Optimized Video Compression with Residual Split Attention and Swin-Block Artifact Contraction[⋆]

Afsana Ahsan Jeny[a], Md Baharul Islam[a,b,*]

[a] *Department of Computer Engineering, Bahcesehir University, Istanbul, Turkey*
[b] *College of Data Science & Engineering, American University of Malta, Bormla, Malta*

Abstract

Research in video compression has seen significant advancement in the last several years. However, the existing learning-based algorithms continue to be plagued by erroneous motion compression and ineffective motion compensation architectures, resulting in compression errors with a lower rate-distortion trade-off. To overcome these challenges, we present an end-to-end video compression method through a set of primary operations (e.g., motion estimation, motion compression, motion compensation, residual compression, and artifact contraction) differently. A deep residual attention split (DRAS) block is introduced for motion compression networks to pay more attention to certain image regions to create more effective features for the decoder while boosting the rate-distortion optimization (RDO) efficiency. A channel residual block (CRB) is proposed in motion compensation to yield a more accurate predicted frame, potentially improving the residual frame. Due to mitigating the compression errors, an artifact contraction module (ACM) by residual swin convolution UNet block is included in this model to improve the reconstruction quality. A buffer is added to fine-tune the previous reference frames to improve the final frame. These modules combine with a loss function by assessing the trade-off and enhancing the decoded video quality. A comprehensive ablation study demonstrates the effectiveness of the proposed blocks and modules for video compression. Experimental results show the competitive

[⋆]This work is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under the 2232 Outstanding Researchers program, Project No. 118C301.
[*]Corresponding author
Email address: bislam.eng@gmail.com (Md Baharul Islam)

performance of the proposed method on four benchmark datasets.

Keywords: Video compression, residual attention, image reconstruction, channel residual block, artifact contraction

1. Introduction

Video compression is crucial for offering high-quality video services under transmission networks and storage limitations. For instance, it is around 93 megabytes (MB) of raw video content (1080p with 30 Hz) for a second using the YUV 420 format. Maintaining these raw materials with such high data rates is challenging in real-time. According to the report [1], video content accounts for more than 80% of internet traffic, which may rise even more in the future. Therefore, developing an effective video compression system capable of producing higher-quality frames within a specific bandwidth limit is fundamental. Furthermore, video compression approaches are also beneficial for action identification [2] and model compression [3].

Many well-known video compression methods, including H.264 [4], H.265 [5], and VVC [6], have been proposed in the last few decades. These methods rely on handcrafted modules such as block motion estimation and Discrete Cosine Transform (DCT) to decrease the high level of redundancy (spatial or temporal) in video sequences. However, undesirable redartifacts, most notably blocking, ringing, and blurring artifacts, are inevitably produced near the boundaries during block-wise operation [7]. This rippling phenomenon adversely impacts video quality degradation and the user experience. Besides, these methods cannot optimize in an end-to-end manner. Therefore, it is essential to increase compression performance, which needs further investigation. Recently, deep neural network (DNN) based image compression algorithms [8, 9, 10, 11] received significant research attention for two reasons: (i) they enable the usage of non-linear transforms, and (ii) they do not require handcrafting features. These studies have established a theoretical foundation for deep autoencoders in image codecs to improve the rate-distortion trade-off (e.g., low MSE or high SSIM). It focuses on developing end-to-end optimized frameworks and simultaneously improving all modules (e.g., transformation, quantization, entropy estimation).

The use of image compression directly to the video domain may result in difficulties with temporal inconsistency. However, there are three main challenges to developing an end-to-end video compression system. Firstly, adding all the modules (e.g., motion estimation, motion compression, motion compensation, residual compression, and entropy coding) is challenging to optimize the whole system. Some works [12, 13, 14, 15, 16, 17, 18], however, substitute one or two modules in the conventional architecture rather than improving the entire compression system in an end-to-end manner. Secondly, the performance of video compression algorithms depends on generating highly accurate motion vector (MV) information. Optical flow-based methods may compute motion information. However, motion-compensated algorithms may raise extra bits that affect the reconstructed frame owing to inaccurate optical flow estimation. Thirdly, the rate-distortion optimization strategy is also required for the learning-based compression to reduce the temporal redundancy in video sequences.

Recent deep learning-based video compression methods, such as [19, 20, 21, 22, 23, 24, 25] concentrate on different modules to enhance its overall performance. For example, to improve the system's performance and visual quality of the reconstructed frame, Lin et al. [19] utilized multi-reference frames and MV fields for a more accurate current frame and two refinement networks to remove the MV and residual errors. Wu et al. [20] introduced a post-processing step in the baseline [26] and Feng et al. [21] focused on enhancing the residual frame for implementing residual autoencoders. Yang et al. [27] proposed a hierarchical technique and a recurrent enhancement module to improve the overall performance. In contrast, Li et al. [24] and Sheng et al. [25] gave priority to improving the residual prediction through the temporal contextual encoder-decoder network. However, no one explores the motion vector compression network, which is the primary factor of the video compression system. The earlier methods employed the variational autoencoder style network [9], where a new latent vector from the encoder is sent to the decoder at each time interval. In this case, the code vector must include all of the input image features in the decoder, which burdens CNN's memory. Though [23, 21] executed residual blocks to acquire additional deep features, unfortunately, this is inadequate to reduce the computational load. As a result, the capacity of the model was limited. Therefore, a low-cost motion compression network is obvious for learning-based video compression. Besides, it is also significant to emphasize the other modules (e.g., motion compensation) and reduce compression errors (caused by the quantization process in the MV and residual encoder-decoder network). Motivated by the issues above, we propose an end-to-end video compression network with deep residual split attention and a swin-block artifact contraction module capable of mitigating the new arrival issues. Our technical contributions are summarized below.

- This paper introduces an end-to-end video compression method including motion estimation, motion compression with deep residual attention split (DRAS) block, motion compensation with channel residual block (CRB), artifact contraction module (ACM), residual compression, and entropy coding employed based on the rate-distortion trade-off with a single loss function. As a result, our proposed method may readily include new state-of-the-art methodologies for optical flow estimations, image compression, motion and residual frame prediction, and rate control.
- An autoencoder-style network with DRAS blocks is proposed for motion vector compression that assigns better attention to certain image regions to create more relevant features for the decoder and reduce the computational cost with superior rate-distortion optimization (RDO) performance.
- A CRB is applied in a motion compensation network to generate an accurately predicted frame by capturing abstract features and boosting the system's efficiency. It also contributes to improving the residual image for the overall visual quality improvement of the video frames.
- A new module ACM is included before the decoded frame buffer using the proposed residual swin convolution UNet (RSwCU) block to enhance the reconstructed frame while significantly boosting the RDO's performance.
- We performed an extensive ablation study and experiments on four datasets (i.e., HEVC, UVG, VTL, and MCL-JCV), showing competitive performance. Besides, it outperforms the first baseline method, DVC, and a recent RLVC, FVC, and DCVC in terms of PSNR and MS-SSIM at lower bit rates.

The remaining paper is structured as below. The earlier works are briefly discussed in the section 2. Section 3 discussed the proposed video compression method, including all modules and blocks. Detailed information about the datasets and experimental settings is presented in section 4. The experimental results, analysis, ablation study, and comprehensive discussion are carried out in section 5. Finally, we conclude the work with the future research direction.

2. Related Works

This section briefly discusses the state-of-the-art image and video compression methods.

2.1. Image Compression Methods

Over the last few decades, JPEG [28], JPEG 2000 [29], and BPG [30] are just a few of the numerous image compression systems. Conventional codecs have the drawback of individually designed modules of image compression works [28, 29, 30], which might lead to poor compression results. Deep learning-based image compression algorithms have recently made significant advances [31, 9, 32, 33, 34]. These approaches may learn a non-linear transform from data and calculate the probabilities necessary for entropy coding in an end-to-end way. For example, to improve image quality, the autoencoders based on Long Short-Term Memory (LSTM) are utilized in [33, 34, 32] to gradually encode the difference between the original image and the reconstructed image without considering the number of bits utilized for compression. Besides, the convolutional neural network (CNN) based autoencoders utilized in [31, 9, 8] for image compression. For example, a non-linear activation function called generalized divisive normalization (GDN) was added to a CNN-based autoencoder and used to compute the distribution of latent representations [31]. However, the autoencoder of this network is simple and does not consider the input-adaptive entropy model. Later, Balle et al. [9] presented an input-adaptive entropy model with a zero-mean Gaussian distribution to characterize each latent representation. Patel et al. [35] conducted a human investigation on perceptual similarity for various image compression algorithms. Blau et al.

[36] then presented the rate-distortion trade-off by including the perception component in the optimization approach. The above-mentioned learning-based image compression algorithms surpass traditional image codecs. However, they reduce the amount of unnecessary spatial information while ignoring the temporal correlation. We would suggest the readers to go through the recent survey on image and video compression [37, 38] for more details.

2.2. Video Compression Methods

Traditional video codecs such as H.264, H.265, and VVC are designed on a predictive coding architecture that requires multiple independent modules' integration. There is a limited possibility of improving the whole system simultaneously. Thus, DNN-based methods are used to increase the performance of classic codecs, including entropy coding [15], mode determination [14], intra-prediction, and residual coding [13]. They only boost the performance of adjacent blocks, requiring the development of different end-to-end compression techniques. The model's border block artifacts caused the poor performance. Additionally, motion information is conveyed via block motion estimation, which is inefficient. [4, 5, 6, 7, 39].

Recently, deep learning-based approaches have been utilized for creating an end-to-end video compression [40, 26, 41, 42, 19, 43, 44, 45, 20, 23]. For example, Kin et al. [40] proposed an overfitted bidirectional recurrent convolutional neural network (RCNN). Due to the low feature channel, RCNNs may learn temporal information from successive frames without exact motion information for the final reconstructed frame. As a result, they can not attain state-of-the-art performance and reduction rates. Following this, Lu et al. [26] introduced the first end-to-end optimal video compression system to substitute all of the essential modules of the H.264/H.265; in particular, to obtain the MV, they employed a pixel-level optical flow network [46], and improved the rate-distortion trade-off. Then, Djelouah et al. [41] presented a network that integrates motion compression with image synthesis using an interpolation-based video compression technique. Accurate motion estimation has been implemented to achieve decent compression performance. The state-of-the-art optical flow estimation networks [47, 46, 48, 49, 50] have been used to accomplish these networks.

On the other hand, Habibian et al. [42] suggested a 3D autoencoder method that does not rely on optical flow to compensate for motion. Their technique is limited to identifying fine-scale movements, and employing an auto-regressive probability model is time-consuming. To further reduce redundancy, Lin et al.[19] employed multiple frames in various modules, utilizing the prior image compression auto-encoder network and pixel-level network to minimize redundancy further. Even for motion compensation and residual compression, they employed the same architecture as in [26]. Their training technique for many frames was quite tricky, yet they could increase the RDO performance. All these methods [26, 41, 42, 19] estimate pixel-level optical flow maps and compress the residual information by using an image compression autoencoder network. Producing reliable pixel-level flow maps for motion compression via an autoencoder-style network for generating the predicted frame and residual frame may be difficult, reducing video compression performance.

Several researchers [43, 44, 45] utilized deep learning and classical methods in an end-to-end manner to alter certain portions for video compression. For example, in [45], the RaFC block compresses the flow maps like the traditional block-based methods. It determines the correct flow map resolution for each local block of motion feature instead of the pixel-wise optical flow map. Lu et al. [44] focused on improving the reconstruction error of the video compression system by changing autoencoder structures. However, the lost encoder features might be necessary for the predicted frame. Later, Agustsson et al. [43] replaced the bilinear warping operation with a scale-space flow algorithm that learns to blur the reference content adaptively for improved warping outcomes. These algorithms [43, 44, 45] are tailored to specific prediction modes, resulting in a lack of flexibility throughout the whole system and producing compression artifact.

Afterward, Wu et al. [20] proposed video compression using a P-frame compression architecture that can be learned end-to-end. They claimed that their MV-Residual prediction network could simultaneously predict the motion vectors and residual information. However, they did not test their technique on common benchmark datasets (i.e., UVG, HEVC) to prove performance as the Rectified Linear Unit (ReLU) (not deep rectifiers) optimizes rate-distortion trade-off. Feng et al. [21] investigated residual frames instead of motion vector frames or predicted frames in their method. However, they only performed their model on an image compression dataset (CLIC). Yang et al. [22] proposed a hierarchical video compression method with a quality enhancement module at the decoder to improve the RD performance and reconstructed frame. The group-of-picture (GOP) structure was established by the Hierarchical Learning Video Compression (HLVC). It has shown excellent efficiency since H.264 scalable coding was launched. In addition, it uses a weighted recurrent quality improvement at the decoder end. The recurrent auto-encoder and the recurrent probability model were used in the recurrent learning video compression (RLVC) method described in [27], which resulted in better MV and residual compression. Their model [22, 27], however, just raised the computing cost, and the RDO performance was not spectacular. On the other hand, their bit savings rate seems relatively high.

Most recently, Hu et al. [23] proposed a video coding network by performing some significant operations (i.e., motion estimation, compression, and compensation) in the feature space to improve the computing cost (i.e., parameters and time complexity). The deformable compensation modules are used instead of optical flow to obtain accurate motion information. Because of it, the feature-level prediction can not adjust to video content and manage a variety of motion contexts. Moreover, the multi-frame feature fusion module utilizes the nonlocal attention mechanism to combine the reconstructed frames. As a result, by using deformable convolution and nonlocal attention mechanisms in every fusion, their suggested video compression model demonstrated the computational complexity in terms of parameters (26M). Yet, their coding time provided remarkable performance overall. Furthermore, to improve the video compression performance in a better way, the authors proposed a contextual encoder and decoder with multilevel [25] and single-level [24] temporal contexts are used to generate and refine the reconstructed frame. However, while their coding time performance is satisfactory, they have not placed sufficient emphasis on enhancing the performance of the motion vector, which is a crucial factor in increasing RDO performance. In addition, it impacts the predicted and the residual frame. In contrast, our network accomplishes motion compression and compensation module in a new way through the deep residual attention mechanisms and channel residual block, respectively, to raise

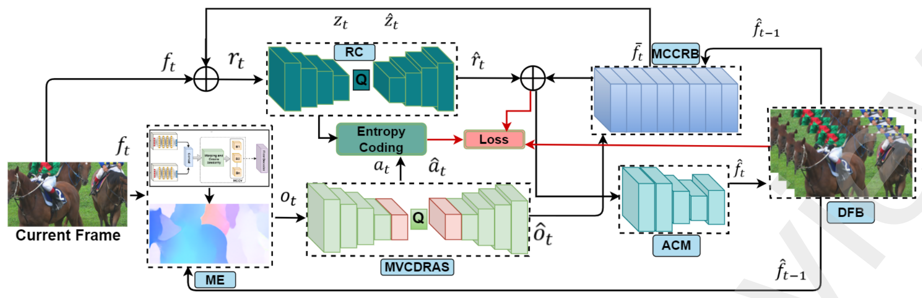

Figure 1: An overview of the proposed video compression architecture. To generate the raw optical flow, o_t, we first send f_t along with $\hat{f}_{t-1}$ to the motion estimation (ME) module. Then it is sent to the motion vector compression network with a deep residual attention split (MVCDRAS) to compress o_t and receive $\hat{o}_t$. Next, $\hat{o}_t$ is passed through the proposed motion compensation with the channel residual block (MCCRB) network to obtain $\bar{f}_t$. After that, f_t and $\bar{f}_t$ are concatenated and obtained as r_t. Additionally, r_t is fed into the residual compression (RC) network, which obtains $\hat{r}_t$. Due to artifact from the quantization process, the concatenated frame, $C(\hat{f}_t, \hat{r}_t)$, is fed into the proposed artifact contraction module (ACM) to get $\hat{f}_t$. A decoded frame buffer (DFB) is employed as an online buffer to create a clear reconstructed frame.

the rate-distortion trade-off for the entire compression system. Furthermore, to refine the decoded or reconstructed frame artifact, we propose a new module ACM using the residual swin convolution UNet block that significantly impacts the final frame to enhance the reconstruction quality.

3. Proposed Method

Fig. 1 depicts the proposed optimized video compression architecture, which includes motion estimation (ME), MV compression (MVC), motion compensation (MC), and residual compression (RC) network. However, the MVC and MC modules [26] are significantly improved through the motion vector compression network with the deep residual attention split (MVCDRAS) block and motion compensation with the channel residual block (MCCRB) to optimize the efficiency of the video compression system. Furthermore, we proposed the artifact contraction module (ACM) that makes the reconstruction frame more pleasing. These components are tuned together and utilize a single rate-distortion loss, as shown via experimental findings and extensive ablation studies. These modules are separately discussed below.

Notations. Let F $= \{f_1, f_2, \dots f_{t-1}, f_t\}$, these symbols represent the original video sequences. When the time step t occurs, f_t, $\bar{f}_t$, and $\hat{f}_t$ represents the original, predicted, and reconstructed/decoded frame, respectively. The difference between the predicted frame $\bar{f}_t$ and the original frame f_t is represented by r_t (the residual frame). The final reconstructed/decoded [1] residual frame is denoted by $\hat{r}_t$. This work uses the reconstructed and decoded words in the same context. Motion information obtained from pixel-wise optical flow estimation is essential for eliminating temporal redundancy. Therefore, motion vector/information (MV) or optical flow value is represented by o_t. The final reconstructed/decoded values of the MV fields at time step t are represented by $\hat{o}_t$. Since an autoencoder illustrates transformation, consequently, the residual frame r_t and the MV frame o_t are transferred to z_t and a_t. After the quantization in MV and residual autoencoder, $\hat{z}_t$ and $\hat{a}_t$ denotes the quantized forms of z_t and a_t, respectively.

3.1. Motion Estimation

Video compression performance significantly depends on motion estimation. Optical flow finds the temporal correlation between video sequences. Several recent learning-based optical flow estimation algorithms [47, 46, 48, 49, 50] have been proposed. These methods can compute the pixel-level motion information precisely. However, more bits are needed to compress motion information for conventional video compression methods because of the higher data capacity. The recent deep recurrent feature pyramid-based network, namely DeepPyNet [51] is used to find optical flow. It includes a pyramid-based feature extractor (not handcrafted), multi-channel cost volume, and flow decoder. In this method, the previous reference frame $\hat{f}_{t-1}$ and current frame f_t are taken as inputs and produce the motion information o_t, which is compressed later through the autoencoder network. This motion estimation network is correctly optimized to reduce the rate-distortion trade-off. Adapting the DeepPyNet [51] optical flow map with our video compression approach makes it much more efficient.

Fig. 2 shows the optical flow estimation maps of two RaceHorse video sequences (sequence 82 in (a) and 83 in (b)) from the HEVC Class C dataset [5] with and without

[1]The reconstructed and decoded words are used in the same context.

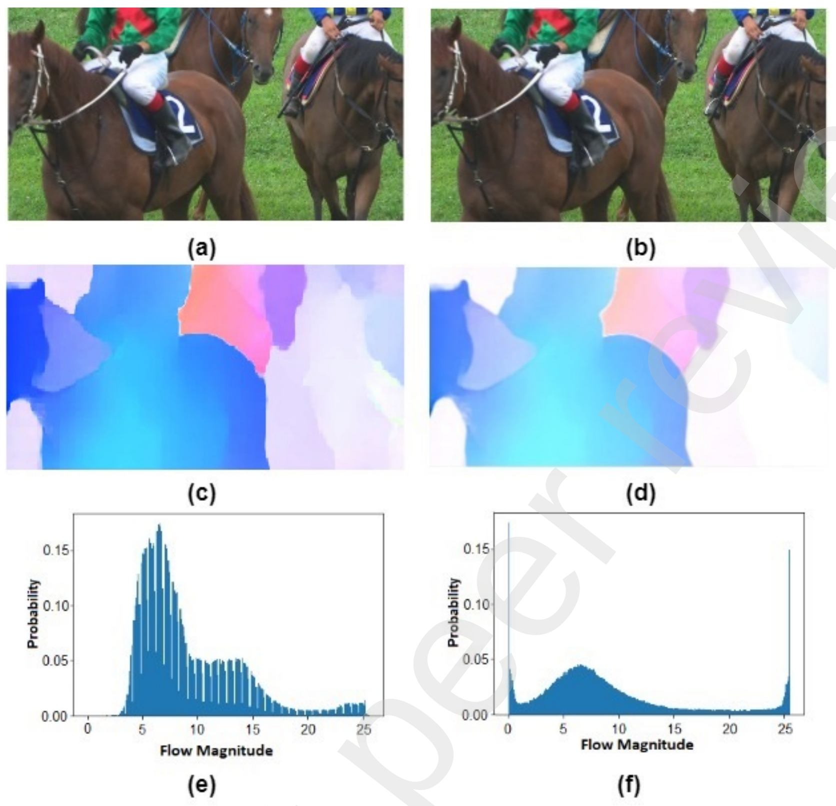

Figure 2: Optical flow with statistical analysis. (a) and (b) represent two consecutive frames from the HEVC Class C dataset, (c) and (e) represent the raw optical flow and its flow magnitude, (d) and (f) optical flow in the video compression system and its flow magnitude.

joint training. The associated probability distributions of the optical flow magnitudes are shown in (e) and (f), respectively. The reconstructed optical flow map (d) obtained by cooperative training of the video compression model includes more pixels with zero value than the original optical flow map (c) counterpart. Thus, the reconstructed optical flow map needs fewer bits for encoding. For example, encoding the optical flow map in Fig. 2(c) requires 0.043 Bpp, whereas encoding the optical flow map in Fig. 2(d) requires just 0.031 Bpp, saving around 27 percent of the bits.

3.2. MV Compression with Deep Residual Attention Split

The key contributor to improving RDO performance is optical flow compression.

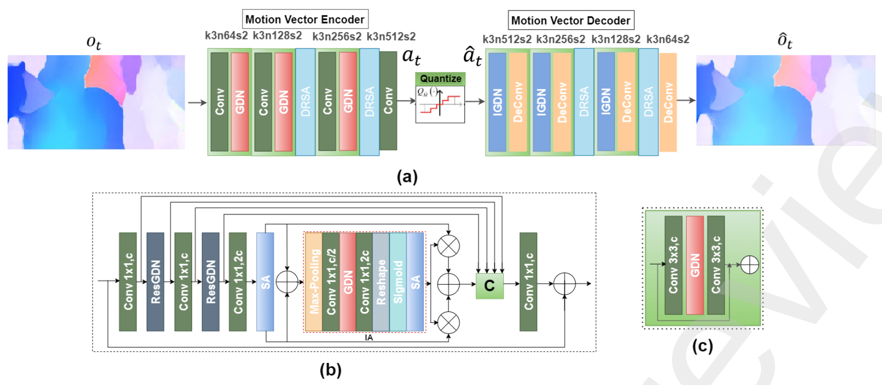

Figure 3: (a) The proposed autoencoder-style network with deep residual split attention (DRSA) block to compress the optical flow. The $k3n64s2$ denotes the convolution layer with the kernel size of 3×3, output feature channel of 64, and stride 2. (b) the proposed DRSA block where ResGDN, SA, and IA define the residual GDN, split attention, and inner attention blocks, respectively. (c) the ResGDN block. For more information, please refer to Section III (B).

To compress it, the earlier works [19, 20, 21, 22, 24, 25] employed the variational autoencoder network [9] for compressing the motion vector. It takes significant time to capture abstract features for the decoder. Although residual blocks are utilized by [23, 21] to acquire more deep features, this was insufficient to lessen the computational burden. However, attention mechanisms [52] with residual blocks [53] may perform better since an attention mechanism is used to assign greater attention to certain image regions to create more relevant features for the decoder and reduce the computational load. Besides, a model with computable linear-nonlinear transformations shows superior efficiency in end-to-end optimization architecture.

Inspired by this, we propose an autoencoder-style network with deep residual attention and split (DRAS) block to compress the optical flow, as shown in Fig. 3 (a). It consists of four DRAS blocks (2 each encoder and decoder), four convolution and deconvolution layers, followed by the Generalized Divisive Normalization (GDN) or inverse GDN, except for the last layer. The convolution and deconvolution layer sizes are considered by the three parameters, k $\times$ n $\times$ s that represent the size of the kernel, filter maps, and the stride, respectively. At first, the motion representation a_t is generated by the MV encoder, which is then quantized via uniform scalar quantization to $\hat{a}_t$. After that $\hat{a}_t$ is sent to the MV decoder, which rebuilds at $\hat{o}_t$. The decoder's architecture is just the reverse of the encoder's.

DRAS Block. A novel part of our autoencoder network is the DRAS block, as shown in Fig. 3 (b). Due to the high computational cost, typical non-local attention [54] is ineffective for this task. Our proposed DRAS block utilizes both split attention (SA) [55] and residual GDN (ResGDN) blocks (in Fig. 3 (c)) to achieve greater efficiency. Here, the SA can capture cross-channel relationships in the features for better representations of the optical flow-based motion vector. On the other hand, the paradigm of identity shortcut connection in the ResNet [56] is added to certain GDN layers, represented by the ResGDN, to enable deeper learning of the network. At first, the input features go through convolution and ResGDN layers and are fed into the SA layer for splitting the features. After that, these features are concatenated and passed to the inner attention (IA) layer (inside the red dotted line in Fig. 3 (b)). This IA layer consists of a Max-Pooling layer, two convolution layers with a 1×1 kernel, two types of rectifiers (i.e., GDN and Sigmoid), and one SA block. The output of SA is concatenated with the element-wise production features of the previous SA. Before concatenating the last single 1×1 convolution layer and input residual connection, the last features from SA layers (the initial SA and SA from IA) are concatenated with some residual skip connections. These residual connections with the IA layer increase the channel dimensions and allow the prediction of accurate motion vectors, which can minimize the bit rate and distortion (please refer to section V (ablation study)).

3.3. Motion Compensation with Channel Residual Block

The predicted frame $\bar{f}_t$ is obtained by the motion compensation (MC) network, which is supposed to be similar to f_t. In the MC module, $\hat{f}_{t-1}$ is warped to f_t depending on $\hat{o}_t$ as Eq. 1.

$$\hat{f}_{t-1}^{w} = \text{Warp}\left(\hat{f}_{t-1}, \hat{o}_t\right) \tag{1}$$

where, $\hat{f}_{t-1}^{w}$ represents the warped frame and $Warp$ is the backward warp operation [57]. However, there are still artifacts in $\hat{f}_{t-1}^{w}$. Therefore, $\hat{f}_{t-1}^{w}$ and $\hat{f}_{t-1}$ are concatenated as the input to eliminate the artifact and passed to a CNN model in MC.

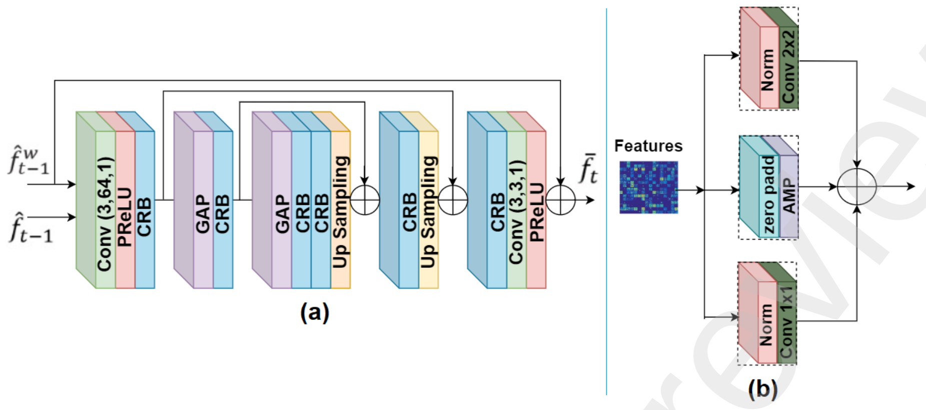

Figure 4: a) The proposed architecture for the motion compensation module, where CRB, GAP, and PReLU define the channel residual block, global average pooling, and parametric ReLU, respectively; (b) indicates the proposed channel residual block (CRB), where Norm and AMP represent normalization and average max-pooling.

Some works [19, 21, 45, 44] utilized a CNN model from the first baseline paper [26] that used a modified UNet [58] for artifact reduction. However, we observe that the generated predicted frame delivers blurry results because of ignoring the intermediate layers in UNet. Therefore, learning may slow down and put the network at risk of skipping the layers that capture abstract features. Later researchers [23, 27, 22] employed multiple reference frames for generating predicted frames and faced time complexity issues. Motivated by these, we propose a CNN model with a channel residual block (CRB) in the MC module, as illustrated in Fig. 4 (a). The MCCRB aims to increase the system's effectiveness and improve compression efficiency while simultaneously reducing the number of parameters. In particular, by including a CRB in feature maps, additional channels are assigned to the features in the lower layers of the network, allowing for the equivalent computational cost. Our proposed MCCRB consists of two convolution layers, two activation functions (PReLU) and global average pooling layers (GAPs), two up-sampling layers, and six CRB blocks. Some residual connections are added to the network, with the final activation after the summation.

Channel Residual Block (CRB). Our special CRB block is depicted in Fig. 4 (b). Three sub-blocks are added to CRB to accept the feature maps separately. The first sub-block contains a normalization layer, followed by a convolution layer with a kernel size of 2×2 and a fixed feature map dimension (64, 128) that executes linear transformation on the input feature map to retain the channel size. An average max-pooling layer and a zero-padding layer are employed in the second sub-block to improve feature map dimensions and shorten the sequence length of feature maps. In the third sub-block, a layer of normalization using a convolution (kernel size of 1×1, 32 filters, and stride of 2 followed by a PReLU) enhances the feature map dimension while decreasing the sequence length of feature maps. To better visualize the predicted frame and boost the RDO performance, we concatenate the outputs of all three sub-blocks using residual connections and their updated feature maps. Moreover, this predicted frame is also able to boost the performance of the residual frame in the system.

3.4. *Residual Compression Network*

The r_t is derived between f_t and $\bar{f}_t$. Then it is encoded and decoded by the image compression network [20]. It is significant to map r_t to a smaller domain to compress data with high efficiency using the residual encoder. Besides, the residual decoder is employed to rebuild $\hat{r}_t$. Furthermore, on the basis of some convolution layers and a GDN/IGDN-based autoencoder type network, the r_t compresses [9]. Similar to the motion compression network, r_t is converted to z_t after the residual encoder. Through the quantization procedure, it is transferred to $\hat{z}_t$, and after the residual decoder, it is turned into $\hat{r}_t$.

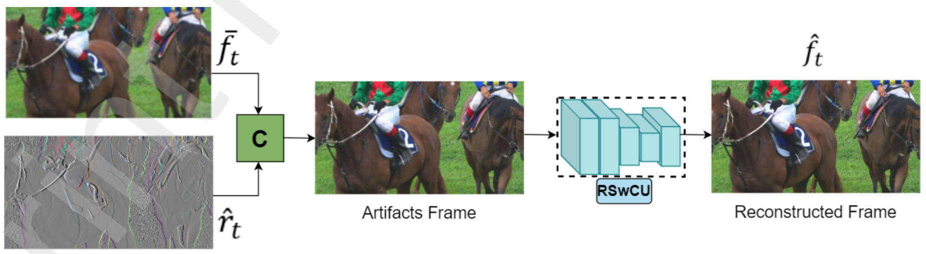

Figure 5: Our artifact contraction module (ACM). The RSwCU represents the proposed residual swin convolution UNet block.

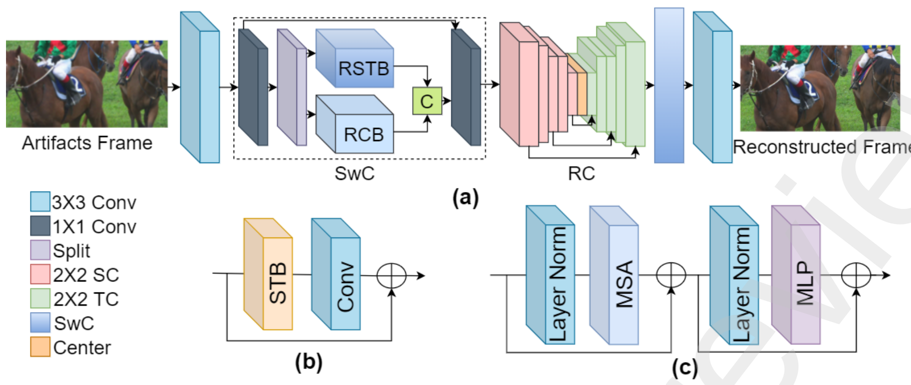

Figure 6: The proposed residual swin convolution UNet (RSwCU) block for ACM. (a) denotes the RSwCU block, (b) residual swin transfer block (RSTB), and (c) swin transfer block (STB) [59]. The terms RCB, SwC, C, RC, MSA, and MLP refer to the residual convolution block, swin convolution block, concatenation, residual connection, multi-head self-attention, and multi-layer perception, respectively. The notations ”SC” and ”TC” refer to a 2×2 strided and transposed convolution with stride 2.

3.5. Artifact Contraction Module

We discovered that there are still compression artifacts in $\hat{f}_t$ and $\hat{r}_t$ owing to quantization in the encoder-decoder network, which resulted in a low-quality reconstructed frame. Aside from that, it is also a key contributor to the enhancement of RDO performance. Therefore, we propose a compression artifact contraction architecture capable of improving the reconstructed frame while increasing the RDO performance. Fig. 5 depicts the proposed artifact Contraction Module (ACM) module. At first, $\hat{f}_t$ and $\hat{r}_t$ are concatenated and produce $C(\hat{f}_t, \hat{r}_t)$ which has compression artifact. After that $C(\hat{f}_t, \hat{r}_t)$ is fed into our proposed artifact contraction architecture to produce $\hat{f}_t$. Below, the proposed artifact contraction architecture is explained in detail.

Residual-Swin-Convolution-UNet. Fig. 6 (a) depicts the proposed artifact contraction architecture, referred to as Residual-Swin-Convolution-UNet (RSwCU). The RSwCU is primarily composed of two 3×3 convolution layers, two swin convolution blocks (SwCs), and one UNet [58] block. There are four levels of the UNet block in our proposed RSwCU. Each level consists of a residual link between 2×2 strided convolution (SC) (down-sampling) and transposed convolution (TC) (up-sampling). There are 16, 32, 64, and 128 channels at each layer from the first to fourth levels.

In Fig. 6 (a), a SwC block connects the residual swin transformer block (RSTB) (see Fig. 6) (b) and the residual convolutional block (RCB) [60] through two 1×1 convolutions, split-concatenation processes, and a residual connection. A 1×1 convolution is first applied to an input feature tensor F of an artifact image. Then, it is divided into two equal feature map groups: F_1 and F_2 by split layer. We define such a procedure as follows.

$$F_1, F_2 = Split(Conv1 \times 1(F)). \tag{2}$$

Then, F_1 and F_2 are passed into an RSTB (consists of a convolution layer and swin transformer block [59], see Fig. 6 (b, c)) and RCB, respectively, resulting in the appearance of

$$M_1, M_2 = RSwinTransformer(F_1), RConv(F_2). \tag{3}$$

Subsequently, M_1 and M_2 are concatenated and utilized as the input of a 1×1 convolution with a residual link to the original input, F. As a result, the SwC block's ultimate output is defined by

$$N = Conv1 \times 1(Concat(M_1, M_2)) + F. \tag{4}$$

It is worth noting that our suggested RSwCU has several advantages over artifact images. First and foremost, the SwC block combines the regional capabilities of the RCB with the non-regional capabilities of the RSTB. Here, the residual connection of RCB and RSTB establishes an identity-based link in the SwC block, enabling the aggregation of features at multiple levels. Second, the variational UNet enhances RSwCU's ability to represent local and non-local phenomena of artifact images. Third, to decrease the computation burden (e.g., parameters), the split and concatenation procedures are employed as group convolution with 2 groups. Please refer to the ablation study to see the effectiveness of the proposed RSwCU.

3.6. Loss Function and Entropy Coding

Loss Function. Our goal is to use the smallest feasible number of bits for video encoding while reducing distortion between f_t and $\hat{f}_t$. For training, we use the following rate-distortion function.

$$\alpha \mathrm{D} + \mathrm{R} = \alpha \mathrm{d}\left(f_t, \hat{f}_t\right) + \left(\mathrm{N}\left(\hat{a}_t\right) + \mathrm{N}\left(\hat{z}_t\right)\right) \tag{5}$$

where the distortion between f_t and $\hat{f}_t$ is denoted by $\mathrm{d}\left(f_t, \hat{f}_t\right)$, and the mean square error (MSE) or multi-scale structure similarity (MS-SSIM) is utilized to measure this distortion. $\mathrm{N}(.)$ denotes the number of bits utilized for encoding. The bitstreams encode both $\hat{a}_t$ and $\hat{z}_t$ representations, and a trade-off between bit numbers and distortion is specified by α, the Lagrange multiplier. The quantization step is needed prior to entropy coding, which needs a_t and z_t. As long as the quantization process is not differentiated, end-to-end training is infeasible. To tackle this issue, we employ the approach in [2] during training where the quantization procedure is substituted by the uniform noise, i.e, $\hat{a}_t = a_t + U$, and $\hat{z}_t = z_t + U$. We directly use the rounding technique in the predicted step, i.e, $\hat{a}_t = round(a_t)$, and $\hat{z}_t = round(z_t)$, respectively.

Entropy Coding. A bit rate estimation network is needed to improve the network by considering both distortion and bit numbers to determine the latent representations. The entropy of the appropriate latent representation is the most precise way to estimate the bit rate. Therefore, we employ the hyperprior entropy model in [9] to precisely estimate the bit rate $(\mathrm{N}\left(\hat{a}_t\right)$ and $\mathrm{N}\left(\hat{z}_t\right))$ of $\hat{a}_t$ and $\hat{z}_t$. In order to ensure that the decoding procedures are compatible with parallelization, we do not employ the autoregressive entropy model [10].

3.7. Decoded Frame Buffer

It is significant to use $\hat{f}_{t-1}$ in the ME and MCCRB networks during f_t compression, as illustrated in Fig. 1. Thus, the encoding operation establishes a chain of dependencies. We have implemented an automated update technique, a decoded frame buffer (DFB), to tackle this issue and make the training process more efficient. More specifically, each iteration of the training stores $\hat{f}_t$ in a buffer. When encoding f_{t+1}, the buffer containing $\hat{f}_t$ is utilized for the ME and MCCRB networks. As a result, every training sample in the DFB will be modified once for each epoch. We can optimize and save the previous frame at each iteration, which allows us to complete the process faster and provide precise reference frames.

4. Dataset and Experiments

4.1. Dataset

In training, we utilize the Vimeo-90K dataset [61] comprises 89,800 video clips, each with seven frames at a size of 448×256 pixels. The video sequences are randomly cropped to 256×256 before training. We used the following four datasets to test our method's compression performance.

- **HEVC Test Sequences [5]:** It is the most common test sequence for assessing video compression performance. Our experiment uses Class B (1920 × 1080), Class C (832 × 480), and Class D (352 × 288) datasets.
- **MCL-JCV Dataset [62]:** It consists of 24 video clips with a 1920 × 1080 resolution. This dataset is used to judge the quality of the video.
- **Ultra Video Group (UVG) Dataset [63]:** It is a video dataset with a high frame rate ($120fps$), in which the motion between adjacent frames is limited. According to the settings described in [64, 42, 26], we conduct our studies of video sequences with 1920 × 1080 resolutions.
- **Video Trace Library (VTL) Dataset [23]:** It comprises many raw YUV sequences utilized for low-level computer vision applications. In our studies, we employ 20 video sequences with 352 × 288 resolutions. The maximum duration of the video clips is set at 300 frames.

4.2. Implementation Details

Different α values are utilized for training. For example, α = 64, 128, 256, and 512 are used for PSNR, and α = 16, 32, 64, and 128 are employed for MS-SSIM. The proposed model is trained in two steps. Firstly, we set α = 1024 and train the model for 2,000,000 steps using mean square error at a high bit rate. We fine-tune the pre-trained model for a further 500,000 iterations for α = 64, 128, 256, and 512. In the second stage, models are further fine-tuned for roughly 80,000 steps using the MS-SSIM criteria as the distortion term to improve MS-SSIM performance. The previous learning rate for newly added modules is $5e-5$, and the final learning rate is $2e-5$ throughout the fine-tuning phases with batch size 4.

4.3. Evaluation Metrics and Settings

To analyze the rate-distortion effectiveness between the reconstructed frame and the ground-truth frame, we employ the PSNR, MS-SSIM [65], and Bjøntegaard delta bit rate (BDBR) [66]. The PSNR/MS-SSIM of each video sequence is obtained by averaging the PSNR/MS-SSIM of all reconstructed frames. The average number of bits needed for motion and residual coding in each frame is measured using Bpp (bits per pixel). The learning-based approaches have a fixed Group of Pictures (GOP) configuration but not traditional codecs. There is no restriction on the GOP size to compare the learning-based video codec to the conventional one. For example, the GOP size for the UVG, MCL-JCV, and VTL datasets is set to 12. Furthermore, the equivalent GOP size for H.265/H.264 in these works is 12 or 10. The x265 LDP high-speed mode is employed for H.265, and the settings are followed from [27]. To provide a fair comparison, the results of the methods (i.e., DVC-CVPR19 [26], Djelouah-CVPR19 et al. [41], Hu-ECCV20 et al. [45], M-LVC-CVPR20 [19], Lu-ECCV20 et al. [44], FVC-CVPR21 [23], DCVC-NeurIPS21 [24], RLVC-JSTSP21 [27], Sheng-21 [25]) are obtained from their respective articles. To demonstrate the effectiveness of our method, experiments with larger GOP sizes were also carried out in the ablation study.

5. Experimental Results and Analysis

In this section, we present experimental findings and detailed analysis to illustrate the efficiency of our method.

5.1. Computational Performance

Our model is built on PyTorch with CUDA support. Experiments are conducted using a Windows 10 workstation with an NVIDIA RTX 2080Ti GPU. The initial training stage takes around four days, the second training stage takes approximately two days, and the fine-tuning step takes 1 day. When encoding videos with a resolution of per 1080P frame, the encoding speed is 0.612s (resp. 0.576s), which is almost 26% faster

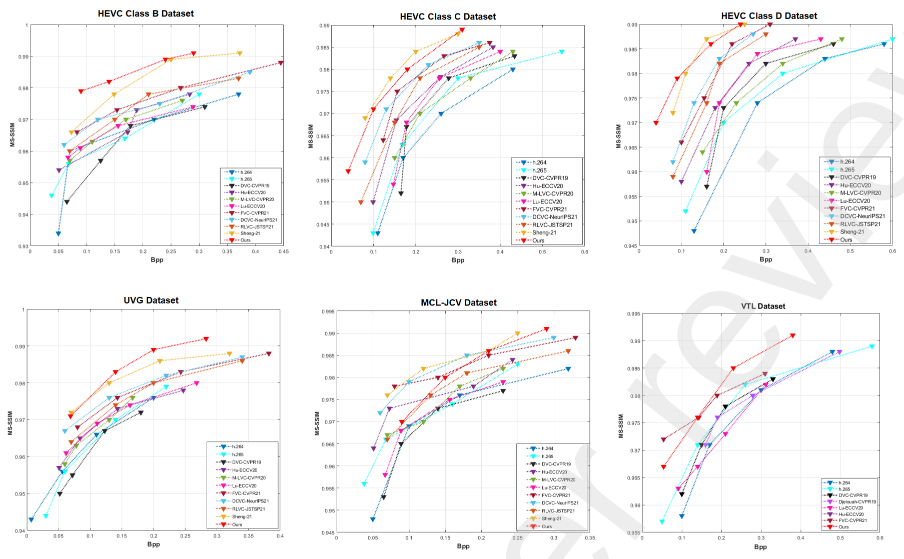

Figure 7: The MS-SSIM comparison of our method with the state-of-the-art learning-based methods, e.g., DVC-CVPR19 [26], Djelouah-CVPR19 et al. [41], Hu-ECCV20 et al. [45], M-LVC-CVPR20 [19], Lu-ECCV20 et al. [44], FVC-CVPR21 [23], DCVC-NeurIPS21 [24], RLVC-JSTSP21 [27], Sheng-21 [25] and conventional methods, H.264/H.265 [4, 5] on HEVC Test Sequences (Class B, C, and D) [5], UVG [63], MCL-JCV [62], and VTL [23] datasets.

than Sheng-21[25] (0.827 vs. 0.612), though the decoding speed has an 18% (0.472 vs. 0.576) gap. On the other hand, in FVC-CVPR21 [23], the authors stated that the overall coding speed is 0.548s. Despite this, they used the auto-regressive entropy model, which is not GPU-friendly. As a result, the coding performance was estimated to take a significant amount of time. They did not mention whether or not the entropy coding speed is included in their overall coding speed. Meanwhile, there are only 10.1M parameters in our model, whereas FVC-CVPR21[23] is 26M and Sheng-21[25] is 10.7M, which is around 157% and 7% higher than ours, correspondingly.

5.2. *Quantitative Performance*

Our video compression method is evaluated using MS-SSIM, PSNR, and BDBR.

5.2.1. MS-SSIM Evaluation

Fig. 7 demonstrates the rate-distortion curves of MS-SSIM for our method with state-of-the-art methods, e.g., conventional H.264/H.265 [4, 5], deep learning-based methods e.g., DVC-CVPR19 [26], Djelouah-CVPR19 et al. [41], Hu-ECCV20 et al. [45], M-LVC-CVPR20 [19], Lu-ECCV20 et al. [44], FVC-CVPR21 [23], DCVC-NeurIPS21 [24], RLVC-JSTSP21 [27], and Sheng-21 [25] in all datasets. Our method shows state-of-the-art/competitive MS-SSIM performance in the HEVC Class C, UVG, VTL, and MCL-JCV datasets/HEVC Class B and D datasets. In the HEVC class B and D datasets, our method received the same MS-SSIM result (around 0.99) as Sheng-21 [25]. However, in every dataset, our method outperforms the popular recent methods, i.e., FVC-CVPR21 [23], DCVC-NeurIPS21 [24], and RLVC-JSTSP21 [27] and the conventional methods, i.e., [4, 5] by a large margin, which reflects the efficacy of our proposed approach.

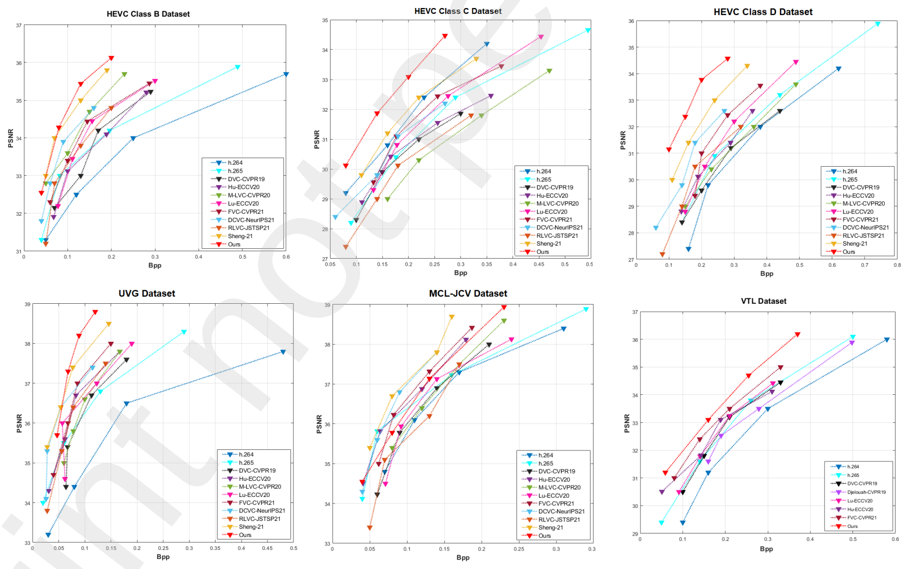

Figure 8: The PSNR comparison of our method with the state-of-the-art learning-based methods, e.g., DVC-CVPR19 [26], Djelouah-CVPR19 et al. [41], Hu-ECCV20 et al. [45], M-LVC-CVPR20 [19], Lu-ECCV20 et al. [44], FVC-CVPR21 [23], DCVC-NeurIPS21 [67], RLVC-JSTSP21 [27], Sheng-21 [25] and conventional methods, H.264/H.265 [4, 5] on HEVC Test Sequences (Class B, C, and D) [5], UVG [63], MCL-JCV [62], and VTL [23] datasets.

Table 1: Comparing the findings for bjontegaard delta bit rate (BDBR), calculated by the PSNR/MS-SSIM of our method with state-of-the-art on H.265 (x265 LDP very fast) in different datasets. The bit-rate savings are shown by negative BDBR values, whereas positive BDBR values indicate higher bit-rate costs. The red and green colors represent the highest and second highest results, respectively.

Dataset	DVC[26]	Hu[45]	Lu[44]	Agustsson[43]	HLVC [22]	M-LVC[19]	RLVC[27]	Liu [67]	FVC[23]	Ours
Class B	5.66/-2.74	-/-	-13.35/-7.93	-/-	-11.75/-37.44	-36.55/-42.82	-24.20/-50.42	-/-	-23.75/-54.51	-37.37/-57.19
Class C	25.88/-6.88	4.94/-32.44	-/-	-/-	7.83/-23.63	-/-	-4.67/-35.94	-/-	-14.18/-43.58	-16.78/-35.94
Class D	15.34/-18.51	-/-32.43	-6.86/-	-/-	-12.57/-52.56	-13.87/-36.27	-27.01/-48.85	-/-	-18.39/-51.19	-19.51/-52.12
MCL-JCV	-/-	-10.60/-34.10	4.21/-	-1.82/-33.61	-/-	-/-	-/-	-/-	-22.48/-52.00	-31.32/-58.19
UVG	10.40/8.05	-/-	-7.56/-25.49	-8.80/-38.04	-1.37/-30.12	-12.11/-25.44	-13.48/-40.62	-49.42/-30.70	-28.71/-45.25	-53.45/-50.21
VTL	-/-	-/-6.04	-16.05/-	-/-	-/-	-/-	-/-	-9.51/2.42	-28.10/-39.44	-40.21/-43.65
Average	8.03/-5.02	-2.83/-26.25	-7.92/-16.71	-5.31/-35.83	-4.47/-35.94	-20.84/-34.84	-17.34/-43.96	-29.4/-14.14	-22.60/-46.66	-33.12/-49.55

5.2.2. *PSNR Evaluation*

From the rate-distortion curves of PSNR in Fig. 8, our method also shows the state-of-the-art/competitive outcomes with the learning-based methods, e.g., DVC-CVPR19 [26], Djelouah-CVPR19 et al. [41], Hu-ECCV20 et al. [45], M-LVC-CVPR20 [19], Lu-ECCV20 et al. [44], FVC-CVPR21 [23], DCVC-NeurIPS21 [24], RLVC-JSTSP21 [27], Sheng-21 [25] and conventional methods, H.264/H.265 [4, 5] in every datasets. In particular, in the HEVC Class D dataset, H.265 outperforms all methods. However, our method overcomes the existing deep learning methods (particularly the recent methods, i.e., FVC-CVPR21 [23], DCVC-NeurIPS21 [24], RLVC-JSTSP21 [27], Sheng-21 [25]). Similarly, in the HEVC Class C dataset, our method is able to yield almost equal performance (around 34.5) to H.265, and Lu-ECCV20 [44]. As a result, we can infer that the proposed blocks in the module have a significant influence on the overall performance of the system.

5.2.3. *BDBR Evaluation*

In Table 1, we present the BDBR [66] findings of our method in comparison to other state-of-the-art methods (DVC-CVPR19 [26], Agustsson et al. [43], Hu-ECCV20 et al. [45], M-LVC-CVPR20 [19], Lu-ECCV20 et al. [44], HLVC-CVPR20 [27], FVC-CVPR21 [23], RLVC-JSTSP21 [27], Liu [67]) where H.265 is set as an anchor. Our method saves around 2%, 28%, 8%, 30%, and 11% bit saving rates in terms of PSNR as well as 5%, 11%, 10%, 10%, and 6% in terms of MS-SSIM on HEVC Class B, MCL-JCV, UVG, and VTL datasets, respectively; these results demonstrate the superiority of our proposed method. It is also clear that our method surpasses the recent deep learning methods, i.e., FVC-CVPR21 [23], RLVC-JSTSP21 [27], Liu [67] and also other existing deep learning methods. On the other hand, in the HEVC Class B dataset, regarding PSNR, it exhibits state-of-the-art results; nevertheless, in MS-SSIM, it has roughly an 18% gap compared to the FVC-CVPR21 [23]. Additionally, in HEVC Class D, it achieves the second highest result (approximately 28% less than RLVC-JSTSP21 [27] and slightly less than HLVC-CVPR20 [27]) in both PSNR and MS-SSIM.

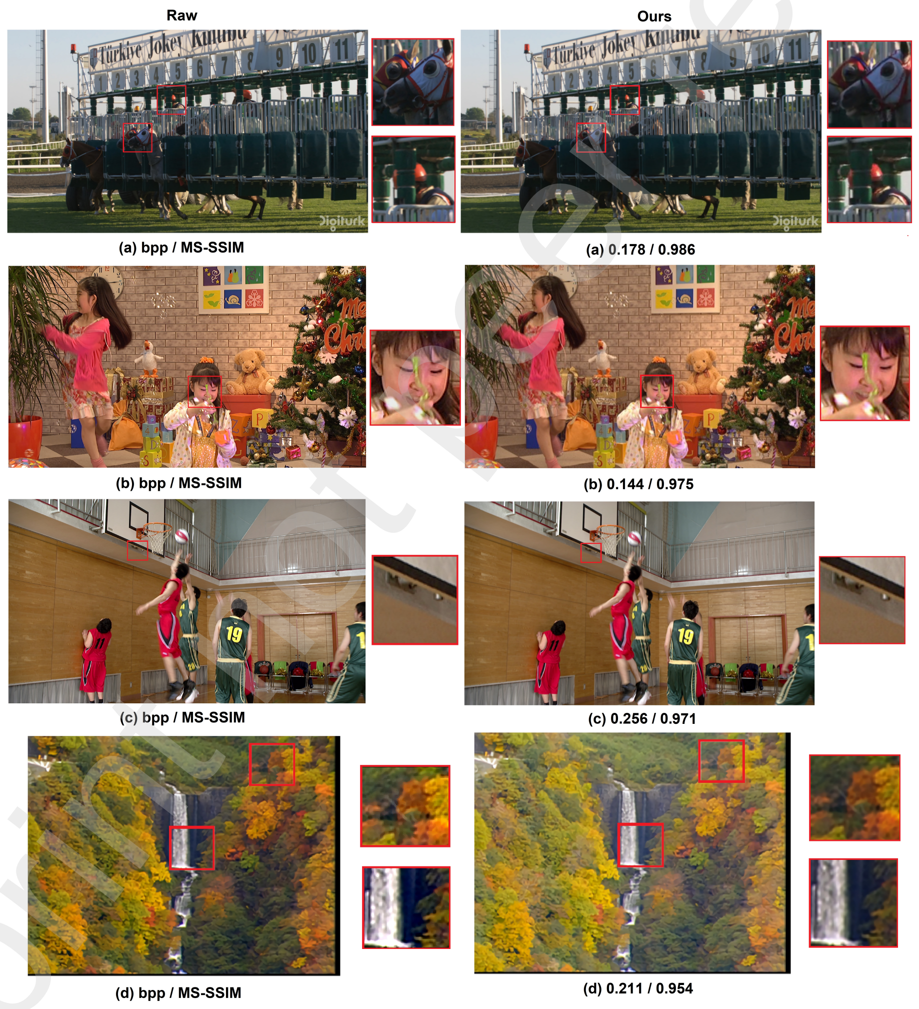

Figure 9: Qualitative comparison between the original and our reconstruction frames on the UVG [63] (a), HEVC [5] (b), MCL-JCV [62] (c), and VTL [23] (d) datasets.

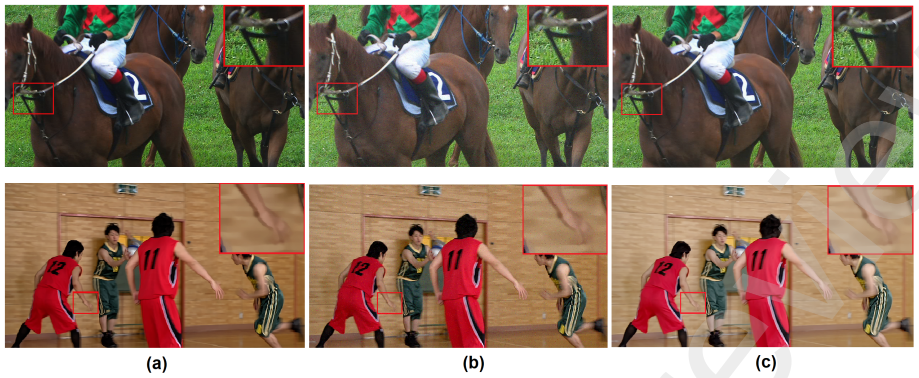

Figure 10: The visualization of the reconstruction frame with the proposed RSwCU. (a) represents the original frames derived from the HEVC Class C and D dataset [5], (b) the artifact frames after the concatenation of the predicted and residual frame, and (c) the final reconstructed frame by applying the ACM module with RSwCU.

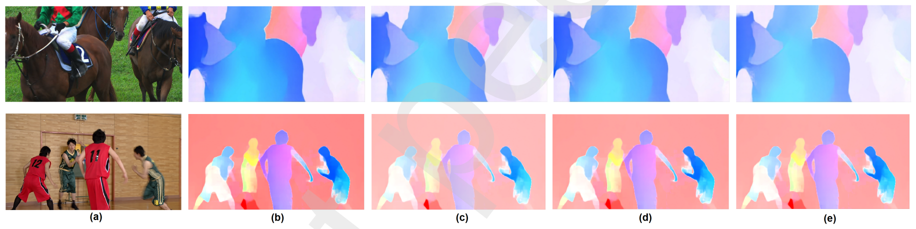

Figure 11: The visualization results of the motion compression using our proposed autoencoder-style network. (a), (b), (c), (d), and (e) represent the raw images from HEVC Class C and D datasets, the raw optical flow from the ME module using DeepPyNet [51], visualization using 1 DRSA, 2 DRSA, and lastly, the visualization of the compressed optical flow/motion vector, respectively.

5.3. Qualitative Result

This sub-section provides qualitative results of predicted, residual, and final reconstructed frames using our proposed blocks (i.e., DRSA, CRB) and the ACM module with RSwCU. The qualitative comparison of the original and our reconstructed frames on all datasets (UVG, HEVC, MCL-JCV, and VTL) is shown in Fig. 9. The Bpp represents the average bits per pixel of each video frame. From the MS-SSIM values and Bpp from Fig. 9, our reconstructed frames exhibit fewer compression artifacts, fewer bit rates, and retain more abstract features. As a result, our method has a higher visual

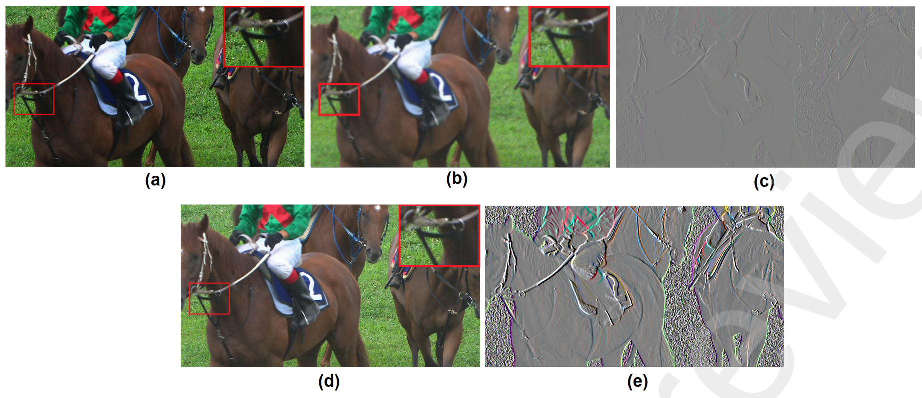

Figure 12: The visualization of the predicted frame from the MC module utilizing the CRB block. (a) represents the raw image, (b) and (c) the predicted frame and the residual frame when there is no CRB, and (d) and (e) the predicted frame and the residual frame when there is CRB, respectively.

quality while creating fewer fuzzy contours. Although the frame in the VTL dataset has a lower quality, our proposed method still produces a 0.954 MS-SSIM with 0.211 Bpp.

Fig. 10 shows the qualitative results of the reconstructed/decoded frame with and without considering the ACM module with RSwCU in the proposed method. It is noticeable that the decoded results without applying the ACM module have more artifacts with noise and are less smooth. In contrast, the outcomes of our technique (by considering the ACM module) are comparatively much smoother (i.e., the grasses in Fig. 10(c) upper part and the hand Fig. 10(c)) lower part with fewer artifacts and better visualization. Besides, the PSNR/MS-SSIM value for (b) is 31.44dB/0.912 (upper part) and (b) is 31.67dB/0.927 (lower part), while it is significantly high for (c) at 35.11dB/0.981 (upper part) and (c) at 33.98dB/0.976. Thus, our ACM module can improve more accurate results with the RSwCU network.

We proposed a DRSA block in the autoencoder network for the MVC module to boost the system's performance. The visualization of DRSA blocks with the raw optical flows is presented in Fig. 11. Our proposed DRSA can learn optical flow with abstract features and reduce unnecessary information (see Fig. 11 (c, d)), which can increase the RDO performance and make it faster. In Fig.12, we also show the MC module's predicted frame using our proposed CRB block. It can be shown that without the CRB block in Fig. 12 (b, c), the objects in the residual frame (c) cannot be identified, and the predicted frame (b) is also of lower quality. However, the predicted frame (d) gets smoother after applying the CRB, while the residual frame becomes more apparent (e), which assists in optimizing compression performance as well. For a more detailed look at the efficacy of DRSA and CRB blocks, please refer to the ablation study in Table 2.

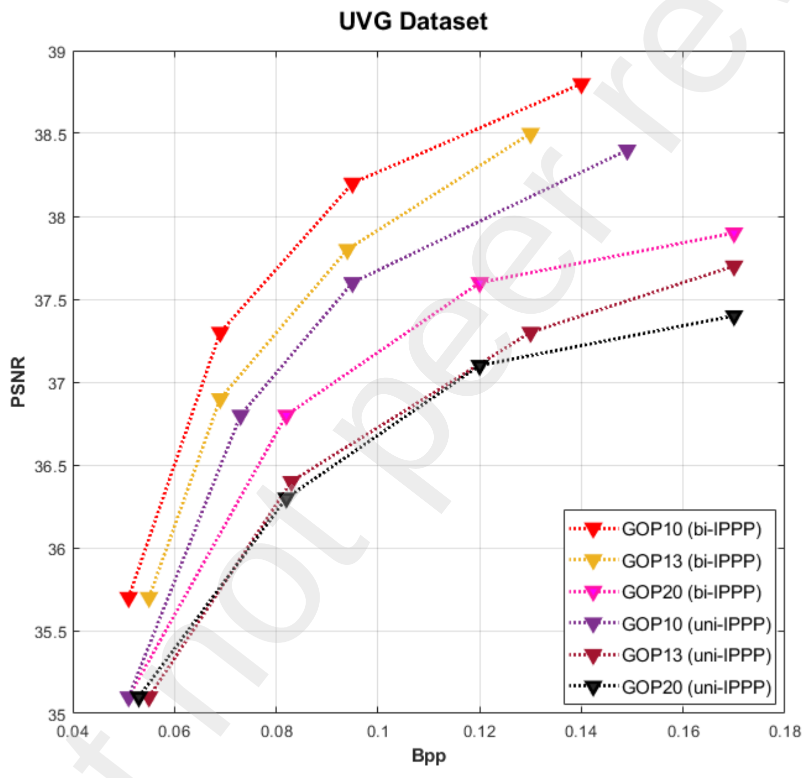

Figure 13: The effectiveness of a variety of GOP configurations on the UVG [63] dataset.

5.4. *Ablation Study*

We conduct an extensive ablation study on different modules and GOP sizes to evaluate the effectiveness of our proposed method.

5.4.1. *Effectiveness of Different Modules*

Table 2 shows the result of the ablation study in terms of PSNR, MS-SSIM, number of parameters, and inference time (encoding and decoding speed in second (s)), on different modules across the entire network on the HEVC Class B dataset. When the baseline (RC, entropy coding, and DFB), ME, the proposed DRSA for MVC, CRB for MC, and ACM module are employed (Case 1) together with joint training, we received the highest PSNR (36.13dB) and MS-SSIM (0.991) of our system with 10.15M parameters and 0.826s. The PSNR is decreased by 1.35dB (from 36.13 to 34.78) and 2.15dB (from 36.13 to 33.98), and MS-SSIM is reduced by 3.5% (0.991 vs. 0.956), and 4% (0.991 vs. 0.951) for omitting the DRSA and CRB separately. However, parameters and inference time are also increased by around 14% (10.15 vs. 11.78) and 22% (10.15 vs. 13.01) in case 2 as well as 19% (0.826 vs. 1.026) and 31% (0.826 vs. 1.196) in case 3, respectively. In contrast, the system delivers too poor performance without DRSA and CRB, as seen by lowered PSNR and MS-SSIM and dramatically higher parameters and inference time (in case 6). To further illustrate the ACM module's effectiveness, we conducted an experiment where the ACM was removed from the architecture. As a result, the PSNR and MS-SSIM decreased significantly (in case 4), though the parameters and inference time were also decreased. An additional experiment is performed with the ME module in case 5. The system would be unable to provide a satisfying RDO performance without the ME module. In conclusion, we can claim that our proposed blocks, DRSA and CRB, as well as the proposed module ACM, have an unrivaled influence on improving the RDO performance of the system.

5.4.2. *GOP Evaluation*

We further analyze the performance of the different group of pictures (GOPs) for our proposed method while compressing video. Two distinct GOP configurations are employed: the bi-directional IPPP (bi-IPPP) and the normal IPPP structure (uni-IPPP) [22]. Fig. 13 demonstrates the PSNR performance of our proposed approach utilizing GOP = 10, 13, and 20 for both bi-IPPP and uni-IPPP. For bi-IPPP configuration, we follow the configuration from [22]. We observe that when the GOP size is increased to 20 (bi-IPPP), the performance remains comparable with that of GOP = 13 (bi-IPPP)

Table 2: The effectiveness of different modules with DRSA, CRB blocks on the proposed network in terms of PSNR, MS-SSIM, number of parameters (Param), and inference time (IT) in second (s) on HEVC Class B Dataset [5].

Case	Baseline	ME	MVC	MC	ACM	PSNR	MS-SSIM	Param (M)	IT (s)
1	✓	✓	with DRSA	with CRB	✓	**36.13**	**0.991**	10.15	0.826
2	✓	✓	no DRSA	with CRB	✓	34.78	0.956	11.78	1.026
3	✓	✓	with DRSA	no CRB	✓	33.98	0.951	13.01	1.196
4	✓	✓	with DRSA	with CRB	×	32.65	0.936	8.45	0.723
5	✓	×	with DRSA	with CRB	✓	30.79	0.926	9.12	0.798
6	✓	✓	no DRSA	no CRB	✓	32.01	0.951	13.22	1.401

and GOP = 10 (bi-IPPP), with just a modest impairment efficiency. It happens when there is a greater gap between the I-frames and the P-frames in bi-IPPP. However, for GOP size 20, the PSNR is decreased by only around 2% from GOP 10 (roughly 38.8 vs. 37.9) and around 1.5% from GOP 13 (approximately 38.5 vs. 37.9). On the other hand, when we analyze the uni-IPPP in terms of GOP = 10, 13, and 20, the performance of the uni-IPPP mode is quite comparable over a range of GOP sizes of the bi-IPPP mode. For example, by around 0.4dB, 1.1dB, and 1.4dB from GOP 10 (bi-IPPP). To summarize, the proposed method is compatible with a wide range of GOP sizes. It is remarkably adaptable to GOP = 20 (bi-IPPP or uni-IPPP) without significantly reducing compression performance.

5.4.3. *Effectiveness of Output Feature Channels with DRSA*

Since our proposed autoencoder network downsamples frames at the encoder and then upsamples them in decoded frames, it must make predictions quickly enough to operate at the frame rate of the video on a low-power edge device. To assess the feasibility of our model, we measure the end-to-end inference time per frame as a function of the number of output feature channels with DRSA and without DRSA in the autoencoder network for the encoder, as shown in TABLE 3. The upper part represents the result without DRSA and the lower part with DRSA. The PSNR, MS-SSIM, inference time, and parameters are raised by increasing the feature channel (FC). However, the inference time and parameters significantly increase at FC of 512, while PSNR

Table 3: The effectiveness of output feature channels with DRSA and without DRSA block for the encoder in MVCA network in terms of PSNR, MS-SSIM, inference time, and the number of parameters on UVG [63] dataset.

Feature Channel (FC)	32	64	128	256	512	1024	2048
PSNR (dB)	33.15	34.11	35.25	36.18	36.32	36.32	36.36
MS-SSIM	0.931	0.943	0.956	0.969	0.971	0.972	0.972
Inference time (ms)	25	27	29	34	40	55	78
Num. of parameters (M)	1.14	2.29	3.22	4.21	5.67	8.99	11.02
PSNR (dB)	35.50	36.22	37.78	38.06	38.80	38.80	38.81
MS-SSIM	0.951	0.962	0.978	0.989	0.991	0.991	0.991
Inference time (ms)	22	23	24	27	31	50	71
Num. of parameters (M)	0.89	2.03	3.01	3.99	4.97	7.85	9.79

and MS-SSIM improve slightly for both parts of the table. For example, for FC of 1024 and 2048, the parameters are increased by around 37% (5.67M vs. 8.99M) and 49% (5.67M vs. 11.02M) while expanding the inference time by roughly 27% (40ms vs. 55ms) and 49% (40ms vs. 78ms), respectively in terms of the upper parts. The PSNR and MS-SSIM are improving slightly for the feature channels of 1024 and 2048. Therefore, we build the autoencoder model up to 512 output feature channels, assuring that inference on a single frame takes just 40 ms for encoding. However, in this case, we proposed the DRSA in an autoencoder network to reduce the computational cost further. Along with the DRSA up to 512 feature channels (lower part of the table), the parameters are minimized by approximately 12% (5.67M vs. 4.97M), and the inference time is reduced by roughly 23% (40ms vs. 31ms). Besides, the PSNR and MS-SSIM are also improved by around 6% (36.32 vs. 38.80) and 2% (0.971 vs. 0.991), respectively. Thus, we can conclude that the effectiveness of our proposed DRSA in the MVC module is unparalleled.

5.5. Limitations

Our proposed approach has not investigated the entropy model and residual compression module. For entropy coding, [9] is applied in our experiment. As a result, the overall coding speed is 1.9s (0.712s for the entropy coding). In addition, the current traditional-based approach H.266/VTM [6] surpassed the recent deep learning-based methods. Therefore, the entropy model needs further investigation in the future in order to reduce our overall coding speed, and a performance comparison using H.266/VTM should be carried out.

6. Conclusion

This article presented a fully end-to-end deep learning-based video compression framework that efficiently reduces temporal redundancy and optimizes the rate-distortion tradeoff. A deep residual attention split (DRAS) block was proposed in a motion compression network to generate more valuable features for the decoder while enhancing RDO efficiency. In motion compensation, a channel residual block (CRB) was suggested to produce a predicted frame accurately and play a role in boosting the residual frame. An artifact contraction module (ACM) via the proposed residual swin convolution UNet block has been included in this model to increase the reconstruction frame's quality. Furthermore, the training phase included an update buffer for the precise reference frames. A single rate-distortion loss function was conducted to optimize all of the modules. Our method received competitive PSNR, MS-SSIM, and bit-saving performance over others with fewer parameters (10.1M) on four datasets. An extensive ablation study was also presented to prove the effectiveness of our proposed approach.

Conflict of interest

The authors declared that there is no known conflict of interest.

References

[1] C. V. Networking, Cisco global cloud index: Forecast and methodology, 2015-2020. white paper, Cisco Public, San Jose (2016) 2016.

[2] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, P. Krähenbühl, Compressed video action recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6026–6035.

[3] S. Yuan, J. Hu, Research on image compression technology based on huffman coding, Journal of Visual Communication and Image Representation 59 (2019) 33–38.

[4] T. Wiegand, G. J. Sullivan, G. Bjontegaard, A. Luthra, Overview of the h. 264/avc video coding standard, IEEE Transactions on circuits and systems for video technology 13 (7) (2003) 560–576.

[5] G. J. Sullivan, J.-R. Ohm, W.-J. Han, T. Wiegand, Overview of the high efficiency video coding (hevc) standard, IEEE Transactions on circuits and systems for video technology 22 (12) (2012) 1649–1668.

[6] B. Bross, J. Chen, S. Liu, Y.-K. Wang, Versatile video coding (draft 5), Joint Video Experts Team (JVET) of ITU-T SG 16 (2019) 3–12.

[7] H. Lim, H. Park, A ringing-artifact reduction method for block-dct-based image resizing, IEEE transactions on circuits and systems for video technology 21 (7) (2011) 879–889.

[8] L. Theis, W. Shi, A. Cunningham, F. Huszár, Lossy image compression with compressive autoencoders, arXiv preprint arXiv:1703.00395.

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

[10] D. Minnen, J. Ballé, G. Toderici, Joint autoregressive and hierarchical priors for learned image compression, arXiv preprint arXiv:1809.02736.

[11] J. Lee, S. Cho, S.-K. Beack, Context-adaptive entropy model for end-to-end optimized image compression, arXiv preprint arXiv:1809.10452.

[12] D. Liu, Y. Li, J. Lin, H. Li, F. Wu, Deep learning-based video coding: A review and a case study, ACM Computing Surveys (CSUR) 53 (1) (2020) 1–35.

[13] T. Chen, H. Liu, Q. Shen, T. Yue, X. Cao, Z. Ma, Deepcoder: A deep neural network based video compression, in: 2017 IEEE Visual Communications and Image Processing (VCIP), IEEE, 2017, pp. 1–4.

[14] Z. Liu, X. Yu, Y. Gao, S. Chen, X. Ji, D. Wang, Cu partition mode decision for hevc hardwired intra encoder using convolution neural network, IEEE Transactions on Image Processing 25 (11) (2016) 5088–5103.

[15] R. Song, D. Liu, H. Li, F. Wu, Neural network-based arithmetic coding of intra prediction modes in hevc, in: 2017 IEEE Visual Communications and Image Processing (VCIP), IEEE, 2017, pp. 1–4.

[16] G. Lu, W. Ouyang, D. Xu, X. Zhang, Z. Gao, M.-T. Sun, Deep kalman filtering network for video compression artifact reduction, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 568–584.

[17] R. Yang, M. Xu, Z. Wang, T. Li, Multi-frame quality enhancement for compressed video, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6664–6673.

[18] S. Andris, P. Peter, R. M. K. Mohideen, J. Weickert, S. Hoffmann, Inpainting-based video compression in fullhd, arXiv preprint arXiv:2008.10273.

[19] J. Lin, D. Liu, H. Li, F. Wu, M-lvc: Multiple frames prediction for learned video compression, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 3546–3554.

[20] X. Wu, Z. Zhang, J. Feng, L. Zhou, J. Wu, End-to-end optimized video compression with mv-residual prediction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 156–157.

[21] R. Feng, Y. Wu, Z. Guo, Z. Zhang, Z. Chen, Learned video compression with feature-level residuals, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 120–121.

[22] R. Yang, F. Mentzer, L. Van Gool, R. Timofte, Learning for video compression with recurrent auto-encoder and recurrent probability model, IEEE Journal of Selected Topics in Signal Processing 15 (2) (2020) 388–401.

[23] Z. Hu, G. Lu, 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.

[24] J. Li, B. Li, Y. Lu, Deep contextual video compression, Advances in Neural Information Processing Systems 34.

[25] X. Sheng, J. Li, B. Li, L. Li, D. Liu, Y. Lu, Temporal context mining for learned video compression, arXiv preprint arXiv:2111.13850.

[26] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, 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. 11006–11015.

[27] R. Yang, F. Mentzer, L. V. Gool, R. Timofte, Learning for video compression with hierarchical quality and recurrent enhancement, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 6628–6637.

[28] G. K. Wallace, The jpeg still picture compression standard, IEEE transactions on consumer electronics 38 (1) (1992) xviii–xxxiv.

[29] A. Skodras, C. Christopoulos, T. Ebrahimi, The jpeg 2000 still image compression standard, IEEE Signal processing magazine 18 (5) (2001) 36–58.

[30] F. Bellard, Bpg image format, URL https://bellard. org/bpg 1 (2015) 2.

[31] J. Ballé, V. Laparra, E. P. Simoncelli, End-to-end optimized image compression, arXiv preprint arXiv:1611.01704.

[32] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. Chinen, S. J. Hwang, J. Shor, G. Toderici, Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4385–4393.

[33] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, R. Sukthankar, Variable rate image compression with recurrent neural networks, arXiv preprint arXiv:1511.06085.

[34] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, M. Covell, Full resolution image compression with recurrent neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5306–5314.

[35] Y. Patel, S. Appalaraju, R. Manmatha, Human perceptual evaluations for image compression, arXiv preprint arXiv:1908.04187.

[36] Y. Blau, T. Michaeli, Rethinking lossy compression: The rate-distortion-perception tradeoff, in: International Conference on Machine Learning, PMLR, 2019, pp. 675–685.

[37] A. J. Hussain, A. Al-Fayadh, N. Radi, Image compression techniques: A survey in lossless and lossy algorithms, Neurocomputing 300 (2018) 44–69.

[38] S. Ma, X. Zhang, C. Jia, Z. Zhao, S. Wang, S. Wang, Image and video compression with neural networks: A review, IEEE Transactions on Circuits and Systems for Video Technology 30 (6) (2019) 1683–1698.

[39] Z. Chen, T. He, X. Jin, F. Wu, Learning for video compression, IEEE Transactions on Circuits and Systems for Video Technology 30 (2) (2019) 566–576.

[40] C. Y. S. Kin, B. Coker, Video compression using recurrent convolutional neural networks (2017).

[41] A. Djelouah, J. Campos, S. Schaub-Meyer, C. Schroers, Neural inter-frame compression for video coding, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 6421–6429.

[42] A. Habibian, T. v. Rozendaal, J. M. Tomczak, T. S. Cohen, Video compression with rate-distortion autoencoders, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 7033–7042.

[43] E. Agustsson, D. Minnen, N. Johnston, J. Balle, S. J. Hwang, 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.

[44] G. Lu, C. Cai, X. Zhang, L. Chen, W. Ouyang, D. Xu, Z. Gao, Content adaptive and error propagation aware deep video compression, in: European Conference on Computer Vision, Springer, 2020, pp. 456–472.

[45] Z. Hu, Z. Chen, D. Xu, G. Lu, W. Ouyang, S. Gu, Improving deep video compression by resolution-adaptive flow coding, in: European Conference on Computer Vision, Springer, 2020, pp. 193–209.

[46] A. Ranjan, M. J. Black, Optical flow estimation using a spatial pyramid network, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4161–4170.

[47] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, T. Brox, Flownet: Learning optical flow with convolutional networks, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 2758–2766.

[48] D. Sun, X. Yang, M.-Y. Liu, J. Kautz, Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8934–8943.

[49] T.-W. Hui, X. Tang, C. C. Loy, Liteflownet: A lightweight convolutional neural network for optical flow estimation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8981–8989.

[50] T.-W. Hui, X. Tang, C. C. Loy, A lightweight optical flow cnn—revisiting data fidelity and regularization, IEEE transactions on pattern analysis and machine intelligence 43 (8) (2020) 2555–2569.

[51] A. A. Jeny, M. B. Islam, T. Aydin, Deeppynet: A deep feature pyramid network for optical flow estimation, in: 2021 36th International Conference on Image and Vision Computing New Zealand (IVCNZ), IEEE, 2021, pp. 1–6.

[52] L. Ning, A. Wang, L. Zhao, W. Xue, D. Bu, Mranet: Multi-atrous residual attention network for stereo image super-resolution, Journal of Visual Communication and Image Representation 77 (2021) 103115.

[53] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, X. Tang, Residual attention network for image classification, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 3156–3164.

[54] H. Liu, T. Chen, P. Guo, Q. Shen, X. Cao, Y. Wang, Z. Ma, Non-local attention optimized deep image compression, arXiv preprint arXiv:1904.09757.

[55] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, H. Lin, Z. Zhang, Y. Sun, T. He, J. Mueller, R. Manmatha, et al., Resnest: Split-attention networks, arXiv preprint arXiv:2004.08955.

[56] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[57] M. Jaderberg, K. Simonyan, A. Zisserman, et al., Spatial transformer networks, Advances in neural information processing systems 28 (2015) 2017–2025.

[58] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical image computing and computer-assisted intervention, Springer, 2015, pp. 234–241.

[59] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, Swin transformer: Hierarchical vision transformer using shifted windows, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 10012–10022.

[60] B. Lim, S. Son, H. Kim, S. Nah, K. Mu Lee, Enhanced deep residual networks for single image super-resolution, in: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, 2017, pp. 136–144.

[61] T. Xue, B. Chen, J. Wu, D. Wei, W. T. Freeman, Video enhancement with task-oriented flow, International Journal of Computer Vision 127 (8) (2019) 1106–1125.

[62] H. Wang, W. Gan, S. Hu, J. Y. Lin, L. Jin, L. Song, P. Wang, I. Katsavounidis, A. Aaron, C.-C. J. Kuo, Mcl-jcv: a jnd-based h. 264/avc video quality assessment dataset, in: 2016 IEEE International Conference on Image Processing (ICIP), IEEE, 2016, pp. 1509–1513.

[63] A. Mercat, M. Viitanen, J. Vanne, Uvg dataset: 50/120fps 4k sequences for video codec analysis and development, in: Proceedings of the 11th ACM Multimedia Systems Conference, 2020, pp. 297–302.

[64] C.-Y. Wu, N. Singhal, P. Krahenbuhl, Video compression through image interpolation, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 416–431.

[65] Z. Wang, E. P. Simoncelli, A. C. Bovik, Multiscale structural similarity for image quality assessment, in: The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003, Vol. 2, Ieee, 2003, pp. 1398–1402.

[66] G. Bjontegaard, Calculation of average psnr differences between rd-curves, VCEG-M33.

[67] B. Liu, Y. Chen, S. Liu, H.-S. Kim, Deep learning in latent space for video prediction and compression, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 701–710.