applied sciences

MDPI

Article

Neural Network-Based Reference Block Quality Enhancement for Motion Compensation Prediction

Yanhan Chu, Hui Yuan *, Shiqi Jiang and Congrui Fu

School of Control Science and Engineering, Shandong University, No. 17923 Jingshi Road, Jinan 250061, China
* Correspondence: huiyuan@sdu.edu.cn

Abstract: Inter prediction is a crucial part of hybrid video coding frameworks, and it is used to eliminate redundancy in adjacent frames and improve coding performance. During inter prediction, motion estimation is used to find the reference block that is most similar to the current block, and the following motion compensation is used to shift the reference block fractionally to obtain the prediction block. The closer the reference block is to the original block, the higher the coding efficiency is. To improve the quality of reference blocks, a quality enhancement network (RBENN) that is dedicated to reference blocks is proposed. The main body of the network consists of 10 residual modules, with two convolution layers for preprocessing and feature extraction. Each residual module consists of two convolutional layers, one ReLU activation, and a shortcut. The network uses the luma reference block as input before motion compensation, and the enhanced reference block is then filtered by the default fractional interpolation. Moreover, the proposed method can be used for both conventional motion compensation and affine motion compensation. Experimental results showed that RBENN could achieve a −1.35% BD rate on average under the low-delay P (LDP) configuration compared with the latest H.266/VVC.

Keywords: H.266/VVC; inter prediction; motion compensation; deep learning; image enhancement

Citation: Chu, Y.; Hui, Y.; Jiang, S.; Fu C. Neural Network-Based Reference Block Quality Enhancement for Motion Compensation Prediction. *Appl. Sci.* **2023**, *13*, 2795. https://doi.org/10.3390/app13052795

Academic Editor: Antonio Fernández-Caballero

Received: 19 December 2022
Revised: 19 February 2023
Accepted: 20 February 2023
Published: 22 February 2023

1. Introduction

As video dominates the current media transmission network, the demand for higher quality video is growing day by day, which brings great challenges to video compression algorithms. Video coding tools have been iterating for decades and, furthermore, eliminating redundant information in the video is becoming more and more difficult.

As one of the most important parts in a hybrid video coding framework, inter prediction predicts the correlation between adjacent frames so that it is not necessary to transmit all the information of each coding frame. As shown in Figure 1, when encoding the current coding unit (CU), the reference CU, which is most similar to the current CU, can be found in the reconstructed reference frame. Only the motion vector (MV) and residual information between them need to be transmitted. Due to the inherent spatial discretization of digital video, block translation may not be aligned with integer pixels. Through motion estimation (ME), as shown in Figure 2, the current CU first finds the reference CU and obtains the integer pixel MV (I_{mv}). To improve the prediction accuracy, the reference CU is further interpolated to achieve a higher resolution; thus, the image block at the best fractional pixel position could be selected by a fractional MV (F_{mv}). It is obvious that the interpolation method plays an important role for fractional motion estimation. The more accurate the predicted block, the smaller the residue, and the higher the coding performance. In the 1980s, researchers adopted simple bilinear and bicubic filters to generate fractional pixels. With the development of signal processing theory, researchers proposed more efficient interpolation filters, such as the Wiener interpolation filter, discrete cosine transform interpolation filter (DCTIF) [1], and the generalized interpolation filter [2]. These filters can perform an overall fractional shift of the image to make the reference block closer to the

current block. In addition, for non-translational motion in video, H.266/VVC also uses affine motion compensation, in which a large block is divided into sub-blocks first before the corresponding MV of each sub-block can be calculated by an affine formula to cope with the motion of zooming, rotation, etc. A typical video coding structure of existing video coding standards is shown in Figure 3.

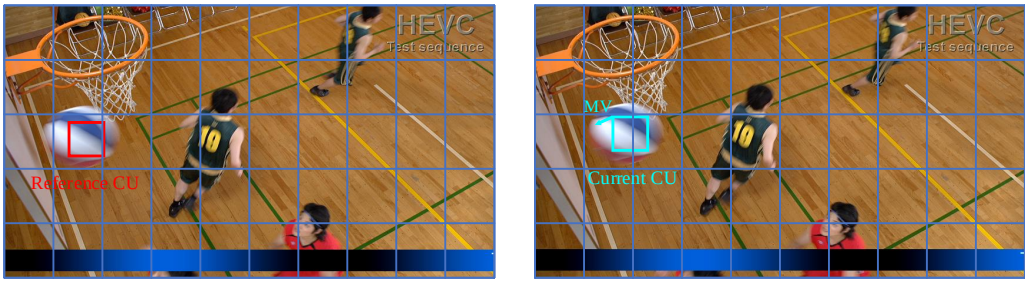


Figure 1. Inter prediction.

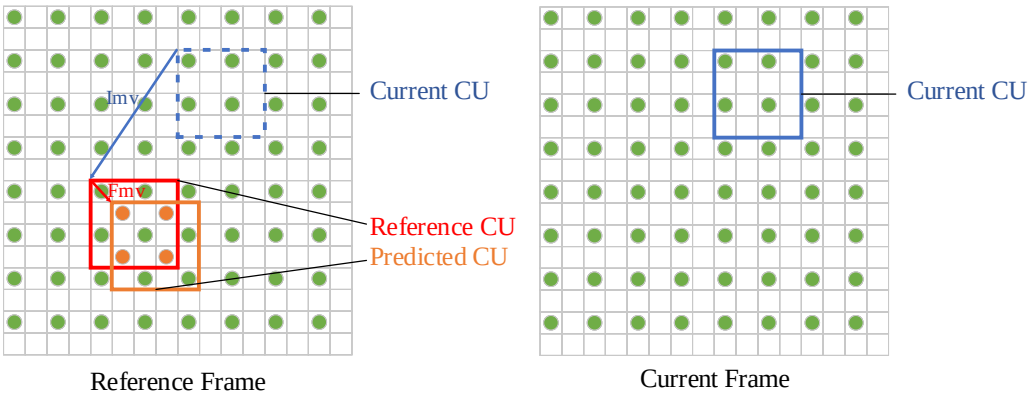


Figure 2. Relationship between the current CU, reference CU, and predicted CU; the block with green dots represents the integer pixel position.

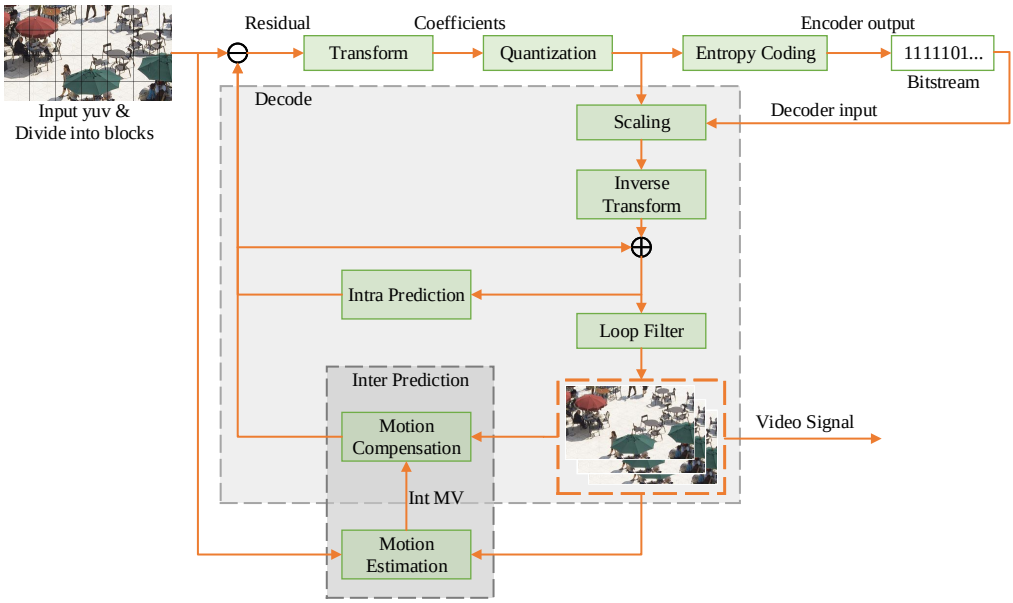


Figure 3. VVC coding framework.

Finally, the optimal MV, the transformed and quantized residual between the current block and the reference block, and some other meta information is entropy coded to

generate a bit stream. The decoder finds the corresponding reference block by the MV and adds residuals to reconstruct the current block.

In recent years, deep learning (DL) has made great achievements in many fields with its strong learning ability and robustness. Especially in the field of computer vision, deep convolutional neural networks have played an interesting role in solving image processing problems. Recently, Moving Picture Expert Group has been investigating deep learning-based coding tools to further improve coding efficiency. DL-based methods can be integrated into a specific video coding module, such as intra prediction, inter prediction, loop filter, and post-processing. Specifically, in inter prediction, DL can be used for prediction, quality enhancement, fractional interpolation, and reference frame generation.

In view of the successful application of DL in image quality enhancement, we proposed a quality enhancement method for CU before MC, which was added to the H.266/VVC framework. Specifically, we designed a reference-block-enhanced neural network (RBENN) to enhance the quality of the reference block. Fractional interpolation and other subsequent operations were then used on the quality-enhanced reference block. Compared with the standard H.266/VVC framework, the proposed method can reduce the Bjøntegaard delta rate (BD rate) [3] in the low-delay P (LDP) configuration by 1.35%. The contributions of this paper are as follows.

- A convolutional neural network with residual structure was used to enhance the quality of reference blocks before MC.
- The proposed method not only enhanced the reference CU before conventional motion compensation, but it also enhanced the reference CU before affine motion compensation.
- Experimental results showed that the proposed method achieved a 1.35% BD rate saving compared with the H.266/VVC standard under LDP test conditions.

The remainder of this paper is organized as follows. Section 2 provides a brief review of the related work, Section 3 describes the proposed methodology and network architecture, Section 4 shows experimental results, and the work is concluded in Section 5.

2. Related Work

In inter prediction, the closer the reference CU is to the current CU, the less the residuals are that are required for coding, and, thus, the lower the bit rate. However, due to the discretization of digital video, the current CU and the reference CU cannot be completely aligned. For this reason, existing video coding standards usually adopt a set of fixed sub-pixel filtering coefficients to interpolate the reference CUs [4]. In H.266/High Efficiency Video Coding (HEVC) and H.266/VVC, discrete cosine transform-based half or quarter interpolation filters are adopted. Specifically, to improve the coding performance, H.266/VVC applies one-sixth sub-pixel precision interpolation filters and affine motion compensation.

Although the fractional interpolation filter achieves good performance in ME and MC, it still cannot adapt to complex video content well. Therefore, many researchers have used DL to refine the sub-pixel blocks. In these methods, [5–13] are for HEVC/H.265 [14], while [15–19] are for H.266/VVC [20].

Yan et al. [5–7] regarded the fractional interpolation filter as an image regression problem and trained neural networks for each fractional interpolation filter. In [5], the author replaced three half-pixel fractional filters with three super-resolution convolutional neural network (SRCNN) [21] models, which resulted in a 0.9% reduction in BD rate under the LDP configuration of H.265/HEVC. In [6], the author used the reference CU as the input, the original CU as the ground truth, and trained 15 networks, whose structures were similar to a variable-filter-size residue learning convolutional neural network (VR-CNN) [22], to replace the fractional filters of half- and quarter-pixel fractional shifts. In this method, there are a total of 120 natural networks for uniprediction, biprediction, and four different quantization parameters (QPs). This method saves 3.9%, 2.7%, and 1.3% in BD rates on average for LDP, low-delay B (LDB), and random-access (RA) configurations of

H.265/HEVC, respectively. Yan et al. [7] also proposed a reversible training method that is based on the assumption that integer pixels and fractional pixels can be derived from each other. This method can save 4.7% and 3.6% in BD rates under the configurations of LDB and RA for H.265/HEVC, respectively. In [8], a group variational transformation neural network (GVTNN) was proposed to replace the fractional interpolation in H.265/HEVC, and it can achieve a 1.9% BD rate reduction under the LDP configuration. Liu et al. [9] optimized a neural network based on [8] and analyzed the influence of a fuzzy kernel on the task. In [10], Huo et al. deemed that fractional interpolation can obtain more information from the reconstructed area around the current CU. The authors then put the current CU, together with the left and top reconstructed areas, into a larger square area as the network input and the original CU as the ground truth label. This method reduces the BD rate by 1.8% in the LDP configuration of H.265/HEVC. Similarly, Wang et al. [11] reshaped the left and the upper reconstructed regions of the current CU into images with the same size, and then input them into the neural network together. This method can achieve 4.6%, 3.0%, and 2.7% BD rate reductions on average under LDP, LDB, and RA configurations, respectively, compared to H.265/HEVC. Chi et al. [12] trained separate networks for luma and chroma channels. The method in [13] is similar to a loop filter; it uses the optical flow network (PWC-Net) [23] to generate an optical flow graph from the video frames before the reference frames, then uses the optical flow graph and the reference frame in a neural network [24] for quality enhancement. T he reference CUs are then extracted from the quality-enhanced reference frame for MC.

In the study of H.266/VVC, Murn et al. [15,16] designed a three-layer neural network that only contains convolutional layers. The reference CUs, and the original CUs corresponding to different fractional filters, are made into different datasets for network training. All convolution kernels in the network are extracted and compressed into a 13×13 matrix to replace the DCTIF for MC. This approach can effectively reduce the complexity of the network; it can finally achieve BD rate savings of 0.77%, 1.27%, and 2.25% on average under RA, LDB, and LDP configurations, respectively, compared to H.266/VVC. Galpin [17] used neural networks to enhance inter bipredictionand replaced bidirectional optical flow (BDOF) with networks. Compared with H.266/VVC, it saves 1.4% and 0.58% of the BD rate under RA and LDB configurations, respectively. Jin [18] combined DL with affine motion compensation. This method takes the reference region, the predicted region, and the motion vectors of sub-CUs of affine motion compensation as the inputs of the network, and then outputs the final predicted region. The authors trained 76 networks for different QPs and CU sizes. It finally achieved a -1.77% BD rate compared to H.266/VVC under LDP configuration. In [19], Katayama used LSTM [25] to enhance the reference frame to generate a virtual frame. This method achieves a BD rate reduction of 0.58% compared to H.266/VVC under the LDP configuration.

Most of the above-mentioned methods use a neural network to replace the DCTIF. Unlike these methods, we embedded the RBENN as a module into inter prediction, which could be used together with these methods. To maximize the coding efficiency, we only enhanced the CU blocks whose size was larger than 16×16, and also enhanced the CU blocks before affine motion compensation.

3. Proposed Method

Section 3.1 describes how the RBENN is embedded into the conventional motion compensation and affine motion compensation of H.266/VVC. The composition of the network is then described in Section 3.2. Finally, the dataset and the training method are described in Sections 3.3 and 3.4.

3.1. An RBENN Integrated in H.266/VVC

In the configuration file of H.266/VVC, CU sizes range from 4×4 to 64×64. As the variation of CUs with small sizes is usually not large and the number of pixels is also small, an RBENN only enhances the CUs that are larger than 16×16. We trained three networks,

denoted as RBENN 1, RBENN 2, and RBENN 3, to process CU blocks with minimum side lengths of 16, 32, and 64, respectively. Specifically, RBENN1 enhances CUs with sizes of {16 × 16, 32 × 16, 16 × 32, 16 × 64, 64 × 16}, RBENN2 enhances CUs with sizes of {32 × 32, 64 × 32, 32 × 64}, and RBENN3 enhances CUs with a size of {64 × 64}. Figure 4 depicts the system diagram of the encoder and decoder when the neural networks are integrated into the H.266/VVC. On the encoder side, the current CU finds the reference CU based on ME. An RBENN enhances the reference CU and replaces the original reference CU with its output. At the decoder, the current CU finds the reference CU in the reconstructed frame based on the decoded MV. The output of the RBENN is then filtered by a DCTIF to obtain the predicted CU.

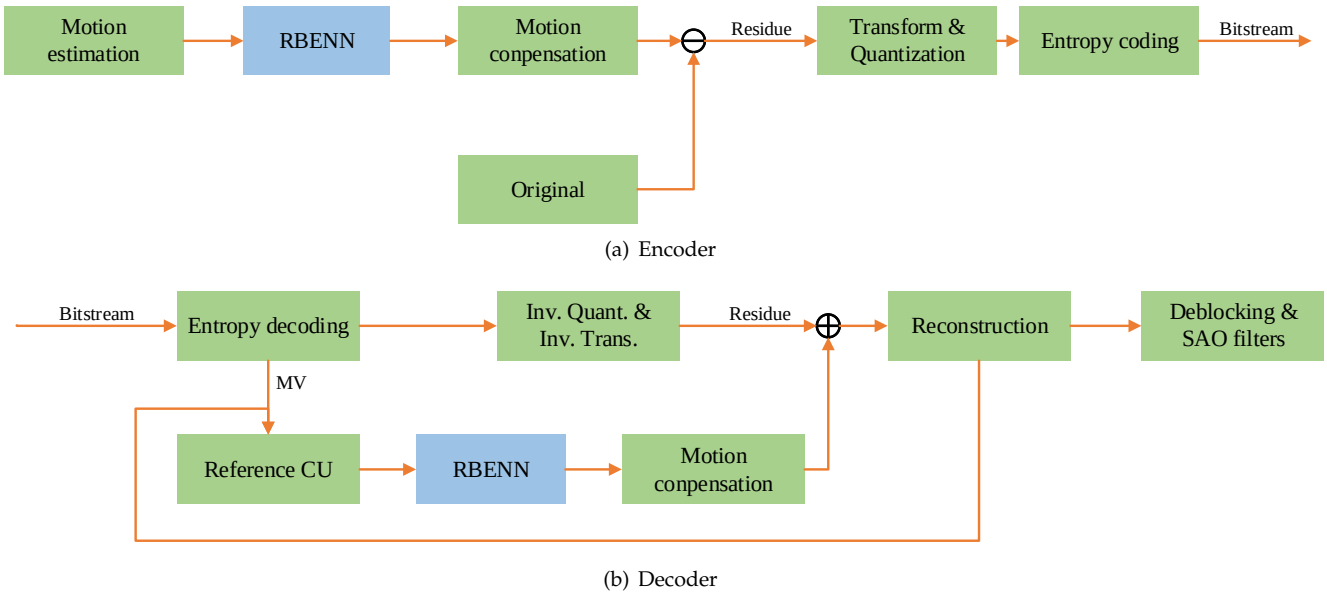
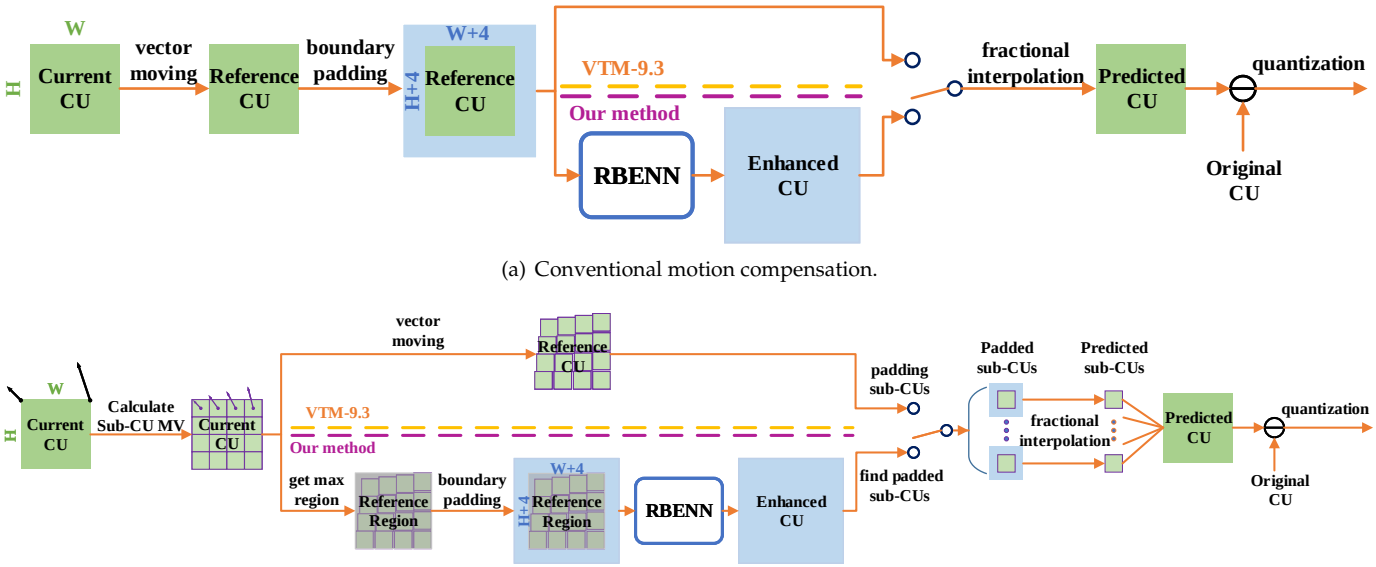


Figure 4. A diagram of RBENN integrated into H.266/VVC.

Figure 5a shows the process of integrating an RBENN into conventional motion compensation, while Figure 5b is the process of integrating an RBENN into affine motion compensation. In conventional motion compensation, the H.266/VVC standard finds the reference CU from the current CU according to the MV. First, the reference CU is expanded by 4 pixels, then a DCTIF is applied to generate the predicted CU. The residual CU is obtained by subtracting the predicted CU from the original CU. It is then transformed and quantized. Our method was designed to input a padded image patch into the proposed RBENN, and the output image patch is then interpolated and filtered by a DCTIF. In affine motion compensation, H.266/VVC splits the current CU into several 4 × 4 sub-CUs and calculates the MV of each sub-CU according to four-parameter or six-parameter affine models. The reference sub-CUs are then found through the corresponding MVs. By performing padding and fractional interpolation on the reference sub-CUs, the final predicted sub-CUs can be obtained. Finally, all the predicted sub-CUs are combined to obtain the predicted CU. In our method, we first obtain the minimum region containing all the reference sub-CUs and then pad the region and feed it to the neural network. Based on the corresponding position of the output image patch, the padded sub-CUs can be filtered by a DCTIF to obtain the predicted sub-CUs.



(a) Conventional motion compensation.

(b) Affine motion compensation.

Figure 5. The detail of an RBENN integrated before MC.

3.2. The Construction of an RBENN

Figure 6 depicts the overall framework of the RBENN. The network consists of 10 residual modules and 2 convolutional layers at the beginning and end. The network takes the padded reference CU before MC as input and passes it through a 3×3 convolution layer, which is followed by a rectified linear unit (ReLU) [26]. This process can be expressed as

$$\mathbf{F}_1 = \max(0, \mathbf{W}_0 \cdot \mathbf{X}), \tag{1}$$

where $\mathbf{W}_0$ represents the convolution kernel of the first layer, $\mathbf{X}$ is the input of the RBENN, and $\mathbf{F}_1$ is the output of the first layer.

The main body of the RBENN is composed of multiple residual convolution blocks (RCB), and the composition of the residual module [27] is also shown in Figure 6. It is composed of two convolutional layers, an ReLU, and a shortcut. The purpose of using a short cut is to make the network converged quickly. This process can be expressed as

$$\begin{cases} \mathbf{F}_{i,1} = \max(0, \mathbf{W}_{i,1} \cdot \mathbf{F}_i), \\ \mathbf{F}_{i,2} = \mathbf{W}_{i,2} \cdot \mathbf{F}_{i,1}, \\ \mathbf{F}_{i+1} = \mathbf{F}_i + \mathbf{F}_{i,2}, \end{cases} \tag{2}$$

where $1 \leq i \leq 10$, $i \in \{1, 2, \ldots, 10\}$, $\mathbf{F}_i$ is the input of the i-th module, $\mathbf{W}_{i,1}$ is the first convolution kernel of the i-th module, and $\mathbf{W}_{i,2}$ is the second convolution kernel of the i-th module, $\mathbf{F}_{i+1}$ is the output of the i-th module and also the input of the $(i+1)$-th module.

Finally, a $3 \times 3 \times 1$ convolution kernel is used to compress all feature maps into a residual image, which is added to the input image to obtain the final output:

$$\begin{cases} \mathbf{F}_{12} = \mathbf{W}_{12} \cdot \mathbf{F}_{11}, \\ \mathbf{Y} = \mathbf{F}_{12} + \mathbf{X}, \end{cases} \tag{3}$$

where $\mathbf{Y}$ is the output, $\mathbf{F}_{11}$ is the output of the last RCB, and $\mathbf{W}_{12}$ is the last convolution kernel.

The number of channels of convolution kernels in the whole network is 64, except in the last layer where it is 1.

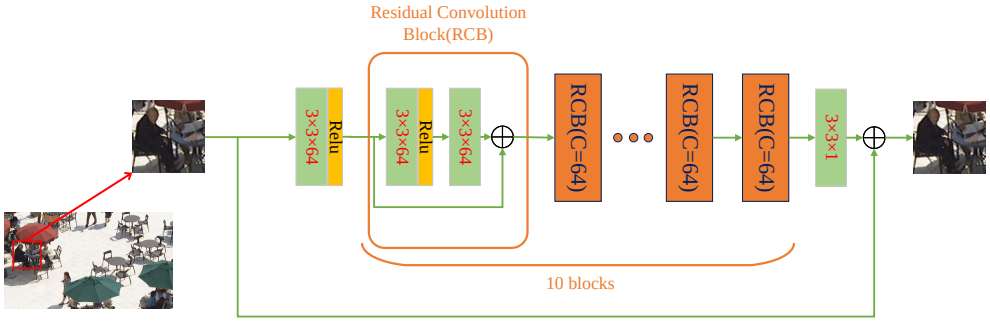


Figure 6. Framework of an RBENN.

3.3. Dataset for Training

In video coding, quantization will cause an irreversible loss of high-frequency image details. The aim of an RBENN is to restore the high frequency details of the image according to the low frequency information of the image.

In H.266/VVC, the relationship between different CUs is shown as

$$\begin{cases} CU_{pre} = DCTIF(CU_{ref}), \\ CU_{res} = CU_{ori} - CU_{pre}, \\ CU_{rec} = CU_{pre} + T^{-1}(Q^{-1}(Q(T(CU_{res})))), \end{cases} \tag{4}$$

in which, CU_{rec}, CU_{pre}, CU_{ref}, CU_{ori}, CU_{res} represent the reconstruction of the current CU at the decoder, the predicted CU, the reference CU, the original CU, and the residual CU, respectively. The reference CU is the CU found on a reference frame according to MV. The predicted CU is obtained from the reference CU through the interpolation filter. The residual CU is transformed $T(\cdot)$, quantized $Q(\cdot)$, inversely quantized $Q^{-1}(\cdot)$, and inversely transformed $T^{-1}(\cdot)$ to obtain the reconstructed CU.

In our proposed method, with the reference CU as input, the $RBENN()$ enhances the quality of the reference CU to replace the initial reference CU:

$$CU_{pre} = DCTIF(RBENN(CU_{ref})). \tag{5}$$

For easy implementation, the input of the training data is the predicted CU, which is extracted from the decoder directly, and the label of the training data is the original CU, which is extracted from the original video sequence.

The specific details are shown in Figure 7, where the purple dots represent the integer pixel position of the picture. The current CU in the position (X, Y) of the current frame finds the reference CU in the reference frame $(X + I_{mv_x}, Y + I_{mv_y})$ position through MV, where I_{mv} represents the integer displacement of MV and F_{mv} is the fractional displacement. The reference CU is then filtered to obtain the predicted CU.

We used 650 video sequences to generate the training data. These 650 video sequences came from the dataset of BVI-DVC [28]. The resolutions were 3840 × 2176, 1920 × 1088, 960 × 544, and 480 × 272. To better adapt to video contents with different resolutions, all these video sequences were encoded using VTM-9.3 with LDP configuration [29]. The quantization parameter was 22. The first 32 frames of each sequence were encoded, the predicted CU of the integer MV was obtained from the compressed bitstream, and the original CU was obtained from the video sequence. The dataset contained 16 × 16, 32 × 32, and 64 × 64 CUs for training RBENN1, RBENN2, and RBENN3, respectively. There were about 10,000,000 blocks for training RBENN1, 2,000,000 blocks for training RBENN2, and 300,000 blocks for training RBENN3.

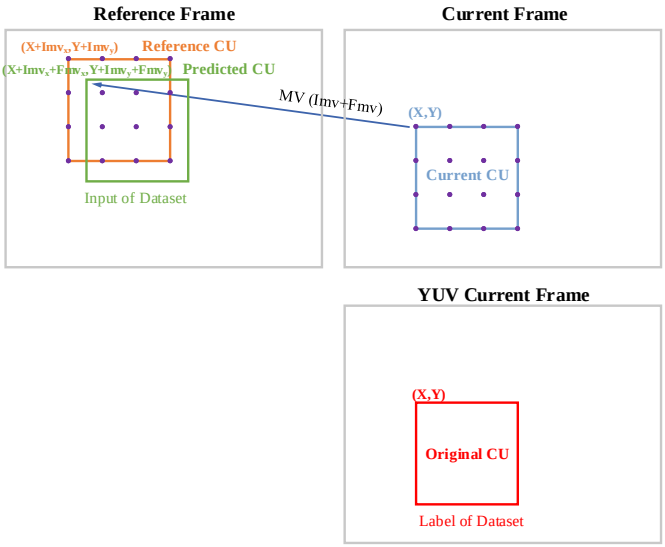


Figure 7. Illustration of the process of generating training data for training the RBENN.

3.4. Training Strategy

Specifically, given a collection of n training samples, $(\mathbf{x}_i, \mathbf{y}_i)$, $i \in \{1, 2, \ldots, n\}$, the mean squared error (MSE) was used to minimize the loss function. The loss function is formulated as

$$L(\Theta) = \frac{1}{n} \sum_{i=1}^{n} ||F(\mathbf{x}_i|\Theta) - \mathbf{y}_i||^2, \tag{6}$$

where Θ is the parameters of the network. The proposed RBENN was trained using PyTorch 1.8.0 on an NVIDIA GeForce GTX 2080Ti GPU. The loss function was minimized by using a first-order gradient based optimization method called Adam [30]. A batch-mode learning method was adopted with a batch size of 64. The momentum of Adam optimization was set to 0.9 and the momentum 2 was set to 0.999. The base learning rate was initially set to decay 0.1 times per 60 epochs from 0.0001. The training lasted 240 epochs.

4. Experimental Results

4.1. Experiment Settings

The proposed method was implemented based on H.266/VVC reference software VTM-9.3. The proposed RBENN was used to improve the quality of all luma CUs above 16×16 before MC. The LDP configuration was tested in the experiment under the H.266/VVC common test conditions. We encoded class D video sequences during the test, where the first 32 frames of each video sequence were encoded and decoded. There was no overlap between the training video sequences and the tested video sequences. The QPs used in our experiments varied among 22, 27, 32, and 37. The BD rate, which represents the bit rate increment at the same reconstruction quality, was used to measure the rate-distortion (RD) performance. The negative BD rate (BD rate reduction) means there is a positive coding gain, while a positive BD rate means there is a negative gain. To evaluate the reconstruction quality, PSNR and SSIM were used. Therefore, there are two kinds of BD rates: a PSNR-based BD rate and an SSIM-based BD rate.

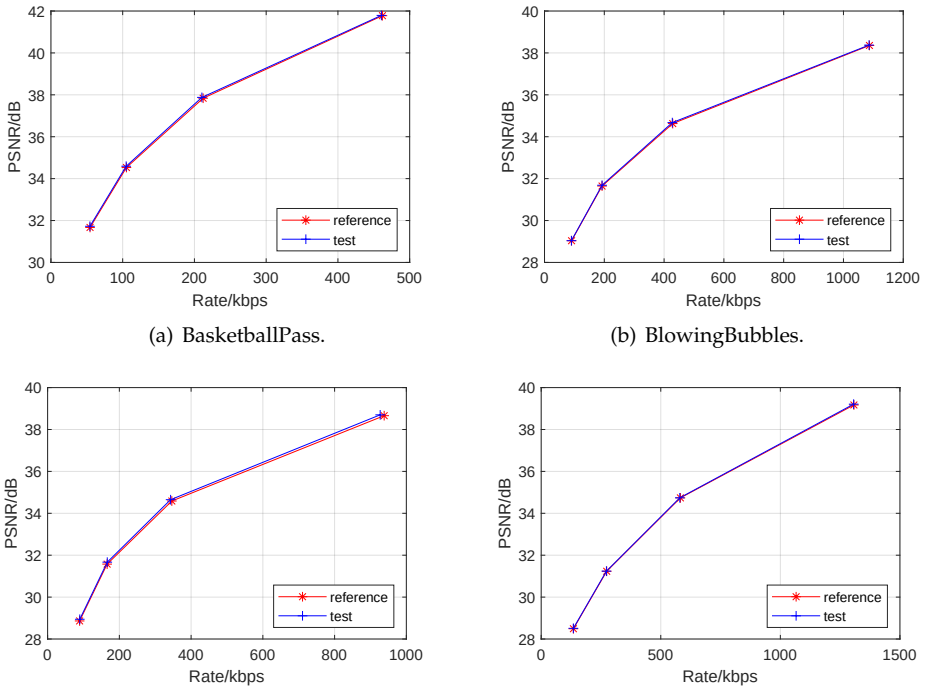
4.2. Compared with VTM 9.3

The experimental results are summarized in Table 1, and RD curves of the several video sequences are shown in Figure 8. We can see that the proposed method achieved, on average, a 1.35% PSNR-based BD rate reduction and a 2.28% SSIM-based BD rate reduction. We can also see that the RBENN can save -2.15% of the PSNR-based BD rate for BQSquare and -3.96% of the SSIM-based BD rate for BlowingBubbles, which contains severe aliasing and more noise, indicating that the RBENN is more efficient for anti-aliasing

and noise reduction. Figure 9 compares the subjective quality of the CUs with and without the proposed method. We can see that the RBENN did not affect the low-frequency information; however, it affected the high-frequency information, thus improving the image quality. Finally, we also tested the complexity of our proposed RBENN, as shown in Table 2. Since the CU is divided iteratively in the video coding process, frequent calls to the neural network lead to a large increase in complexity. In the future, we will reduce the complexity by designing lightweight networks.

Table 1. Results of Our RBENN Compared to VTM9.3 under LDP.

Class D	QP	VTM-9.3			Our Method			BD Rate Y	
		kbps	Y PSNR	Y SSIM	kbps	Y PSNR	Y SSIM	PSNR	SSIM
BasketballPass	22	462.02	41.77	0.9702	460.61	41.79	0.9704	−1.49%	−2.25%
	27	212.12	37.83	0.9402	210.39	37.88	0.9407		
	32	105.22	34.52	0.8965	104.87	34.59	0.8984		
	37	54.50	31.66	0.8420	54.34	31.73	0.8434		
BQSquare	22	938.15	38.65	0.9435	927.06	38.70	0.9439	−2.15%	−2.01%
	27	346.29	34.58	0.8972	342.84	34.65	0.8986		
	32	165.94	31.57	0.8381	166.29	31.66	0.8401		
	37	89.23	28.86	0.7660	89.65	28.94	0.7676		
BlowingBubbles	22	1086.89	38.35	0.9622	1085.67	38.37	0.9637	-1.14%	−3.96%
	27	428.88	34.62	0.9263	427.47	34.67	0.9280		
	32	192.49	31.64	0.8758	192.46	31.68	0.8778		
	37	90.44	29.03	0.8341	89.94	29.03	0.8371		
RaceHorses	22	1307.56	39.17	0.9614	1306.50	39.21	0.9619	−0.61%	−0.91%
	27	582.08	34.73	0.9144	579.64	34.74	0.9146		
	32	273.95	31.23	0.8471	272.80	31.24	0.8481		
	37	134.72	28.49	0.7725	134.48	28.51	0.7719		
Average								−1.35%	−2.28%



(a) BasketballPass. (b) BlowingBubbles.

(c) BQSquare. (d) RaceHorses.

Figure 8. RD curves of class D video sequences under LDP.

Table 2. Complexity of our method.

Class D	LDP	
	EncTime	**DecTime**
BasketballPass	70,977%	17,959%
BQSquare	64,455%	30,334%
BlowingBubbles	58,523%	26,708%
RaceHorses	47,806%	26,507%
Average	56,653%	24,021%

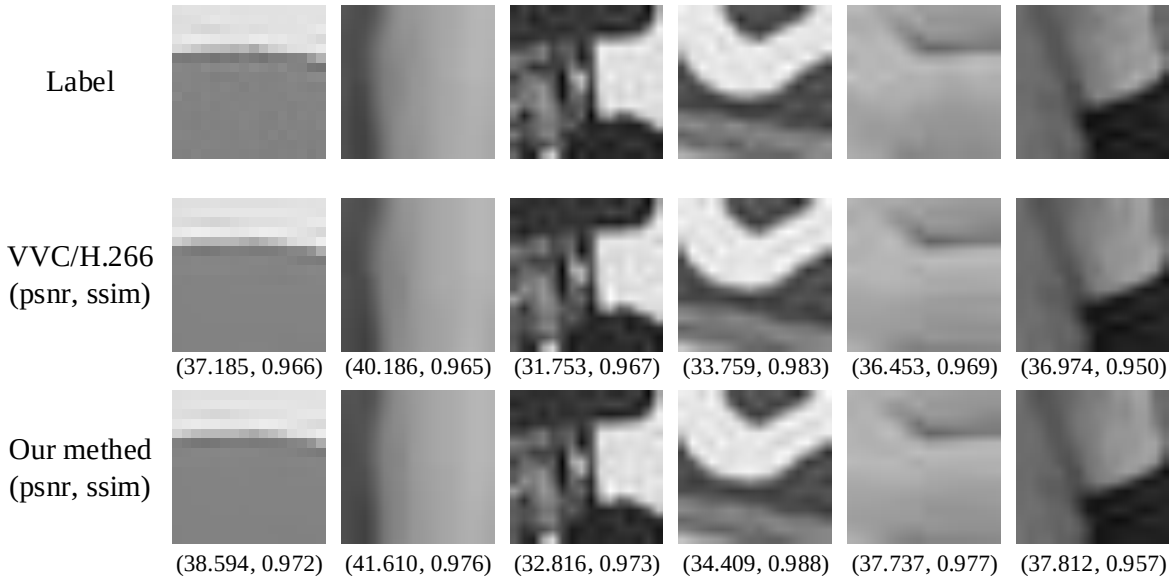


Figure 9. Quality enhancement results on reference CU performed by the RBENN.

4.3. Result Analysis

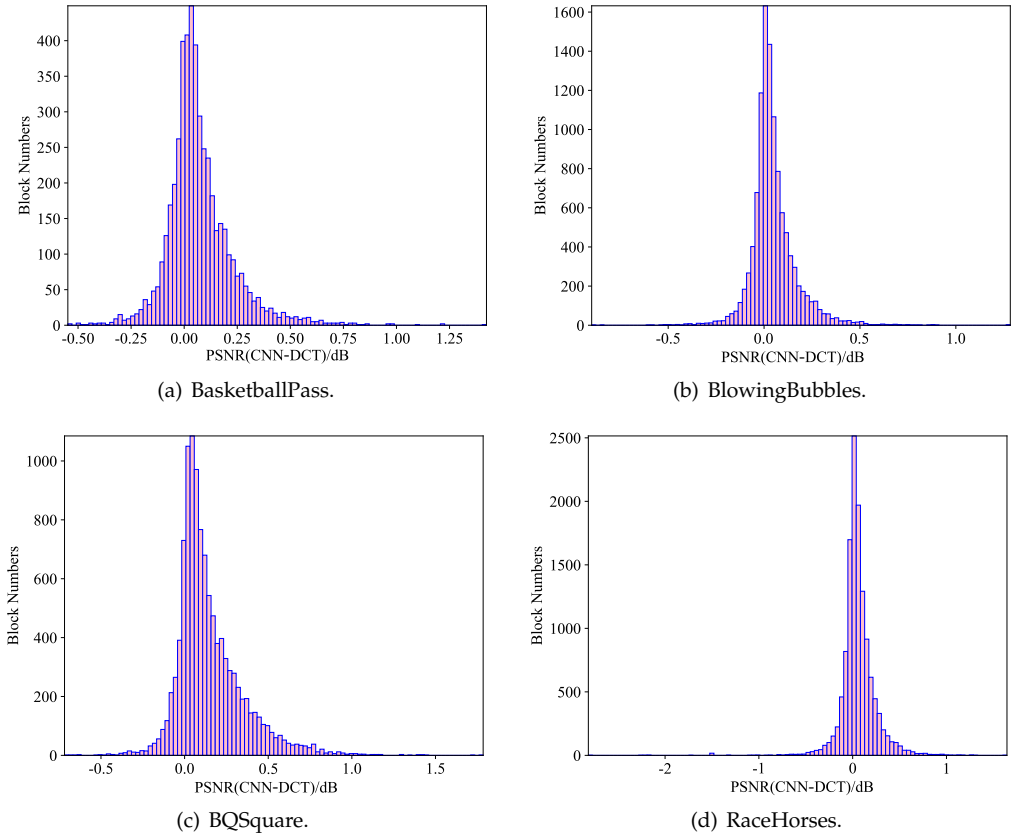
Since the RBENN enhances the reference CU before MC, the performance of the RBENN can be more clearly represented by comparing the PSNR of the reference CU and the original CU before and after the RBENN. Table 3 shows the average gains of the CU with different sizes in the four video sequences. The PSNR gains were calculated as the average of all CUs. Considering the different sizes and numbers of each CU, the PSNR gains were calculated as

$$PSNR\ gain = \frac{\sum_{W,H\in\{16,32,64\}}\left[(W\times H)\times\sum PSNR_{W,H}\right]}{\sum_{W,H\in\{16,32,64\}}\left[(W\times H)\times\sum NUM_{W,H}\right]}, \quad (7)$$

where $(W \times H)$ is the area of a CU, $\sum PSNR_{W,H}$ represents the sum of PSNR gains for each CU after the RBENN, and $\sum NUM_{W,H}$ denotes the total number of CUs. According to Table 3, we can see that there was a 0.052 dB PSNR gain for all CUs larger than 16×16, on average. For *BQSquare*, the PSNR gain reached 0.110 dB, which was consistent with the coding results. We can also see that the RBENN performed better for CUs with sizes of $\{16 \times 16, 32 \times 16, 16 \times 32, 16 \times 64, 64 \times 16\}$. We believe that this was because the training set used for training RBENN1 was large, leading to a higher generalization ability. Figure 10 visually displays the distribution of the PSNR gains. We can see that, in the *RaceHorses* the RBENN achieved significant negative gains for a few CUs. This is the reason why limited coding improvement was obtained for it. In the future, we will add flags in the coding process to further improve the coding efficiency.

Table 3. PSNR gain for RBENN for different CUs.

Class D	Average (CNN−DCT) PSNR for Each Block									
	16 × 16	**16 × 32**	**32 × 16**	**32 × 32**	**16 × 64**	**64 × 16**	**32 × 64**	**64 × 32**	**64 × 64**	**PSNR Gain**
BasketballPass	0.125	0.088	0.066	0.006	0.080	0.009	0.012	0.028	0.005	0.036
BQSquare	0.175	0.177	0.179	0.051	0.203	0.150	0.068	0.038	0.055	0.110
BlowingBubbles	0.067	0.050	0.040	0.011	0.029	0.028	0.014	0.008	0.000	0.030
RaceHorses	0.072	0.047	0.014	0.003	0.027	−0.040	0.007	−0.042	−0.15	0.030
Average	0.110	0.091	0.075	0.018	0.085	0.037	0.025	0.008	−0.023	0.052



(a) BasketballPass. (b) BlowingBubbles.

(c) BQSquare. (d) RaceHorses.

Figure 10. Number of CUs with different gains.

5. Conclusions

We proposed a neural-network-based quality enhancement for the reference CU to improve the accuracy of inter prediction. The network takes the luma reference CU before motion compensation as input. The output quality-enhanced blocks of the network are then interpolated by a DCTIF to generate a fractional motion compensation block. The proposed method was also implemented for affine motion compensation. Experimental results demonstrated that the proposed method could achieve a 1.35% reduction in the BD rate on average for the luma component, compared with H.266/VVC. In the future, we will design a more effective neural network to improve the coding efficiency and lightweight neural networks to reduce the complexity, while preserving the coding efficiency.

Author Contributions: Conceptualization, Y.C., H.Y. and S.J.; methodology, Y.C., H.Y., S.J. and C.F.; software, Y.C.; validation, Y.C.; formal analysis, Y.C.; investigation, Y.C.; resources, H.Y., S.J. and C.F.; data curation, Y.C., S.J. and C.F.; writing—original draft preparation, Y.C.; writing—review and editing, Y.C., H.Y., S.J. and C.F.; visualization, Y.C. and S.J.; supervision, S.J. and C.F.; project administration, H.Y. and S.J.; funding acquisition, H.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the National Natural Science Foundation of China under Grants 62222110 and 62172259, the Taishan Scholar Project of Shandong Province (tsqn202103001), the open project program of the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, under Grant VRLAB2021A01, the Natural Science Foundation of Shandong Province of China under Grant (ZR2022ZD38), the Central Guidance Fund for Local Science and Technology Development of Shandong Province under Grant YDZX2021002, and the Major Scientific and Technological Innovation Project of Shandong Province under Grant 2020CXGC010109.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used and analyzed during the current study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

RBENN	reference-block-enhancement neural network;
LDP	low-delay P;
MC	motion compensation;
ME	motion estimation;
DCTIF	discrete cosine transform interpolation filter;
CU	coding unit;
BD rate	Bjøntegaard delta rate;
PSNR	peak signal to noise ratio.

References

1. Rao, K.R.; Ahmed, N.; Natarajan, T. Discrete Cosine Transfom. *IEEE Trans. Comput.* **1974**, *23*, 90–93.
2. Delogne, P.; Cuvelier, L.; Maison, B.; Van Caillie, B.; Vandendorpe, L. Improved interpolation, motion estimation, and compensation for interlaced pictures. *IEEE Trans. Image Process.* **1994**, *5*, 482–491. [CrossRef]
3. Bjøntegaard, G. Calculation of Average PSNR Differences between RD-Curves. ITU-T SG.16 Q.6 VCEG-M33. 2001. Available online: https://www.itu.int/wftp3/av-arch/video-site/0104_Aus/VCEG-M33.doc (accessed on 18 December 2022).
4. Ugur, K.; Alshin, A.; Alshina, E.; Bossen, F.; Han, W.; Park, J.; Lainema, J. Interpolation filter design in HEVC and its coding efficiency—Complexity analysis. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–31 May 2013; pp. 1704–1708.
5. Yan, N.; Liu, D.; Li, H.; Wu, F. A Convolutional Neural Network Approach for Half-pel Interpolation in Video Coding. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 28–31 May 2017; pp. 1–4.
6. Yan, N.; Liu, D.; Li, H.; Li, B.; Li, L.; Wu, F. Convolutional Neural Network-Based Fractional-Pixel Motion Compensation. *IEEE Trans. Circuits Syst. Video Technol.* **2019**, *29*, 840–853. [CrossRef]
7. Yan, N.; Liu, D.; Li, H.; Li, B.; Li, L.; Wu, F. Invertibility-driven Interpolation Filter for Video Coding. *IEEE Trans. Image Process.* **2019**, *28*, 4912–4925. [CrossRef]
8. Xia, S.; Yang, W.; Hu, Y.; Ma, S.; Liu, J. A Group Variational Transformation Neural Network for Fractional Interpolation of Video Coding. In Proceedings of the 2018 Data Compression Conference, Snowbird, UT, USA, 27–30 March 2018; pp. 127–136.
9. Liu, J.; Xia, S.; Yang, W.; Li, M.; Liu, D. One-for-All: Grouped Variation Network-Based Fractional Interpolation in Video Coding. *IEEE Trans. Image Process.* **2019**, *28*, 2140–2151. [CrossRef]
10. Huo, S.; Liu, D.; Wu, F.; Li, H. Convolutional Neural Network-Based Motion Compensation Refinement for Video Coding. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018; pp. 1–4.
11. Wang, Y.; Fan, X.; Xiong, R.; Zhao, D.; Gao, W. Neural Network-Based Enhancement to Inter Prediction for Video Coding. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *32*, 826–838. [CrossRef]

12. Pham, C.D.; Zhou, J. Deep Learning-Based Luma and Chroma Fractional Interpolation in Video Coding. *IEEE Access* **2019**, *7*, 112535–112543. [CrossRef]
13. Prette, N.; Valsesia, D.; Bianchi, T. Deep Multiframe Enhancement for Motion Prediction in Video Compression. In Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems (ICECS), Dubai, United Arab Emirates, 28 November–1 December 2021; pp. 1–6.
14. Sullivan, G.J.; Ohm, J.R.; Han, W.J.; Wieg, T. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Trans. Circuits Syst. Video Technol.* **2012**, *22*, 1649–1668. [CrossRef]
15. Murn, L.; Blasi, S.; Smeaton, A.F.; O'Connor, N.E.; Mrak, M. Interpreting CNN For Low Complexity Learned Sub-Pixel Motion Compensation In Video Coding. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Abu Dhabi, United Arab Emirates, 25–28 October 2020; pp. 798–802.
16. Murn, L.; Blasi, S.; Smeaton, A.F.; Mrak, M. Improved CNN-Based Learning of Interpolation Filters for Low-Complexity Inter Prediction in Video Coding. *IEEE Open J. Signal Process.* **2021**, *2*, 453–465. [CrossRef]
17. Galpin, F.; Bordes, P.; Dumas, T.; Nikitin, P.; Le Leannec, F. Neural Network based Inter bi-prediction Blending. In Proceedings of 2021 International Conference on Visual Communications and Image Processing (VCIP), Munich, Germany, 5–8 December 2021; pp. 1–5.
18. Jin, D.; Lei, J.; Peng, B.; Li, W.; Ling, N.; Huang, Q. Deep Affine Motion Compensation Network for Inter Prediction in VVC. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *32*, 3923–3933. [CrossRef]
19. Katayama, T.; Song, T.; Shimamoto, T.; Jiang, X. Reference Frame Generation Algorithm using Dynamical Learning PredNet for VVC. In Proceedings of the 2021 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 10–12 January 2021; pp. 1–5.
20. Bross, B.; Chen, J.; Ohm, J.R.; Sullivan, G.J.; Wang, Y.K. Developments in International Video Coding Standardization After AVC, With an Overview of Versatile Video Coding (VVC). *Proc. IEEE* **2021**, *109*, 1463–1493. [CrossRef]
21. Dong, C.; Loy, C.C.; He, K.; Tang, X. Image Super-Resolution Using Deep Convolutional Networks. *Trans. Pattern Anal. Mach. Intell.* **2016**, *38*, 295–307. [CrossRef]
22. Dai, Y.; Liu, D.; Wu, F. A Convolutional Neural Network Approach for Post-Processing in HEVC Intra Coding. In Proceedings of the 2017 International Conference on Multimedia Modeling (MMM), Reykjavik, Iceland, 4–6 January 2017; pp. 28–39.
23. Sun, D.; Yang, X.; Liu, M.; Kautz, J. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8934–8943.
24. Zhang, K.; Zuo, W.; Chen, Y.; Meng, D.; Zhang, L. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE Trans. Image Process.* **2017**, *26*, 3142–3155. [CrossRef]
25. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
26. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 807–814.
27. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
28. Ma, D.; Zhang, F.; Bull, D.R. BVI-DVC: A Training Database for Deep Video Compression. *IEEE Trans. Multimed.* **2022**, *24*, 3847–3858.
29. Liu, S.; Segall, A.; Alshina, E.; Liao, R. JVET Common Test Conditions and Evaluation Procedures for Neural Network-Based Video Coding Technology. Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29, Document no. JVET-T2006. January 2021. Available online: https://www.itu.int/wftp3/av-arch/jvet-site/2021_01_U_Virtual/JVET_Notes_d1.docx (accessed on 18 December 2022).
30. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the 2015 International Conference on Learning Representations(ICLR), San Diego, CA, USA, 7–9 May 2015; pp. 1–15.