A Convolutional Neural Network Approach for Half-Pel Interpolation in Video Coding

Ning Yan, Dong Liu*, Houqiang Li, and Feng Wu
CAS Key Laboratory of Technology in Geo-Spatial Information Processing and Application System, University of Science and Technology of China, Hefei 230027, China
nyan@mail.ustc.edu.cn, {dongeliu,lihq,fengwu}@ustc.edu.cn

Abstract—Motion compensation is a fundamental technology in video coding to remove the temporal redundancy between video frames. To further improve the coding efficiency, sub-pel motion compensation has been utilized, which requires interpolation of fractional samples. The video coding standards usually adopt fixed interpolation filters that are derived from the signal processing theory. However, as video signal is not stationary, the fixed interpolation filters may turn out less efficient. Inspired by the great success of convolutional neural network (CNN) in computer vision, we propose to design a CNN-based interpolation filter (CNNIF) for video coding. Different from previous studies, one difficulty for training CNNIF is the lack of ground-truth since the fractional samples are actually not available. Our solution for this problem is to derive the “ground-truth” of fractional samples by smoothing high-resolution images, which is verified to be effective by the conducted experiments. Compared to the fixed half-pel interpolation filter for luma in High Efficiency Video Coding (HEVC), our proposed CNNIF achieves up to 3.2% and on average 0.9% BD-rate reduction under low-delay P configuration.

1. INTRODUCTION

Motion compensation is a fundamental technology in video coding to remove the temporal redundancy between video frames. The existing video coding standards, including High Efficiency Video Coding (HEVC), mostly adopt block-based motion compensation for inter prediction, which assumes the block to be coded can be retrieved from the previously coded frames, and the corresponding location of the retrieved block is indicated by a motion vector (MV). Due to the inherent spatial sampling of digital video, MV is probably not integer, and thus the corresponding block may need to be generated instead of simply retrieved. Typically, generating non-integer samples is performed by kinds of interpolation, which can be viewed as fitting a continuous curve through a set of discrete samples, and picking out the new values at specific positions on the curve.

The problem of fractional sample interpolation for video coding has been widely studied [1]. A theoretical analysis is conducted in [2] about the influence of fractional-pel accuracy on the efficiency of motion compensated prediction, using a Gaussian power spectral density model. Practically, the video coding standards mostly adopt fixed interpolation filters. For example, MPEG-4 AVC/H.264 uses the 6-tap filter to perform half-pel interpolation and the simple average filter to perform quarter-pel interpolation for luma component [3]. In HEVC, the DCT-based interpolation filter (DCTIF) is adopted [4]. HEVC uses a “7q+8h” DCTIF for luma component, that is a 7-tap DCTIF used for quarter-pel samples, and a 8-tap DCTIF for half-pel samples. Lv et al. [5] give the derivation process of DCTIF in detail, and compare the frequency responses between the interpolation filters in HEVC and H.264.

The fixed interpolation filters have been designed according to the signal processing theory, with the premise that the video signal is an ideal low-pass one. However, the video signal is indeed not low-pass and not stationary. Further study has been conducted to design different interpolation filters. For example, a motion compensated hybrid video coding scheme using an adaptive filter is presented in [6], where the filter coefficients are estimated during motion compensation for each frame. Wittmann et al. [7] present a separable adaptive interpolation filter to reduce the computational cost while maintaining the coding efficiency of non-separable adaptive filter. Nonetheless, the previous works adopted hand-crafted filters, leaving a space for further improving the accuracy of motion compensation.

Recently, deep learning has achieved great success in computer vision. Convolutional neural network (CNN) based models led to a series of breakthroughs in high-level computer vision tasks, such as image classification [8] and object detection [9]. Later on, CNN is also utilized in some low-level computer vision tasks. For example, Dong et al. [10] propose a CNN approach for image super-resolution, termed SRCNN, which learns an end-to-end mapping between low- and high-resolution images. SRCNN has achieved significant boost of performance in both subjective and objective quality, compared to the previous methods without CNN. SRCNN is then extended to cope with the problem of artifact reduction [11]. More recently, Dai et al. [12] propose to learn a CNN for post-processing in video coding, and demonstrate on average 4.6% bit-rate reduction than HEVC baseline. All these works seem to open up a new direction that adopts CNN into video coding to further improve the coding efficiency.

In this paper, we present a CNN approach for fractional sample interpolation in video coding. We expect the CNN to automate the discovery of interpolation filters rather than to design them manually. However, a key difficulty here is how to generate training data for CNN. In previous studies concerning CNN, the ground-truth labels are provided comprehensively in the training data. However, for fractional sample interpolation, we have no ground-truth label because the fractional samples
actually do not exist. Our solution for this problem is to derive the labels by smoothing high-resolution images, which is verified to be effective in experiments. We then reuse the network architecture of SRCNN, considering the similarity between super-resolution and fractional interpolation, but train the network with our derived training data. After training, the CNN-based interpolation filter (CNNIF) is integrated into HEVC for testing its performance in video coding. Currently, the CNNIF is applied only to half-pel samples of luma component. Experimental results show that the proposed CNNIF leads to up to 3.2% and on average 0.9% bits saving under low-delay P configuration.

The remainder of this paper is organized as follows. Section II provides the details of the proposed CNNIF. Section III gives the experimental results, and Section IV concludes the paper.

II. PROPOSED METHOD

A. Overview of the Proposed Method

In this work, we propose a CNN approach for half-pel interpolation of luma component. Fig. 1 illustrates the integer and half-pel positions during interpolation. The positions labeled with \( A_{i,j} \) stand for the available luma samples at integer locations, whereas the other positions labeled with \( b_{i,j}, h_{i,j} \) and \( j_{i,j} \) represent samples at half-pel locations, which need to be interpolated from integer-location samples. In HEVC, these three positions are derived using uniform 8-tap DCTIF. Taking into account the positional relation between the integer and the half-pel samples, we propose to train three CNN models to perform the interpolation of horizontal half-pel, vertical half-pel, and diagonal half-pel samples, respectively. That is, we train CNNIF_H, CNNIF_V and CNNIF_D, for the interpolation of \( b_{i,j}, h_{i,j} \), and \( j_{i,j} \), respectively.

It is very important to have abundant training data in store for supervised deep learning. However, in this fractional interpolation task, we cannot obtain the real ground-truth for training, i.e., the fractional samples do not exist actually. Without the ground-truth, it is not possible to carry out the training task, let alone expecting performance improvement in video coding. To overcome this contradiction, in this work, we design a method for generating training data, which will be introduced in Section II-C.

B. Network Architecture

In this work, we reuse the existing Convolutional Neural Network for Super-Resolution (SRCNN) in [10] to carry out the half-pel interpolation task. Fig. 2 depicts the architecture of SRCNN.

SRCNN consists of three convolutional layers. The output of \( i \)-th layer \((i = 1, 2)\) is the result of a linear transform of the previous layer followed by a Rectified Linear Unit (ReLU) [13], and this process can be expressed as:

\[
F_i = \max(0, W_i \ast F_{i-1} + B_i) \tag{1}
\]

where \( W_i \) and \( B_i \) are the convolutional filter kernel and bias of the \( i \)-th layer, respectively, and ‘\( \ast \)’ means convolutional operation. For SRCNN, the three layers are claimed to perform three steps, respectively [10]:

- The first layer is used for patch extraction and representation, extracting the features from low-resolution image. Here, \( W_1 \) is of size \( 9 \times 9 \times 64 \) and \( B_1 \) is a 64-dimensional vector.
- The second layer can be seen as non-linear mapping, which converts the features of low-resolution image to those of high-resolution. Here, \( W_2 \) is of size \( 1 \times 1 \times 32 \) and \( B_2 \) is a 32-dimensional vector.
- The third layer, where \( W_3 \) is of size \( 5 \times 5 \times 1 \), is used to recover the high-resolution image from the high-resolution features.

In the super-resolution task, a low-resolution image is firstly up-scaled to the desired size using bicubic interpolation, the interpolated image is denoted as \( Y \). The goal is to recover an image \( F(Y) \) to be as similar as possible to the ground-truth image \( X \). While in our fractional interpolation task, the input of the network is the image consisting of integer-position pixels, denoted by \( Y_{int} \), and the output is the interpolated image of fractional positions, \( F_h \), which has the same size with the input image:

\[
F_h = W_3 \ast F_2 + B_3 \tag{2}
\]

Please note that fractional sample interpolation is related to but different from super-resolution: the former tries to...
CNNIF, and the labels are denoted as \( \{Y^i\}_{i=1}^N \). Here, we use the Euclidean distance as the loss function:

\[
L = \frac{1}{N} \sum_{i=1}^{N} \| F^i - Y^i \|^2
\]

where \( N \) is the total number of training data items.

III. EXPERIMENTS

A. Training Stage

We use the deep learning framework Caffe [14] to train the CNNIFs on an NVIDIA Tesla K40C graphical processing unit (GPU). The training set we use is the same as that in [11], which is a collection of 400 natural images. All the images are processed using the method depicted in Fig. 3 and after that we have 400 sub-images as input and 1200 sub-images for the labels (400 sub-images for each CNNIF). In this implementation, each sub-image as input is compressed by HEVC intra coding at four different QPs: 22, 27, 32, and 37. For each QP and each half-pel position, a separate network will be trained. Therefore, we finally train 12 CNNIFs. During the process of compression, a CNNIF will be selected according to the slice QP and the corresponding half-pel position. The nearest QP among 22, 27, 32, and 37 to the current slice QP will be considered.

B. Comparison with HEVC Baseline

The proposed method is implemented based on HEVC reference software HM 16.7. Currently, only the process of half-pel interpolation of the luma component is replaced by CNNIFs. The low delay P (LDP) configuration is tested in the experiment under the HEVC common test conditions. BD-rate is used to measure the rate-distortion (RD) performance. The experimental results are summarized in Table I. As can be observed, the proposed method achieves on average 0.9% BD-rate reduction. For the test sequence BQTerrace, the BD-rate reduction can be as high as 3.2%, 1.6%, and 1.6% for Y, U, and V components, respectively. Since the fractional interpolation of chroma components is still DCTIF, the performance of chroma components is not prominent. In the future work, we will train CNN models for the chroma components.
| Class   | Sequence           | BD-rate          |
|---------|--------------------|------------------|
|         |                    | Y (%) | U (%) | V (%) |
| Class B | Kimono             | -1.1  | 0.1   | 0.2   |
|         | ParkScene          | -0.4  | -0.3  | -0.3  |
|         | Basketballs        | -0.8  | 0.0   | 0.2   |
|         | BGTerrace          | -1.3  | -0.2  | -0.1  |
| Class C | BasketballDrill    | -3.2  | -1.6  | -1.6  |
|         | BGMail             | -1.2  | 0.6   | 0.2   |
|         | PartyScene         | -0.9  | 0.2   | 0.7   |
|         | RaceHorses         | -1.5  | -0.5  | -0.1  |
| Class D | BasketballPass     | -1.3  | -0.4  | 0.3   |
|         | BGSquare           | 1.2   | -2.9  | 4.1   |
|         | BlowingBubbles     | -0.3  | 0.4   | 0.8   |
|         | RaceHorses         | -0.8  | -0.9  | 0.0   |
|         | FourPeeples        | -1.3  | -0.4  | 0.1   |
|         | Johnny             | -1.2  | -0.4  | 0.7   |
|         | KristenAndSara     | -1.0  | 0.3   | 0.2   |
| Class F | BasketballDrilltex | -1.4  | -0.2  | 0.1   |
|         | ChinaSpeed         | -0.6  | -0.5  | -0.3  |
|         | SliderEditing      | 0.0   | 0.3   | 0.4   |
| Class D | Class B            | -1.4  | -0.4  | -0.3  |
|         | Class C            | -0.9  | -0.1  | 0.3   |
|         | Class D            | -0.3  | 0.5   | 1.0   |
|         | Class E            | -1.2  | -0.2  | -0.1  |
|         | Class F            | 0.7   | -0.1  | 0.0   |
| Overall  | All                | -0.9  | -0.1  | 0.2   |

| Class   | Sequence           | BD-rate          |
|---------|--------------------|------------------|
|         |                    | Y (%) | U (%) | V (%) |
| Class C | BasketballDrill    | 0.8   | 1.2   | 2.1   |
|         | BGMail             | 2.8   | 2.7   | 3.0   |
|         | PartyScene         | 3.6   | 3.8   | 3.7   |
|         | RaceHorses         | 2.4   | 2.1   | 2.0   |
| Class D | BasketballPass     | 1.7   | 1.3   | 2.0   |
|         | BGSquare           | 8.2   | 7.9   | 6.8   |
|         | BlowingBubbles     | 3.2   | 3.5   | 4.2   |
|         | RaceHorses         | 3.6   | 2.9   | 2.3   |

**C. Comparison with Super-Resolution**

We also compare our proposed method with super-resolution method. As an anchor, the reconstructed frame is up-scaled to a larger one by the pre-trained SRCNN model [15] with a factor of 2. From the enlarged frame, the phase one, phase two and phase three pixels are used as the interpolated half-pel samples. We integrate this anchor method into HM 16.7. Table II shows the results of the HEVC test sequences Class C and Class D of the anchor method compared to HEVC baseline. It can be observed that all the sequences suffer from significant loss. For the test sequence BGSquare, the loss can be as high as 8.2% for luma component. Therefore, despite the similarity between fractional interpolation and image super-resolution, they are indeed not the same task, and it is not appealing to directly apply super-resolution method to fractional interpolation, as validated by the experimental results.

**IV. Conclusion**

This paper presents a convolutional neural network based fractional interpolation for inter prediction in HEVC. We use the existing SRCNN structure, but retrain interpolation CNN models for the three half-pel positions of luma component. A blurring followed by extracting method is proposed to generate training data, especially the missing labels. Experimental results show that the proposed CNNF can achieve on average 0.9% bits saving. Our further work will focus on two aspects. First, to design a more efficient network architecture that is more suitable for the interpolation task. Second, we will investigate how to generate better labels for training, especially for the quarter-pel interpolation.

**REFERENCES**

[1] H. Lakshman, B. Bross, H. Schwarz, and T. Wiegand, “Fractional-sample motion compensation using generalized interpolation,” in Picture Coding Symposium (PCS), 2010. IEEE, 2010, pp. 530–533.

[2] B. Girod, “Motion-compensating prediction with fractional-pel accuracy,” IEEE Transactions on Communications, vol. 41, no. 4, pp. 604–612, 1993.

[3] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the h. 264/avc video coding standard,” IEEE Transactions on circuits and systems for video technology, vol. 13, no. 7, pp. 560–576, 2003.

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

[5] H. Lv, R. Wang, X. Xie, H. Jia, and W. Gao, “A comparison of fractional-pel interpolation filters in hevc and h. 264/avc,” in Visual Communications and Image Processing (VCIP), 2012 IEEE, 2012, pp. 1–6.

[6] T. Weli, “Adaptive interpolation filter for motion compensated hybrid video coding,” in Picture Coding Symposium (PCS 2001), Seoul, Korea, 2001.

[7] S. Wittmann and T. Wedi, “Separable adaptive interpolation filter for video coding,” in 2008 15th IEEE International Conference on Image Processing. IEEE, 2008, pp. 2500–2503.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Advances in neural information processing systems, 2012, pp. 1097–1105.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[10] C. Dong, C. C. Loy, K. He, and X. Tang, “Learning a deep convolutional network for image super-resolution,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[11] C. Dong, Y. Deng, C. Change Loy, and X. Tang, “Compression artifacts reduction by a deep convolutional network,” in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 576–584.

[12] Y. Dai, D. Liu, and F. Wu, “A convolutional neural network approach for post-processing in hevc intra coding,” arXiv preprint arXiv:1608.00690, 2016.

[13] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 807–814.

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in Proceedings of the 22nd ACM international conference on Multimedia. ACM, 2014, pp. 675–678.

[15] C. Dong, C. C. Loy, K. He, and X. Tang. (2015) Image super-resolution using deep convolutional networks. [Online]. Available: http://mmlab.ie.cuhk.edu.hk/projects/SRCNN.html