IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 25, 2019, accepted August 5, 2019, date of publication August 14, 2019, date of current version August 27, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2935378

Deep Learning-Based Luma and Chroma Fractional Interpolation in Video Coding

CHI DO-KIM PHAM[1] **AND JINJIA ZHOU**[1,2]
[1]Graduate School of Science and Engineering, Hosei University, Tokyo 184-8584, Japan
[2]JST, PRESTO, Tokyo 332-0012, Japan

Corresponding author: Jinjia Zhou (jinjia.zhou.35@hosei.ac.jp)

This work was supported by the JST, PRESTO, Japan, under Grant JPMJPR1757.

ABSTRACT Motion compensated prediction is one of the essential methods to reduce temporal redundancy in inter coding. The target of motion compensated prediction is to predict the current frame from the list of reference frames. Recent video coding standards commonly use interpolation filters to obtain sub-pixel for the best matching block located in the fractional position of the reference frame. However, the fixed filters are not flexible to adapt to the variety of natural video contents. Inspired by the success of Convolutional Neural Network (CNN) in super-resolution, we propose CNN-based fractional interpolation for Luminance (Luma) and Chrominance (Chroma) components in motion compensated prediction to improve the coding efficiency. Moreover, two syntax elements indicate interpolation methods for the Luminance and Chrominance components, have been added to *bin-string* and encoded by CABAC using regular mode. As a result, our proposal gains 2.9%, 0.3%, 0.6% Y, U, V BD-rate reduction, respectively, under low delay P configuration.

INDEX TERMS Convolution neural network (CNN), fractional interpolation, video coding, motion compensated prediction.

I. INTRODUCTION

H.265/High Efficiency Video Coding (HEVC) [1] has outperformed its predecessor H.264/AVC [2] to become the state-of-the-art video coding standard. Compared to H.264/AVC, H.265/HEVC has been improved its coding techniques and achieves a 25-50% better data compression at the same image quality [3]. One of the critical technologies that significantly contributes to the high coding performance of HEVC is motion compensated prediction (MCP). MCP aims to predict the current frame from the reference frames which are previously reconstructed and store the residual along with the motion vector between the corresponding blocks, which benefits for reducing the temporal redundancy in inter coding. However, the converting signals from the analog domain to the digital domain may omit some data, which could make prediction worse. Therefore, if the best matching block does not fall into integer samples, fractional pixels and fractional motion vector are required for these movements. Widely used, MCP applies interpolation filters on the reference frame, considered as integer samples, to obtain fractional samples. For interpolation filters, H.265/HEVC offers 7-tap quarter and 8-tap half Discrete Cosine Transform interpolation filters (DCTIF) for fractional interpolation while they are the 6-tap filter for half-pixel and average filters for quarter-pixel interpolation in H.264/AVC.

Refer to fractional interpolation in video coding, there have been many works that focus on improving fixed filters [4], [5], or designing adaptive filters [6], [7], or hardware design [8] for fractional interpolation. Due to the covered area is limited, the correlation between neighboring pixels may not be fully exploited. Moreover, the input signal is not always ideal and stable for handcraft filters. For these reasons, designed filters may not be able to adapt to the diversity of natural video content. In some aspects, fractional interpolation in MCP can be considered as a specific task of super-resolution where a high-resolution image is reconstructed given a low-resolution image (images).

Recently, deep learning-based methods have been widely used and obtained remarkable results in image and video processing. Convolutional Neural Network (CNN), a most representative model of deep learning, well improves the performance of the traditional method in high-level computer vision such as classification [9], detection [10], [11] to low-level computer vision tasks like image denoising [12], and mostly super-resolution. SRCNN [13], a very first CNN

The associate editor coordinating the review of this article and approving it for publication was Huimin Lu.

model in learning-based super-resolution, learns the mapping between an input of low resolution-image and output of the high-resolution image, outperforms traditional method bicubic. The work [14] aims to learn image detail on a 20-layer CNN model VDSR to improve the quality of low-resolution input. Despite the robust of CNN in improving super-resolution, they can not be directly applied for fraction interpolation in video coding because of two main problems. First, CNN-based super-resolution may change integer pixel after convolution. Second, and more importantly, the training sets of super-resolution and fractional interpolation in video coding are different. While the former aims to recover the high-resolution image by ''enhancing'' quality of the low-resolution image, the latter focuses on producing fractional samples that close to the current block to be encoded from the reference frames.

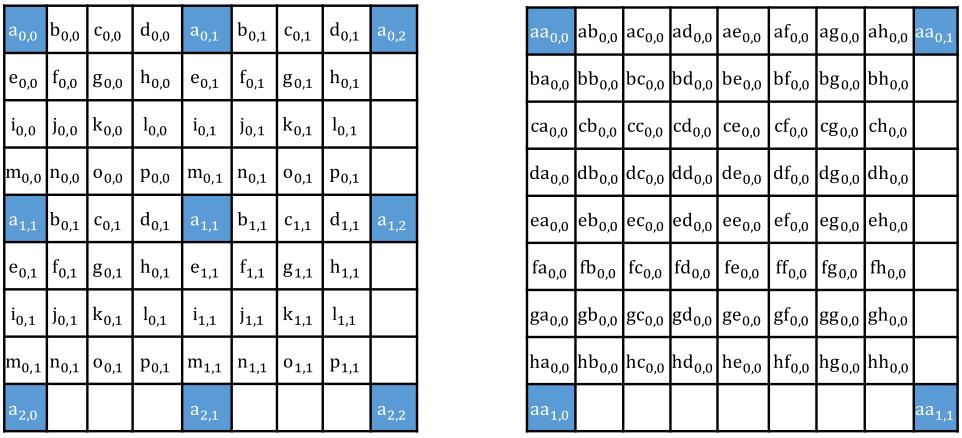
Inspired by the contributions of deep learning in video coding, recent studies have implemented diverse approaches to deep learning-based fractional interpolation works to improve the performance of MCP. To handle the problem of changing integer pixel after convolution, [15] first proposes three CNN models CNNIF_H, CNNIF_V, and CNNIF_D for horizontal, vertical, and diagonal half-pixel positions, respectively. By producing three half pixel independently, they keep integer pixels for the later process of HEVC. Later on, Zhang *et al.* introduces a CNN model followed by a Constrained mask with different weights for the integer pixels and three half pixels [16]. Similar to [15], Yan *et al.* trains 15 models for three half samples and 12 quarter samples in [17]. In [18], half and quarter pixels share nine convolution layers and be separately produced in group variational transformation in GVTCNN. Similarly, the work [19] designs switch mode based fractional interpolation to reduce the drawback of the motion shift in [18]. Liu *et al.* implements GVCNN that supports sub-pixel (half-pixel or quarter-pixel) under different QPs in a model [20].

For the second problem, besides the issue of different training sets, another difficulty that needs to be solved is fractional pixel does not exist in the real image. Generally, existing works assume integer and fractional pixels in the original frame, encode integer video, and learn the mapping between the reconstructed integer and fractional pixels [15], [18]–[20] or the mapping between the interpolated frame of the reconstructed reference frame and the original reference image [16]. Another way is encoding the original video and extracting the inter-coding block and its reference block to be ground-truth label and input of CNN [17].

Although prior research generally confirms that CNN-based fractional interpolation improves coding performance, there are some drawbacks that could be improved in these approaches: only half-pixel are supported [15], [16], many models need to be trained for fractional positions [17], or predicting fractional pixel from integer pixel may not good because the motion shift between integer and fractional pixel are not always stable [15], [17], [18], [20]. In paper [21], although the one models for all fractional samples and the prepossessing helps to reduce the motion shift problem, Chroma components are not well treated. In this paper, we take the next step towards the CNN-based fractional interpolation in video coding: all components can be processed by CNN. In our proposal, the Y, U and V components of the reference frame is interpolated by DCTIF before feeding into CNN to avoid the motion shift problem. To take full advantage of CNN and DCTIF, we implement an RDO-based interpolation method selection for each CU: Luma and Chroma components are interpolated by either DCTIF or CNN. Note that U and V components share one model for all fractional interpolation. For this selection, we encode two flags that indicate the interpolation method for Luma and Chroma components. As a result, we archive 2.9%, 0.3%, 0.6% BD-rate reduction compared to the anchor HEVC under low delay P configuration. Our work makes the following three contributions:

1) We present two CNN models for Luma and Chroma fractional pixels interpolation in video coding. A reconstructed frame is first interpolated by DCTIF to get 15 fractional samples. Our models take an input of a fractional sample and output corresponding fractional sample. Only one model is trained for 15 fractional samples at each QP. We use one model for Y component interpolation and a shared Chroma model for U and V components interpolation. Totally, we train eight models for four QPs in Luma and Chroma components.
2) We investigate a dataset generation method for our Y, U, V fractional interpolation training. As commonly, we generate our training set by assuming integer and fractional pixel in each video frame and encoding integer video. Each reconstructed video frame is interpolated by DCTIF to be CNN input and the fractional pixels extracted from the original frame are ground-truth labels for CNN.
3) We implement an RDO-based selection for Luma and Chroma fractional interpolation. In this selection, we define two new syntax elements to HEVC bin-string and encode them under CABAC regular mode. Each syntax element indicates the interpolation method for Luma or Chroma components for each CU that chooses inter coding with the fractional motion vector.

The proposal can be integrated to the existing video coding standards that support fractional motion search such as H.262, MPEG-4 Part 10 (as known as H.264/AVC), H.265/HEVC, AV1, etc., to further improve the coding efficiency. Moreover, network architecture also can be utilized for half and quarter-pixel interpolation for the coming video coding standard VVC/H.266. Since the different coding standards provide different noises, re-train our network on other standards reconstructed frames is required. The interpolation CNN output can replace the output of the interpolation filters in motion compensation to further improve the prediction of interpolation filters.



(a) Luma sub-samples (b) Chroma sub-samples

FIGURE 1. Luma (a) and Chroma (b) sub-samples in HEVC. In (a) Luma sub-samples, $a_{i,j}$ present for integer pixel, $c_{i,j}$, $i_{i,j}$,$k_{i,j}$ are the half pixels and others are quarter pixels. Half and quarter pixels in the Luma component are interpolated by a 7-tap quarter and an 8-tap half DCTIF. At (b) Chroma sub-samples, $aa_{i,j}$ presents for integer samples and other pixels are fractional samples which are interpolated by four-tap filters.

The rest of this paper is organized as follow. Section II present fractional interpolation in video coding standards. Our proposals are discussed in section III followed by the experiments in section refexperiments. Our conclusions are drawn in section V.

II. FRACTIONAL INTERPOLATION IN VIDEO CODING STANDARDS

In HEVC, motion search searches for the best matching fractional Luma samples in the list of previously reconstructed reference frames and stores the motion vector points to the best fractional pixel. Fig. 1 presents the positions of integer and fractional samples of Luma and Chroma components in HEVC. In Luma sub-samples, $a_{i,j}$ presents integer pixels, $c_{i,j}$, $i_{i,j}$, and $k_{i,j}$ are half pixel, and others are quarter pixels. In Luma fractional interpolation, HEVC applies horizontal filters on integer samples for $b_{i,j}$, $c_{i,j}$, and $d_{i,j}$. For $e_{i,j}$, $i_{i,j}$, and $m_{i,j}$, vertical filters are applied on integer samples. For the other samples, the vertical filter of its row is applied on the fractional sample at its column in the top row. For example, to obtain $g_{0,0}$, vertical filter of $e_{0,0}$ is applied on $c_{0,0}$.

HEVC supports fractional samples and motion vector for the Luma component up to quarter accuracy for the common used YUV format 420. Chroma component, whose resolution is equal to the half of the Luma component's, holds a motion vector that accurate to one eight samples. Therefore, eight samples are interpolated using 4-tap filters [1] in motion compensation. Motion vector at Chroma component is calculated as:

$$fracMV_{Chroma} = MV \bmod 8 \tag{1}$$

Fig. 1 (b) presents fractional-sample in Chroma component. If the fractional MV points to $ab_{0,0}$, $ab_{0,0}$ and $ab_{i,j}$ are interpolated with the resolution is equal to Chroma resolution of the current block to be encoded. In Luma and Chroma component fractional interpolating, the applied area is restricted in the tap size of the interpolation filters, which may lead the predicted block not good enough.

III. PROPOSED CNN-BASED LUMA AND CHROMA FRACTIONAL INTERPOLATION

A. OVERVIEW

As mentioned above, despite the fact that CNN-based techniques have acquired outstanding performance compared to traditional super-resolution methods, they can not be directly applied to fractional interpolation in video coding. In this paper, we design a training set and propose two Convolutional Neural Networks for Luma and Chroma fractional interpolation in video coding. To take full advantage of DCTIF, we offer an RDO-based selection for Luma and Chroma components by CNN. In Fig. 2, we present our proposal in integrating CNN-based fractional interpolation and the interpolation method for Luma and Chroma Components to HEVC. In searching for the best fractional pixels at encoding, Y component of the reference frame is interpolated in both DCTIF and CNN. At motion compensation, we interpolate Y component by the method used in motion search, U and V components are separately interpolated by CNN and DCTIF. At encoding residual and calculating RDO cost for CU that chooses inter coding with a fractional motion vector, we also encode two flags indicate fractional interpolation method for Luma and Chroma components by choosing the smallest RDO cost between four possible interpolation methods: DCTIF and DCTIF, DCTIF and CNN, CNN and CNN, and CNN and DCTIF for Luma and Chroma, respectively.

This section outlines our training set generation followed by our network architecture and ends with Luma and Chroma interpolation selection.

B. TRAINING SET GENERATION

In training any network, the training set plays a vital role and decides network at testing. For CNN-based super-resolution,

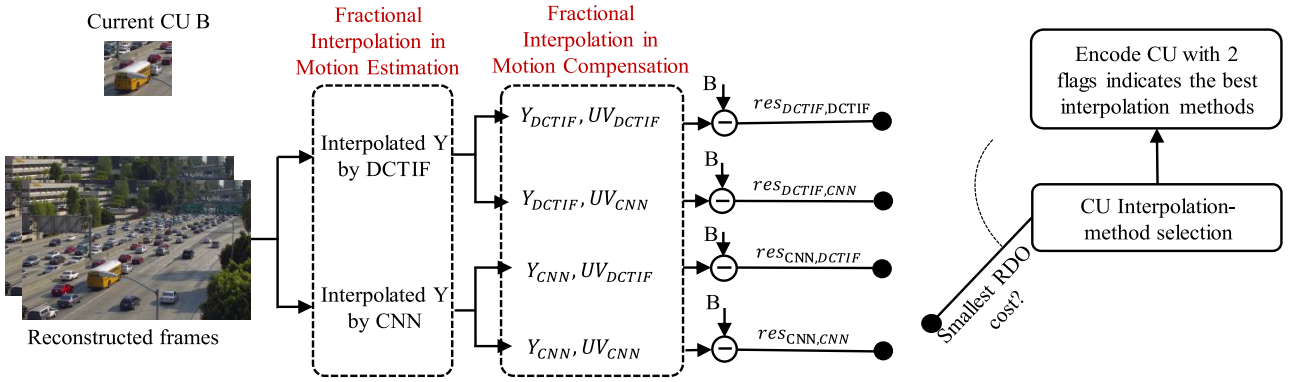


FIGURE 2. Visualization of our proposal. In Motion Search, the Y component of the reference frame is interpolated by DCTIF and CNN. In Motion compensation, U and V components of the reconstructed frame are interpolated by both CNN and DCTIF, and the Y component is interpolated by the method in motion search. The residual between current CU B and predicted CU are calculated and encoded with 2 bits indicate interpolation methods for Luma and Chroma components. Finally, an RDO-based fractional interpolation selection is implemented to decide which interpolation method should be used for Luma and Chroma fractional interpolation.

a popular training set includes a low-resolution image as input and the corresponding original high-resolution image as the ground-truth label. The prevailing method for creating input of CNN is doing down-sample the original image and up-sample the result for a low-resolution image [13], [14]. Due to the goal of reducing bitrate, MCP needs to predict the fractional-pixel image from the reconstructed frames, which makes the training set of super-resolution does not fit with the fractional interpolation task. Moreover, integer pixels could change after convolution, and fractional pixels do not exist in the real image are also the problems in creating a training set for applying the idea CNN-based super-resolution for fractional interpolation in video coding.

Frequently implemented, prior works encode the integer-position image and learn the mapping between reconstructed integer-position video and the fractional-position video. To solve the problem of integer pixel change after convolution, the work [16] generates a double-resolution image that includes integer and half-pixel and a mask that restricts integer pixel after convolution. The works [15], [17], [18], [20] separately generate half and quarter pixels based on integer pixels rather than generate double- or four-time-resolution image that includes integer and fractional pixels and keeps the integer pixel for MCP.

We implement an experiment to answer the question of what should be the ground-truth label for our training set. The target of MCP is to predict the current block to be encoded from the reference frames. However, the pair of the current block and reference block are achieved only in the encoding process and could be changed under different encoding configuration. Therefore, a training set of the current block and reference block may be restricted. In testing for ground-truth, we assume and extract integer and fractional pixels from the original video frames, do the encoding for integer images and set fractional images extracted from the original image as the interpolated image for encoding the down-sampled video. For this experiment, we test the first five frames of every sequence in class B, C, and E. For sequences in class B and E; the down-sampled videos do not fit the CU partition where HM cannot encode the down-sampled video, we then duplicate some rows and columns at the original video before downsampling. Experimental settings will be sufficiently described in IV-A. The results in Table 1 shows our BD-rate compare to the anchor HM under low delay P configuration.

TABLE 1. BD-rate reduction (%) of the replacing interpolated frame by the original frame when encoding down-sampled video compared to the anchor HEVC.

Class	Y	U	V
Average B	-39.3	-16.2	-16.5
Average C	-19.9	-8.4	-8.1
Average E	-23.7	-5.5	-5.6
Average all sequences	-27.6	-10.0	-10.0

Follow the successful of our experiment in finding a training set for our CNN, we then create a training set that takes the reconstructed integer video as input and the extracted fractional pixel as ground-truth, integer and fractional pixels are assumed from the original image. The process of training set generating visualized in Fig. 4 can be described as follow:

1) We extract the integer and fractional-position video by assuming integer and fractional pixels in every 4-by-4 non-overlapping blocks of each frame. Integer, half and quarter positions are pixels at similar positions to fractional samples in Fig. 1. We then obtain a low-resolution video of integer pixels (integer-position video) and 15 low-resolution videos including three half- and 12 quarter-position videos corresponding to 15 fractional samples.
2) Encode low-resolution video under low delay P configuration with QPs of 22, 27, 32 and 37 to get reconstructed down-sampled video.
3) Extract the Y component from reconstructed frames and interpolate them to 15 fractional samples by DCTIF. These 15 fractional samples are used as training input for our CNN.

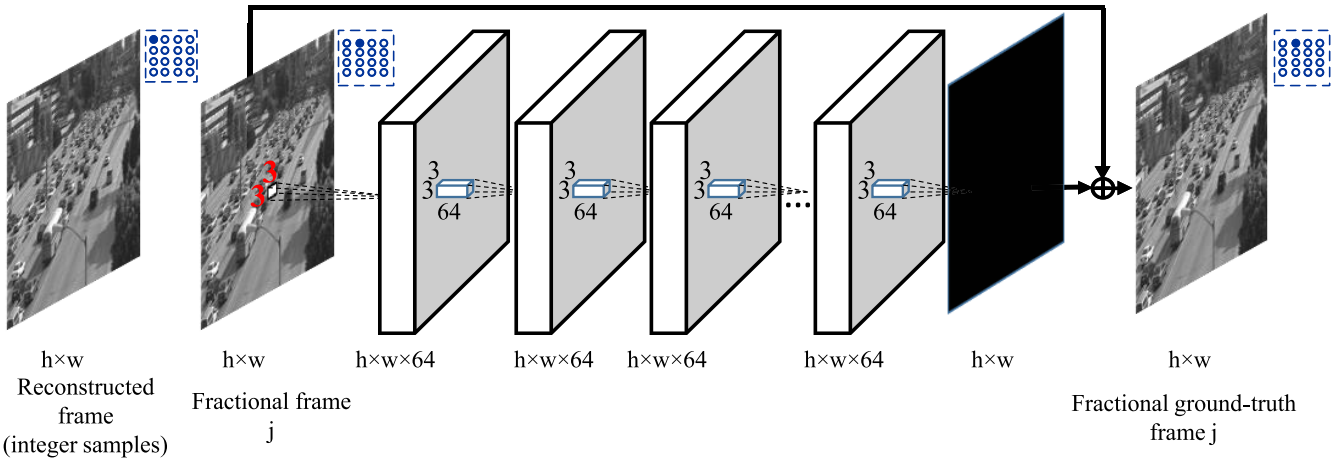


FIGURE 3. Our CNN architecture for learning all the fractional samples in Luma and Chroma components. For each feed, one fractional samples is fed into our CNN. For Luma component, *j* is from 1 to 15 in Luma component and 1 to 63 in Chroma component.

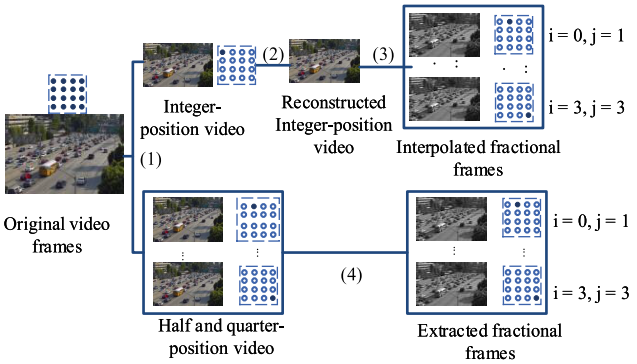


FIGURE 4. Our training set generation for learning fractional interpolation in video coding.

4) Extract the Y component from each fractional-position video frame to be the CNN ground-truth label for training. Each pair of the fractional sample interpolated by DCTIF and the fractional sample extracted from the original frame is considered as a training sample.

For generating the training set of the Chroma component, we do the down-sampling to one eight for all components because Chroma components require one eight fractional samples. All the processes for generating the Chroma training set are the same as the Luma Y component, except we have 63 fractional positions for Chroma components. Note that we use the default interpolation method DCTIF for the Y component when we generate a training set for Chroma components and vice versa.

We note that training a deep network with a small training set can cause overfitting which is not good for predicting test data. We then do the data augmentation for our training set. For each and every fractional-position frame, we flip and rotate input and the respective ground-truth label to avoid training biases.

C. NETWORK ARCHITECTURE

In inter coding, MCP interpolates reconstructed frames to fractional accuracy and finds a fractional pixel that is closest to the current frame to be encoded. To further improve the interpolation filter DCTIF in HEVC, in this paper, we design two CNN models for fractional pixel interpolation in Luma and Chroma components. As mentioned above, despite the facts that both of super-resolution and fractional interpolation in video coding need to increase the resolution of the input and CNN has remarkable results in super resolution, we cannot directly apply CNN-based super resolution for fractional interpolation in video coding. In this subsection, we introduce our network architecture, inspired by the Very Deep Super Resolution VDSR [14] for Luma and Chroma fractional interpolation. The network architecture (Fig. 3) includes 20 convolutional layers. Each layer does convolution by applying 64 3×3 filters with the stride of one. Padding is set as one to keep the size of the input image after convolution. For each convolutional layer, we set a ReLU activation layer after convolution except for the final layer. We set a learning rate that starts at 0.1 and decreases it by ten after ten epochs. Training stops at 50 epochs with a batch size of 128.

At training, our network takes an input of interpolated fractional sample and an ground-truth label of real fractional sample extracted from original frame, which are described in III-B. Given the reconstructed image x contains the integer pixels and the fractional-position frames y_j assumed and extracted from the original video frames, we apply DCTIF on reconstructed image x to obtain x'_j fractional samples where j is from one to 15 in Luma and one to 63 in Chroma components.

Each interpolated image x' by DCTIF are fed to into CNN for the output y'. Our network aims to learn the mapping between x'_j and y'_j by minimizing the loss function:

$$L(\theta) = \frac{1}{2}\sum_{i=1}^{n} ||y'_i - y_i||^2 \tag{2}$$

where n is number of training samples obtained from training set generation III-B. At testing, reconstructed images are interpolated to 15 fractional-samples image in Luma

component and 63 fractional-samples in Chroma components before feeding into CNN.

D. RDO-BASED INTERPOLATION MODE SELECTION

In video coding, selection between different encoding modes is an efficient tool to improve the image quality, decrease bitrate, or reduce the computational complexity of video coding standard such as HEVC. We note that DCTIF and CNN have the abilities to deal with different signals. To take advantage of DCTIF and CNN, we define two context models for a Luma and Chroma interpolation method selection. Our selection is based on RDO cost and be described as follows. In motion estimation, we do the interpolation for the Y component by DCTIF and CNN. For each interpolation method, we choose the best fractional sample and send the best fractional MV. In motion compensation, we then do the interpolation for all components by both DCTIF and CNN. There are four possibilities: DCTIF and DCTIF, DCTIF and CNN, CNN and DCTIF, CNN and CNN for Luma and Chroma component, respectively. After calculating residual and encoding all parameters including two bits for Luma and Chroma interpolation method flags, an RDO-based selection will choose among four possibilities the interpolation methods for each CU coded with fractional motion vector. Each and every sub-CU in the same CU, if coded with fractional motion vector, share the same interpolation methods.

IV. EXPERIMENTS

A. EXPERIMENTAL SETTINGS AND EVALUATION METHOD

For our experiments, we focus on two main parts: training and coding. HEVC Test Model (HM) version 16.18 is used for training set generation and demonstration of CNN-based fractional interpolation in video coding. The format of all the test sequences is YUV 4:2:0.

In our training, we use PyTorch 1.0.0 with the support of the NVIDIA Tesla V100 GPU. As mentioned earlier, we have trained two models for Luma and Chroma components for each quantization parameter (QP) value. Eight models have been trained for four QPs: 22, 27, 32, and 37. Training set for QP 32 and 37 models are acquired from three sequences *Pedestrian*, *Traffic* and *PeopleOntheStreet* [22]. For QP 22 and 27's models, we produce training set from *Traffic* and *PeopleOnTheStreet*. These sequences are obtained from HEVC standard test sequences. For training set generation, we encode our training sequences under Low Delay P configuration with full search is enable and an Intra period of 16 for a stable training set. Other parameters are set as default. We benefit from the residual training, which takes 10 hours to train Luma and eight hours for Choma component models. In generating fractional pixels for the Chroma component, some pixels are duplicated to match with the HM requirements. For training, the input image and ground-truth label from III-B are split into 41×41 patches with a stride of 16.

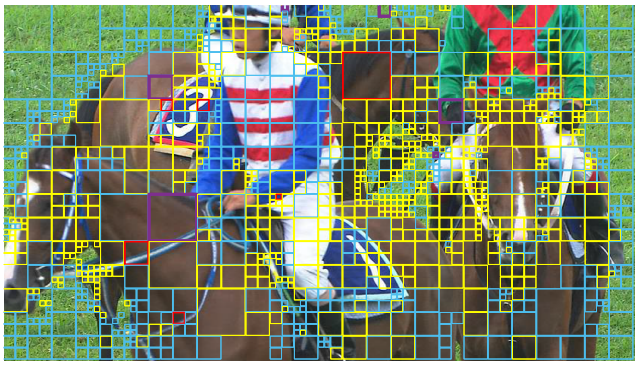


FIGURE 5. Visualization of our CU selection on *RaceHorseC*. In this figure, cyan, magenta, yellow, and red blocks indicate CUs that choose DCTIF for YUV interpolation, DCTIF for Y and CNN for UV interpolation, CNN for Y and DCTIF for UV interpolation, and CNN for YUV interpolation, respectively. The rest parts are CUs that choose inter coding with integer motion vector or intra coding.

The test is conducted under Low Delay P, Low Delay B and Random Access configurations. We set all coding configuration as default except full search is enabled. Because Bitrate and Peak signal-to-noise ratio (PSNR) are all need to be considered in evaluating our method, we use a well-known measuring metric, Bjøtegaard-Delta bit-rate (BD-rate) [23]. BD-rate takes at least four samples from different video coding techniques and tells how many bits are reduced between them at the same video quality. In our case, we compare our proposals obtained from four QPs.

B. EXPERIMENTAL RESULTS

1) RDO-BASED INTERPOLATION METHOD SELECTION RESULT

In our proposal, RDO cost-based interpolation-method selection for each CU has been implemented in encoding. When a CU is coded with a motion vector that has the fractional part, two bits indicate the flags for Luma and Chroma interpolation methods are encoded. Fig. 5 shows our visualization on the first P frame of RaceHorseC under the Low Delay P configuration. In our visualization, cyan blocks indicate CU that choose DCTIF for interpolating all components, magenta blocks indicate CUs that choose DCTIF for Y and CNN for UV components, yellow blocks indicate CUs that choose CNN for Y and DCTIF for UV components, and red blocks indicate CUs that choose CNN for interpolating all components. The rest parts are CUs coded with integer motion vector or intra coding. As our visualizations, CUs at the static background tend to choose DCTIF for fractional interpolation and CUs at moving object tend to choose CNN for fractional interpolation.

On another hand, we measure the ratio of choosing interpolation methods for each CU. In HEVC, CU size is variety, which makes the calculating hitting ratio should be on the area than on count. Since these two flags are dependent, we calculate the hitting ratio on two flags as one than separately counting. In Table 2, we show the hitting ratio of using CNN

TABLE 2. Hitting ratio (%) of two interpolation methods for Luma and Chroma component under Low Delay P configuration.

Class	$DCTIF_Y$ $DCTIF_{UV}$	$DCTIF_Y$ CNN_{UV}	CNN_Y $DCTIF_{UV}$	CNN_Y CNN_{UV}
B	54.51	0.51	44.36	0.62
C	53.18	0.78	45.07	0.97
D	57.75	0.47	41.34	0.44
E	64.18	0.32	34.71	0.79
F	71.84	1.51	26.14	0.51
All	60.29	0.72	38.32	0.67

and DCTIF for Luma and Chroma components at CU-level. Class F, where Luma and Chroma components rarely choose CNN for fractional interpolations, obtains the lowest bitrate saving compare to other classes.

2) COMPARISON WITH EXISTING WORKS

We also experiment to compare our proposal to existing CNN-based fractional interpolation works. For a fair comparison, we reimplement our proposal on HM 16.7 and disable the CNN-based fractional interpolation and the context model for the Chroma component. In this experiment, only Y component is interpolated by CNN, and DCTIF is used for interpolating Chroma components. The selection of CNN/DCTIF fractional interpolation is also implemented for the Y component, and the flag for the interpolation method is encoded in CABAC regular mode. In this comparison, we use the reimplemented results from paper [19].

The results (Table 3) shows that our proposal surpasses the GVTCNN [18] and the switch mode-based fractional interpolation [19] on HM-16.7. In general, we achieve a 3.7 BD-rate reduction on average and rank first all Classes from B to E.

TABLE 3. Comparison of CNN-based fractional interpolation and our proposal under Low Delay P configuration.

Class	[18]	[19]	Ours
Average B	-3.0	-3.0	-3.8
Average C	-1.7	-2.7	-3.0
Average D	-1.5	-2.5	-3.5
Average E	-1.9	-2.8	-4.6
Average all sequences	-2.1	-2.8	-3.7

We also compare our work with the works [15], [16]. In these works, only half pixels are interpolated by CNN, quarter pixels are interpolated by DCTIF. In this comparison, we also re-implement our work on HM 16.7 and only half pixels are interpolated by CNN. The results (Table 4) shows that our half-pixel interpolation outperforms CNNIF [15] and *Zhang*'s work [16] in average.

3) OVERALL RESULTS

Our experiments under Low Delay P, Low Delay B, and Random Access configurations are shown in Table 5. Generally, we obtain the highest BD-rate reduction on Low Delay P

TABLE 4. Comparison of CNN-based half-pixel interpolation and our proposal under Low Delay P configuration (anchor: HM 16.7).

Class	Sequence	CNNIF [15]	[16]	Ours (half)	Ours (half and quarter)
B	Kimono	-1.1	-	-4.4	-4.9
	ParkScene	-0.4	-	-0.8	-0.1
	Cactus	-0.8	-	-3.2	-4.6
	BasketbalDrive	-1.3	-	-3.4	-3.7
	BQterrace	-3.2	-	-3.9	-5.7
C	BasketballDrill	-1.2	-0.9	-3.7	-4.8
	BQMall	-0.9	-0.3	-1.8	-2.5
	PartyScene	0.2	-0.1	-1.2	-1.8
	RacehorsesC	-1.5	-0.4	-2.4	-3.0
D	BasketballPass	-1.3	-1.2	-2.3	-3.5
	BQSquare	1.2	0.3	-1.8	-3.9
	BlowingBubbles	-0.3	-0.2	-2.6	-3.1
	RaceHorses	-0.8	-0.7	-2.9	-3.7
E	FourPeople	-1.3	-0.6	-3.8	-4.3
	Johnny	-1.2	-0.7	-4.7	-5.4
	KristenAndSara	-1.0	-0.4	-3.6	-3.9
F	BasketballDrillText	-1.4	-0.4	-3.4	-4.2
	ChinaSpeed	-0.6	-0.6	-0.6	-0.8
	SlideEditing	0.0	-0.0	-0.2	-0.1
	SlideShow	-0.7	-0.2	-0.2	-0.6
	Average B	-1.4	-	-3.2	-3.8
	Average C	-0.9	-0.4	-2.3	-3.0
	Average D	-0.3	-0.5	-2.4	-3.5
	Average E	-1.2	-0.6	-4.0	-4.6
	Average F	-0.7	-0.3	-1.1	-1.4

configuration and the lowest saving bitrate belongs to the test under Random Access configuration.

We obtain a 2.9, 0.3, and 0.6 % Y, U, and V BD-rate saving compared to the original HM under Low Delay P configuration and up to 6.5%, 2.1%, 2.9 % Y, U, V BD-rate reduction on sequence *BQTerrace*. Results show that the proposal can deal with high-resolution videos such as sequences in class A and E where average BD-rate reductions for Y component are over 4%. We can obtain a high and stable BD-rate reduction for all components of class E sequences where backgrounds are static. Although Y, U, and V components are trained, and an RDO-based interpolation method selection has been implemented, there is some space that our methods cannot improve. For example, it can be seen that our proposal does not work well on screen-content sequence *ChinaSpeed*, *SlideEditing*, and *SlideShow* under Low Delay P configuration since no data for these content has been trained. The future work may include training for the screen- content video data.

For Low Delay B configuration results, the best performance belongs to class E, where 3.3%, 0.6%, and 0.7% Y, U, and V BD-rate reductions are obtained, respectively. Although our models do not work well on screen-content videos of class F under Low Delay P configuration, they acquire the average BD-rate reduction of 0.4%, 0.6%, and 0.8% on ChinaSpeed, SlideEditing, and SlideShow under Low Delay B configuration.

For Random Access configuration, we achieve 1.2%, 0.2%, and 0.2% BD-rate reduction on Y, U, and V components, respectively. This experiment obtains a Y BD-rate

TABLE 5. BD-rate (%) of our proposal compared to HEVC under Low Delay P, Low Delay B and Random Access configurations.

Class	Sequence	Low Delay P			Low Delay B			Random Access		
		Y (%)	U (%)	V (%)	Y (%)	U (%)	V (%)	Y (%)	U (%)	V(%)
B	Kimono	-4.7	1.1	0.6	-1.1	1.3	0.6	-0.7	0.6	0.0
	ParkScene	-1.3	1.1	0.2	-0.4	0.7	0.4	-0.5	0.2	0.0
	Cactus	-4.2	-1.6	-1.9	-3.2	-1.1	-0.6	-2.3	-0.5	-0.9
	BasketbalDrive	-4.2	-1.6	-1.9	-1.4	0.1	-0.8	-1.7	-0.1	0.3
	BQterrace	-6.5	-2.1	-2.9	-2.5	-0.2	-0.6	-1.6	-0.2	-0.2
C	BasketballDrill	-4.0	-0.2	-1.4	-3.1	0.5	-0.2	-1.5	-0.6	-0.9
	BQMall	-2.0	0.0	-0.3	-1.7	-0.1	-0.4	-0.9	-0.2	-0.4
	PartyScene	-1.8	-0.9	-0.6	-1.1	-0.2	-0.1	-0.6	-0.2	-0.5
	RacehorsesC	-2.6	-0.7	-0.3	-2.1	-0.1	-0.5	-1.6	-0.6	-1.5
D	BasketballPass	-2.6	-0.8	-0.7	-2.2	-1.6	-1.8	-1.1	0.2	-0.2
	BQSquare	-4.2	-1.6	-1.3	-2.2	-0.7	-0.4	-0.9	0.4	0.1
	BlowingBubbles	-2.5	-0.5	-1.3	-2.7	-0.8	-0.6	-1.0	-0.9	0.1
	RaceHorses	-2.9	0.3	-1.0	-2.8	-0.1	-1.4	-1.4	-0.8	-0.9
E	FourPeople	-4.3	-0.3	-0.4	-3.8	-0.2	-0.7	-2.9	-0.1	-0.4
	Johnny	-5.7	-2.3	-1.5	-2.6	-0.5	-0.5	-2.1	-0.1	-0.4
	KristenAndSara	-4.1	-0.9	-1.6	-3.4	-1.1	-0.8	-2.2	-0.3	-0.5
F	BasketballDrillText	-3.3	1.1	-0.1	-3.0	0.7	0.4	-1.0	-0.5	-1.2
	ChinaSpeed	1.2	1.6	1.4	-0.6	-0.2	-0.4	-1.0	-0.9	-0.8
	SlideEditing	1.3	1.0	1.0	-0.2	-0.2	-0.2	0.2	0.4	0.2
	SlideShow	1.0	0.1	1.6	-0.5	-1.6	-2.0	-0.4	0.2	4.7
Class Sumarry	Average B	-4.3	-0.5	-1.1	-1.7	0.2	-0.2	-1.4	0.0	-0.2
	Average C	-2.6	-0.5	-0.6	-2.0	0.0	-0.3	-1.1	-0.4	-0.8
	Average D	-3.0	-0.6	-1.1	-2.5	-0.8	-1.0	-1.1	-0.3	-0.2
	Average E	-4.7	-1.1	-1.2	-3.3	-0.6	-0.7	-2.4	-0.2	-0.5
	Average F	0.1	1.0	1.0	-1.1	-0.3	-0.5	-0.6	-0.2	0.7
	Average all sequences	-2.9	-0.3	-0.6	-2.0	-0.3	-0.5	-1.2	-0.2	-0.2

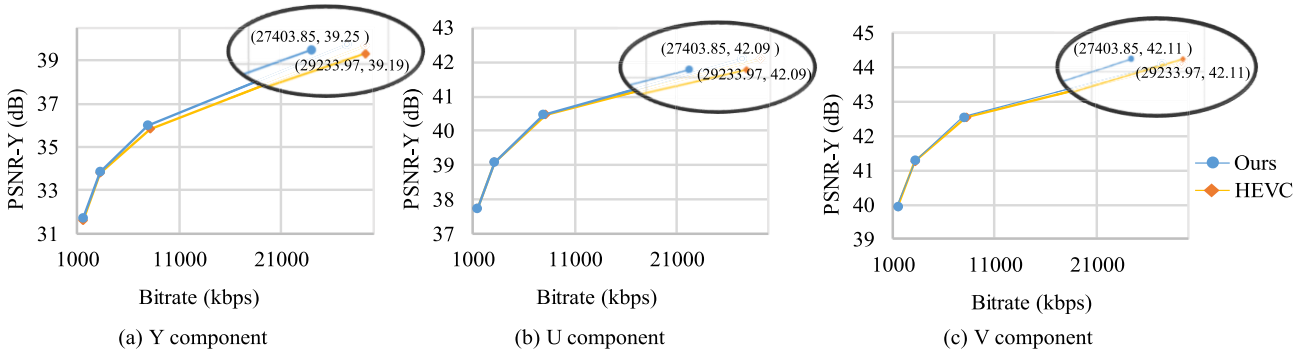


FIGURE 6. R-D curves of sequence BQTerrace under Low Delay P configuration on (a) Y component, (b) U component and (c) U component.

reduction up to 2.9% at sequence *FourPeople*. Along with the configurations that we do experiments, Random Access obtains the lowest bit rate saving at all components.

To further investigate the quality of our method compare to HEVC, we visualize RD-curves (Fig. 6) of our method on Y, U, and V components of sequence BQTerrace. It can be seen that our proposal can significantly increase the PSNR at the high bit rates more than at lower bit rates.

We also experiment to figure the effect of the separate network on Chroma component. In this experiment, we train four models for each Chroma component at the separate test. Eight models have been trained in the combining experiment, and twelve models have been trained on the separate experiment. The result of testing training models (Table 6) shows that our separate test archives better BD-rate reduction on U and V components than the combining experiment.

TABLE 6. BD-rate (%) of our proposals in seperating models for U and V compared to HEVC under Low Delay P configuration.

Class	Combining models			Separate models		
	Y (%)	U (%)	V (%)	Y (%)	U (%)	V (%)
Average B	-4.3	-0.5	-1.1	-4.3	-1.1	-3.2
Average C	-2.6	-0.5	-0.6	-2.4	-0.2	-2.6
Average D	-3.0	-0.6	-1.1	-3.0	-1.0	-3.0
Average E	-4.7	-1.1	-1.2	-4.7	-1.2	-2.8
Average F	0.1	1.0	1.0	0.0	0.7	0.0
Average all sequences	-2.9	-0.3	-0.6	-2.8	-0.6	-2.3

However, Y BD-rate is decreased by 0.1% at separate models even models for Y component are the same. For more detail, training separate models can archive up to 6.6% for Y, 3.0% for U, and 5.1% for V BD-rate reduction in sequence BQTerrace.

V. CONCLUSIONS

In this paper, we propose a deep learning-based method for fractional interpolation in video coding and design a training set for our CNN models. In our proposal, Luma and Chroma components are interpolated by both DCTIF and CNN, and an RDO cost-based interpolation method selection chooses among them the best fractional interpolation methods for Luma and Chroma components at CU level. As a result, we obtain an average BD-rate reduction of 2.9%, 0.3%, and 0.6% on Y, U, and V component, respectively, under low Delay P configuration. We also show the effects of training the separate models or combining models for U and V components and a comparison of our method compared to the existing works.

REFERENCES

[1] 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.

[2] 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.

[3] J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards—Including high efficiency video coding (HEVC)," *IEEE Trans. circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1669–1684, Dec. 2012.

[4] H. Lakshman, B. Bross, H. Schwarz, and T. Wiegand, "Fractional-sample motion compensation using generalized interpolation," in *Proc. 28th Picture Coding Symp.*, Dec. 2010, pp. 530–533.

[5] H. Lakshman, H. Schwarz, and T. Wiegand, "Generalized interpolation-based fractional sample motion compensation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 3, pp. 455–466, Mar. 2013.

[6] Y. Ye, G. Motta, and M. Karczewicz, "Enhanced adaptive interpolation filters for video coding," in *Proc. Data Compress. Conf.*, Mar. 2010, pp. 435–444.

[7] S. Wittmann and T. Wedi, "Separable adaptive interpolation filter for video coding," in *Proc. 15th IEEE Int. Conf. Image Process.*, Oct. 2008, pp. 2500–2503.

[8] Z. Guo, D. Zhou, and S. Goto, "An optimized MC interpolation architecture for HEVC," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2012, pp. 1117–1120.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[10] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 2147–2154.

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[12] 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.

[13] 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.* Zürich, Switzerland: Springer, 2014, pp. 184–199.

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

[15] 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. (ISCAS)*, May 2017, pp. 1–4.

[16] 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. (VCIP)*, Dec. 2017, pp. 1–4.

[17] N. Yan, D. Liu, H. Li, B. Li, L. Li, and F. Wu, "Convolutional neural network-based fractional-pixel motion compensation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 3, pp. 840–853, Mar. 2018.

[18] S. Xia, W. Yang, Y. Hu, S. Ma, and J. Liu, "A group variational transformation neural network for fractional interpolation of video coding," in *Proc. Data Compress. Conf.*, Mar. 2018, pp. 127–136.

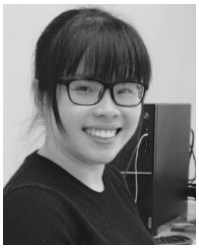
[19] S. Xia, W. Yang, Y. Hu, W.-H. Cheng, and J. Liu, "Switch mode based deep fractional interpolation in video coding," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.

[20] J. Liu, S. Xia, W. Yang, M. Li, and D. Liu, "One-for-all: Grouped variation network-based fractional interpolation in video coding," *IEEE Trans. Image Process.*, vol. 28, no. 5, pp. 2140–2151, May 2018.

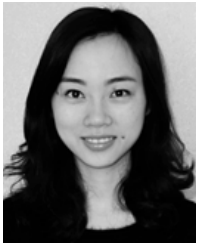
[21] C. Pham and J. Zhou, "ICNN: A convolutional neural network for fractional interpolation in video coding," in *Proc. Int. Symp. Artif. Intell. Robot.*, to be published.

[22] F. Bossen,*Common Test Conditions Software Reference Configurations*, document JCTVC-J1100, Stockholm, Sweden, Jul. 2012.

[23] G. Bjontegaard, *Calculation of Average PSNR Differences Between RD-Curves*, document VCEG-M33, 2001.



CHI DO-KIM PHAM received the B.S degree from the University of Information Technology (UIT), Vietnam National University, Ho Chi Minh City, Vietnam, in 2017. She is currently pursuing the IIST master's degree with Hosei University, Japan, where she is currently with the MMLab-Hosei. Her research interests include deep learning, video coding, and image processing.



JINJIA ZHOU received the B.E. degree from Shanghai Jiao Tong University, China, in 2007, and the M.E. and Ph.D. degrees from Waseda University, Japan, in 2010 and 2013, respectively, where she was a Junior Researcher, from 2013 to 2016. From 2017 to 2019, she was also a Senior Visiting Scholarship with the State Key Laboratory of ASIC and System, Fudan University, China. She is currently an Associate Professor with Hosei University, Tokyo, Japan. She was selected as a Research Fellow by the Japan Society for the Promotion of Science, during 2010 to 2013. She is selected as JST PRESTO Researcher, during 2017 to 2021. Her research interests include algorithms and VLSI architectures for multimedia signal processing.