iCNN: A Convolutional Neural Network for Fractional Interpolation in Video Coding

Chi Do-Kim Pham and Jinjia Zhou

September 4, 2019

iCNN: A Convolutional Neural Network for Fractional Interpolation in Video Coding

Chi Do-Kim Pham[1], and Jinjia Zhou[1,2]

[1] Graduate School of Science and Engineering, Hosei University, Japan
chi.kim.pham.do.94@stu.hosei.ac.jp
[2] JST, PRESTO, Tokyo, Japan
jinjia.zhou.35@hosei.ac.jp.

Abstract. Motion compensated prediction has significantly contributed to the temporal redundancy in video coding by predicting the current frame from the list of previously reconstructed frames. Later video coding standard HEVC uses DCTIF to interpolate fractional pixels for more accurate motion compensated prediction. Although the fixed interpolation filters have been improved, they are not able to adapt to the diversity of video content. Inspired by super-resolution, we design the interpolation Convolutional Neural Network for fractional interpolation in video coding. Our work also solves two main problems in applying Convolutional Neural Network to fractional interpolation in video coding: there is no training set for fractional interpolation and integer pixels change after processing. As a result, this work achieves a 2.6% BD-rate reduction compared to the baseline HEVC.

Keywords: Deep Learning, fractional interpolation, video coding, Motion Compensated Prediction.

1 Introduction

In video coding, Motion Compensated Prediction (MCP) is one of the critical factors in removing temporal redundancy between video frames. MCP searches in a list of previously reconstructed frames to find the best matching block of the current block to be encoded. The difference between the best-matching block and the current block to be encoded, as known as residual, is coded beside the motion vector that indicates the movement of the current block to the position of the best matching block in the reference frame. If the best-matching block does not fall into integer pixels, reference blocks in the previously reconstructed frame are interpolated to get fractional samples, and fractional motion vectors are also used for representing 11 movements. The widely used video coding standard H.265/HEVC [1] uses 7-tap and 8-tap Discrete Cosine Transform interpolation filter (DCTIF) for the quarter and half-pixel positions, respectively. Although DCTIF improved the accuracy of MCP, these fixed filters are not flexible enough for the variety of content in natural videos.

The past decades have witnessed the tremendous success of artificial intelligent, especially machine learning, in various aspects of the social life [2], [3]. In machine learning, Convolutional Neural Networks (CNN) is one of the most famous techniques that outperforms the traditional methods in image processing task. Recently, CNN has been widely used in super-resolution where a high-resolution image is obtained given a low-resolution image. Chao Dong proposed SRCNN that learns a mapping between an input of low-resolution image and an output of high-resolution image [4], Jiwon Kim designs VDSR that reconstructs a high-resolution image by estimating image details given a low-resolution image [5]. Although both super-resolution and fractional interpolation in video coding increases the resolution of input image, CNN-based super-resolution cannot directly be used for fractional interpolation in video coding because of two problems: (1) super-resolution tends to change the pixel values while interpolation needs to keep the integer pixels for integer motion search and (2) there is no trainset for fractional interpolation in video coding because fractional pixels do not really exist.

To keep the integer pixels, the authors of [6] train CNNIF_H, CNNIF_V, and CNNIF_D for three half-pixel positions and keep all integer pixels for integer motion search. Similar to [6], Ning Yan proposed 15 Fractional-pixel Reference generation CNN (FRCNN) [7] for half- and quarter-pixel interpolation in uni-directional and bi-directional MCP. The work [8] proposes GVTCNN to learn 15 fractional pixels from an input of integer pixels. Different from above the works, Han Zhang added a constraint mask after their CNN for half-pixel interpolation to make sure the integer pixels are kept [9].

To solve the problem (2), the work [6] uses a low-pass filter for blurring images to get the correlation between integer and fractional pixels and extracts them. They then encode the integer pixel to get the reconstructed frame and treat them as the input of CNN, and the fractional pixels are ground truth. In [7], the authors extract the reference blocks from reconstructed video and the corresponding blocks from the original video to be CNN input and label, respectively. Totally, 120 CNN models are trained for four values of QPs in uni- and bi-directional prediction. [9] trains a mapping between DCTIF interpolated frame of the low-resolution frame and the corresponding original video frame. They also trained four CNN models that are compatible with four input QPs in encoding.

In this paper, we propose interpolation CNN (iCNN) that enhances fraction-position images interpolated by DCTIF for MCP. Moreover, we create a train set for our iCNN and an iCNN/DCTIF selection for improving coding efficiency. The rest of this paper presents our proposal in data generation and our network for fractional-pixel interpolation in section 2 follows by experiments in section 3 and ends with the conclusion in section 4.

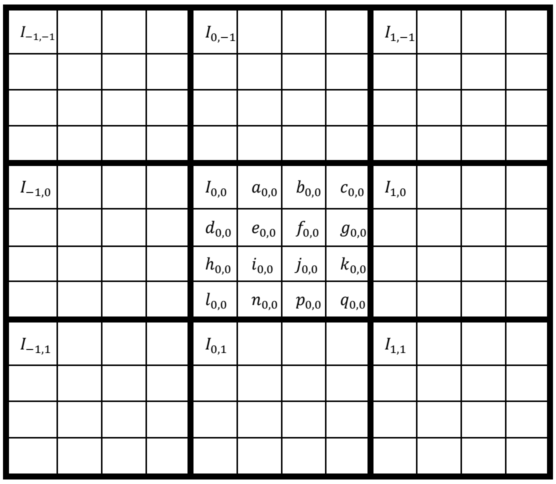

Fig. 1. Integer and fractional samples of Luma Component in HEVC. $I_{i,j}$ presents for integer pixels, $b_{i,j}$, $h_{i,j}$, and $j_{i,j}$ presents for half samples and others are quarter samples.

2 Proposals

As mention in the introduction, there are two problems in applying CNN to fractional interpolation in video coding: CNN-based super-resolution tends to change pixel values while interpolation needs to keep integer-pixel values for integer motion search, and there is no trainset for interpolation in video coding. This section presents our proposals to improve DCTIF by using CNN and a dataset generation method for our CNN.

2.1 Ground Truth Selection

Fig. 1 presents integer and fractional samples in HEVC. In HEVC fractional interpolation, half pixels $b_{i,j}$, $h_{i,j}$, and $j_{i,j}$ are obtained by applying 7-tap DCTIF, and 8-tap DCTIF is applied on to get quarters positions. Although more taps and bigger paddings are added into DCTIF for better prediction, DCTIF may not be able to adapt the diversity of video contents, and the covered area is still limited. Meanwhile, CNN has remarkable contributions in super-resolution where global and local information are needed for reconstructing a high-resolution image.

In training any CNN model, training set is one of the essential elements to obtain the expected output. In CNN-based super-resolution, it is clear that the expected output is the original high-resolution image which is set as CNN ground-truth labels. Although super-resolution seems similar to fractional interpolation, super-resolution trainset cannot be directly applied to fractional-position interpolation because one's input is low-resolution images and the other's input is the reconstructed images, and no ground truth for half and quarter samples because they do not exist. Hence, there is a question that needs to be answered: What is best the ground truth for training CNN-based fractional interpolation in MCP?

In order to generate the train set, a typical way is to assume integer and fractional position in each video frame, extract integer and fractional videos, do encoding for integer frames, then learn the mapping between the reconstructed integer frame and the fractional blurred frame [6], [8] or the original frame [11].

We conduct an experiment to find which should be the ground truth for training. In this experiment, we assume and extract integer videos from the original video. Supposing that if DCTIF can exactly generate the original video frames from the reconstructed integer-video frames and this original frame can improve the coding efficiency on the integer-video, the reconstructed integer-video frame and the corresponding original video frame can be a pair of input and ground truth for training. Our experiment is described as: encode integer video and replace DCTIF's interpolated frame by the original video frame. This experiment achieves a 7.1% Y BD-rate reduction on the first three frames of extracted integer sequences of class B and C. We then use reconstructed integer-video frame and the original video frame as our training set.

2.2 Dataset Preparation Method

After the ground-truth selection experiment, we design a training set for our iCNN. This section presents the trainset generation in detail. Given a YUV video, our training-data generation (Fig. 2) can be described as follow:

(1) We assume the integer and fractional pixels in a video by dividing each frame into 4-by-4 non-overlapping blocks, and each top-left pixel of a block is treated as integer pixel, others are fractional pixels. Half and quarter positions are pixels at a similar position 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 with QPs of 22, 27, 32 and 37 under low delay P configuration to get reconstructed downsampled video.

(3) Extract Y components from reconstructed frames and interpolate them to 15 fractional samples by DCTIF. These 15 fractional samples are used as training input for iCNN.

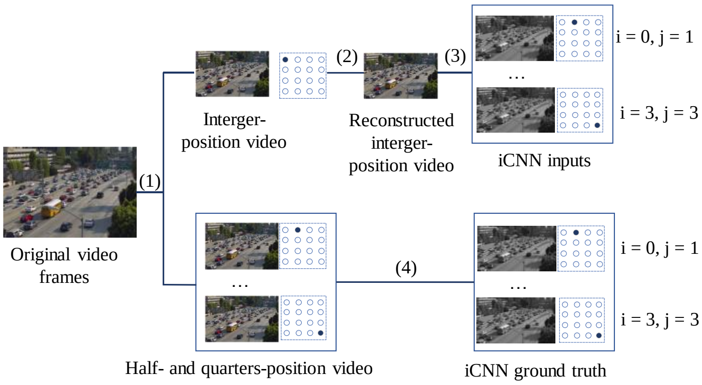

Fig. 2. Visualization of input and ground-truth generation method from the original video frames. In the end, 15 samples for each frame are trained.

(4) Extract Y component from each fractional-position video frame. The original Y component of fractional-position videos is iCNN ground truth 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.

In training any deep networks, small trainset could cause overfitting. We then do some data augmentation techniques including flipping and rotating training images to avoid spatial bias training (such as objects and movements only distribute in some positions) and overfitting.

2.3 Interpolation Convolution Neural Network

Our proposal in improving DCTIF by using CNN (Fig. 3) is twofold: an iCNN for enhancing fractional samples by DCTIF and an iCNN/DCTIF selection for taking full advantage of iCNN and DCTIF. In fractional motion search and motion compensation, the integer sample from the reconstructed frame is interpolated by DCTIF to get 15 fractional samples. Each fractional sample is fed into iCNN to get iCNN fractional sample after convolution. For iCNN/DCTIF selection, iCNN's and DCTIF's fractional samples are checked for R-D cost at CU level to decide which method should be used for interpolating fractional samples of that CU.

Interpolation Convolutional Neural Network (iCNN). Our iCNN takes an input from fractional frames which are interpolated by DCTIF (as 2.2) and a ground-truth of fractional sample extract from the original video frame. Only one network is trained for 15 fractional samples. Our network architecture (inside dash border of Fig. 3), inspired by VDSR, contains 20 convolution layers, each layer has 64 filters size 3x3 and do convolution with a stride of 1. During convolution, we set a padding

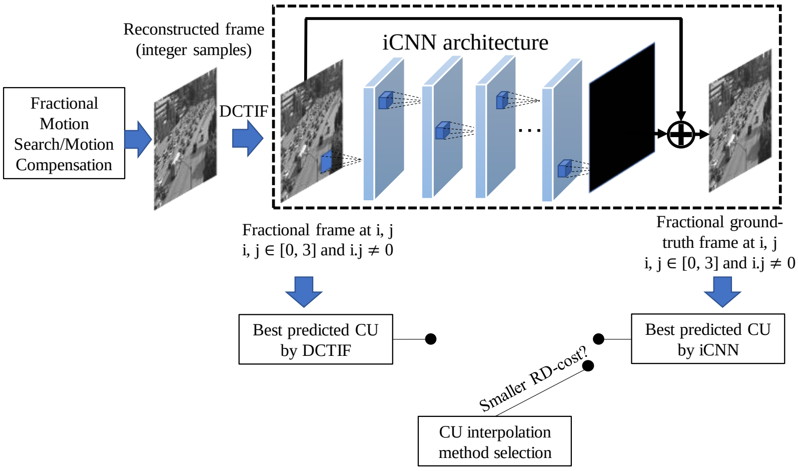
Fractional Motion Search/Motion Compensation
Reconstructed frame (integer samples)
DCTIF
iCNN architecture
Fractional frame at i, j
i, j ∈ [0, 3] and i.j ≠ 0
Fractional ground-truth frame at i, j
i, j ∈ [0, 3] and i.j ≠ 0
Best predicted CU by DCTIF
Best predicted CU by iCNN
Smaller RD-cost?
CU interpolation method selection

Fig. 3. Our proposal in improving DCTIF by using CNN. Firstly, reconstructed frame which includes integer samples are interpolated by DCTIF to get 15 fractional samples. Each fractional sample is then fed into iCNN which outputs a corresponding fractional sample. At CU level, interpolation method is chosen if its R-D cost is smaller than the other method's.

of 1 to keep the input size not to change over the layers. All the convolution layers are followed by a ReLU layer except for the final one. For training, learning rate is first started at 0.1 and decreased by a factor of 10 after ten epochs. We set a batch size of 128 and finish training after 50 epochs.

In training, fractional-position image x interpolated from the reconstructed integer frames and fractional-position image y extracted from the original video are used as CNN inputs and ground-truth label, respectively. Let w_l, b_l denotes the learned weights and biases at layer l^{th}, output z at layer l^{th} is:

$$z_l = ReLU(w_l * z_{l-1} + b_l) \tag{1}$$

where $ReLU(X) = max(X, 0)$, and '*' means convolutional operator. The first layer takes the input image x for z_{l-1}. The final layer has no ReLU layer and produces a residual $f(x, \theta)$ which will be add onto input. The network training aims to minimize the loss function:

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left\|y - \left(x \oplus f(x,\theta)\right)\right\|^2 \tag{2}$$

where n is number of training samples, and θ is the set of learned weights and biases for all layers. In testing, reconstructed frames are first interpolated to 15 fractional samples by DCTIF before feeding iCNN one by one. Motion search then compares the reconstructed integer, and fractional iCNN outputs to the original frame to be coded to find the best fractional matching block. By producing only fractional samples, we can keep integer pixels for later processes.

iCNN/DCTIF selection. In some cases, DCTIF can generate a better-predicted block than iCNN. To take both advantages of iCNN and DCTIF, we implement a Rate-Distortion cost-based (R-D cost) iCNN and DCTIF interpolation-method selection at CU level. At CU level, if a CU is coded with fractional motion, an interpolation method is decided base on R-D cost. All PUs in the same CU will be interpolated by CU's interpolation method. For each CU that has fractional MV, a syntax element for the interpolation method is defined in bypass mode and extra bit indicates the interpolation method will be stored without compressing.

3 Experiments

In our experiments, the reference software HEVC Test Model (HM) [10] 16.18 is used for generating iCNN trainset and demonstrating CNN-based interpolation. For trainset generation, we extract 60 frames from standard HEVC test sequence `Traffic`, `PeopleOnTheStreet`, and `Pedestrian` [11] for QP 32 and 37, and 40 frames from `Traffic` and `PeopleOnTheStreet` for QP 22 and 27. After producing input and ground truth by section 2.2, input and ground-truth images are divided into 41x41 patches. Our iCNN is trained by PyTorch 1.0.0 platform [13] on NVIDIA Tesla V100 GPU.

All the experiments are conducted under low delay P configuration. Quantization parameter (QP) is set to 22, 27, 32, and 37, the full search is activated, and other parameters are set to default. In evaluating our results, we compare our method with HM by Bjøntegaard-Delta bit-rate (BD-rate) method [12]. BD-rate indicates how many bits a coding method reduces compared to an anchor method at the same image quality. Since each QP significantly affects the image quality, four models for four QP values are trained. U and V components are interpolated by the default DCTIF of HEVC.

Table 1 shows our BD-rate on standard test sequences compared to HEVC. The first column is our results without iCNN/DCTIF selection. Since the model is trained only on the Y component, we obtain a 0.2% BD-rate reduction on the Y component and up to 4.5% for sequence Kimono. For more information, U and V BD-rate are increased to 2.6% and 2.8% over the sequences compared to HEVC. The second column shows our results for iCNN/DCTIF selection without an extra bit for each CU interpolation flag. We acquire Y, U, and V BD-rate reduction of 4.4%, 0.6%, 0.5%, respectively. The third column shows our CU level selection results when an extra bit for every CUs that choose fractional interpolation. We get a Y BD-rate reduction of 2.6%, but U and V do not good enough to deal with one extra bit added

Table 1. Y BD-rate (%) of proposal with and without iCNN/DCTIF selection. For iCNN/DCTIF selection, result with and without extra bit for indicating interpolation method is shown. (Anchor: HEVC) Note that the lower negative BD-rate, the better result.

Class	Sequence	Without iCNN/DCTIF selection	With iCNN/DCTIF selection	
			Without extra bits	With extra bits
B	Kimono	-4.5	-7.1	-4.5
	ParkScene	4.4	-3.6	0.1
	Cactus	-0.8	-6.7	-4.7
	BasketbalDrive	-2.4	-6.0	-3.6
	BQterrace	-0.1	-8.3	-5.6
C	BasketballDrill	-1.5	-5.5	-3.6
	BQMall	-0.5	-3.3	-1.9
	PartyScene	2.6	-2.9	-1.4
	RacehorsesC	-0.3	-4.5	-2.5
D	BasketballPass	-0.9	-4.7	-2.8
	BQSquare	4.8	-5.6	-3.5
	BlowingBubbles	2.3	-4.6	-2.5
	RaceHorses	-0.4	-5.6	-3.1
E	FourPeople	-3.3	-5.2	-4.1
	Johnny	-0.9	-7.0	-4.9
	KristenAndSara	-1.4	-5.4	-3.5
F	BasketballDrillText	-0.9	-5.1	-3.2
	ChinaSpeed	-0.3	0.2	1.2
	SlideEditing	0.1	1.6	1.7
	SlideShow	-0.5	0.7	1.1
Average B		-0.7	-6.3	-3.7
Average C		0.1	-4.0	-2.3
Average D		1.4	-5.2	-3.0
Average E		-1.9	-5.9	-4.1
Average F		-0.4	-0.6	0.2
Average all sequences		-0.2	-4.4	-2.6

for each CU which leads U and V BD-rate increase to 1.3% and 1.4%, respectively. The future works include training models for U and V and compressing the added bit for each CU interpolation method.

We also do some visualizations for our Rate-Distortion Optimization-based iCNN/DCTIF selection for each CU. Fig. 4 is the visualization of CU-level interpolation-method selection on POC 7 of `BQMall` including iCNN interpolation (red CUs), DCTIF interpolation (blue CUs), and other CUs without border are CUs coded in intra mode or move with integer MV.

In detail, we synthesize the ratio of choosing iCNN and DCTIF over the test sequences. It is unfair to say which method is chosen more than the other one based on the number of CUs since CU size can be 64-by-64, 32-by-32, 16-by-16 or 8-by-

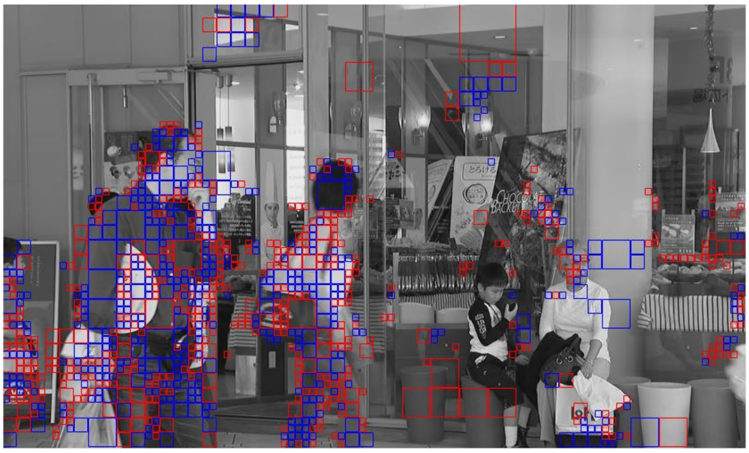

Fig. 4. Visualization iCNN/DCTIF interpolation selection on POC7 of sequence BQMall. Blue, red blocks represent for CU that choose DCTIF, iCNN, respectively. The other parts include intra-coding CUs and CUs have integer MVs.

8. We then calculate the hitting ratio of iCNN and DCTIF on the area. The following equation calculates the area in pixel of a method:

$$area_{method} = \sum_{j=0}^{3} n_j \left(\frac{64}{2^j}\right)^2 \quad (3)$$

where n_j is the number of CU that is coded at depth j. The hitting ratio statistics of choosing iCNN or DCTIF is shown in Fig. *5*. For our experiment, the higher QP is used, the less iCNN is chosen. In conclusion, our iCNN gets a hitting ratio of 43.27% over the test sequences.

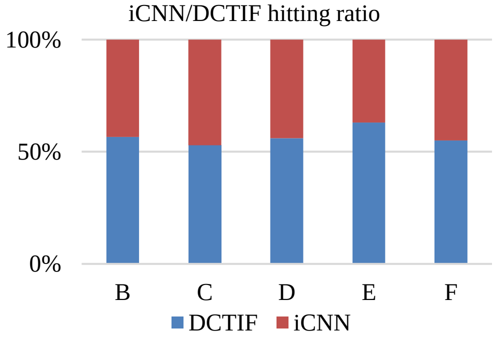

Fig. 5. Hitting ratio (%) of iCNN and DCTIF over class B, C, D, E, and F.

4 Conclusion

Inspired by CNN-based super-resolution, we design interpolation Convolutional Neural Network (iCNN) to adapt the diversity of video contents where the fixed interpolation filters face limitations. In this work, we solve two main problems in improving HEVC's interpolation filters with CNN-based super-resolution: training set are different, and CNN will change the integer pixels which is required for integer motion search in HEVC. For the training set problem, we create a training set of encoded integer video and fractional videos extracted from the original video frame. For the problem of changing integer pixels, we produce fractional pixels one-by-one and keep the integer pixels. Besides, an R-D cost-based iCNN/DCTIF selection is implemented to further improve the coding efficiency. Current work achieves a 2.6% Y BD-rate reduction over the standard test sequences compared to HEVC.

Acknowledgements. This work is supported by JST, PRESTO Grant Number JPMJPR1757 Japan.

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] H. Lu, Y. Li, M. Chen, H. Kim, and S. Serikawa, "Brain Intelligence: Go beyond Artificial Intelligence," *Mobile Netw Appl*, vol. 23, no. 2, pp. 368–375, Apr. 2018.

[3] H. Lu *et al.*, "CONet: A Cognitive Ocean Network," *arXiv:1901.06253 [cs]*, Jan. 2019.

[4] C. Dong, C. C. Loy, K. He, and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, Feb. 2016.

[5] 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.

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

[8] 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.

[9] 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.

[10] "High Efficiency Video Coding (HEVC) | JCT-VC." [Online]. Available: https://hevc.hhi.fraunhofer.de/. [Accessed: 27-Nov-2018].

[11] F. Bossen, "Common test conditions and software reference configurations," *Joint Collaborative Team on Video Coding (JCT-VC) of ITUTSG16 WP3 and ISO/IEC JTC1/SC29/WG11, Doc. JCTVC-J1100*, Stock-holm, Sweden, Jul-2012.

[12] G. BJONTEGAARD, "Calculation of average PSNR differences between RD-curves," *VCEG-M33*, 2001.