Variations

Learning-based interpolation methods [11], [20], [21] aim to estimate the mapping $f(\cdot)$ between an input low-resolution (LR) image $\mathbf{y}$ and its corresponding high-resolution (HR) image $\mathbf{x}$:

$$\mathbf{x} = f(\mathbf{y}). \tag{1}$$

These approaches that directly model the mapping between $\mathbf{y}$ and $\mathbf{x}$ usually suffer from several drawbacks: 1) the learned model over-fits to the regularity between low-frequency parts of image signals and high-frequency details are neglected; 2) the priors are imposed on the whole $\mathbf{x}$, thus the method is hard to learn useful guidance for recovering the high frequency image signal; 3) useful structural correspondences in the high frequency domain may be neglected when directly modeling $\mathbf{x}$.

Our method also attempts to estimate a mapping function, except that the mapping estimated is between integer pixels I^{A} and sub-pixel samples $I^{\mathrm{h^k}}$ (or $I^{\mathrm{q^k}}$). On the other hand, we propose a novel image model, *i.e.* grouped variation image representation, to overcome the aforementioned drawbacks.

Motivated by wavelet transformation [22], [23] and variation image statistics [24]–[26], we make use of the local redundancy to remove the auto-correlation of I^{A} and $I^{\mathrm{h^k}}$ (or $I^{\mathrm{q^k}}$) as well as their correlation, and to construct more useful structural correspondences between adjacent pixels. Specifically, a pixel can be represented by the summation of one of its nearest neighbors and a very small difference value, *i.e.* the variation. This operation removes much of auto-correlation within adjacent pixels. For a reference block I^{A}, the variations between its pixels and sub-pixels are demonstrated below:

$$\Delta I^{\mathrm{h^k}} = I^{\mathrm{h^k}} - I^{\mathrm{A}}, \quad \mathrm{k} \in \{1, 2, 3\}, \tag{2}$$

where $\Delta I^{\mathrm{h^k}}$ denotes the variations of half-pixels.

Similarly, the variations of quarter-pixels are constructed as:

$$\Delta I^{\mathrm{q^k}} = I^{\mathrm{q^k}} - I^{\mathrm{A}}, \quad \mathrm{k} \in \{1, 2, \cdots, 12\}. \tag{3}$$

Thus, in our work, we focus on estimating the variations $\Delta I^{\mathrm{h^k}}$ and $\Delta I^{\mathrm{q^k}}$, which naturally leads to a modified mapping:

$$\begin{aligned} I^{\mathrm{h^k}} - I^{\mathrm{A}} &= f_h(I^{\mathrm{A}}), \quad \mathrm{k} \in \{1, 2, 3\}, \\ I^{\mathrm{q^k}} - I^{\mathrm{A}} &= f_q(I^{\mathrm{A}}), \quad \mathrm{k} \in \{1, 2, \cdots, 12\}, \end{aligned} \tag{4}$$

where $f_h(\cdot)$ and $f_q(\cdot)$ represent the learned mapping between integer pixels and grouped variations of half- and quarter-pixel positions, respectively.

This image representation (4) has three major advantages compared with (1):

- *Fitting to high frequency image signals*. According to the local dependency, low frequency part is removed, and the regularity between the high frequency is focused on.
- *Direct priors*. The priors are imposed on the missing part of the reference block I^{A}, which benefits learning useful guidance for recovering the high frequency part of the image signal.
- *Plenty of structural correspondences*. Compared with an image signal, the variation signal significantly increases

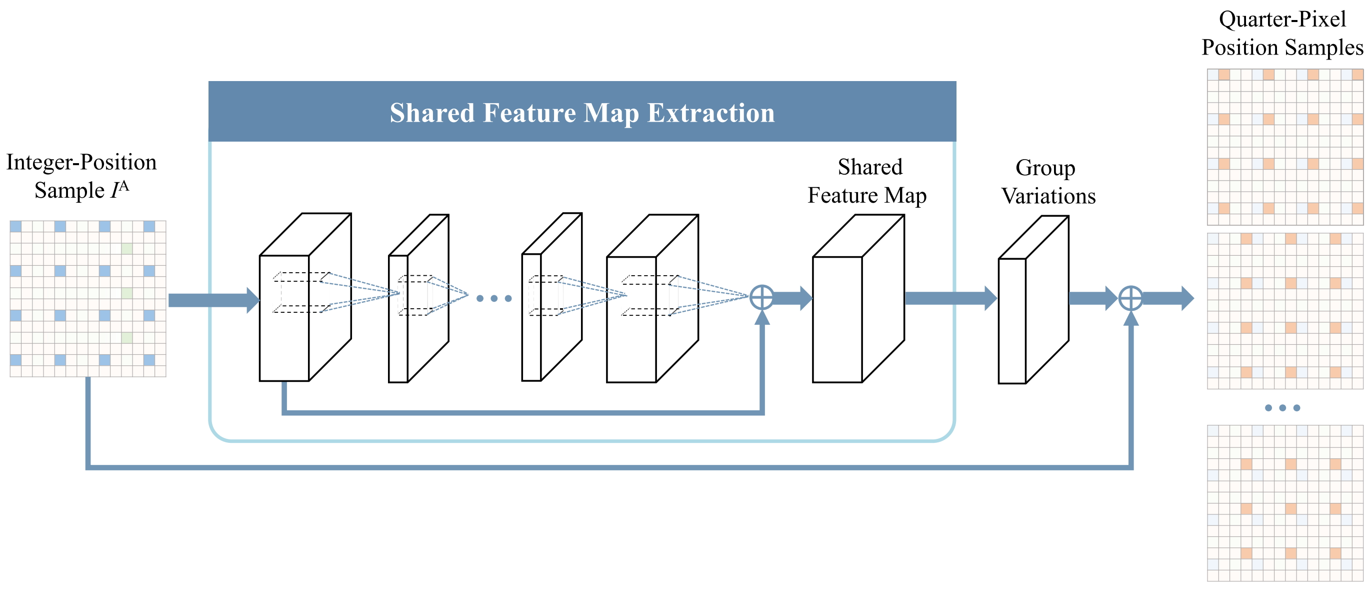

Shared Feature Map Extraction
Integer-Position Sample I^A
Shared Feature Map
Group Variations
Quarter-Pixel Position Samples

Fig. 2. Framework of the proposed GVCNN. The network first extracts feature maps from the integer-position sample. Then the group variations that identify the differences between different sub-pixel position samples and the integer-position sample are inferred using the same feature maps. Final results of sub-pixel position samples are naturally obtained by adding the variations back to the integer-position sample.

the patch repetitiveness, which benefits the inference of missing details in the restoration process.

In summary, for fractional interpolation, compared with predicting sub-pixel values, it is more simple to predict sub-pixel variations based on integer pixels. Thus, the model training for learning-based methods becomes easier than directly end-to-end learning from integer pixels to sub-pixels. Therefore, the modeling capacity of a model also becomes larger.

C. Grouped Variation Neural Network

As discussed in Sec. II, deep convolutional neural network based methods have obtained much success in different kinds of low-level image processing problems. With the help of large-scale training data, deep CNN based methods attempt to learn a mapping from the input signal x to the target result y' as follows:

$$y' = f(x, \Theta), \tag{5}$$

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

In this subsection, we introduce the proposed one-for-all fractional interpolation method based on grouped variation convolutional neural network (GVCNN). It first extracts a feature map from integer-position samples I^{A} and then infers residual maps of different sub-pixel positions sharing the same feature map.

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. The 10-th layer later derives a 48 channel feature map. This *shrinkage in the middle* design makes the network lightweight and cost less to train. The residual learning technique is utilized in shared feature map extraction to accelerate the convergency of the network. So that we add the 1-st layer to the 10-th layer and then activate their sum with parametric rectified linear unit (PReLU) [27] function to obtain the shared feature map. After 9 convolutional layers with 3×3 kernel size, the receptive field of each pixel in the shared feature map is 19×19 in the input space, which means that a large nearby area in the integer-pixel position sample has been considered for the feature extraction of each pixel.

Compared with the method in [11], which investigates several networks to predict each subpixel position samples separately, our GVCNN is more compact and effective. We argue that there is no need to separately extract different feature maps to predict sub-pixel position samples, once the spatial correlation and continuity of the sub-pixels are fully considered. In our network, a shared feature map is generated and then used to infer sub-pixel samples at different locations. The shared feature map is followed by a specific convolutional layer, which generates a residual map for each sub-pixel position. The final results of sub-pixels can be obtained by adding the integer-position sample to the residual maps.

Fig. 2 shows the architecture of GVCNN. The network takes the integer-position sample I^{A} as its input. PReLU is 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:

$$f_k^{out} = P_k\left(W_k * f_{k-1}^{out} + B_k\right), \tag{6}$$

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 its bias. f_0^{out} is the input integer-position sample. The function $P_k(\cdot)$ is the PReLU function of the k-th layer:

$$P_k(x) = \begin{cases} x, & x > 0, \\ a_k * x, & x \le 0. \end{cases} \tag{7}$$

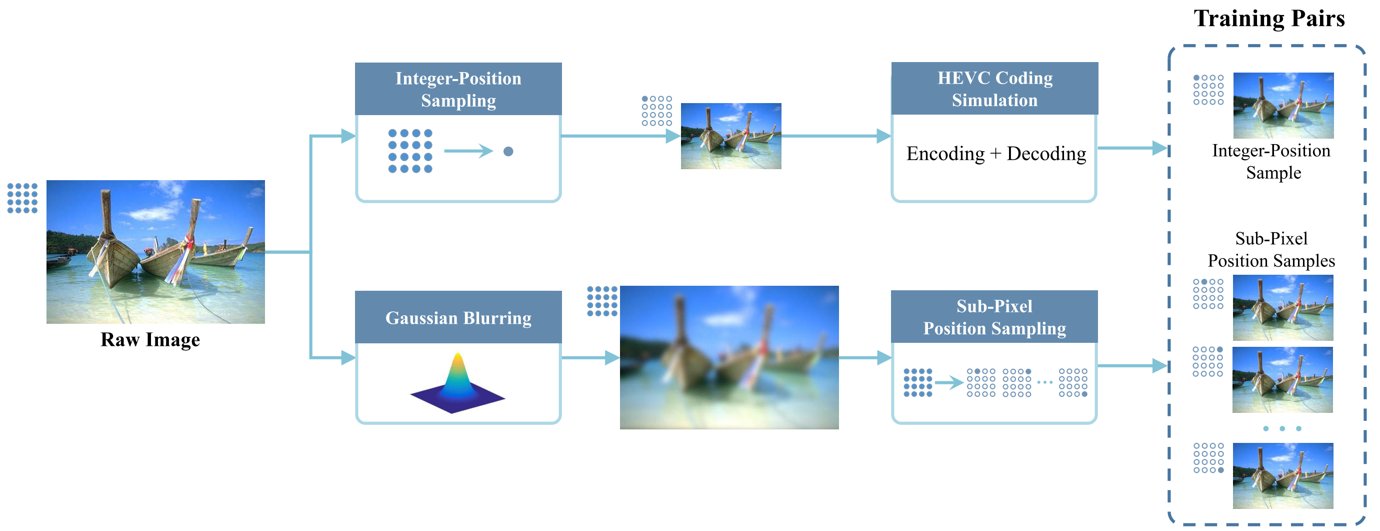


Fig. 3. Flow chart of the training data generation for GVCNN. For the integer position sample generation, pixels at integer positions are first sampled and then coded for simulation. Sub-pixel position samples are sampled from the raw image blurred by a Gaussian kernel to prevent aliasing effects.

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.

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 layer. The loss function can be formulated as follows:

$$L(\Theta) = \frac{1}{n}\sum_{i=1}^{n} \|F(x_i, \Theta) - y_i\|^2, \tag{8}$$

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

IV. Training Data Generation for Motion Compensation

In this section, we focus on the generation of training data that is going to be fed to the proposed GVCNN. We present a framework to simulate the practical process to generate ground truth for fractional interpolation. Specifically, we encode the integer-position pixels with HEVC and perform blurring operations before down-sampling. In addition, we provide theoretical analysis of the performance bound for motion estimation with quantization to choose the appropriate size of blur kernels.

A. Training Data Generation

In most video coding standards, motion compensation is used to compress redundant information between adjacent frames. Intuitively, corresponding pixels projected by predicted motion vectors between frames are likely to be located at sub-pixel positions, and this is where fractional interpolation is used. It generates reference samples in sub-pixel positions for motion vectors to project on. The effectiveness of fractional interpolation methods can be quantitatively evaluated by the final coding performance, which is affected by a great many factors in the complex video coding system. It means there is no exact ground truth for fractional interpolations.

Thus, in our work, we attempt to generate simulated ground truth for our GVCNN. Similar to previous works, we also obtain sub-pixel position samples by down-sampling from high-resolution pixel grids. Nevertheless, our training data generation framework has two key considerations:

1) Since fractional interpolation is performed on reconstructed coded frames, which contains reconstruction error especially under high quantization parameters, the input of our network is coded and reconstructed by video coding system to simulate practical situations. As shown in the lower part of Fig. 3, input integer-pixels are coded and reconstructed by HEVC after down-sampled from the raw image.
2) High-frequency components in the original high-resolution signal become aliasing after down-sampling, which leads to different appearance than the original signal. To deal with the problem, we perform blurring operations to alleviate the aliasing effect caused by down-sampling. The upper part of Fig. 3 illustrates the process. The high-resolution signal is first blurred and then down-sampled to generate simulated ground-truth sub-pixels. The size of the blur kernel is essential. Generally, a large blur kernel effectively reduces the influence of aliasing. However, it also introduces more noise in reconstruction. In the next subsection, we discuss the optimal size of blur kernels theoretically.

B. Performance Bound for Motion Estimation With Quantization

We analyze the performance bound for motion estimation with quantization and give suggestions for choosing the best blur kernel size in training data generation. Specifically, we consider the Cramer-Rao bounds (CRB) for

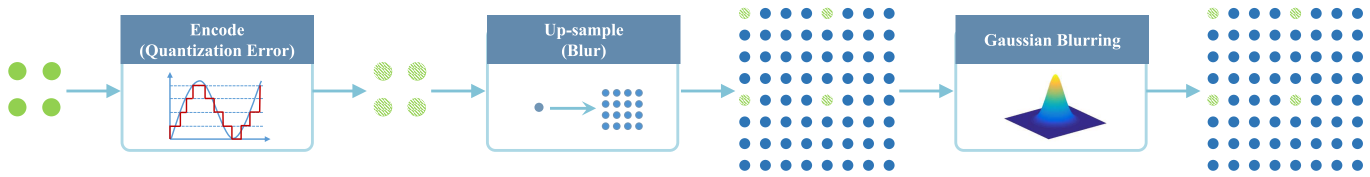


Fig. 4. Illustration of the degradations caused by quantization, up-sampling and Gaussian blurring during sub-pixel interpolation.

motion estimation, which provides the minimum mean square error that an unbiased estimator can achieve. The CRB can be obtained by constructing the maximum likelihood estimator and Fisher information matrix.

The following derivation refers to [28] and [29]. Our analysis has two distinct properties: 1) Our analysis considers quantization errors; 2) In the analysis of [28], the original and shift signals go through the same degradation. However, in our analysis, the shifted signals go through an additional quantization degradation process. For simplicity, we only consider 1-D signal.

1) Frequency Signals With Motion Shift: Consider two signals: a low frequency signal ω_1, and a high-frequency aliasing signal ω_2, where $\omega_2 = \omega_1 + N_L$. The original signal with these two frequency components in the time domain is presented as follows,

$$I_{t_1}(n) = \frac{A_1}{N_L} e^{-\frac{i2\pi\omega_1 n}{N_H}} + \frac{A_2}{N_L} e^{-\frac{i2\pi\omega_2 n}{N_H}}, \tag{9}$$

where N_L and N_H are the lengths of down-sampled and original signals, respectively, and N_H serves as a normalization constant. n is an integer indexes a certain frequency band. A_1 and A_2 are magnitudes of low frequency signal and high-frequency signals, respectively. Note that, the magnitudes of signals are assumed to follow a power law [30], thus the magnitude of ω_2 is much larger than those of other high-frequency components. That is why we can assume that, there is only one high frequency component in the given signal. Assuming a motion u_2 happens on the original grid in t_2 time, the shifted signal can be presented as follows,

$$\begin{aligned} I_{t_2}(n) &= \frac{A_1}{N_L} e^{-\frac{i2\pi\omega_1(n-u_2)}{N_H}} + \frac{A_2}{N_L} e^{-\frac{i2\pi\omega_2(n-u_2)}{N_H}}, \\ &= \frac{A_1}{N_L} W^{-w_1(n-u_2)} + \frac{A_2}{N_L} W^{-w_2(n-u_2)}, \end{aligned} \tag{10}$$

where $W = \exp\left(\frac{i2\pi}{N_H}\right)$.

We then conduct the further analysis in DFT domain, where the shifts in time naturally become changes in the phase of the signal. The DFTs of the original signals are

$$\begin{aligned} \widetilde{I}_{t_1}(\omega) &= Z_0 \left[A_1\delta(\omega-\omega_1) + A_2\delta(\omega-\omega_2)\right], \\ \widetilde{I}_{t_2}(\omega) &= Z_0 \left[A_1\delta(\omega-\omega_1)W^{u_2\omega_1} + A_2\delta(\omega-\omega_2)W^{u_2\omega_2}\right], \end{aligned} \tag{11}$$

where $Z_0 = N_H/N_L$.

2) Up-Sampled Quantized Signals: We then deduce the degradation version of Eqs. (10) and (11). In our sub-pixel interpolation for motion estimation, the current block is given in the high-resolution space, and the reference block is assumed to be at first quantized and then up-sampled by our algorithm. The whole process is shown in Fig. 4. To suppress the negative effect of the potential aliasing and quantization noises on motion compensation, blurring operations can be conducted on the current block and reference blocks. There are two kinds of blurring operations. The first one is implicitly conducted by our interpolation method. The second one is set in the training data synthesis. Thus, in the formulation, the two signals are degraded by rescaling operation (modeled as Gaussian blur), quantization (modeled as random noise) and contaminated by random noises. The DFTs of these two degraded signals are presented as follows,

$$\begin{aligned} \widetilde{J}_{t_1}(\omega) &= \left[G_{\sigma_p}(\omega_1)A_1 + G_{\sigma_p}(\omega_2)A_2\right]\delta(\omega-\omega_1) + n_1(\omega), \\ \widetilde{J}_{t_2}(\omega) &= \left\{\left[G_{\sigma_q}(\omega_1)G_{\sigma_k}(\omega_1)A_1W^{u_2\omega_1} + G_{\sigma_q}(\omega_1)n_1^{q_1}(\omega_1)\right]\right. \\ &\quad \left. + G_{\sigma_q}(\omega_2)G_{\sigma_k}(\omega_2)A_2W^{u_2\omega_2}\right\}\delta(\omega-\omega_1) + n_2(\omega), \end{aligned} \tag{12}$$

where n_1 and n_2 are assumed to be additive Gaussian noise with variances $\sigma_{n_1}^2$ and $\sigma_{n_2}^2$, respectively. DFTs of the Gaussian blur kernels that are used to suppress aliasing and quantization are denoted by $G_{\sigma_p}(\omega) = \exp\left(-\frac{\omega^2\sigma_p^2}{2}\right)$ and $G_{\sigma_q}(\omega) = \exp\left(-\frac{\omega^2\sigma_q^2}{2}\right)$, where σ_p and σ_q are the standard deviations. DFT of the Gaussian blur kernel caused by up-sampling is denoted by $G_{\sigma_k}(\omega) = \exp\left(-\frac{\omega^2\sigma_k^2}{2}\right)$, where σ_k is the standard deviation. $n_1^{q_1}$ is the quantization error, which follows a uniform distribution ranging between $[-\frac{q}{2}, \frac{q}{2}]$, where q is assumed to be linearly correlated to A_1. The expectation and variance of $n_1^{q_1}$ are:

$$\begin{aligned} \mathrm{E}(n_1^{q_1}) &= 0, \\ \mathrm{var}(n_1^{q_1}) &= C_1A_1^2, \end{aligned} \tag{13}$$

where C_1 is a constant correlated to the peak and range of the signal, as well as the quantization step. Since A_1 and A_2 follow a power law [30], the magnitude of quantization error of ω_1 is much larger than that of ω_2. Thus, we here neglect the quantization error of ω_2.

Similar to [28], the aliasing signals can be regarded as additive white Gaussian noise. Hence, the problem becomes

$$\begin{aligned} \widetilde{J}_{t_1}(\omega) &= \left[G_{\sigma_p}(\omega_1)A_1 + G_{\sigma_p}(\omega_2)A_2\right]\delta(\omega-\omega_1) + n_1(\omega), \\ &= G_{\sigma_p}(\omega_1)A_1\delta(\omega-\omega_1) + n_1'(\omega), \\ \widetilde{J}_{t_2}(\omega) &= G_{\sigma_q}(\omega_1)G_{\sigma_k}(\omega_2)A_1\delta(\omega-\omega_1)W_{N_H}^{u_2w_1} + n_2'(\omega), \end{aligned} \tag{14}$$

where $n'_1 = n_1 + G_{\sigma_p}(\omega_2)A_2$ has the variance

$$\sigma^2_{n'_1} = G^2_{\sigma_p}(\omega_2)A^2_2 + \sigma^2_{n_1}, \tag{15}$$

and $n'_2 = n_2 + G_{\sigma_q}(\omega_2)G_{\sigma_k}(\omega_2)A_2 + G_{\sigma_q}(\omega_1)n^q_1$ has the variance

$$\sigma^2_{n'_2} = G^2_{\sigma_k}(\omega_2)G^2_{\sigma_q}(\omega_2)A^2_2 + \sigma^2_{n_2} + G^2_{\sigma_q}(\omega_2)C_1A^2_1. \tag{16}$$

3) Cramer-Rao Bounds for Motion Estimation: The Cramer-Rao bounds [31] provides a bound for unbiased estimators. It is obtained by calculating the inverse of the Fisher information matrix. The Fisher information matrix is calculated by taking derivatives of a log-likelihood in terms of the unknown parameters.

The negative log likelihood function for the input down-sampled image is

$$\begin{aligned} &-\log p\left(\widetilde{J}_{t_1}, \widetilde{J}_{t_2}|A_1, u_2\right) \\ &= \frac{1}{2\sigma^2_{n'_2}}||\widetilde{J}_{t_2}(\omega_1) - G_{\sigma_q}(\omega_1)G_{\sigma_k}(\omega_1)A_1W^{u_2\omega_1}_{N_H}||^2_2 \\ &\quad + \frac{1}{2\sigma^2_{n'_1}}||\widetilde{J}_{t_1}(\omega_1) - G_{\sigma_p}(\omega_1)A_1||^2_2, \end{aligned} \tag{17}$$

where $||*||^2_2$ evaluates the ℓ_2 norm for complex signals.

Then, we can get Fisher Information Matrix for the unknown parameters $\theta = \{\mathrm{Re}\{A_1\}, \mathrm{Im}\{A_1\}, u_2\}$ as follows,

$$I_\theta = \begin{pmatrix} t_1+t_2 & 0 & st_2 \\ 0 & t_1+t_2 & st_2 \\ st_2 & st_2 & s^2t_2 \end{pmatrix}, \tag{18}$$

where $t_1 = G^2_{\sigma_k}(\omega_1)/\sigma^2_{n'_1}$, $t_2 = G^2_{\sigma_q}(\omega_1)G^2_{\sigma_k}(\omega_1)/\sigma^2_{n'_2}$, and $s = 2\pi A_1\omega_1/N_H$. Then, its inverse is,

$$I^{-1}_\theta = \begin{pmatrix} \frac{t_1}{t^2_1 - t^2_2} & \frac{t_2}{t^2_1 - t^2_2} & \frac{-1}{st_1 - st_2} \\ \frac{t_2}{t^2_1 - t^2_2} & \frac{t_1}{t^2_1 - t^2_2} & \frac{-1}{st_1 - st_2} \\ \frac{-1}{st_1 - st_2} & \frac{-1}{st_1 - st_2} & \frac{t_1+t_2}{s^2(t_1-t_2)t_2} \end{pmatrix}. \tag{19}$$

Thus, we obtain the following Cramer-Rao bounds for estimating motion u_2 as

$$\mathrm{var}[\hat{u}_2] \geq I^{-1}_\theta(3,3) = \frac{t_1+t_2}{s^2(t_1-t_2)t_2} = \frac{1}{s^2t_2}\mathbf{H}_1, \tag{20}$$

where

$$\begin{aligned} \mathbf{H}_1 &= \frac{t_1+t2}{t_1-t2} = \frac{G^2_{\sigma_k}(\omega_1)\sigma^2_{n'_2} + G^2_{\sigma_q}(\omega_1)G^2_{\sigma_k}(\omega_1)\sigma^2_{n'_1}}{G^2_{\sigma_k}(\omega_1)\sigma^2_{n'_2} - G^2_{\sigma_q}(\omega_1)G^2_{\sigma_k}(\omega_1)\sigma^2_{n'_1}}, \\ &= \frac{\sigma^2_{n'_2} + G^2_{\sigma_q}(\omega_1)\sigma^2_{n'_1}}{\sigma^2_{n'_2} - G^2_{\sigma_q}(\omega_1)\sigma^2_{n'_1}}. \end{aligned}$$

Since the power spectral of natural images follows a power low, *i.e.* $|A_1| \gg |A_2|$, the quantization error term related to $|A_1|$ in $\sigma^2_{n'_2}$ makes $\sigma^2_{n'_2}$ dominate $\sigma^2_{n'_2} + G^2_{\sigma_q}(\omega_1)\sigma^2_{n'_1}$ in magnitude. Then, $1 \leq \mathbf{H}_1 \leq 1+\alpha$, and $1+\alpha$ is the upper-bound of $\mathbf{H}_1$. Therefore, we can replace $\mathbf{H}_1$ with a constant h_1. After that, we have

$$\begin{aligned} \mathrm{var}[\hat{u}_2] &\geq I^{-1}_\theta(3,3) = \frac{1}{s^2t_2}\frac{t1+t2}{t_1-t_2} \geq \frac{h_1}{s^2t_2} \\ &= \frac{N^2_Hh_1}{4\pi^2A^2_1\omega^2_1}\frac{\sigma^2_{n'_2}}{G^2_{\sigma_q}(\omega_1)G^2_{\sigma_k}(\omega_1)}, \\ &= \frac{N^2_Hh_1}{4\pi^2A^2_1\omega^2_1}Z_1(\omega_1,\omega_2), \\ &= \frac{N^2_Hh_1}{4\pi^2A^2_1\omega^2_1}Z_2(\omega_1,\omega_2), \\ &= \frac{N^2_Hh_1}{4\pi^2A^2_1\omega^2_1}(\mathbf{H}_2 + \mathbf{H}_3), \end{aligned} \tag{21}$$

where

$$\begin{aligned} Z_1(\omega_1,\omega_2) &= \frac{G^2_{\sigma_k}(\omega_2)G^2_{\sigma_q}(\omega_2)A^2_2 + \sigma^2_{n_2} + G^2_{\sigma_q}(\omega_2)C_1A^2_1}{G^2_{\sigma_q}(\omega_1)G^2_{\sigma_k}(\omega_1)}, \\ Z_2(\omega_1,\omega_2) &= e^{\frac{(\omega^4_1-\omega^4_2)\sigma^4_q}{2}}e^{\frac{(\omega^4_1-\omega^4_2)\sigma^4_k}{2}}A^2_2 \\ &\quad + e^{\frac{(\omega^4_1-\omega^4_2)\sigma^4_q}{2}}e^{\frac{\omega^4_1\sigma^4_k}{2}}C_1A^2_1 + e^{\frac{\omega^4_1\sigma^2_q}{2}}e^{\frac{\omega^4_1\sigma^2_k}{2}}\sigma^2_{n_2}, \end{aligned}$$

and

$$\begin{aligned} \mathbf{H}_2 &= e^{\frac{(\omega^4_1-\omega^4_2)\sigma^4_q}{2}}e^{\frac{(\omega^4_1-\omega^4_2)\sigma^4_k}{2}}A^2_2 + e^{\frac{(\omega^4_1-\omega^4_2)\sigma^4_q}{2}}e^{\frac{\omega^4_1\sigma^4_k}{2}}C_1A^2_1, \\ \mathbf{H}_3 &= e^{\frac{\omega^4_1\sigma^2_q}{2}}e^{\frac{\omega^4_1\sigma^2_k}{2}}\sigma^2_{n_2}. \end{aligned}$$

We can observe the effects of the blur kernel (σ_k) on motion estimation. The effects are two-fold. A small kernel preserves low-frequency component, boosts less noises (in $\mathbf{H}_3$), and suppresses less aliasing signals and quantization errors (in $\mathbf{H}_2$). A large kernel lose effective low-frequency component, boosts noises (in $\mathbf{H}_3$), but suppresses more aliasing signals and quantization errors (in $\mathbf{H}_2$). An intermediate size blur kernel achieves best performance.

V. Experimental Results

A. Implementation Details

200 training images and 200 testing images in *BSDS500* [32] at size 481 × 321 and 321 × 481 are used for training data generation. Different from [11], which codes and reconstructs input integer samples with one single QP value to obtain a specific model for each QP, we utilize multiple QPs from 0 to 51 for training data generation to acquire a model that can be applicable for all QPs.

Besides, due to different sub-pixel levels of half-pixel samples and quarter-pixel samples, we adopt different settings for training data generation for half-pixel and quarter-pixel position samples, and we train two models for these two sub-pixel levels (GVCNN-H for half-pixel positions and GVCNN-Q for quarter-pixel positions). Specific settings are demonstrated below.

For GVCNN-H, 3 × 3 Gaussian kernels with random standard deviations in the range [0.4, 0.5] are used for blurring. By dividing the images into 2 × 2 patches without overlapping,

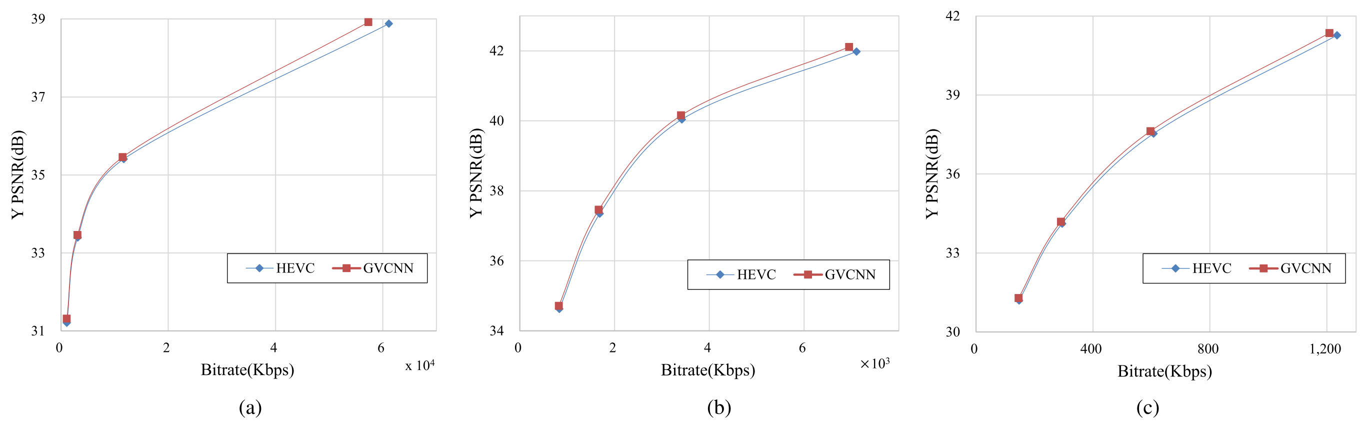


Fig. 5. Three example R-D curves of the sequences (a) *BQTerrace*, (b) *Kimono* and (c) *BasketballPass* under LDP configuration.

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

As for GVCNN-Q, the inferred samples are at a smaller sub-pixel level. 3×3 Gaussian kernels with random standard deviations in the range $[0.5, 0.6]$ are utilized. The sampling is 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.

During the integration into HEVC, we interpolate an entire coded reference frame with our trained GVCNN-H and GVCNN-Q in advance. Specifically, we take the reference frame as the integer-position sample and input it into the trained networks to acquire 3 half-pixel position samples $(I^{h^1}, I^{h^2}, I^{h^3})$ and 12 quarter-pixel position samples $(I^{q^1}, I^{q^2}, \ldots, I^{q^{12}})$. The interpolated sub-pixel position samples are cached in the system and we can later obtain the needed blocks during inter prediction. For the sake of better performance, a CU level RDO is used for the interpolation method selection. Two passes of encoding that respectively interpolate the sub-pixel position samples with and without the deep based method are performed at the encoder side. A flag is set based on the rate-distortion costs of the two passes to indicate whether to use the deep based method for sub-pixel position samples interpolation. The flag is coded with one bit and integrated at CU level. All the prediction units (PU) in a CU share the same flag.

B. Experimental Settings

During the training process, the training images are decomposed into 32×32 sub-images with a stride of 16. The GVCNN is trained on Caffe platform [33] via Adam [34] with standard back-propagation. The learning rate is initially set as a fixed value 0.0001. The batch size is set as 128. Models after $100,000$ iterations are used for testing. The network is trained on a single NVIDIA GeForce GTX 1080.

The proposed method is tested on HEVC reference software HM 16.4 under the low delay P (LDP) configuration. BD-rate is used to measure the rate-distortion. The QP values are set to be 22, 27, 32 and 37. As mentioned above, we train one GVCNN-H and GVCNN-Q for all QPs based on the corresponding training data. 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.

C. Experimental Results and Analysis

1) Overall Performance: Performance of the proposed method for classes A, B, C, D, E and F is shown in Table I. For the luma component, our method achieves on average 2.2%, 1.2% and 0.9% BD-rate saving respectively under LDP, LDB and RA conditions. And for the sequence *BQTerrace*, the BD-rate saving under LDP can be as high as 5.2%. For further verification, some example rate-distortion (R-D) curves are shown in Fig. 5. It can be seen that under most QPs our method is superior to HEVC.

Besides, the JVET ultra high definition (UHD) sequences (3840×2160) are also tested under the RA configuration. Results are shown in Table II. The BD-rate reduction is less significant on high-resolution test sequences compared with that on other sequences. Our explanation is that, the sampling precision of the high-resolution test sequences is high enough and the signals of adjacent pixels are more continuous. Hence, it is not very beneficial to generate sub-pixel position samples for motion compensation.

2) Comparison With Existing Methods: We also compare our method with two existing deep based fractional interpolation methods. Two methods (respectively called CNNIF [11] and VDIF [12]) only replace the half-pixel interpolation without rate-distortion optimization in HEVC and is tested on HM 16.4. For fair comparison, we also integrate our method into HM 16.7 by only replacing the half-pixel interpolation algorithm with the GVCNN-H without rate-distortion optimization. Three methods are all tested under the HEVC common test conditions. The BD-rate reduction comparison for Y component between the three methods are shown in Table III. Our method obtains gain over CNNIF and VDIF in most cases. And our performance will be further improved after adding the GVCNN-Q and rate-distortion optimization.

3) Verification of the Grouped Variation Network: With the grouped variation network, only two networks corresponding to half-pixel and quarter-pixel position samples need

TABLE I
BD-Rate Reduction of the Proposed Method Compared to HEVC

Class	Sequence	BD-rate of LDP			BD-rate of LDB			BD-rate of RA		
		Y	U	V	Y	U	V	Y	U	V
Class A	Traffic	-	-	-	-	-	-	-1.1%	0.1%	0.3%
	PeopleOnStreet	-	-	-	-	-	-	-0.9%	-0.1%	-0.8%
	Nebuta	-	-	-	-	-	-	-0.1%	-0.3%	-0.4%
	SteamLocomotive	-	-	-	-	-	-	-0.2%	-0.5%	-0.5%
	Average	-	-	-	-	-	-	-0.6%	-0.2%	-0.3%
Class B	Kimono	-4.1%	2.1%	1.6%	-1.7%	0.9%	0.4%	-1.3%	0.2%	-0.2%
	BQTerrace	-5.2%	-3.4%	-3.9%	-1.3%	0.3%	0.2%	-2.5%	0.0%	-1.1%
	BasketballDrive	-3.3%	0.2%	-0.5%	-0.5%	0.5%	0.1%	-1.4%	-0.8%	-0.8%
	ParkScene	-1.3%	0.0%	-0.5%	-0.8%	0.7%	0.2%	-0.7%	0.3%	-0.5%
	Cactus	-2.5%	-0.5%	-0.8%	-1.0%	0.0%	-0.1%	-1.1%	-0.2%	-1.0%
	Average	-3.3%	-0.3%	-0.8%	-1.1%	0.8%	0.4%	-1.4%	-0.1%	-0.7%
Class C	BasketballDrill	-2.2%	-1.3%	-0.6%	-1.0%	0.0%	-0.1%	-0.7%	0.0%	0.1%
	BQMall	-2.9%	-2.1%	-1.7%	-1.3%	-0.9%	-1.0%	-1.1%	-0.2%	-1.0%
	PartyScene	-1.6%	-0.5%	-1.4%	-0.9%	-0.2%	-0.4%	-0.7%	-0.4%	-1.1%
	RaceHorsesC	-2.0%	-1.4%	-1.6%	-1.5%	-0.4%	-0.8%	-1.6%	-1.5%	-1.3%
	Average	-2.2%	-1.3%	-1.3%	-1.1%	-0.4%	-0.6%	-1.0%	-0.5%	-0.8%
Class D	BasketballPass	-3.3%	-1.7%	-1.4%	-1.8%	-0.9%	-1.5%	-0.9%	-1.0%	-1.5%
	BlowingBubbles	-2.1%	-0.9%	-0.3%	-1.0%	0.2%	-0.2%	-0.8%	-0.7%	0.2%
	BQSquare	-0.6%	1.2%	3.1%	-0.7%	1.6%	0.9%	-0.7%	-0.3%	-0.5%
	RaceHorses	-2.7%	-1.7%	-0.8%	-1.9%	0.0%	-0.9%	-1.4%	-0.9%	-1.1%
	Average	-2.2%	-0.7%	0.2%	-1.4%	0.2%	-0.4%	-1.0%	-0.7%	-0.7%
Class E	FourPeople	-1.6%	-0.5%	-0.2%	-2.8%	5.9%	0.0%	-	-	-
	Johnny	-2.9%	0.2%	0.4%	-1.2%	1.1%	1.4%	-	-	-
	KristenAndSara	-2.2%	1.1%	0.2%	-1.3%	1.6%	1.2%	-	-	-
	Average	-2.2%	0.3%	0.2%	-1.6%	2.5%	0.8%	-	-	-
Class F	BasketballDrillText	-1.8%	-0.9%	0.1%	-1.0%	-0.6%	-0.8%	-0.9%	0.2%	-0.5%
	ChinaSpeed	-1.4%	-1.9%	-1.4%	-1.0%	-1.3%	-1.5%	-1.4%	-2.0%	-1.8%
	SlideEditing	0.0%	-0.1%	-0.2%	0.0%	-0.1%	-0.1%	0.6%	0.9%	0.9%
	SlideShow	-0.5%	0.2%	-0.6%	-0.9%	0.5%	1.0%	-0.8%	0.3%	-0.9%
	Average	-0.9%	-0.7%	-0.5%	-0.7%	-0.9%	-0.6%	-0.6%	-0.2%	-0.6%
All Sequences	Overall	-2.2%	-0.6%	-0.5%	-1.2%	0.4%	-0.1%	-0.9%	-0.3%	-0.6%

TABLE II
BD-Rate Reduction of the Proposed Method for UHD Sequences Under RA Configuration Compared to HEVC

Class	Sequence	Y	U	V
Class SDR-A	FoodMarket4	-0.8%	-0.7%	-0.7%
	CatRobot1	-0.3%	-0.4%	-0.4%
	DaylightRoad2	-0.4%	-0.7%	-0.8%
	ParkRunning3	-0.6%	-1.3%	-1.7%
	Campfire	-0.1%	-0.2%	-0.1%
	Average	-0.4%	-0.7%	-0.7%

to be trained, which is convenient and meaningful for real application. In order to further identify the effectiveness of our grouped variation method, we additionally train 15 networks for each sub-pixel position to identify that the convenience is achieved without the loss of coding performance. We define the method that trains networks separately as GVCNN-Separate and the proposed uniform method as GVCNN-Uniform. The average BD-rate reduction comparison for some testing classes are shown in Table IV. Note that in this comparison only the quarter-pixel interpolation method is selected by RDO. As can be observed, results of training the networks uniformly are comparable to the results obtained by training networks separately.

Besides the coding performance, we additionally compare the implementation time of the two methods. The overall encoding time complexity in CPU mode for GVCNN-Separate is 1495% and for GVCNN-Uniform is 629%. It can be seen that the time complexity of GVCNN-Separate is much larger than GVCNN-Uniform since GVCNN-Separate needs more passes of network forwarding.

4) Analysis of Blurring in Training Data Generation: The effect of the blurring process in training data generation is also further tested and analyzed. Table V has shown the results which are derived by the models trained without blurring. The proposed model obtains obvious gain over the model without blurring.

We define the half-pixel interpolation model trained with Gaussian blurring kernels whose standard deviations are in range $[x, x + 0.1]$ as GVCNN-H(x) and similarly the

TABLE III
BD-RATE REDUCTION COMPARISON FOR CLASSES C, D AND E UNDER LDP CONFIGURATION

Class	Sequence	CNNIF [11]	VDIF [12]	GVCNN-H
Class C	BasketballDrill	-1.2%	-1.8%	-1.8%
	BQMall	-0.9%	-0.6%	-1.0%
	PartyScene	0.2%	-0.2%	0.0%
	RaceHorsesC	-1.5%	-0.8%	-1.8%
	Average	-0.9%	-0.9%	-1.1%
Class D	BasketballPass	-1.3%	-2.4%	-2.1%
	BlowingBubbles	-0.3%	-0.4%	-0.6%
	BQSquare	1.2%	0.5%	2.7%
	RaceHorses	-0.8%	-1.0%	-1.1%
	Average	-0.3%	-0.8%	-0.3%
Class E	FourPeople	-1.3%	-1.3%	-2.0%
	Johnny	-1.2%	-1.4%	-1.0%
	KristenAndSara	-1.0%	-1.0%	-2.2%
	Average	-1.2%	-1.2%	-1.7%

TABLE IV
AVERAGE BD-RATE REDUCTION ACHIEVED BY TRAINING THE NETWORKS SEPARATELY AND UNIFORMLY

Class	GVCNN-Separate			GVCNN-Uniform		
	Y	U	V	Y	U	V
Class C	-1.8%	-0.8%	-0.9%	-1.8%	-1.1%	-0.9%
Class D	-1.3%	0.2%	0.1%	-1.5%	-0.3%	-0.1%
Class E	-2.0%	0.7%	0.9%	-1.9%	0.5%	0.4%

TABLE V
BD-RATE REDUCTION OF GVCNN WITH AND WITHOUT BLURRING FOR CLASS D

Sequence	GVCNN w/o blurring			GVCNN		
	Y	U	V	Y	U	V
BasketballPass	-3.1%	-0.6%	-0.7%	-3.4%	-1.4%	-1.9%
BlowingBubbles	-1.5%	0.1%	-0.8%	-1.6%	-0.9%	-1.0%
BQSquare	1.1%	3.6%	4.6%	1.1%	2.2%	3.9%
RaceHorses	-1.8%	-1.3%	-1.4%	-2.2%	-1.1%	-1.3%
Average	-1.3%	0.5%	0.4%	-1.5%	-0.3%	-0.1%

quarter-pixel interpolation model as GVCNN-Q(x). For further analysis, we have trained models with different blurring levels and tested them on class D. By fixing the blurring level of one model (GVCNN-H or GVCNN-Q) and changing the others', two line charts are derived as shown in Fig. 6 and Fig. 7. It is obvious that the models of median blurring levels obtain the largest gain, which accords with the theory proposed in Sec. IV-B. Note that, in this analysis, only the quarter-pixel interpolation method is selected by RDO.

5) Implementation Time: Experiments shown in this paper are all run in CPU mode. We additionally test the class D in GPU mode for tesing. The Intel i7 7700 CPU and NVIDIA GeForce GTX 1070 GPU are used for testing. In GPU mode, the GPU is alternatively used for the forward operation of CNN and other operations are still performed by CPU. The overall encoding time complexity in CPU mode is 629% and in GPU mode is 282%. And the decoding time complexity in CPU mode is 154858% and in GPU mode is 11922%. It can be seen that the coding efficiency is well improved by GPU acceleration. There are advanced techniques to accelerate our method, such as a fixed point implementation of our model. Based on recent progresses of deep model quantization [35], all 32-bit float point multiplications can be replaced by 8-bit fixed point operations. After the optimization, the running time of our neural network model in theory can reduce to 1/4 of that before optimization. The deduced encoding time of our method in GPU mode can reduce from about 280% to 145%. Moreover, the complexity of our work can be further reduced by hardware acceleration when being integrated into the chip in real applications.

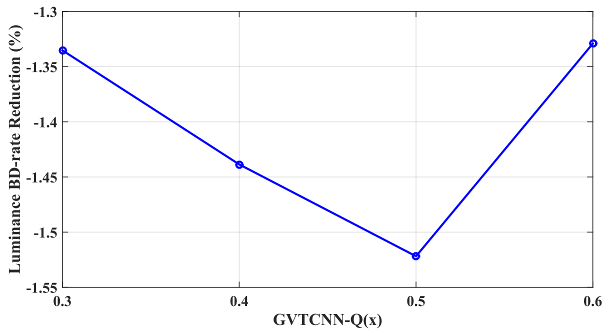


Fig. 6. BD-rate reduction obtained with fixed GVCNN-H model ($x = 0.4$) and various GVCNN-Q models ($x = 0.3, 0.4, 0.5, 0.6$).

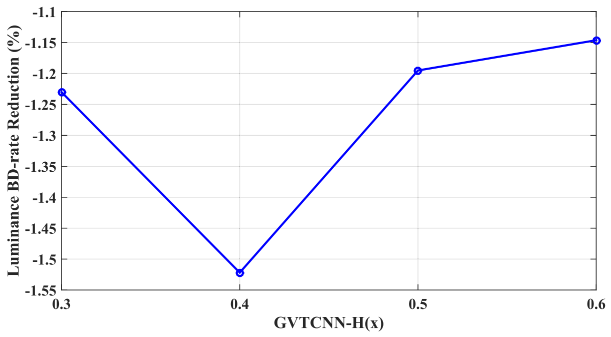


Fig. 7. BD-rate reduction obtained with fixed GVCNN-Q model ($x = 0.5$) and various GVCNN-H models ($x = 0.3, 0.4, 0.5, 0.6$).

6) Rate Distortion Optimization Results: During the integration, a CU level RDO is performed to select the method for interpolating sub-pixel position samples. PUs with sub-pixel level motion vectors in one CU will choose the same interpolation method. Some example RDO results of the sequences in class D are shown for verification.

Fig. 8 shows the visualization results of the interpolation method selection for a frame in *RaceHorses*. In the figure, the marked blocks are the CUs of inter mode. Color red indicates that PUs in the CU choose GVCNN for interpolation and color blue indicates that PUs choose the interpolation method DCTIF of HEVC.

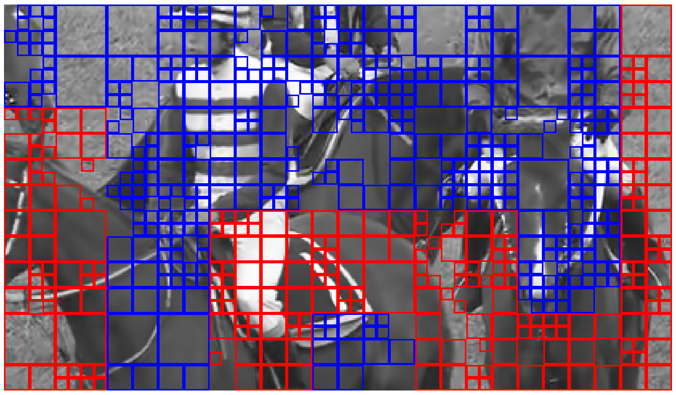


Fig. 8. Visualization of RDO results of a frame in *RaceHorses*. The test condition is LDP, QP32, POC8. Red Blocks indicate that the CU chooses GVCNN for interpolation and the blue ones represent CUs that choose DCTIF.

TABLE VI
RDO RESULTS FOR SUB-PIXEL POSITION SAMPLES INTERPOLATION

Sequence	GVCNN	HEVC	Ratio	Y BD-rate
BasketballPass	28580	100466	22.15%	-3.3%
BlowingBubbles	20167	158391	11.29%	-2.1%
BQSquare	10435	259432	3.87%	-0.6%
RaceHorses	35342	102604	25.62%	-2.7%

We also count the total numbers of the two kinds of CUs for the sequences in class D. We test the sequences for 2s with QPs 22, 27, 32 and 37. The number of CUs that choose GVCNN and DCTIF for interpolation during the encoding process are calculated and shown in Table VI. The choosing ratio is calculated for each sequence to indicate the ratio of the CUs that choose GVCNN for interpolation. It can be seen that the choosing ratio is highly related with the coding performance of our method over the sequence.

VI. CONCLUSION

In this paper, we propose a one-for-all grouped variation convolutional neural network for fractional interpolation in motion compensation of video coding. The network first uniformly extracts a feature map from the input integer-position sample, and then variations at different sub-pixel positions are inferred sharing the same feature map. Training data generation of the proposed network is also carefully analyzed and designed. Experimental results show that our method has obtained on average a 2.2% BD-rate saving on the test sequences compared with HEVC. Effectiveness of the grouped variation 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 Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[2] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[3] R. Wang, C. Huang, J. Li, and Y. Shen, "Sub-pixel motion compensation interpolation filter in AVS [video coding]," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, vol. 1, Jun. 2004, pp. 93–96.

[4] H. Lakshman, B. Bross, H. Schwarz, and T. Wiegand, "Fractional-sample motion compensation using generalized interpolation," in *Proc. Picture Coding Symp.*, Dec. 2010, pp. 530–533.

[5] A. Kokhazadeh, A. Aminlou, and M. R. Hashemi, "Edge-oriented interpolation for fractional motion estimation in hybrid video coding," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2011, pp. 1–6.

[6] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017.

[7] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser prior for image restoration," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 2808–2817.

[8] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2536–2544.

[9] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 184–199.

[10] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1646–1654.

[11] 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. Symp. Circuits Syst.*, May 2017, pp. 1–4.

[12] H. Zhang, L. Song, Z. Luo, and X. Yang, "Learning a convolutional neural network for fractional interpolation in HEVC inter coding," in *Proc. IEEE Vis. Commun. Image Process.*, Dec. 2017, pp. 1–4.

[13] R. Lin, Y. Zhang, H. Wang, X. Wang, and Q. Dai, "Deep convolutional neural network for decompressed video enhancement," in *Proc. Data Compress. Conf.*, Mar./Apr. 2016, p. 617.

[14] Y. Dai, D. Liu, and F. Wu, "A convolutional neural network approach for post-processing in HEVC intra coding," in *Proc. Int. Conf. Multimedia Modeling*, 2017, pp. 28–39.

[15] T. Wang, M. Chen, and H. Chao, "A novel deep learning-based method of improving coding efficiency from the decoder-end for HEVC," in *Proc. Data Compress. Conf.*, Apr. 2017, pp. 410–419.

[16] W.-S. Park and M. Kim, "CNN-based in-loop filtering for coding efficiency improvement," in *Proc. IEEE Image, Video, Multidimensional Signal Process. Workshop*, Jul. 2016, pp. 1–5.

[17] T. Laude and J. Ostermann, "Deep learning-based intra prediction mode decision for HEVC," in *Proc. Picture Coding Symp.*, Dec. 2016, pp. 1–5.

[18] Z. Liu, X. Yu, Y. Gao, S. Chen, X. Ji, and D. Wang, "CU partition mode decision for HEVC hardwired intra encoder using convolution neural network," *IEEE Trans. Image Process.*, vol. 25, no. 11, pp. 5088–5103, Nov. 2016.

[19] M. Xu, T. Li, Z. Wang, X. Deng, R. Yang, and Z. Guan. (2017). "Reducing complexity of HEVC: A deep learning approach." [Online]. Available: https://arxiv.org/abs/1710.01218

[20] W. Dong, L. Zhang, R. Lukac, and G. Shi, "Sparse representation based image interpolation with nonlocal autoregressive modeling," *IEEE Trans. Image Process.*, vol. 22, no. 4, pp. 1382–1394, Apr. 2013.

[21] J.-J. Huang, W.-C. Siu, and T.-R. Liu, "Fast image interpolation via random forests," *IEEE Trans. Image Process.*, vol. 24, no. 10, pp. 3232–3245, Oct. 2015.

[22] H. Huang, R. He, Z. Sun, and T. Tan, "Wavelet-SRNet: A wavelet-based CNN for multi-scale face super resolution," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Oct. 2017, pp. 1689–1697.

[23] T. Guo, H. S. Mousavi, T. H. Vu, and V. Monga, "Deep wavelet prediction for image super-resolution," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jul. 2017, pp. 1100–1109.

[24] F. Guichard and F. Malgouyres, "Total variation based interpolation," in *Proc. Eur. Signal Process. Conf.*, 1998, pp. 1–4.

[25] S. D. Babacan, R. Molina, and A. K. Katsaggelos, "Total variation super resolution using a variational approach," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2008, pp. 641–644.

[26] Q. Yuan, L. Zhang, and H. Shen, "Regional spatially adaptive total variation super-resolution with spatial information filtering and clustering," *IEEE Trans. Image Process.*, vol. 22, no. 6, pp. 2327–2342, Jun. 2013.

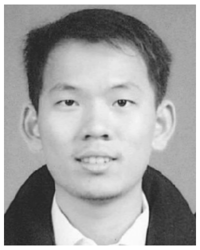
[27] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1026–1034.

[28] C. Liu and D. Sun, "On Bayesian adaptive video super resolution," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 2, pp. 346–360, Feb. 2014.
[29] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1999.
[30] S. Roth, "High-order Markov random fields for low-level vision," Ph.D. dissertation, Dept. Comput. Sci., Brown Univ., Providence, RI, USA, 2007.
[31] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Upper Saddle River, NJ, USA: Prentice-Hall, 1993.
[32] 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. 8th IEEE Int. Conf. Comput. Vis. (ICCV)*, vol. 2. Jul. 2001, pp. 416–423.
[33] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM Int. Conf. Multimedia*, 2014, pp. 675–678.
[34] D. P. Kingma and L. J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–15.
[35] P. Gysel, J. Pimentel, M. Motamedi, and S. Ghiasi, "Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5784–5789, Nov. 2018.

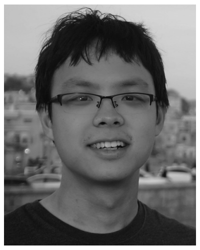
Jiaying Liu (S'08–M'10–SM'17) received the Ph.D. degree (Hons.) in computer science from Peking University, Beijing, China, in 2010. She was a Visiting Scholar with the University of Southern California, Los Angeles, from 2007 to 2008. She was a Visiting Researcher with Microsoft Research Asia in 2015 supported by the Star Track Young Faculties Award. She is currently an Associate Professor with the Institute of Computer Science and Technology, Peking University. She has authored over 100 technical articles in refereed journals and proceedings. She holds 28 granted patents. Her current research interests include multimedia signal processing, compression, and computer vision.

Dr. Liu is a Senior Member of CCF. She served as a member for the Multimedia Systems and Applications Technical Committee, the Education and Outreach Technical Committee in the IEEE Circuits and Systems Society, and the Image, Video, and Multimedia Technical Committee in APSIPA. She has also served as the Publicity Chair for IEEE ICIP-2019/VCIP-2018, the Grand Challenge Chair for IEEE ICME-2019, and the Area Chair for ICCV-2019. She was the APSIPA Distinguished Lecturer from 2016 to 2017.

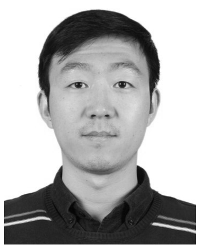
Sifeng Xia received the B.S. degree in computer science from Peking University, Beijing, China, in 2017, where he is currently pursuing the master's degree with the Institute of Computer Science and Technology. His current research interests include deep learning-based image processing and video coding.



Wenhan Yang (M'18) received the B.S. and Ph.D. degrees (Hons.) in computer science from Peking University, Beijing, China, in 2012 and 2018, respectively. He was a Visiting Scholar with the National University of Singapore from 2015 to 2016. His current research interests include deep-learning based image processing, bad weather restoration, and related applications and theories.



Mading Li received the B.S. and Ph.D. degrees in computer science from Peking University, Beijing, China, in 2013 and 2018, respectively. He was a Visiting Scholar with McMaster University in 2016. His current research interests include image and video processing, image interpolation, image restoration, and low-light image enhancement.



Dong Liu (M'13) received the B.S. and Ph.D. degrees in electrical engineering from the University of Science and Technology of China (USTC), Hefei, China, in 2004 and 2009, respectively.

He was a member of the Research Staff with the Nokia Research Center, Beijing, China, from 2009 to 2012. He joined USTC as an Associate Professor in 2012. He has authored or co-authored more than 80 papers in international journals and conferences. He has 14 granted patents. His research interests include image and video coding, multimedia signal processing, and data mining.

Dr. Liu received the 2009 IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY Best Paper Award and the Best 10% Paper Award at VCIP 2016. He and his team received four challenges that were held in ACM MM 2018, CVPR 2018, ECCV 2018, and ICME 2016, respectively.