Screen Content Encoding Network Based on Deep Contextual Information

Tianyu Gong[1], Tao Zhang[2,*], Ye Zhong[1], Mengmeng Zhang[3] and Huihui Bai[1]
[1]Institute of Information Science, Beijing Jiaotong University, Beijing 100044, China
[2]Beijing Polytechnic College, Beijing 100042, China
[3]The Faculty of Smart City, Beijing Union University, Beijing 102200, China
{24110103,21125295,hhbai}@bjtu.edu.cn, 2022001707@bgy.edu.cn, mengmeng@buu.edu.cn

***Abstract*—This paper proposes a deep contextual video coding network that integrates transformational jumps. The network takes a sequence of original in-screen content video frames as input and extracts motion features through motion estimation and a motion information encoder, with motion compensation occurring during decoding. It also extracts contextual information in the feature domain through a context encoder. This contextual information assists in both context encoding and decoding, as well as entropy encoding. By using contextual information as a condition for conditional encoding, the network transitions from predictive coding to conditional coding, aiding in the high-quality reconstruction of high-frequency content. The motion information encoder introduces a transformational jump branch into the analysis and synthesis process. This branch has the ability to extract coarse features and reconstruct them, thereby enhancing the encoding and decoding of visual signals. Finally, experiments confirm that this algorithm effectively improves the encoding performance of screen content videos.**

I. Introduction

In recent years, digital device advancements have made screen content images and videos central to communication. The growing data volume challenges image and video encoding technologies. Emerging needs in online education, virtual meetings, and cloud gaming highlight the urgency for efficient screen content encoding. This encoding aims to represent images and videos compactly with minimal visual quality loss, reducing storage and bandwidth use.

Camera-captured images often contain sensor noise and complex textures, whereas screen content images, typically computer-generated, are noise-free with discrete tones and fewer colors. They have finer lines, sharper edges, and more repetitive patterns. Traditional video encoding, designed for camera footage, uses a less effective hybrid prediction-transformation structure for screen content, risking detail loss and edge distortion. Current screen content encoding schemes, based on traditional methods, independently encode segmented blocks, ignoring overall content and causing high complexity. This paper contrasts video screen content encoding with traditional and deep video encoding algorithms H.264[1], H.265[2], DVC[3], Hu_ECCV[4], Lu_ECCV[5], FVC[6], DVCpro[7], EA_CVPR[8], and RLVC[9].

Existing end-to-end compression schemes, focused on natural images, overlook screen content image compression. The latter, being computer-generated, are noise-free with unique patterns and textures, affecting compression efficiency. Directly applying existing learning-based image compression approaches to screen content results in decreased rate and distortion performance. This paper aims to enhance the compression performance of screen content images. Inspired by deep contextual information and the concept of transform-skipping, we propose an end-to-end compression scheme for video screen content. Specifically, the proposed model thoroughly considers the characteristics of screen content, utilizing deep contextual learning to capture the high-frequency content in screen images, and applies deep contextual knowledge for the compression and reconstruction of video frames. Furthermore, a transform-skipping branch is integrated into the motion codec process. This branch, capable of coarse feature extraction and reconstruction, effectively extracts multidimensional, high-dimensional features from screen content images. As a result, visual signals can be more simply interpreted at the encoder end and restored at the decoder end.

II. Overall Network Structure

To enhance screen content compression, this paper introduces a deep video compression algorithm tailored for screen content's high texture contrast and sharp edges. Utilizing deep contextual information and transform-skipping, the algorithm's structure, depicted in Figure 1, comprises a context codec and a motion information codec.

The algorithm processes video screen content frame sequences. Initially, motion estimation extracts motion information v_t from input and reference frames. The motion information codec, using a convolutional neural network with transform-skipping, compresses this information. The transform-skipping branch, adept at coarse feature extraction and reconstruction, simplifies screen content perception in its two branches and finely processes it in the main branch. This allows for straightforward encoding and precise reconstruction at the decoder end into $\hat{v}_t$. The second part utilizes the reconstructed motion information $\hat{v}_t$ and high-dimensional features of the reconstructed reference frame $\hat{x}_{t-1}$, or $\tilde{x}_{t-1}$. These undergo warping and context mapping to produce deep contextual information $\overline{x}_t$. This context information then aids the context encoder and decoder in reconstructing high-frequency content, enhancing video quality. The context decoder outputs the reconstructed frame $\hat{x}_{t-1}$, with distortion

calculated against the input frame x_t. The bitrate, derived from the high-dimensional features m_t and y_t quantized and entropy coded into a bitstream, is calculated alongside distortion. These metrics form the rate-distortion loss function, crucial for network optimization during training.

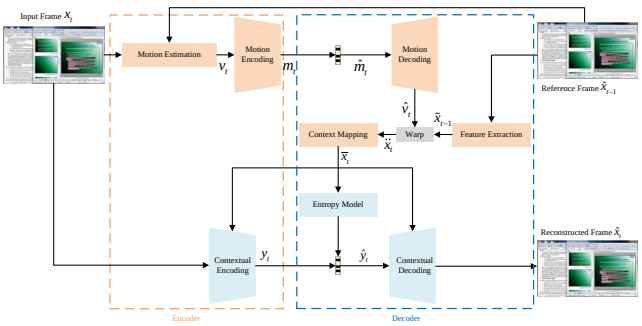

Fig. 1. Deep Context-Based Screen Content Coding Network.

The network framework of this paper can be represented as follows:

$$\hat{x}_t = f_{context}(\hat{x}_{t-1}) = f_{dec}(f_{enc}(x_t \mid \overline{x}_t) \mid \overline{x}_t) \otimes \overline{x}_t \quad (1)$$

In this representation, x_t denotes the input frame, $\hat{x}_t$ represents the predicted or reconstructed frame, $\overline{x}_t$ signifies the generated high-dimensional context, and $\hat{x}_{t-1}$ is the reference frame, i.e., the reconstructed previous frame. The function $f_{context}()$ is responsible for generating the context $\overline{x}_t$. $f_{enc}()$ and $f_{dec}()$ are the context encoder and decoder, distinct from the motion encoder and decoder. The deep contextual information is generated in a higher-dimensional feature domain, containing multidimensional, high-dimensional features. This provides a richer and more relevant set of conditions for encoding x_t, particularly for its high-frequency content.

III. Motion Encoding Network

A. Motion Encoding Network Based on Transform Skipping

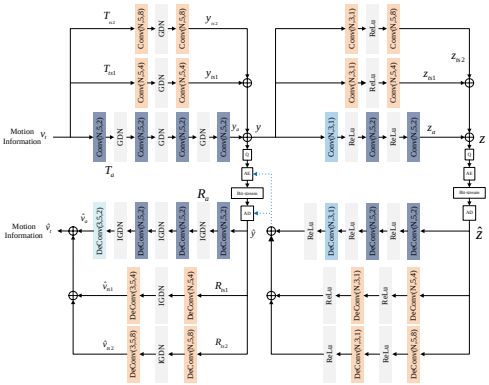

Fig. 2. Motion coding networks based on transform jumps.

This section details how integrating transform skipping into an end-to-end encoding framework can boost screen content image compression. Traditionally, transform skipping effectively handles sparse motion compensation residuals and intra prediction in screen content compression [10]. The motion encoding network's structure, based on transform skipping, is depicted in Figure 2. Our video screen content compression framework features an autoencoder-style encoder and decoder, with the multi-scale transform skipping encoder built upon the hyperprior model from Balle et al [11].

Figure 2 shows that the transform-skipping-based motion encoding network comprises convolutional, GDN, and ReLU layers. It features three branches in both the encoder and decoder, with most convolutional layers having a 5x5 kernel size and 128 channels (N=128). In the main branch, convolutional layers have a stride of 2, while the two side branches have strides of 4 and 8, respectively. This setup allows for local feature extraction while transform skipping captures larger-scale features, efficiently reducing spatial redundancy.

At the encoder, motion information v_t feeds into three scale analysis branches: a forward transform branch T_a and two transform skipping branches T_{ts1} and T_{ts2}. The forward transform branch T_a, resembling existing analysis transform structures, employs four sequential convolutional layers with a stride of 2. The transform skipping branches T_{ts1} and T_{ts2} use two convolutional layers each, with strides of 4 and 8, and a 5x5 kernel size with N channels. These branches collectively process the visual signal at different scales, extracting high-dimensional features across various scales. The T_a, T_{ts1}, and T_{ts2} branches generate the latent codes y_a, y_{ts1} and y_{ts2}, as follows:

$$\begin{aligned} y_a &= T_a(v_t, \theta_a) \\ y_{ts1} &= T_{ts1}(v_t, \theta_{ts1}) \\ y_{ts2} &= T_{ts2}(v_t, \theta_{ts2}) \end{aligned} \quad (2)$$

In this context, θ_a, θ_{ts1} and θ_{ts2} represent the model parameters of the main transform branch and the two transform skipping branches, respectively. The high-dimensional feature y is a combination of y_a, y_{ts1} and y_{ts2} as shown below.

$$y = y_a + y_{ts1} + y_{ts2} \quad (3)$$

Quantization is applied to the noise representation of the high-dimensional features, where $\widetilde{y}$ denotes the high-dimensional feature, and $\hat{y}$ is encoded into the bitstream as an entropy value.

$$\hat{y} = Q(\widetilde{y}) \quad (4)$$

This paper assumes that the distribution of the high-dimensional feature $\widetilde{y}$ follows a zero-mean Gaussian distribution. The motion encoding network based on transform skipping predicts the scale of each distribution conditioned on the hyperprior z. In this section, two transform skipping branches are designed for the encoding and decoding processes of motion encoding. During the encoding process, the main branch and the two transform skipping branches separately

generate z_a, z_{ts1} and z_{ts2}. The hyperprior feature represented by z can be expressed as follows:

$$z = z_a + z_{ts1} + z_{ts2} \tag{5}$$

After quantization, z is compressed using the entropy model [12]. At the decoder end, it is input into the hyperprior decoder. The main reconstruction branch and the two transform skipping reconstruction branches in the hyperprior decoder yield the standard deviation $\hat{\sigma}$ of the Gaussian distribution. Then, $\hat{\sigma}$ is sent to the arithmetic encoder and decoder to generate the probability distribution of $\hat{y}$ for encoding and decoding.

Similarly, reconstructing the motion information $\hat{v}_t$ from the high-dimensional feature $\hat{y}$ involves three branches. The main reconstruction path and the two transform skipping reconstruction paths are described as follows:

$$\begin{aligned} \hat{v}_a &= R_a(\hat{y}, \delta_a) \\ \hat{v}_{ts1} &= R_{ts1}(\hat{y}, \delta_{ts1}) \\ \hat{v}_{ts2} &= R_{ts2}(\hat{y}, \delta_{ts2}) \end{aligned} \tag{6}$$

Here, R_a, $\hat{v}_{ts1}$ and $\hat{v}_{ts2}$ respectively represent the main reconstruction branch and the two transform skipping branches in the decoder. δ_a, δ_{ts1}, and δ_{ts2} respectively denote the network parameters of the main reconstruction branch and the two transform skipping branches in the decoder. Finally, the outputs of the main reconstruction branch and the two transform skipping branches are summed to obtain the reconstructed motion information $\hat{v}_t$, as expressed in Equation 7:

$$\hat{v}_t = \hat{v}_a + \hat{v}_{ts1} + \hat{v}_{ts2} \tag{7}$$

Finally, the quantized high-dimensional feature $\hat{y}$ and the hyperprior feature $\hat{z}$, along with their formed bitstream, are used by the entropy model to calculate the bitrate. This is used in the training process's loss function to optimize the overall video screen content encoding network.

B. Feature Extraction and Contextual Refinement Networks

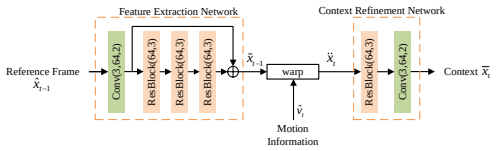

Fig. 3. Feature Extraction and Contextual Refinement Network.

This paper introduces a feature extraction network to generate contextual information in the feature domain. It consists of a convolutional layer with 64 channels, a 3x3 kernel size, and a stride of 2, followed by three 64-channel residual blocks with a 3x3 kernel size. The reference frame $\hat{x}_{t-1}$ inputs into this network, and through these layers, high-dimensional features $\widetilde{x}_{t-1}$ are obtained, shifting reference coordinates from the pixel to the feature domain. A $warp$ function, using motion information $\hat{v}_t$, distorts $\widetilde{x}_{t-1}$ to produce the preliminary context $\ddot{x}_t$. However, this initial context is coarse, as the warping may cause spatial misalignment. To enhance the preliminary context $\ddot{x}_t$, a context refinement network is employed. It comprises a 64-channel residual block with a 3x3 kernel size and a convolutional layer with 64 channels, a 3x3 kernel size, and a stride of 2. This network refines the initial context, addressing spatial discontinuities from warping, resulting in the final deep contextual information $\overline{x}_t$. Figure 3 illustrates the specific structures of both the feature extraction and context refinement networks. The process of feature extraction and context generation is summarized as a context generation function, detailed in Equation 8.

$$f_{context}(\hat{x}_{t-1}) = f_{cr}(warp(f_{fe}(\hat{x}_{t-1}), \hat{v}_t)) \tag{8}$$

$f_{context()}$ uses motion information $\hat{v}_t$ to guide context extraction and expand the learning area for the context. This generation process is akin to motion compensation in the feature domain. Without feature extraction, applying motion compensation directly in the pixel domain would mean that the learned context remains tied to the pixel domain. This could limit the relevance between the context's corresponding positions and the input frame x_t, reducing its efficacy in compressing x_t. Therefore, the feature extraction module is crucial. Deep contextual information in the feature domain holds extensive information, significantly enhancing the compression of high-frequency content. Furthermore, feature-domain context, as opposed to pixel-domain motion compensation, allows the network to utilize a larger reference area for extracting useful features.

IV. Context Coding Network

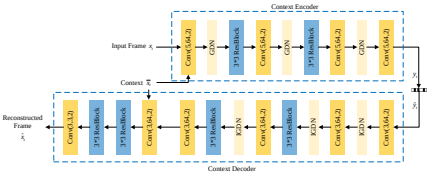

Fig. 4. Context coding network.

The context encoding network is mainly composed of convolutional layers, GDN layers, and residual blocks. In the context encoder, the key components are 64-channel convolutional layers with a 5x5 kernel size and a stride of 2. Conversely, the context decoder primarily utilizes 64-channel convolutional layers with a 3x3 kernel size and a stride of 2, as detailed in Figure 4. The context encoder's input includes the current frame x_t and context $\overline{x}_t$, which it encodes into 96-channel, 16x downsampled high-dimensional latent features y_t. The context decoder begins by upsampling these latent features $\hat{y}_t$ to the original resolution, then concatenates them with the context $\overline{x}_t$ to produce the final reconstructed frame $\hat{x}_t$.

V. Entropy Model

According to [13], the cross-entropy between the estimated probability distribution and the actual distribution of the latent codes represents a tight lower bound of the actual bitrate, which can be expressed as:

$$R(\hat{y}_t) \geq E_{\hat{y}_t \sim q_{\hat{y}_t}}[-log_2 p_{\hat{y}_t}(\hat{y}_t)] \tag{9}$$

In this paper, $p_{\hat{y}_t}(\hat{y}_t)$ and $q_{\hat{y}_t}(\hat{y}_t)$ represent the estimated and actual probability density functions of the quantized $\hat{y}_t$, respectively. Arithmetic coding is employed to encode latent codes at a bitrate closely matching the cross-entropy, resulting in a negligible discrepancy between the actual bitrate $R(\hat{y}_t)$ and the cross-entropy bitrate. The primary goal is to develop an entropy model that precisely estimates the probability distribution $p_{\hat{y}_t}(\hat{y}_t)$.

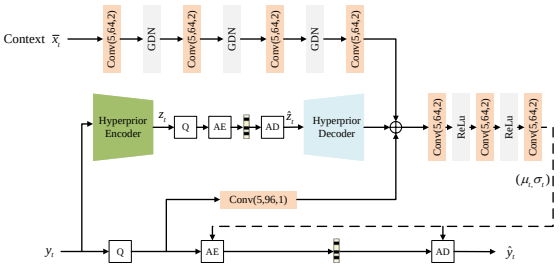

Fig. 5. Structure diagram of entropy model.

Figure 5 illustrates the entropy model's structure. Initially, a hyperprior model [11] learns hierarchical priors within y_t. Next, an autoregressive network with a $mask$ convolutional layer learns spatial priors in y_t's quantized bitstream. While hierarchical and spatial priors are standard in deep image compression, video features also exhibit temporal correlations, necessitating the removal of temporal redundancy. To address this, a temporal prior encoder analyzes the temporal correlations in $\overline{x}_t$, generating temporal priors. Subsequently, a prior fusion network combines these three priors: hierarchical, spatial, and temporal. It estimates the mean and variance of $\hat{y}_t$'s distribution. These estimates assist the arithmetic encoder and decoder in efficient entropy encoding and decoding of y_t. This paper assumes the mean and variance of $p_{\hat{y}_t}(\hat{y}_t)$ follow a Laplacian distribution. The expression for $p_{\hat{y}_t}(\hat{y}_t)$ is:

$$\begin{aligned} p_{\hat{y}_t}(\hat{y}_t \mid \overline{x}_z, \hat{z}_t) &= \prod_i (L(\mu_{t,i}, \sigma_{t,i}^2) * U(-\frac{1}{2}, \frac{1}{2}))(\hat{y}_{t,i}) \\ \mu_{t,i}, \sigma_{t,i} &= f_{pf}(f_{hpd}(\hat{z}_t), f_{ar}(\hat{y}_{t,<i}), f_{tpe}(\overline{x}_t)) \end{aligned} \tag{10}$$

In Equation 10, the index i represents the spatial position. $f_{hpd}()$ is the function form of the hyperprior decoding network, $f_{ar}()$ is the function form of the autoregressive network, $f_{tpe}()$ represents the function of the temporal prior encoder network used for learning the temporal correlations in the contextual information, and $f_{pf}()$ is the function form of the prior fusion network. $f_{ar}(\hat{y}_{t,<i})$ and $f_{tpe}(\overline{x}_t)$ respectively learn the spatial and temporal priors, while $f_{hpd}(\hat{z}_t)$ learns the supplementary information that cannot be derived from spatial or temporal correlations.

VI. Experimental Results and Analysis

A. Comparison and analysis of experimental results

To validate our proposed algorithm, we tested it on three videos from the HEVC Class F dataset, focusing on key foreground targets and overall content. We compared its performance against advanced video compression algorithms, including H.264[1], H.265[2], DVC[3], Hu_ECCV[4], Lu_ECCV[5], FVC[6], DVCpro[7], EA_CVPR[8], and RLVC[9], using PSNR and MS-SSIM metrics. Despite the limited availability of end-to-end video screen content encoding algorithms, we included notable natural image video encoding algorithms in our comparison.

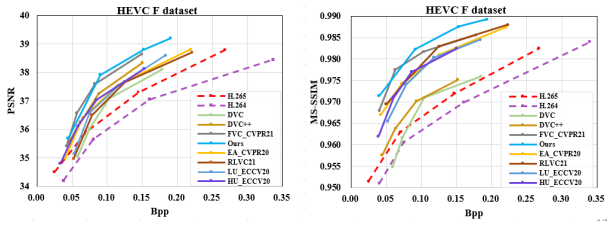

Fig. 6. Algorithm test results.

Our results, illustrated in Figure 6, show the proposed algorithm's superiority on the HEVC Class F dataset. The left side of the figure compares PSNR-bpp rate-distortion curves, while the right side focuses on MS-SSIM-bpp rate-distortion curves. The proposed algorithm outperforms others in both PSNR and MS-SSIM at higher bpp values, especially in MS-SSIM. It maintains this advantage even at lower bpp values, indicating its overall superior performance. Compared to natural image video encoding, our algorithm's PSNR and MS-SSIM values are higher, likely due to screen content characteristics like absence of noise, high contrast, sharp edges, and repetitive areas, which reduce spatial redundancy and complexity, enhancing performance in screen content encoding.

B. Ablation Experiment

To validate the effectiveness of the subnetworks in the deep video compression algorithm proposed in this paper, experimental tests were conducted on the HEVC Class F dataset, focusing on the motion codec network, feature extraction network, and context refinement network. The results of these experiments were then compared with the overall performance of the deep video compression algorithm proposed in this paper, as shown in Figure 7.

Figure 7 displays, on the left, the PSNR-bpp rate-distortion curve comparisons between the proposed algorithm and several ablation algorithms. On the right, it presents the MS-SSIM-bpp rate-distortion curve comparisons. The curve labeled "Ours" represents the test results of the video compression algorithm proposed in this paper.

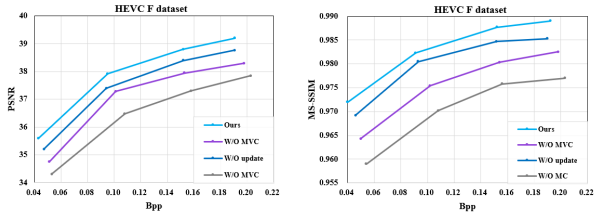

Fig. 7. Comparison of ablation results

The curve "W/O CR," shows results without the context refinement network. Using distorted high-dimensional features directly for motion information, it forgoes context refinement, leading to spatial discontinuities. This impacts the context codec network and entropy model, reducing PSNR by about 0.4dB and MS-SSIM by 0.002. The curve "W/O FE," represents results lacking the feature extraction network. It directly employs motion information to distort the reference frame, leaving context in the pixel domain with limited information depth. This approach decreases PSNR by roughly 0.9dB and MS-SSIM by 0.007. The curve "W/O MVC," indicates results without the motion codec network. Foregoing our CNN-based motion codec network for direct motion encoding, this method fails to adequately reduce spatiotemporal redundancy, decreasing PSNR by approximately 1.5dB and MS-SSIM by 0.013. We also tested the efficacy of transform skipping branches in our motion encoding network on the HEVC Class F dataset. These tests, shown in Figure 8, compared configurations with only the main branch, one transform skipping branch, and the two original transform skipping branches from our method.

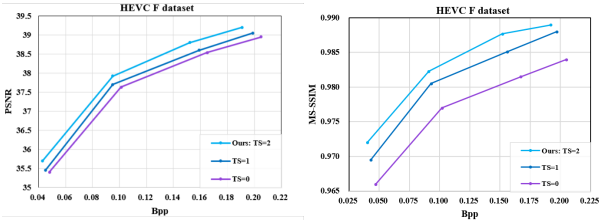

Fig. 8. Transform skip branch verification results

Figure 8 displays rate-distortion curve comparisons: PSNR-bpp on the left and MS-SSIM-bpp on the right. The "TS=1" curve shows results with one transform skipping branch, specifically the one with a convolution stride of 4. This setup slightly reduces performance, with an approximate 0.1dB decrease in PSNR and 0.002 in MS-SSIM. The minor impact suggests that the remaining branch effectively learns larger-scale features. Additionally, the algorithm's multi-layer convolutional networks contribute to learning larger-scale features. The "TS=0" curve, indicating results without transform skipping branches and relying solely on the main branch, shows a more significant reduction in performance: about 0.4dB in PSNR and 0.005 in MS-SSIM. This underlines the importance of transform skipping branches in enhancing screen content encoding.

VII. Conclusion

This paper focuses on the underexplored area of screen content images, distinct from natural video images, by developing a deep contextual screen content encoding network. It specifically addresses screen content characteristics like noiseless patterns, high repetitiveness, high texture contrast, and sharp edges. The network includes a motion encoding network featuring transform skipping, highly effective in compressing screen content motion information. Initially, the paper outlines the network's overall structure. It then describes the functions and principles of its various modules: motion encoding network, feature extraction network, context refinement network, context encoding network, and entropy model. Experimental tests on the HEVC Class F dataset and comparative analyses with other methods demonstrate the proposed algorithm's superior performance in objective evaluation metrics. Ablation studies further affirm the significance of each sub-module, validating the innovation and effectiveness of this approach.

References

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

[2] 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, 2012.

[3] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, "Dvc: An end-to-end deep video compression framework," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 006–11 015.

[4] Z. Hu, Z. Chen, D. Xu, G. Lu, W. Ouyang, and S. Gu, "Improving deep video compression by resolution-adaptive flow coding," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, Springer, 2020, pp. 193–209.

[5] G. Lu, C. Cai, X. Zhang, *et al.*, "Content adaptive and error propagation aware deep video compression," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, Springer, 2020, pp. 456–472.

[6] Z. Hu, G. Lu, and D. Xu, "Fvc: A new framework towards deep video compression in feature space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1502–1511.

[7] G. Lu, X. Zhang, W. Ouyang, L. Chen, Z. Gao, and D. Xu, "An end-to-end learning framework for video compression," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 10, pp. 3292–3308, 2020.

[8] E. Agustsson, D. Minnen, N. Johnston, J. Balle, S. J. Hwang, and G. Toderici, "Scale-space flow for end-to-end optimized video compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8503–8512.

[9] R. Yang, F. Mentzer, L. Van Gool, and R. Timofte, "Learning for video compression with recurrent auto-encoder and recurrent probability model," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 2, pp. 388–401, 2020.

[10] M. Mrak and J.-Z. Xu, "Improving screen content coding in hevc by transform skipping," in *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, IEEE, 2012, pp. 1209–1213.

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

[12] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *arXiv preprint arXiv:1611.01704*, 2016.

[13] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE mobile computing and communications review*, vol. 5, no. 1, pp. 3–55, 2001.