IEEE Open Journal of
Circuits and Systems

Received 5 July 2021; revised 30 September 2021; accepted 15 October 2021. Date of current version 22 November 2021.

Digital Object Identifier 10.1109/OJCAS.2021.3124243

Codec-Simulation Network for Joint Optimization of Video Coding With Pre- and Post-Processing

KAITIAN QIU, LU YU (Senior, Member, IEEE), AND DAOWEN LI

College of Information Science & Electronic Engineering, Zhejiang University, Hangzhou 310058, China

This article was recommended by Associate Editor Z. Chen.

CORRESPONDING AUTHOR: L. YU (e-mail: yul@zju.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 62071427, and in part by the Key Research and Development Program of Zhejiang Province, China, under Grant 2021C01119.

ABSTRACT In real video coding systems, there might be pre-processing before the encoding and post-processing after the decoding. Those pre- and post-processing should be jointly optimized considering the existing of distortions introduced by lossy coding in between. This paper proposes a differentiable network to simulate the non-differentiable traditional coding to make the joint optimization of pre- and post-processing possible. The proposed codec-simulation network can achieve up to 49.4dB simulation accuracy for HEVC codec in intra mode. When we use down-sampling and up-sampling as examples of pre- and post-processing of a video codec, the joint optimization of these two processing modules in a video coding system can result in 46.88% and 55.15% BD-rate reduction based on PSNR and SSIM compared with that without joint optimization.

INDEX TERMS codec-simulation, pre- /post-processing, down-sampling, super-resolution, HEVC.

I. INTRODUCTION

WITH the development of technology, the demand for video transmission continues to increase, which urgently requires us to improve the coding efficiency of video compression algorithms. For now, the mainstream video compression standard in the market is HEVC/H.265, whose algorithm is complicated. Therefore, there are specific hardware to accelerate the calculation process.

In order to adapt to the existing hardware, there are many methods [1]–[6] proposed to improve the overall coding efficiency through pre- and post-processing. For example, to mitigate blocking artifacts caused by transform and quantization in video compression, [1], [2], [3] add simple time-domain pre- and post-processing modules to DCT (Discrete Cosine Transform)-based infrastructures to improve coding efficiency. The pre-filter acts as a flattening operator, trying to make the input data of each DCT block as homogeneous as possible, while the post-filter acts as a smooth interpolation, reducing blocking artifacts. Reference [4] proposed a pre-/post-filtering method based on global motion estimation (GME) and global motion compensation (GMC) to improve video coding efficiency. The pre-processing module estimates the global motion information between the coded frame and the key frame, which is transmitted to the decoder through supplemental enhancement information (SEI). Then in the post-processing module, the frame is reconstructed by using the global motion information. In order to make the codec better compress the depth images, which have richer changes in the edge area compared with color images, [5] proposes to modify the edge block in the depth image into a plane block in the pre-processing, so that it can be better compressed by the encoder, while in the post-processing, the decoded depth image will be enhanced by edge information to reconstruct the final output. To reuse existing low resolution codec design and keep high coding efficiency in the meantime for high resolution content, LCEVC [6] introduces down-sampling as pre-processing and up-sampling as post-processing for the existing codec working in low resolution. These methods all improve the overall coding efficiency through pre- and post-processing without changing the core codec.

However, the above pre- and post-processing methods are all traditional methods designed and optimized based on

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

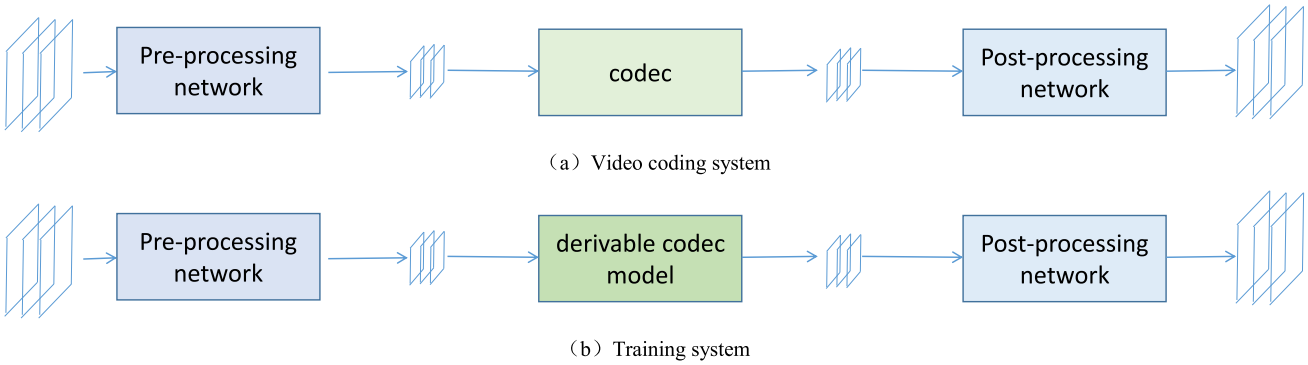


FIGURE 1. System frameworks.

experience. With the emergence of neural networks, data-driven machine learning methods can have better performance than traditional methods in various fields. Taking up-sampling as an example, super-resolution technology aims to recover high-resolution objects from the observed low-resolution objects (including videos and images). Different from the traditional method that uses bicubic interpolation to achieve super-resolution, the idea of using neural networks to solve the super-resolution problem was first proposed by He's team in 2014 named as SRCNN [7]. When the scale of up-sampling is 4 times, the PSNR of the network reconstructed image compared to the original image can be 2 dB higher than the PSNR of the bicubic interpolation reconstructed image compared to the original image. Later, the ResNet structure proposed in [22] has also been applied in the field of super-resolution as EDSR [8] and SRResNet [9]. They both have realized the further improvement of super-resolution performance. Similarly, RDDB [11] and RCAN [10] achieved better performance by using DenseNet [23] and attention method [24] for reference. Then the objective quality of the super-resolution image has almost reached into a bottleneck period when it is difficult to make a huge breakthrough. It is pointed out that bicubic down-sampling is usually used to generate lower resolution image which introduces visual artifacts, such as blur and aliasing. This may largely limit the performance of the super-resolution reconstruction result [12], [13], [14]. So researchers treat the down-sampling image and up-sampling image as a two-way problem, i.e., reducing resolution may cause information loss and increasing resolution try to recover the lost information. Therefore, considering these two issues together by jointly optimizing down-sampling network and up-sampling network, the information lost in the resolution reduction process might be rebuildable by the super-resolution network. Experimental results in [12] show that the joint optimization can not only improve the objective quality of the final reconstructed image, but also greatly improve the subjective quality of the reconstructed image to a certain extent. Therefore, using jointly optimized neural networks as the pre- and post-processing of video codec may have some performance improvement space.

Since the traditional video coding process is non-differentiable, it is worth noting that joint optimization of the pre- and post-processing networks will meet a serious problem. The non-differentiable coding process prevents the backward propagation of gradient from the post-processing neural network to the pre-processing one. Therefore, this paper proposes to design a neural network to simulate the traditional video coding between the pre- and post-processing so that the joint optimization of the whole system can be realized.

The remainder of the paper is organized as follows. Section II will analyze how to solve the joint optimization problem. Section III will explain the detailed design of a codec-simulation network to make the codec differentiable and the predicted distortion approximated by the simulator close to ground truth. Section IV provides experimental results to prove the effectiveness of the whole design. And finally, we will summarize the solution in Section V.

II. PROBLEM ANALYSIS

A diagram of a video coding system is shown in Fig. 1(a). Firstly, the video sequence is pre-processed with a pre-processing network. Then the processed video is fed into a codec and the decoded video sequence with lossy is processed by a post-processing network which provides the final reconstructed video. In most cases, the pre-processing network and the post-processing network should match with each other, so joint training is indispensable. The joint optimization of networks is based on backward propagation of gradient through the whole pipeline of the coding system. Suppose

$$L = PRE(X) \tag{1}$$

$$M = Codec(L) \tag{2}$$

$$Y = POST(M) \tag{3}$$

where X represents for the input. $PRE()$, $Codec()$ and $POST()$ represent for pre-processing network, codec and post-processing network respectively. L, M and Y represent for the outputs from the corresponding modules. The optimization of the pre-processing network, considering the existence of the codec and post-processing, needs

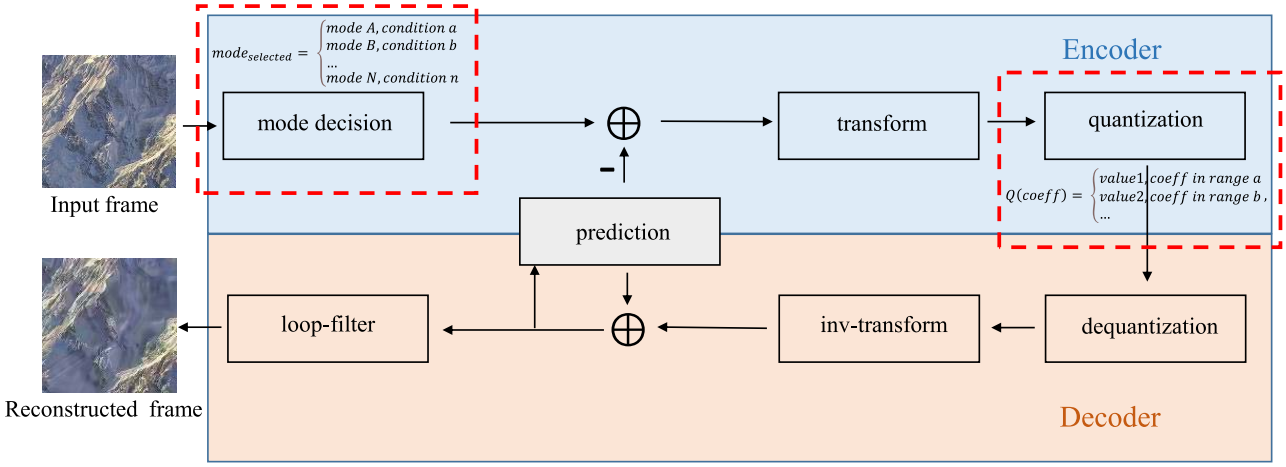


FIGURE 2. Basic framework of hybrid video coding.

the following gradient propagation process in joint training,

$$\frac{\partial Loss}{\partial PRE_\theta} = \frac{\partial Loss}{\partial Y} * \frac{\partial POST(M)}{\partial M} * \frac{\partial Codec(L)}{\partial L} * \frac{\partial PRE(X)}{\partial PRE_\theta} \quad (4)$$

where, $Loss$ represents the loss function of training, and PRE_θ represents parameters of the pre-processing network. This means the codec in the pipeline must be differentiable.

Next, we will first analyze the differentiability of popular video codec, clarify the necessity and target of codec-simulation and then present two possible ways of codec simulation: simulating coding procedures and simulating coding results.

A. ANALYSIS OF DIFFERENTIABILITY OF VIDEO CODEC

A hybrid video coding framework is shown in Fig. 2. The most popular video codecs, such as AVC, HEVC, VVC, AVS1/2/3 etc. all follow this framework. The major coding modules include mode decision, inter/intra prediction, transformation, quantization and in-loop filter. An input picture will first be divided into fixed-sized coding units. For each coding unit, the encoder will select one optimal coding mode from multiple block partition modes and prediction modes considering better coding performance. Then the encoder derives residuals between the original coding unit and the predicted ones from the selected prediction mode, and performs transform and quantization on the residuals to generate syntax elements for lossless entropy encoding. At the decoding side, the inverse procedure, including dequantization, inverse transform, prediction compensation with signaled coding mode and in-loop filter, will be conducted to generate the reconstructed picture.

Each intra prediction mode can be regarded as a finite impulse responses extrapolation filter, while inter prediction mode as a translational shift with finite impulse responses interpolation filter (one hypothesis prediction) or a combination of such filters (two hypotheses prediction). And a given block-sized transform used in video coding can also be regarded as linear combination of pixel values in that block. So it is obvious that all these modules are differentiable.

The two major issues of differentiability in the codec are caused by quantization and mode decision.

The quantization is performed on transformed coefficients and results in discrete quantized values. A generic quantization equation is

$$Q(coeff) = \begin{cases} value1, & coeff \text{ in range } a \\ value2, & coeff \text{ in range } b \\ & \dots \end{cases}, \quad (5)$$

in which $Q(coeff)$ represents for the quantized value of coefficient $coeff$ with discrete value. The mode selection can also be described by a generic equation as

$$mode_{selected} = \begin{cases} mode\ A, & condition\ a \\ mode\ B, & condition\ b \\ & \dots \\ mode\ N, & condition\ n \end{cases}, \quad (6)$$

in which $mode_{selected}$ represents for selected mode. No matter how many candidate modes there are, what kind of mode decision strategy and what kind of criteria the encoder uses for the mode selection, the selected mode is a discrete value representing for discrete coding unit partition mode and prediction mode. The discrete results of quantization and mode selection caused the non-derivability of the codec.

According to above analysis, it is obvious that a traditional hybrid video codec is non-differentiable. That means $\frac{\partial Codec(L)}{\partial L}$ in equation (4) does not exist and the optimization of pre-processing considering the existence of lossy coding and post-processing is impossible.

B. TARGET OF CODECSSIMULATION AND POSSIBLE SOLUTIONS

To overcome the difficulty which prevents the gradient backward propagation for joint training of pre- and post-processing networks before and after the codec, a training framework (Fig. 1(b) and Fig. 3) replacing the inference one is proposed in this paper. The novelty of this training framework is introduction of a codec simulator which is expected

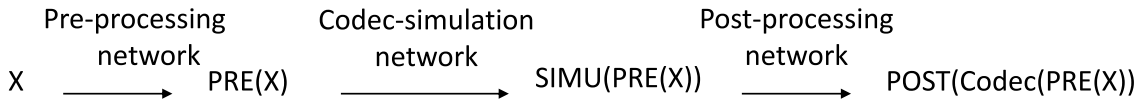


FIGURE 3. Joint optimization framework for pre- and post- processing networks.

to be differentiable and able to estimate similar output as that of the codec. Now, the gradient propagation process for the optimization of the pre-processing network, considering the existence of the codec and post-processing can be modified to,

$$\frac{\partial Loss}{\partial PRE_\theta} = \frac{\partial Loss}{\partial Y} * \frac{\partial POST(M)}{\partial M} * \frac{\partial SIMU(L)}{\partial L} * \frac{\partial PRE(X)}{\partial PRE_\theta} \quad (7)$$

where $SIMU()$ represents for the codec-simulation. The goal of the simulator design is to make $SIMU(L)$ be differentiable and approach to $Codec(L)$. The introduction of the codec-simulation makes the joint optimization possible.

Therefore, we set the major target of codec-simulation as that the mismatch between the simulator model and the real codec should be much smaller than typical coding distortion. Additionally, since the simulator will be used in the joint optimization of a pre- and post-processing network for a specific codec design, the simulation network should be general enough to adapt to different codec designs and simple enough to avoid being too time-consuming for joint optimization.

There are two possible ways to design a simulator of codec. One is a straightforward approach which simulates the whole coding procedures by assembling coding modules with differentiable networks. And the other one is directly simulating the coding results which may not follow the coding procedures. Part C and D will check if these two ways can meet the above-mentioned criteria of codec-simulation.

C. SIMULATION OF CODING PROCEDURES

As we analyzed in part A of this section, the major challenges of the traditional hybrid video codecs are the quantization and mode selection modules.

The differentiable approximation of quantization of jpeg compression [15] and in the training of end-to-end image compress network [16], [17], [18] have been raised and solved. There are three main methods for the approximation.

1. Distribution approximation [17], [18] by adding a uniformly distributed random noise to the input: the data distribution after adding the noise could be equal to the data distribution after quantizing, thereby the simulation of quantization could be realized.

2. Function approximation by replacing the quantization function with a similar differentiable function. For example, [14] proposes

$$\lfloor x \rfloor = \lfloor x \rfloor + (x - \lfloor x \rfloor)^3 \quad (8)$$

to achieve the approximation of the quantization function, where $\lfloor\rfloor$ means quantization. Similarly, [19] proposes to replace the non-differentiable step function with differentiable sigmoid function thus replace the entire quantization function.

3. Gradient approximation: Reference [16] proposes to keep the quantization function unchanged during the forward inference, and set the gradient of the quantization function to be 1 during the gradient back propagation, so as to solve the non-differentiation problem.

The differentiable approximation of mode selection has not been touched except that [25], [26], [27] try to realize block partition by using neural networks. Those networks predict probabilities of division of all possible block partition boundaries. In order to make a soft switching among different block partitions, we try to use a weighted combination of all possible partition results by taking the probability of each partition mode as the weighting factor. Similarly, we need to design a prediction mode estimation network to provide probability distribution among all prediction modes and describe each prediction mode by a neural network model. Then we can weighted combine all prediction modes by using probability distribution of prediction modes.

We tried to assemble the above-mentioned networks together with other differentiable coding tools as a simulator of HEVC intra codec and trained such a simulator. It was found that the reconstructed image from the simulator is far from the actual decoded image. The reason leading to the failure of the simulator might be that there are too many sub-networks and it is hard to control the performance of each sub-network.

It can be seen that design of such a coding procedure simulation network is seriously dependent on the knowledge about all quantization methods, prediction modes and potential partition modes and thus it's an extremely high manpower demand and time-consuming way. That will seriously limit the flexibility and usability of the simulator.

D. SIMULATION OF CODING RESULTS

Direct simulation of results of codec is a substituted way to generate a differentiable network approximating video codec. Different from the simulation of coding procedures, the simulation of coding results does not need to separate compress and decompress of the input, and does not output the coded bitstream. It only needs to make the distribution of distortion from the simulator similar with that from the real codec. Deep neural networks borrowed from low-level image processing could be powerful to learn the characteristics of decoded image by sufficient training data. In fact, there are some data-driven methods to replace a certain module of the hybrid coding framework through simulating the results directly in previous study. For example, [28] proposed to do intra prediction

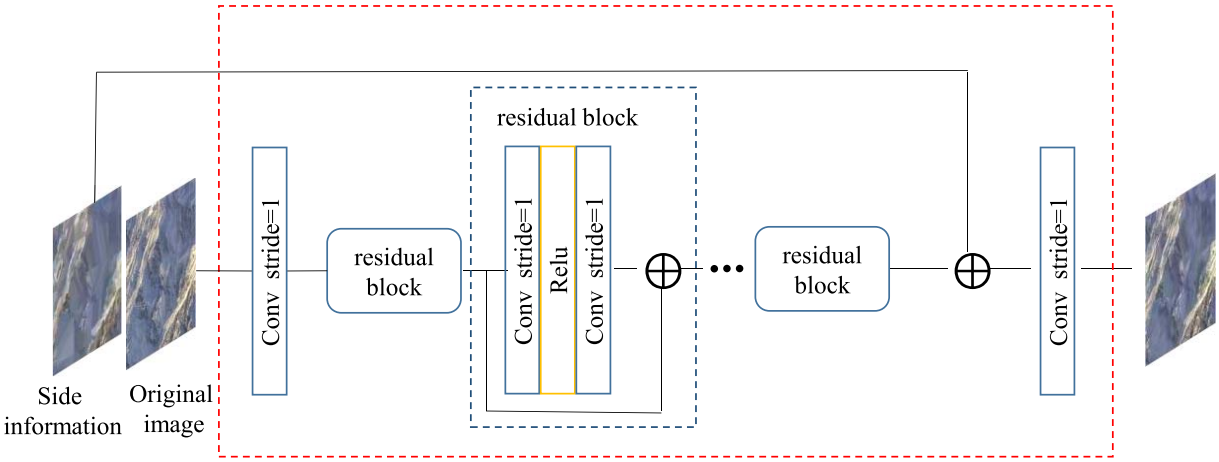


FIGURE 4. Codec-simulation network.

through using a single-layer filter to simulate the results, and [29] aims at inter prediction. Refereence [30], [31], [32] use network to simulate the result in the loop filter module so as to improve the coding efficiency. These methods are all based on data-driven approaches, and it is proved that the network can fit the results well according to the data. Compared to simulating only one part of the coding process, we simulates the results of the entire coding process, but their essence is the same.

Therefore, the generation of a simulation network can be easy.

In real application, the same encoder working in different conditions or different encoders working in the same condition will produce different distortions. Correspondingly, by establishing different databases for different coding distortions, we can train simulation networks to adapt to those encoders in different situations. Considering the optimization of pre-processing coupled with the specific design of codec and post-processing, information about the pre-processing or post-processing might be necessary to be communicated from encoder side to decoder side. Existing video coding standards have the way to implement such communication via SEI messages in bitstream. Therefore, the design of a differentiable network simulating coding results has practical value.

The detailed design of the coding results simulation network will be described in Section III.

III. CODEC-SIMULATION NETWORK

In the previous section, we analyzed that it is too complicated to simulate the coding procedures and obtain a differentiable simulator. Therefore, we try the other way which directly simulate the coding results instead of coding procedures in this section. The final holistic framework of the codec-simulation network is shown in Fig. 4. The inputs to the simulation network include not only the original image for compression but also supplementary side information. The network structure is a ResNet, which is very popular in low-level image processing area. Further discussions on network structure and inputs are in the following part A and B. Part C will introduce how to use the simulated coding model to realize joint optimization of the pre- and post-processing networks.

A. CODEC-SIMULATION NETWORK STRUCTURE

To simulate diversity of coding distortions which may happen for different video contents, the codec-simulation network should be deep enough to learn the characteristics from a dataset which includes samples with input and output of a real codec.

Since such a simulation network accepting original image as input and estimating reconstructed image of a codec deals with input image as a low-level pixel-to-pixel processing problem, network structures of image denoising and super-resolution can be borrowed. ResNet is a relatively mature network structure when dealing with such low-level image problems. Its variations, SRResNet [9] and DnCNN [20] have achieved stable performance at least in super-resolution and denoising. ResNet structure has a strong generalization ability and is easy to converge. Its structure is a stack of residual blocks each of which uses a residual structure. Feature before convolution operations is added to that after convolution operations so that the backbone of the network only processes sparse residuals, which improving convergence and stability of the network. Therefore, we decide to use the backbone of SRResNet as structure of the codec-simulation network, shown in Fig. 4 in the red box.

We have also tried many different network structures, such as DenseNet, channel attention structure, and the combination of ResNet and a differentiable approximation of quantization operation using method in [15], etc. Preliminary experiments training these potential network structures simulating HEVC intra coding are conducted. Inference results shown in Table 1 indicate that different network structures and scale of the models do not make significant differences of simulation performance. Here we use Peak-Signal-to-Noise Ratio (PSNR) of simulated output over the real codec output

(a) original image. (b) decoded image. (c) block-based intra prediction image. (d) estimated output from ResNet

FIGURE 5. An example of subjective quality of the simulation results.

TABLE 1. Performance of different network structure.

	Network structures			
	ResNet	*DenseNet*	*channel attention*	*ResNet+ quantization*
PSNR (dB)	36.30	36.44	36.25	36.29
Network size(kB)	1287	3491	1305	1287

as the measurement of simulation accuracy. The DenseNet structure can bring slight gain of simulation accuracy compare to other structures. This may stem from about 3-times larger scale of DenseNet. Of course there is almost no difference in their subjective quality. Considering that the codec-simulation network will be used for the subsequent joint optimization of other modules in the video coding system, network complex should be concerned. As a tradeoff between simulation accuracy and network complexity, ResNet is selected as the structure of the codec-simulation network.

B. SUPPLIMENTARY WITH SIDE INFORMATION

You may notice that the simulation accuracy in PSNR illustrated in Table 1 is about 36.x dB, which might not be high enough. An example of subjective quality of the simulation is shown in Fig. 5. It is obvious that the estimated output from the simulation ResNet is much smooth and lacking of angular texture details (Fig. 5(d)) than the actual decoded image from an HEVC intra codec (Fig. 5(b)). These angular texture details may reflect typical features of coding distortion, especially when coding bit rate is lower. Therefore, we need to find the way to enhance the codec-simulation network to better approximate coding results.

Since the codec-simulation network plays its role only in the joint optimization of pre- and post- processing networks in Fig. 3, apparently, it can use side information available from the real codec other than the original image. So that we can extend the codec-simulation network with side information and the gradient propagation procedure of equation (7) converts to

$$\frac{\partial Loss}{\partial PRE_\theta} = \frac{\partial Loss}{\partial Y} * \frac{\partial POST(M)}{\partial M} * \frac{\partial SIMU(L, SIDE)}{\partial L} * \frac{\partial PRE(X)}{\partial PRE_\theta} \tag{9}$$

accordingly, where $SIDE$ is the side information fed into the codec-simulation network.

Our goal is to make $SIMU(L, SIDE)$ approach $Codec(L)$ by introducing side information REF. In the meantime, it should be noticed that:

1) If the side information cannot bring sufficient supplementary information which means the simulator $SIMU(L, SIDE)$ cannot reflect the result of coding $Codec(L)$, and then the pre-processing cannot be ideally optimized with the existence of coding distortion and post-processing.
2) On the other hand, if the side information is too strong, e.g., using the expected output of the simulator, the decoded image, as side information, then the codec-simulation network can be degenerated to a function $SIMU(L, SIDE) = SIDE$ where the partial gradient $\frac{\partial SIMU(L,SIDE)}{\partial L}$ is always 0. The pre-processing network also cannot be optimized anymore.

Therefore, we need to find a reasonable balance in this contradiction.

In terms of using convolutional network to simulate video coding distortion, the difficulty is that video coding distortion is position-dependent, which means the distortion varies depending on its position in the picture. For example, the distortion at the border or in the center of a block could be different. What's more, blocks with different prediction modes also present different distortion even in the same relative position of the block. However, the convolutional network structure is translation invariant. For every pixel in the picture, when they are processed by codec-simulation network, the parameters of the network are fixed, resulting in the simulated distortion position-independent. As we can see from Fig. 5(d), it seems that every position pixel has been blurred to the same degree. In order to overcome this problem for the convolutional network, partition information and mode information are indispensable to send to the network as side information. By combining two information as one, prediction image is chosen as the side information of the codec-simulation network.

After deciding to use the prediction image as the side information of the network, we use the cascade method to

send the prediction image to the codec-simulation network. Cascade is a very common way to send side information into the network. As long as the network is well trained, the convolution kernels of different channel can adaptively perform different calculations on the inputs of different channel layers, so that the network can use input information and reference information reasonably and separately. The final structure of codec-simulation network is shown in Fig. 4 and the simulation results with prediction image as supplementary side information will be showed in Section IV-A.

C. JOINT OPTIMIZATION OF PRE- AND POSTPPROCESSING

Let's introduce in detail how to use the codec-simulation network to jointly optimize the pre- and post-processing networks in the coding system.

Firstly, pre- and post-processing are trained without the codec. A post-processing network is trained with a loss function of L2 norm between the output of the network and the ground truth. Then a pre-processing network followed by the post-processing network is trained with the following loss function[12],

$$loss_{pre} = \alpha l_2\big(post(pre(x)), x'\big) + l_2(pre(x), y), \quad (10)$$

where x and y represent for the input and ground truth output to and from the pre-processing networks, $pre(\cdot)$ and $post(\cdot)$ represent for pre- and post-processing, and x' represents for the ground truth output of post-processing network. α is a hyperparameter. For a coding system with pre- and post-processing before and after a codec, the ground truth output of the post-processing network x' is usually set to be the same as the input x of the coding system. The first part of the loss function is the reconstruction loss of pre- and post-processing, which is the l2 norm between the predicted result and the ground truth after pre- and post-processing. The second part of the loss function is measuring spatial structure maintenance of pre-processing, which is the l2 norm between the predicted output from the pre-processing network and the ground truth output of that network. The second part of the loss function is important for keeping major information in the input and avoiding serious information loss from the pre-processing.

It should be noticed that there are quantization operations after the pre- and post-processing to match with ability of traditional video coding of 8 bits or 10 bits fixed-point input and output. So, a differentiable approximation function of quantization is introduced in the training stage. Methods described in Section II-C can be used.

Secondly, we insert the codec-simulation network developed in part B of this Section between the pre-processing network and the post-processing network and retrain the pre-processing network again in the full pipeline of the coding system. The same loss function as equation (10) is used with the only exception that the pipeline of pre- and post-processing will be replaced by the full pipeline of the coding system,

$$loss_{pre} = \alpha l_2\big(post(sim(pre(x))), x'\big) + l_2(pre(x), y), \quad (11)$$

where $sim(\cdot)$ represents for the codec-simulation network.

It is worth noting that the input to the codec-simulation network includes the prediction image generated by encoding process. Theoretically, when the pre-processing network is continuously optimizing, the pre-processed image, i.e., the input fed into the codec is also changing. Therefore, the pre-processing network should be optimized iteratively by continuously updating the dataset of training.

As the matter of fact, a simplified procedure with fixed prediction image set generated from a pre-defined pre-processing method is used in our case. We take down-sampling and up-sampling as example of pre- and post-processing. And a bicubic down-sampling filter is used to generate pre-processed low-resolution images as input into codec and corresponding prediction images can be obtained accordingly as side information of the codec-simulation network. Such a set of original video and corresponding prediction images compose the training dataset for joint optimization of pre- and post-processing for a specific codec.

IV. EXPERIMENTS AND RESULTS

In this section, we will first show the accuracy of the codec-simulation network. Then using the down-sampling and up-sampling networks as the pre- and post-processing networks of the coding system, we compare the coding efficiency of the jointly optimized coding system and the non-jointly optimized coding system, thereby demonstrating the effectiveness of our codec-simulation network. Finally, we compared our scheme with anchor which is the codec without pre- and post- processing.

A. ACCURACY OF THE CODEC-SIMULATION NETWORK

In this experiment, we use x265 as the real codec. With all intra mode at very-slow level, we make four datasets at 4 CRF (Constant Rate Factor) points (CRF12, 17, 23, 28 respectively) and traine four corresponding codec-simulation networks. The training datasets is based on div2k, and the test dataset is composed of several video sequences on the https://media.xiph.org/video/derf/.

To evaluate the accuracy of the codec-simulation network, We calculate the PSNR of simulated output with output of the true codec and the results are shown in Table 2. PSNR is calculated as shown in formula (12).

$$PSNR = 10 * \log \frac{255^2}{\sum_{k\in Y,U,V} \sum_{i=0}^{w_k} \sum_{j=0}^{h_k} \left(x_{i,j,k} - y_{i,j,k}\right)^2} \quad (12)$$

where k represents for one of the YUV channel, w_k and h_k represent for the width and height of the image in channel k, and $x_{i,j,k}, y_{i,j,k}$ are corresponding pixel values of two compared images.

It can be seen from the Table 2 that as the CRF value increases, the objective quality becomes worse. However,

TABLE 2. The simulation results of the codec-simulation network.

Video sequence	PSNR at different rate points			
	CRF=28	*CRF=23*	*CRF=17*	*CRF=12*
aspen	44.83	46.25	47.73	48.44
Boat	40.65	41.78	43.21	44.23
controlled_burn	38.27	40.31	43.19	45.33
crowd_run	37.11	38.83	40.81	42.06
CSGO	50.85	53.45	57.45	59.58
dinner	52.77	54.57	56.23	57.05
DOTA2	44.18	45.80	47.99	49.87
ducks_take_off	38.94	39.99	41.12	42.23
factory	44.39	45.60	47.38	48.92
Fallout4	43.31	44.89	46.74	48.29
Hearthstone	42.66	44.27	46.19	48.02
life	44.57	45.26	46.40	48.66
old_town_cross	42.16	43.96	44.08	44.11
park_joy	38.68	40.19	42.20	43.68
pedestrian_area	47.83	49.12	48.66	48.60
red_kayak	42.72	44.38	46.59	47.91
rush_field_cuts	41.06	42.57	43.90	45.87
rush_hour	48.49	49.39	48.53	48.20
RUST	44.89	45.93	48.01	49.75
snow_mnt	38.09	40.50	43.71	46.01
speed_bag	48.42	48.85	48.79	48.35
station2	46.29	47.15	47.93	47.88
sunflower	45.43	46.85	48.73	48.84
Tango	47.34	47.27	46.57	46.06
tractor	41.41	42.77	44.55	46.48
west_wind_easy	43.76	45.46	47.73	49.33
WindAndNature	47.78	47.75	47.49	47.22
Animal	40.24	41.33	42.54	43.45
Face	48.01	48.14	47.51	47.63
average	**43.99**	**45.24**	**46.62**	**47.66**

despite the worst case, the average PSNR can reach more than 43dB, greatly higher than 36dB in the preliminary experiments in Section III-A. The subjective quality of the codec-simulation network is shown in Fig. 6. It can be found that the simulated coded image is highly similar to the actual coded image, which means the codec-simulation network is accurate.

TABLE 3. Performance of different down- and up-sampling method.

Down-sample	*Bicubic*	*Bicubic*	*CNN-CR[12]*	*ResNet-CR*
Up-sample	*Bicubic*	*RCAN[10]*	*CNN-SR[12]*	*ResNet-SR*
Set5	33.68	38.27	38.57	39.80
Set14	30.24	34.12	34.79	36.85
Div2k	32.45	36.63	-	40.46

B. EFFECTIVENESS OF THE CODECSIMULATION NETWORK

To further verify the effectiveness of the codec-simulation network, we take down-sampling and up-sampling as examples of the pre- and post-processing of the coding system.

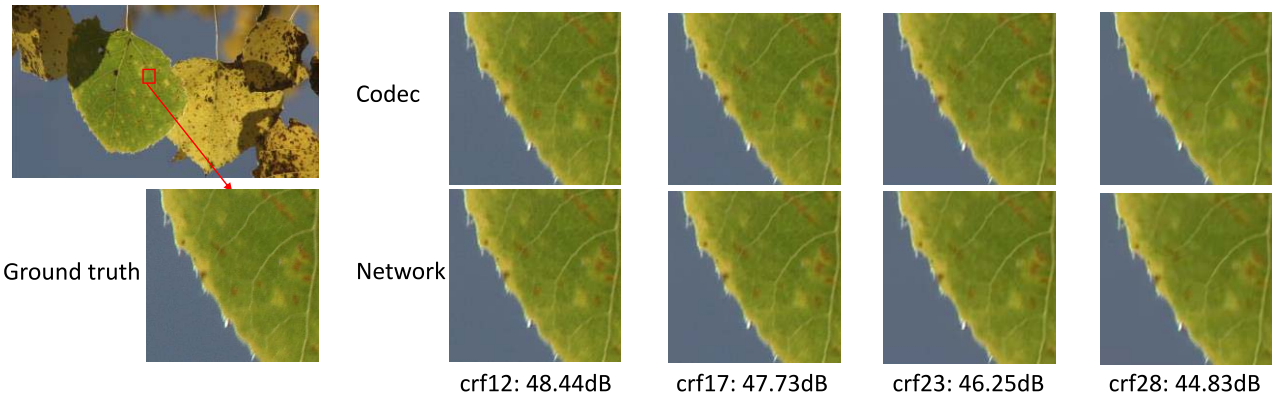
We designed down-sampling network ResNet-CR and up-sampling network ResNet-SR based on ResNet. ResNet-CR is composed of 5 residual blocks and a down-sampling module, and the up-sampling network ResNet-SR is composed of 16 residual blocks and an up-sampling module. Then we trained them on div2k dataset.

Image reconstruction performance of the combination of ResNet-CR and ResNet-SR are provided first. It is compared with bicubic+bicubic, bicubic+RCAN[10] and CNN-CR+CNN-SR methods. The performance of different down- and up- sampling methods are shown in Table 3. ResNet-CR and ResNet-SR obviously outperform the compared down- and up-sampling solutions.

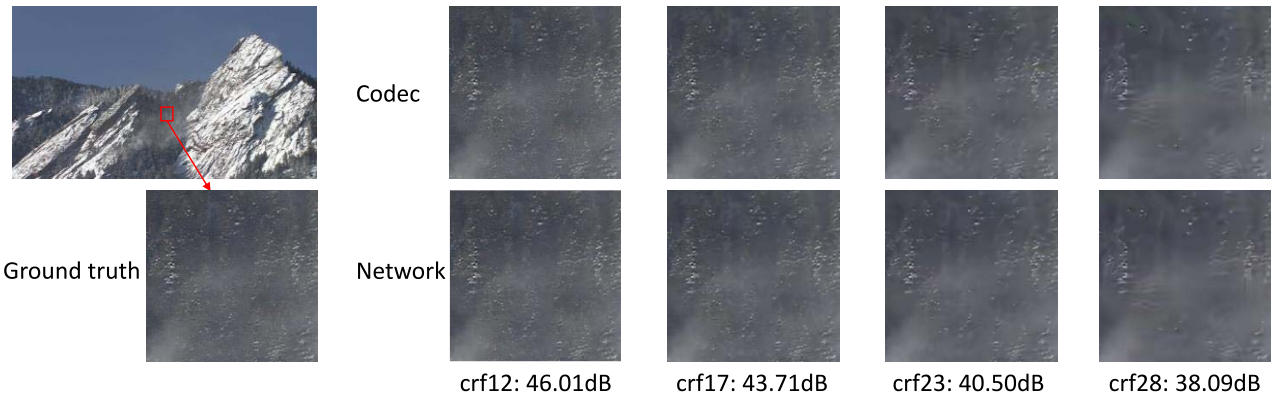
Next, let us provide the coding performance of the combination of ResNet-CR and ResNet-SR with x265 as the codec in between. X265 is set as an all intra codec with very slow mode working at 4 CRF points, 12, 17, 23 and 28. We name the coding system with jointly-optimized down- and up-sampling network as CSWJ and the coding system with non-jointly-optimized down- and up-sampling network as CSWNJ. The coding efficiency of CSWJ and CSWNJ is shown in Fig. 7. We present the RD performances of 10 specific video sequences. The blue curves are the performance of CSWJ while the yellow curves are that of CSWNJ. The average PSNR BD-rate gain of all 10 sequences is 46.88% and the average SSIM BD-rate gain is 55.15%.

Fig. 8 shows the subjective quality of output frames from CSWJ and CSWNJ. Results of the minimum CRF point 12 and the maximum CRF point 28 in our test are demonstrated. The bit rates of these two coding systems are similar at the same CRF points.

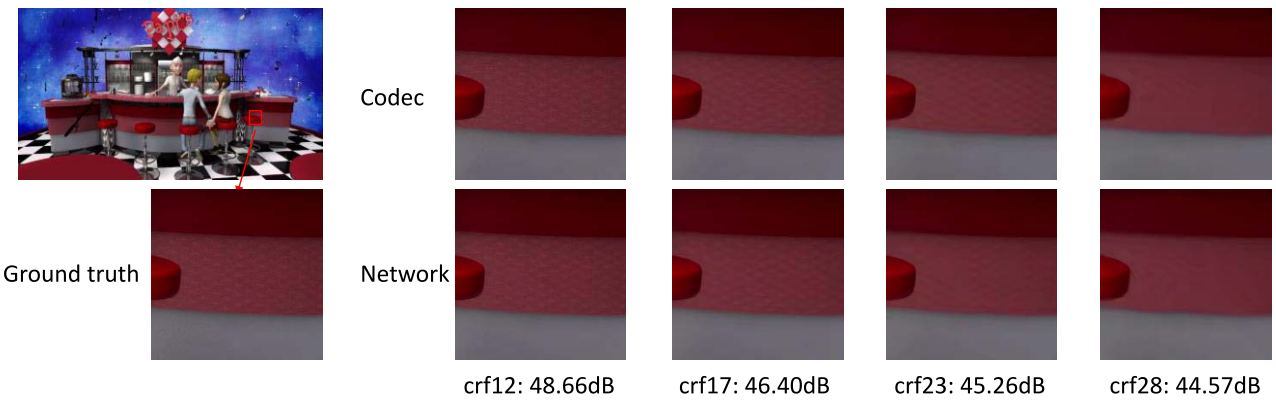
It can be observed that the reconstructed frames of CSWJ have more high frequency details than the ones of CSWNJ. In the case of high bit rates, the frames reconstructed by



(a) Video sequence: aspen



(b) Video sequence: snow_mnt



(c) Video sequence: life

FIGURE 6. **Simulation result (upper row: reconstructed image by codec at different rate points, lower row: output from codec-simulation network at different rate points.).**

CSWNJ have faked texture, which can be seen in Fig. 8 (b). After joint optimization, on the one hand, the down-sampling network can suppress the possible faked texture in the low-resolution image as much as possible. On the other hand, the up-sampling network can further remove the remaining faked texture. In the case of low bit rates, coding distortion causes excessive blurring of low-resolution image, resulting in that the up-sampling network cannot add details based on it. With the joint optimization, the down-sampling network can enhance the information, which is beneficial to up-sampling, in the low-resolution image as much as possible. Then the up-sampling network to restore more details for the reconstructed image based on the enhanced information. These effects cannot be achieved without joint optimization. Therefore, the reconstructed image of jointly optimized network has better subjective quality. It further proves that our codec-simulation network can fully improve the coding efficiency of the coding system.

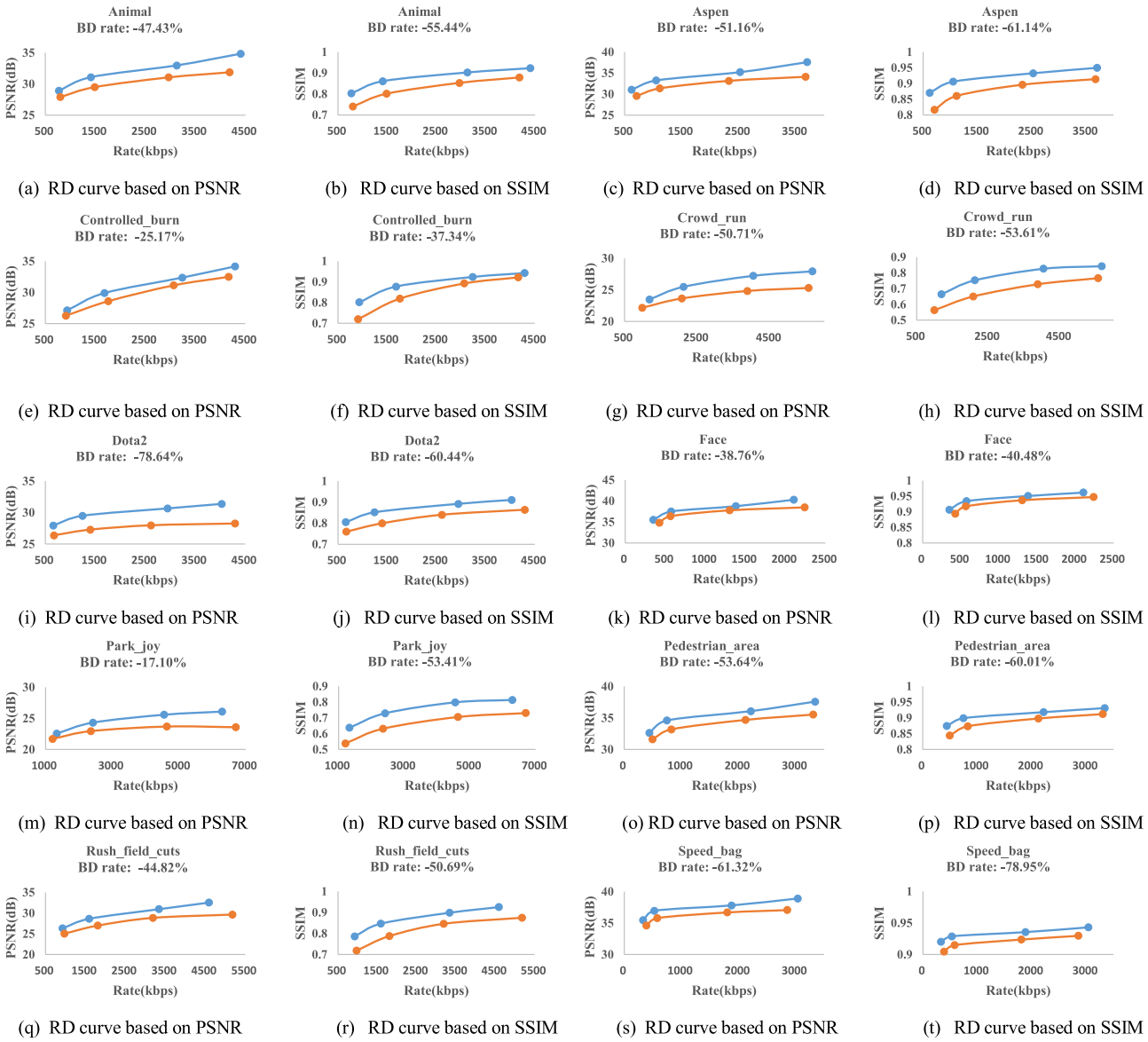


FIGURE 7. Coding efficiency of CSWJ and CSWNJ.

TABLE 4. Objective quality of CSWJ and anchor.

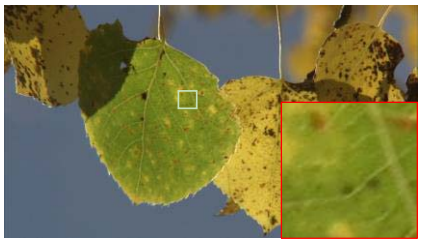
CRF of CSWJ	*12*		*17*		*23*		*28*	
metrics	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**
CSWJ	38.80	0.981	37.40	0.975	34.53	0.956	31.52	0.923
anchor	41.14	0.985	39.04	0.977	35.56	0.955	32.20	0.914

C. COMPARE WITH THE ACTUAL CODEC

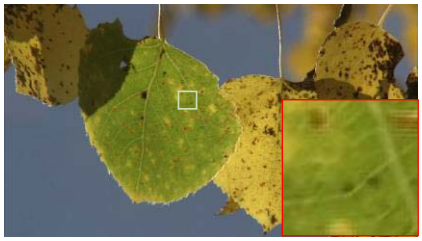
In this part, we will check whether CSWJ can outperform full resolution coding. For CSWJ, setting CRF points to 12, 17, 23, 28, we obtain the reconstructed videos. To ensure the bit rate alignment, we adjust the CRF point to encode and decode the video with original resolution to obtain the corresponding reconstructed video as the anchor. The fluctuation of the bit rate is controlled within 3%. The objective quality of CSWJ and anchor are presented in Table 4.

The PSNR of our scheme will be reduced by 1 to 2 dB on average compared to that of the anchor, which is reasonable. Down-sampling causes a lot of high frequency information to be lost. Though the up-sampling adds some detailed information, which may result in sufficient visual similarity, there is still huge differences at the signal level. As we can see in Table 3, even though no coding noise is introduced in, the upper bound of the down- and up-sampling can only achieve 40dB, while the reconstructed image and the original image can be hardly distinguished by people in fact. In order to further demonstrate the visual performance of our scheme, we did the following subjective experiments.

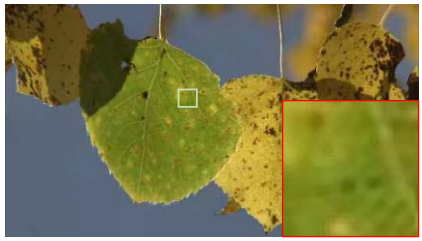
When testing subjective quality, the video reconstructed by CSWJ and anchor are played side by side in a blind test according to the standard in [33]. The manual score of each



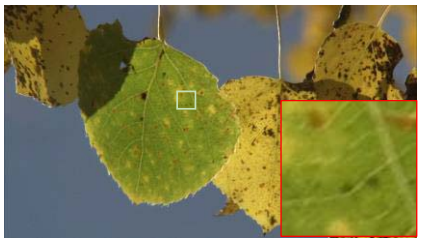
(a) ground truth



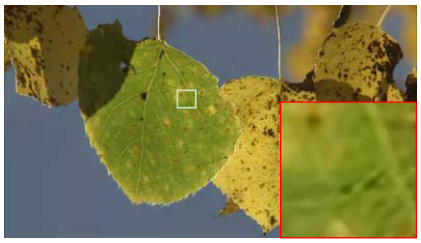
(b) image reconstructed by CSWNJ at CRF points 12



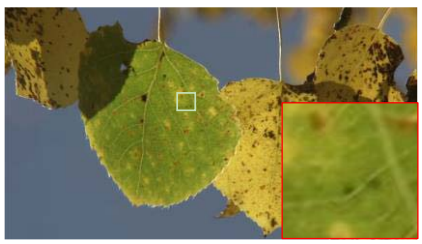
(c) image reconstructed by CSWNJ at CRF points 28



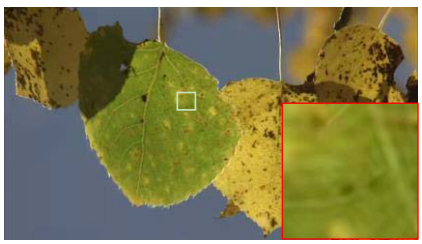
(d) image reconstructed by CSWJ at CRF points 12



(e) image reconstructed by CSWNJ at CRF points 28



(f) image reconstructed by anchor at CRF points 17



(g) image reconstructed by anchor at CRF points 33

FIGURE 8. Subjective quality of output from coding system.

video sample is 1, 0 or −1, which respectively represent that the reconstructed video of CSWJ has better subjective quality, similar subjective quality, or worse subjective quality compared with anchor. We record the mean of the scores and the final experimental results are shown in the Table 5. It can be seen that our scheme can achieve significant subjective quality improvement in large bitrate range for most of tested sequences.

TABLE 5. Subjective quality of CSWJ and anchor.

CRF of CSWJ / Video sequence	*12*	*17*	*23*	*28*
controlled_burn	0	0.2	0.6	0.6
crowd_run	-0.2	0.4	1	1
Animal	0	0	0	1
Face	0	0.6	0.6	1
DOTA2	-0.6	0.2	0.2	0.4
park_joy	0	0.4	1	0.6
pedestrian_area	0	0	0.8	0.6
rush_field_cuts	0.2	1	1	0.4
speed_bag	0	0	0.4	0.2
aspen	0.2	0.8	0.6	0.2
average	**-0.04**	**0.36**	**0.62**	**0.60**

V. CONCLUSION

Joint optimization of pre- and post-processing networks across the non-differentiable hybrid video codec is studied in this paper. The idea of developing a differentiable neural network to simulate coding results is novel. A ResNet-based convolutional neural network with input of side information is developed as a simulator of traditional hybrid video codec. Up to 47.6 dB objective simulation accuracy and sufficient good subjective simulation quality are reported for HEVC intra coding. Additionally, joint optimization of pre- and

post-processing across the differentiable codec-simulation is discussed. 46.88% and 55.15% BD-rate gain based on PSNR and SSIM have been demonstrated when we use down- and up-sampling as the pre- and post-processing in an HEVC coding system and jointly optimized. Compared with codecs without pre- and post-processing, our scheme achieves higher subjective quality under most condition.

Future work may include: simulation of the state-of-the-art video codec, such as VVC and AVS3 including inter coding, with similar network structure and more proper side information, optimization of pre- and post-processing methods other than down-sampling and up-sampling to improve efficiency of coding system, etc.

REFERENCES

[1] T. D. Tran, J. Liang, and C. Tu, "Lapped transform via time-domain pre- and post-filtering," *IEEE Trans. Signal Process.*, vol. 51, no. 6, pp. 1557–1571, Jun. 2003.

[2] G. Sun, U. Samarawickrama, J. Liang, C. Tian, C. Tu, and T. D. Tran, "Multiple description coding with prediction compensation," *IEEE Trans. Image Process.*, vol. 18, pp. 1037–1047, 2009.

[3] C. Tu, T. D. Tran, and J. Liang, "Error resilient pre/post-filtering for DCT-based block coding systems," *IEEE Trans. Image Process.*, vol. 15, pp. 30–39, 2006.

[4] H. H. Ryu, K. Y. Choi, and B. C. Song, "Prefiltering and postfiltering based on global motion compensation for improving coding efficiency in H.264 and HEVC codecs," *IEICE Trans. Inf. Syst.*, vol. 100, no. 1, pp. 160–165, 2017.

[5] C. Lan, J. Xu, and F. Wu, "Improving depth compression in HEVC by pre/post processing," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops*, Melbourne, VIC, Australia, 2012, pp. 611–616.

[6] S. Ferrara *et al.*, "MPEG-5 part 2: Low complexity enhancement video coding (LCEVC): Overview and performance evaluation," *Proc. Appl. Dig. Image Process. XLIII*, 2020, Art. no. 115101C.

[7] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016.

[8] B. Lim, S. Son, H. Kim, S. Nah and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Honolulu, HI, USA, 2017, pp. 1132–1140.

[9] C. Ledig *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, 2017, pp. 105–114.

[10] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proc. ECCV*, 2018, pp. 294–310.

[11] X. Wang *et al.*, "ESRGAN: Enhanced super-resolution generative adversarial networks," in *Proc. ECCV Workshops*, 2018, pp. 63–79.

[12] Y. Li, D. Liu, H. Li, L. Li, Z. Li, and F. Wu, "Learning a convolutional neural network for image compact-resolution," *IEEE Trans. Image Process.*, vol. 28, pp. 1092–1107, 2019.

[13] W. Sun and Z. Chen, "Learned image downscaling for upscaling using content adaptive resampler," *IEEE Trans. Image Process.*, vol. 29, pp. 4027–4040, 2020.

[14] H. Kim, M. Choi, B. Lim, and K. M. Lee, "Task-aware image downscaling," in *Proc. ECCV*, 2018, pp. 419–434.

[15] R. Shin and D. Song, "JPEG-resistant adversarial images," in *Proc. NIPS Workshop on Machine Learning and Computer Security*, 2017, pp. 1–6.

[16] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," 2017, *arXiv:1703.00395*.

[17] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," 2018, *arXiv:1802.01436*.

[18] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimization of nonlinear transform codes for perceptual quality," in *Proc. Pict. Coding Symp. (PCS)*, Nuremberg, Germany, 2016, pp. 1–5.

[19] J. Yang *et al.*, "Quantization networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, 2019, pp. 7300–7308.

[20] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, pp. 3142–3155, 2017.

[21] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, pp. 600–612, 2004.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770–778.

[23] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, 2017, pp. 2261–2269.

[24] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. ECCV*, 2018, pp. 3–19.

[25] T. Li, M. Xu, and X. Deng, "A deep convolutional neural network approach for complexity reduction on intra-mode HEVC," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Hong Kong, 2017, pp. 1255–1260.

[26] M. Xu, T. Li, Z. Wang, X. Deng, R. Yang, and Z. Guan, "Reducing complexity of HEVC: A deep learning approach," *IEEE Trans. Image Process.*, vol. 27, pp. 5044–5059, 2018.

[27] Z. Chen, J. Shi, and W. Li, "Learned fast HEVC intra coding," *IEEE Trans. Image Process.*, vol. 29, pp. 5431–5446, 2020.

[28] J. Pfaff *et al.*, "Data-driven intra-prediction modes in the development of the versatile video coding standard," *ITU J. ICT Discoveries*, vol. 3, no. 1, pp. 25–32, 2020.

[29] J. Mao and L. Yu, "Convolutional neural network based bi-prediction utilizing spatial and temporal information in video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 7, pp. 1856–1870, Jul. 2020.

[30] D. Li and L. Yu, "An in-loop filter based on low-complexity CNN using residuals in intra video coding," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Sapporo, Japan, 2019, pp. 1–5.

[31] W.-S. Park and M. Kim, "CNN-based in-loop filtering for coding efficiency improvement," in *Proc. IEEE 12th Image Video Multidimensional Signal Process. Workshop (IVMSP)*, Bordeaux, France, 2016, pp. 1–5.

[32] Y. Zhang, T. Shen, X. Ji, Y. Zhang, R. Xiong, and Q. Dai, "Residual highway convolutional neural networks for in-loop filtering in HEVC," *IEEE Trans. Image Process.*, vol. 27, pp. 3827–3841, 2018.

[33] "Methodology for the subjective assessment of quality of television pictures," Int. Telecommun. Union, Geneva, Switzerland, ITU-Recommendation BT 500-13, 2012.