Research on Deep Learning-based Fractional Interpolation in Video Coding

Research on Deep Learning-based Fractional Interpolation in Video Coding

17R8101 Pham Do Kim Chi
Supervisor: Jinjia Zhou

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 CNN in super-resolution, we propose Convolutional Neural Network-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.

Keywords— *Video coding, motion compensated prediction, fractional interpolation*

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. 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, 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. However, the HEVC filters are not flexible enough for video data.

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 detection to low-level computer vision tasks like image denoising, de-blurring, and super-resolution. Despite the robust of CNN in improving super-resolution, they cannot 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, 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.

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, the work [5]–[9] aims to produce fractional pixels from an input of integer pixels, and then, integer pixels are kept. Zhang et al. [10] introduces a CNN model followed by a Constrained mask with different weights for the integer pixels and three half pixels.

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 [3]–[6] or the mapping between the interpolated frame of the reconstructed reference frame and the original reference image [7]. 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 [8].

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 [6], [7], many models need to be trained for fractional positions [8], or predicting fractional pixel from integer pixel may not good because the motion shift between integer and fractional pixel are not always stable [3], [5], [6], [8]. In this paper, we take the next step towards the CNN-based fractional interpolation in video coding: all components can be processed by CNN. Our work makes the following three contributions: First, we present two CNN models for Luma and Chroma fractional pixels

interpolation in video coding. Only one model is trained for 15 fractional samples at each QP. Totally, there are eight models for four QPs in Luma and Chroma components. Secondly, we investigate a dataset generation method for our Y, U, V fractional interpolation training. Finally, we implement an RDO-based selection for Luma and Chroma fractional interpolation. Each syntax element indicates the interpolation method for Luma or Chroma components for each CU that chooses inter coding with the fractional motion vector. As a result, we archive 2.9%, 0.3%, 0.6% BD-rate reduction compared to the anchor HEVC under low delay P configuration.

II. Proposed CNN-based Luma and Chroma Fractional Interpolation

A. Training set generation

Training set plays a vital role in training any network. We then do the experiment to figure out what should be ground truth for our training. In testing for ground-truth, we down-sample 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 experimentwe achieve 27.6%, 5.5%, 5.6% BD-rate reduction on Y, U, and V components compare to the anchor HM under low delay P configuration. Follow the successful of our experiment in finding a training set for our CNN, our training set generation method (Figure 2) can be described as follow:

- 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. We then obtain a low-resolution video of integer pixels (integer-position video) and 15 low-resolution videos.
- Encode low-resolution video to get reconstructed down-sampled video. 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.
- 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.

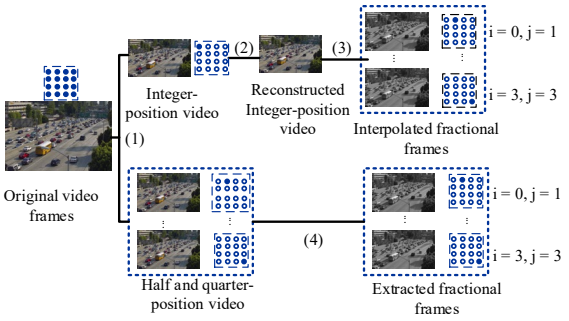

Figure 2: Our training data generation method.

For generating the training set of the Chroma component, we do the down-sampling to one eight 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.

For each and every fractional-position frame, we flip and rotate input and the respective ground-truth label to avoid training biases and overfitting.

B. Network Architecture

In this paper, we design two CNN models for fractional pixel interpolation in Luma and Chroma components. We train a CNN model for all the fractional samples in the Luma-component model and keep the same architecture but a different training set for the Chroma-component model.

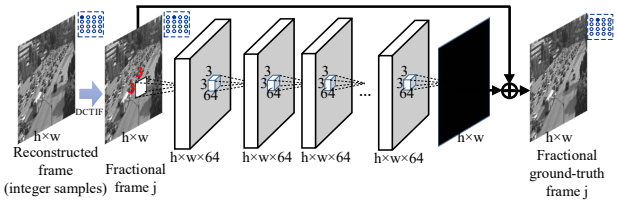

Figure 1. Our network architecture for Luma and Chroma fractional interpolation in video coding.

Our network, inspired by VDSR [9], 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.

At testing, reconstructed images are interpolated to 15 fractional-sample images in Luma component and 63 fractional-samples in Chroma components before feeding into CNN.

C. RDO-based Luma and Chroma interpolation selection

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. One bit indicates interpolation method for each context model will be store in CABAC regular mode. Each and every sub-CU in the same CU, if coded with fractional motion vector, share the same interpolation method for Luma and Chroma components.

III. Results

For our experiments, we use HEVC Test Model (HM) version 16.18 and 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,

TABLE 1. BD-RATE (%) OF OUR PROPOSAL COMPARED TO HEVC UNDER LOW DELAY P, LOW DELAY B AND RANDOM ACCESS CONFIGURATIONS ON CLASS B, C, D, E

Class	Sequence	Low Delay P			Low Delay B			Radom 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
	Avg	-3.6	-0.6	-1.0	-2.3	-0.2	-0.5	-1.4	-0.2	-0.4

TABLE 2. BD-RATE (%) OF OUR PROPOSAL COMPARED TO HEVC UNDER LOW DELAY P, LOW DELAY B AND RANDOM ACCESS CONFIGURATIONS ON CLASS F

Sequence	Low Delay P			Low Delay B			Radom Access		
	Y	*U*	*V*	*Y*	*U*	*V*	*Y*	*U*	*V*
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
Avg	0.1	1.0	1.0	-1.1	-0.3	-0.5	-0.6	-0.2	0.7

and 37. Training set for QP 32 and 37 models are acquired from three sequences Pedestrian, Traffic and PeopleOntheStreet. For QP 22 and 27's models, we produce training set from Traffic and PeopleOnTheStreet. We set all coding configuration as default except full search is enabled. We use a well-known measuring metric Bjøtegaard-Delta bitrate (BD-rate) to evaluate our proposal. 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. For BD-rate, the low negative number means the better result.

A. RDO-based Luma and Chroma fractional interpolation results

Our experiments under Low Delay P, Low Delay B, and Random Access configurations are shown in TABLE *1*. Generally, we obtain the highest BD-rate reduction on Low Delay P configuration and the lowest saving bitrate belongs to the test under Random Access configuration.

We obtain 3.6%, 0.6% and 1% Y, U, and V BD-rate saving compared to the original HM under Low Delay P configuration. Results show that 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%.

For Low Delay B configuration results, the best performance belongs to FourPeople class E, where 3.8%, 0.2%, and 0.7% Y, U, and V BD-rate reductions are obtained, respectively.

For Random Access configuration, we achieve 1.4%, 0.2%, and 0.4% BD-rate reduction on Y, U, and V components, respectively.

Generally, we obtain the highest BD-rate reduction on Low Delay P configuration and the lowest bitrate saving belongs to the test under Random Access configuration.

In TABLE *2*, 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 screen-content has been trained. The future work may include training for the screen-content video data for Low Delay P configuration. Although our models do not work well on screen-content videos under Low Delay P

configuration, they acquire the average BD-rate reduction of 1.1%, 0.3%, and 0.5% on over class E under Low Delay B configuration.

B. RDO-based Luma and Chroma fractional interpolation results

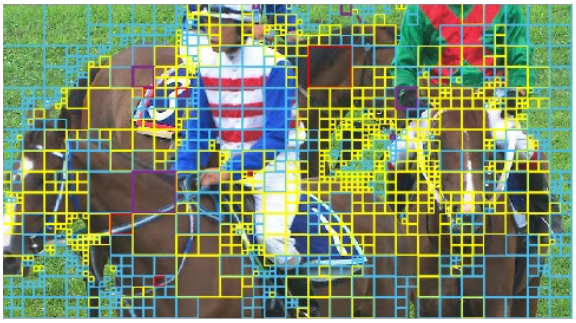

Figure 3. Visualization on Luma and Chroma interpolation method selection.

Figure 2 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.

TABLE 3. HITTING RATIO (%) OF THE TWO INTERPOLATION METHODS FOR LUMA AND CHROMA COMPONENTS

Class	$\mathbf{DCTIF_Y}$ $\mathbf{DCTIF_{UV}}$	$\mathbf{DCTIF_Y}$ $\boldsymbol{CNN_{UV}}$	$\mathbf{CNN_Y}$ $\mathbf{DCTIF_{UV}}$	$\boldsymbol{CNN_Y}$ $\mathbf{CNN_{UV}}$
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

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 3, we show the hitting ratio of using CNN 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.

C. Compare with the 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 comparison, we use the reimplemented results from paper [4].

TABLE 4. BD-RATE COMPARISON OF CNN-BASED FRACTIONAL INTERPOLATION AND OUR PROPOSAL .

Class	[5]	[4]	Ours
B	-3.0	-3.0	-3.8
C	-1.7	-2.7	-3.0
D	-1.5	-2.5	-3.5
E	-1.9	-2.8	-4.6
Avg.	-2.1	-2.8	-3.7

The results (TABLE *4*) shows that our proposal surpasses the GVTCNN [5] and the switch mode-based fractional interpolation [4] 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.

IV. CONCLUSION

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 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 Transactions on Circuits and Systems for Video Technology*, 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. Cir. and Sys. for Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[3] J. Liu, S. Xia, W. Yang, M. Li, and D. Liu, "One-for-All: Grouped Variation Network-Based Fractional Interpolation in Video Coding," *IEEE Transactions on Image Processing*, vol. 28, no. 5, pp. 2140–2151, May 2019.

[4] S. Xia, W. Yang, Y. Hu, W. Cheng, and J. Liu, "Switch Mode Based Deep Fractional Interpolation in Video Coding," in *2019 IEEE Int. Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[5] S. Xia, W. Yang, Y. Hu, S. Ma, and J. Liu, "A Group Variational Transformation Neural Network for Fractional Interpolation of Video Coding," in *2018 Data Compression Conference*, 2018, pp. 127–136.

[6] N. Yan, D. Liu, H. Li, and F. Wu, "A convolutional neural network approach for half-pel interpolation in video coding," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.

[7] H. Zhang, L. Song, Z. Luo, and X. Yang, "Learning a convolutional neural network for fractional interpolation in HEVC inter coding," in *2017 IEEE Visual Communications and Image Processing (VCIP)*, 2017, pp. 1–4.

[8] N. Yan, D. Liu, H. Li, B. Li, L. Li, and F. Wu, "Convolutional Neural Network-Based Fractional-Pixel Motion Compensation," *IEEE Trans. Circuits and Systems for Video Technology*, pp. 1–1, 2018.

[9] J. Kim, J. K. Lee, and K. M. Lee, "Accurate Image Super-Resolution Using Very Deep Convolutional Networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1646–1654.