Signal Processing: Image Communication 103 (2022) 116633

Contents lists available at ScienceDirect

Signal Processing: Image Communication

journal homepage: www.elsevier.com/locate/image

Learning to compress videos without computing motion

Meixu Chen [a,*], Todd Goodall [b,1], Anjul Patney [b,2], Alan C. Bovik [a]

[a] Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, 78712, USA
[b] Facebook Reality Labs, Redmond, 98052, USA

ARTICLE INFO

Keywords:
Video compression
Deep learning
Motion

ABSTRACT

Video has become an increasingly important part of our daily digital communication. With the development of higher resolution contents and displays, its significant volume poses significant challenges to the goals of acquiring, transmitting, compressing and displaying high quality video content. In this paper, we propose a new deep learning video compression architecture that does not require motion estimation, which is the most expensive element of modern hybrid video compression codecs like H.264 and HEVC. Our framework exploits the regularities inherent to video motion, which we capture by using displaced frame differences as video representations to train the neural network. In addition, we propose a new space–time reconstruction network based on both an LSTM model and a UNet model, which we call LSTM-UNet. The combined network is able to efficiently capture both temporal and spatial video information, making it highly amenable for our purposes. The new video compression framework has three components: a Displacement Calculation Unit (DCU), a Displacement Compression Network (DCN), and a Frame Reconstruction Network (FRN), all of which are jointly optimized against a single perceptual loss function. The DCU *removes the need for motion estimation found in hybrid codecs,* and is less expensive. In the DCN, an RNN-based network is utilized to compress displaced frame differences as well as retain temporal information between frames. The LSTM-UNet is used in the FRN to learn space time differential representations of videos. Our experimental results show that our compression model, which we call the MOtionless VIdeo Codec (MOVI-Codec), learns how to efficiently compress videos without computing motion. Our experiments show that MOVI-Codec outperforms the Low-Delay P (LDP) *veryfast* setting of the video coding standard H.264 and exceeds the performance of the modern global standard HEVC codec, using the same setting, as measured by MS-SSIM, especially on higher resolution videos. In addition, our network outperforms the latest H.266 (VVC) codec at higher bitrates, when assessed using MS-SSIM, on high resolution videos. The MOVI-Codec project page can be found at https://github.com/Meixu-Chen/MOVI-Codec.

1. Introduction

Video traffic is predicted to reach 82 percent of all consumer Internet traffic by 2021 [1], and to continue this rapid growth even further. The increasing share of video in Internet traffic is being driven by several factors, including the great diversity and extraordinary popularity of streaming and social media services, the rise of video teleconferencing and online video education (accelerated by the Coronavirus Crisis), and significant increases in video resolution. Indeed, it is estimated that by 2023, two-thirds of installed flat-panel television sets will be UHD, up from 33 percent in 2018 [2]. Given significant strains on available bandwidth, it is crucial to continue and greatly accelerate the evolution of video compression systems.

Traditional video compression codecs, like H.264, HEVC and the latest VVC/H.266 process videos through a sequence of hand-designed algorithms and modules, including block motion estimation, and local decorrelating decompositions like the Discrete Cosine Transform (DCT). Although the component modules of modern hybrid codecs have been carefully designed over several generations, the overall codecs have not been globally optimized other than by visual examination or post-facto objective measurement of results, typically by the highly fallible PSNR [3]. Naturally, one could expect the performances of video codecs to be improved by collective, end-to-end optimization. Because of their tremendous ability to learn efficient visual representations, deep learning models are viewed as highly promising vehicles of developing alternative, globally optimal video codecs, and a variety of deep learning based image compression architectures have been proposed [4–19]. These new models have deployed Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), autoencoders, and Generative Adversarial Networks (GAN) yielding rate–distortion

* Corresponding author.
E-mail addresses: chenmx@utexas.edu (M. Chen), beyondmetis@gmail.com (T. Goodall), rwebb@fb.com (A. Patney), bovik@ece.utexas.edu (A.C. Bovik).
[1] T. Goodall was with Facebook Reality Labs when this work was conducted, and now he is with Apple, Inc.
[2] A. Patney was with Facebook Reality Labs when this work was conducted, and now he is with Nvidia.

https://doi.org/10.1016/j.image.2022.116633
Received 5 July 2021; Received in revised form 29 December 2021; Accepted 3 January 2022
Available online 6 January 2022

efficiencies that are reportedly comparable to those of traditional image compression codecs like JPEG, JPEG 2000, and BPG. Encouraged by these advances, several authors have devised deep video compression models that suggest the considerable promise of this general approach. Wu et al. [20] proposed the first end-to-end trained deep video codec, using a hierarchical frame interpolation scheme. A block-based deep video compression codec was proposed by Chen et al. [19]. Lu et al. proposed an end-to-end video compression model (DVC) [21] which replaces each component of the traditional hybrid video codec with a deep learning model, which are jointly trained as a global hybrid architecture against a single loss function. Another hierarchical video compression architecture called HLVC (Hierarchical Learned Video Compression) was proposed by Yang et al. [22].

In both traditional video codecs and recent deep learning-based ones, motion estimation and compensation has occupied a significant portion of the system resources. Motion estimation requires an expensive search process that we avoid, by instead training the network to efficiently represent the residuals between each current frame and a set of spatially-displaced neighboring frames. Computing a set of frame differences, even over many displacement directions is much cheaper than effective search processes. Moreover, while the statistics of motion are generally not regular, the intrinsic statistics of frame differences exhibit strong regularities [23], including those of differences between spatially displaced frames [24]. The strong internal structure of these frame differences makes them easier to efficiently represent in a deep architecture.

Our idea is inspired by the way the human vision system processes natural time-varying images. Many studies have produced strong evidence suggesting that the early stages of the vision system are primarily implicated in reducing redundancies in the sensed input visual signals [25,26]. Indeed, much of early visual processing along the retino-cortical pathway appears to be devoted to process of spatial and temporal decorrelation [27–32]. We have found that sets of spatially displaced frame differences, which are space–time processes, supply a rich and general way to exploit space–time redundancies [23,24]. Importantly, our idea is also related to recent theories of the role of microsaccades in human visual information processing [28–31]. Microsaccades create small spatial displacements of visual field from moment to moment. While microsaccades have been theorized to play roles in avoiding retinal saturation, maintaining accurate fixation in the presence of drifts, and preserving the perception of fine spatial details [31], they are more recently thought to play an important role in efficiently representing locally changing and shifting space–time visual information [28–31]. We believe that micro-saccadic eye movements deployed by the human eye have adapted to the local regularities induced by small spatial displacements over time, in order to achieve more efficient visual (neural) representations. This has inspired us to, in like manner, trained a deep coder–decoder network to compress videos using regular displaced residual representations as inputs.

By capturing displaced frame differences from a large database of videos, and feeding them into a deep space–time coding–decoding network, we have formulated a new breed of deep video compression algorithms that are motion computation free, statistically motivated, and have perceptual relevance. The contributions of this work can be summarized as follows:

- We innovate the use of displaced frame differences to capture efficient representations of structures induced by motion.
- Our method avoids the computational overhead of motion estimation and motion compensation.
- A combined LTSM-UNet efficiently captures both spatial and temporal information which it uses to recreate video frames from the abstracted video code.
- The entire video compression system is collectively jointly optimized using a single loss function.

Our results show that video compression can be efficiently accomplished without explicitly computing motion predictions. We trained the new MOVI-Codec architecture end-to-end on the Kinetics-600 dataset and the Vimeo-90K dataset, using a single perceptual loss function (MS-SSIM), and tested it on the UVG dataset, the VTL dataset, and the HEVC Standard Test Sequences (Class B, Class C, Class D, and Class E). Our experimented results show that our new model outperforms the widely used video codec H.264 in LDP *veryfast* setting, and exceeds the performance of the latest standard video codec H.265 using the same setting. In addition, our network outperforms the latest H.266 (VVC) codec at higher bitrates, as assessed by the perceptually relevant MS-SSIM algorithm, on high resolution videos.

The rest of the paper is organized as follows. Section 2 briefly introduces current progress on learning-based methods for image/video compression and motion estimation. Section 3 describes details of the architecture and training protocol of the new MOVI-Codec model. Section 4 discusses the experiments we conducted and their outcomes, along with a data analysis along several dimensions. Section 5 concludes the paper with a discussion of future research directions.

2. Related works

2.1. Deep image compression

A variety of standardized image compression engines have been proposed over the years to meet the needs of increasingly picture-centric technologies. JPEG algorithm [33], and later challengers JPEG 2000 [34], BPG [35], and VP9 [36]. These methods have proven to be quite practical, and in the case of JPEG, ubiquitous. Yet they are all handcrafted, highly modularized without the benefit of collective optimization of all their elements. Each of these standards maps pixels to a less correlated representation, regardless of the attributes of the input image. These transformed values are then non-uniformly quantized, typically with reference to a human visual sensitivity model.

A variety of authors have recognized the potential of deep learning to advance progress on the image compression problem (a still timely goal given the senectitude of the prevailing JPEG standard), and many learning-based architectures have been devised [4–18]. Given that Convolutional Neural Networks (CNN) [37] were the first deep learning models to obtain standout performance on image analysis problems, it was natural that it be the first deep architecture to be applied to learning-based image compression. Ballé et al. [8] proposed a CNN-based image compression framework that was optimized end-to-end, which was shown to outperform JPEG2000 with respect to both MS-SSIM and PSNR image quality measures. Their framework was later extended by incorporating a hyperprior to capture spatial dependencies in the latent representation for entropy estimation [6]. In [14], Minnen et al. further enhanced the entropy model, by combining autoregressive and hierarchical priors to exploit the probabilistic structure in the latents. The resulting model was reported to outperform BPG with respect to both PSNR and MS-SSIM. Another architecture favored for learning-based image compression are Recurrent Neural Networks (RNN), because of their ability to exploit representative memories. Long Short-Term Memory (LSTM) models were proposed [38] to address the vanishing gradient problem of RNNs. Toderici et al. [4,5] was the first to deploy a deep RNN-based architecture for image compression by utilizing a scale-additive framework. This architecture allows for variable bit rates and only needs to be trained once. The authors also presented results using different types of RNNs, including LSTM, associative LSTM and a hybrid of a Gated Recurrent Unit (GRU) [39] and a ResNet, reporting that the performance of the model was better than JPEG. Generative Adversarial Networks (GAN) have been applied in several learning-based image compression models. Early on, Rippel et al. [11] proposed a GAN-based image compression framework that they claim outperformed all existing codecs with respect to MS-SSIM, while being lightweight and deployable. In [12], a GAN framework is presented to build an extreme image compression system which the authors report as achieving state-of-the-art performance, especially at very low bit rates, based on a user study.

2.2. Deep video compression

It is natural to also consider learning-based methods for video compression [19–22,40–43]. Wu et al. [20] proposed a video compression architecture based on the idea that video compression is repeated image compression. They define two types of frames: key frames and other frames. Key frames are compressed using an RNN-based image compression network [5], while the other frames are interpolated in a hierarchical manner. Another hierarchical video compression architecture, called Hierarchical Learned Video Compression (HLVC), was proposed by Yang et al. [22]. In this method, there are three quality layers: an image compression layer, a Bi-Directional Deep Compression (BDDC) layer, and a Single Motion Deep Compression (SMDC) layer. In an attempt to match the pipeline structure of hybrid codecs, Lu et al. proposed an end-to-end video compression model (DVC) [21] that replaces each traditional hybrid component, with deep learning models, then jointly optimized all the components against a single loss function. This work was further extended to two models, a lightweight version called DVC_Lite, and an advanced version called DVC_Pro, by adjusting various components of the architecture. Later, Habibian et al. [40] proposed a deep generative model for video compression using an autoregressive prior to conduct entropy coding. Generally, all learning-based video compression models implement traditional block-based motion estimation or optical flow, both of which have a high computational overhead. The most related work to ours is [43], whereby an interpolation loop is used as an alternative to motion estimation/compensation. However, the frame interpolation network still requires training, which adds to the complexity of the overall method.

2.3. Motion estimation and motion compensation

Motion estimation (ME) and motion compensation (MC) are crucial components in modern hybrid video codecs. These are used to exploit the temporal redundancy of video frames via inter-frame prediction. In traditional hybrid video codecs like H.264 and H.265, video frames are first partitioned into blocks, then motion vectors (MV) associated with each block are estimated with respect to predictions of neighboring reference frames via expensive block search methods, which is the most intensive aspect of video compression. A few deep learning methods have been proposed to solve the ME problem. For example, Choi *et al.*[44] trained a CNN to measure the similarity of pairs of image patches and used this to estimate MVs. However, this method still requires a search process to find the best match. In [45], the authors developed a CNN that was trained to conduct both uni- and bi-directional ME, using separate networks so that motion information need not be transferred from the encoder to the decoder. The CNN does require two frames from the decoded picture buffer and their temporal indices as inputs, which it uses to produce filter coefficients that synthesize patches of a new frame, which is then used to predict the current frame. A drawback of this approach is that it requires the CNN to be resident at both the encoder and the decoder, which reduces decoding efficiency.

Another popular alternative to block matching algorithms are optical flow routines, which seek to obtain a dense vector field mapping the movements of pixel. A variety of deep learning based optical flow estimation methods have been proposed to reduce the computational overhead of dense optical flow vectors [46]. FlowNet [47] showed that it was possible to train a network from two input images to predict optical flow while matching or exceeding the accuracies of traditional methods. Later improvements introduced a stacked architecture that included warping of the second image via intermediate optical flow estimates, and a sub-network specialized to predict small motions [48]. Other approaches have tried to combine networks with traditional methods. Ranjan et al. [46] proposed such a network called SPyNet, which adopted a traditional coarse-to-fine computational hierarchy using a spatial pyramid. Later, another network competitive with FlowNet2 was proposed, called LiteFlowNet [49], but with a significantly decreased model size. Our approach avoids even these methods of deep flow computation, by instead feeding the network a set of directional inter-frame residuals containing adequate information for the network to seek the most efficient perceptual representation.

3. Proposed method

3.1. Framework

Fig. 1 exemplifies the flow of our deep video compression network. A current frame is input to the network, along with multiple displaced frame differences from adjoining, previously coded and then decoded frames (lower part of figure). This is similar to the classic hybrid coding loop, which also includes the decoder as part of the encoder loop, to reduce reconstruction errors. The key components in our network is: Displacement Calculation Unit (DCU), Displacement Compression Network (DCN), and Frame Reconstruction Network (FRN). The details of each key component in our network will be discussed in the following sections.

The flow of our network is: Given an input video with frames $x_1, x_2, \ldots, x_T$, for every frame x_t, displaced frame differences between the current frame x_t and previous reconstructed frame $\hat{x}_{t-1}$ are calculated via the DCU, after which the displaced frame differences d_t are input into the DCN. The DCN compresses the incoming displaced frame differences which are used to capture statistical redundancies. An illustration of displaced frame differences, i.e. differences between spatially displaced frames, is shown in Fig. 2. Given a compressed output $\hat{d}_t$ from the DCN, FRN uses the reconstructed displaced frame differences $\hat{d}_t$ and the reconstructed previous frame $\hat{x}_{t-1}$ to reconstruct a current frame $\hat{x}_t$. Every frame is processed following this except for the first frame. The first frame x_1 is processed differently as it does not have previous reconstructed frame. As a result, an all-zero image is chosen as its previous reconstructed frame and it is otherwise processed the same as other frames. Pseudo code of the flow is shown in Algorithm 1. By using this architecture, we are able to reconstruct the videos without the use of motion.

Algorithm 1 Flow of MOVI-Codec for an Input Video

x_1 to x_T: video frames.
$\hat{x}_0$: previous reconstructed frame for x_1.
d_t, $\hat{d}_t$: displaced frame differences and corresponding reconstructed ones, respectively.
d_1, $\hat{d}_1$: displaced frame differences between x_1 and $\hat{x}_0$, and corresponding reconstructed ones, respectively.

```
procedure MOVI-Codec
    for t in 1 to T do
        if t is 1 then
            x̂_0 = all zero frame
            d_1 ← DCU(x_1, x̂_0)
            d̂_1 ← DCN(d_1)
            x̂_1 ← FRN(d̂_1, x̂_0)
        else
            d_t ← DCU(x_t, x̂_{t−1})
            d̂_t ← DCN(d_t)
            x̂_t ← FRN(d̂_t, x̂_{t−1})
        end if
    end for
end procedure
```

3.2. Displacement Calculation Unit (DCU)

The DCU removes the need for any kind of motion vector search. Instead, it allows the DCU network to learn to optimally represent time-varying images as sets of spatially displaced frame differences.

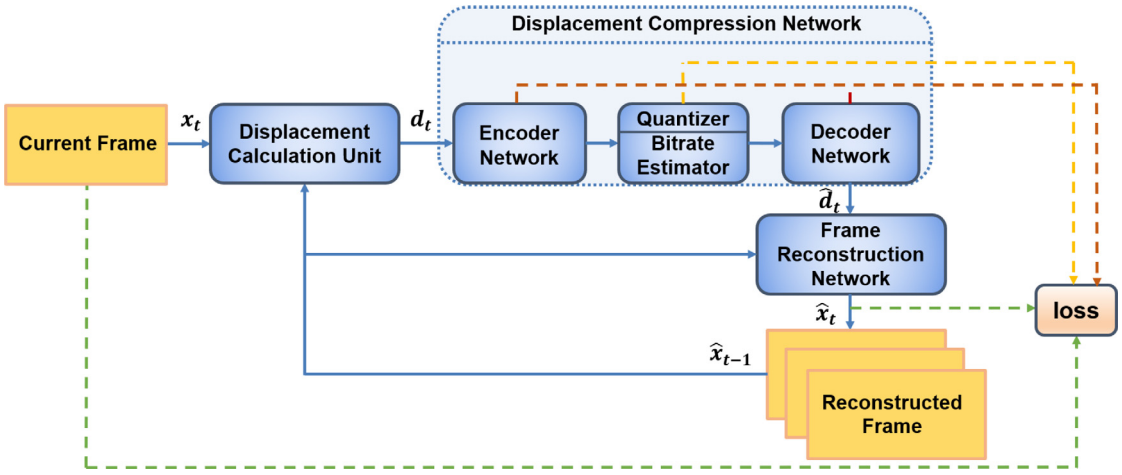


Fig. 1. The overall network architecture of MOVI-Codec, which consists of three components: a Displacement Calculation Unit, a Displacement Compression Network and a Frame Reconstruction Network.

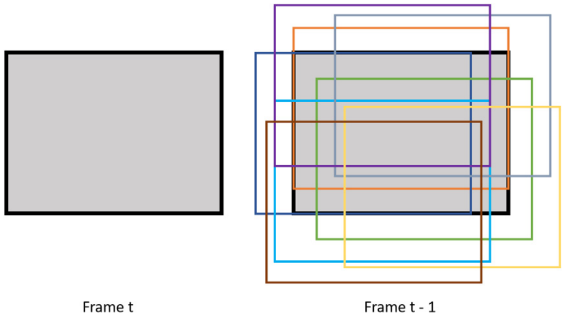


Fig. 2. Concept of displaced frame differences, showing a frame t and previous frame $t-1$, and multiple spatially displaced versions of frame $t-1$.

Given a video with T frames $x_1, x_2, \ldots, x_T$ of width W and height H, two directional (spatially displaced) temporal differences are computed between each pair of adjacent frames, as shown in Fig. 2. In the DCU, the inputs are a current frame x_t and the reconstructed previous frame $\hat{x}_{t-1}$. Then, at each spatial coordinate (i, j), a set of spatially displaced differences is calculated as:

$$d_H(i,j)_t = x_t(i,j) - \hat{x}_{t-1}(i, j-s), \tag{1}$$

$$d_V(i,j)_t = x_t(i,j) - \hat{x}_{t-1}(i-s, j), \tag{2}$$

where $s = 0, \pm3, \pm5, \pm7$ in our experiment. The set of 13 displaced frame differences (residuals) is then fed into the Displacement Compression Network, which delivers as output the reconstructed set of displaced residuals $\hat{d}_t$. As mentioned in Section 2, the statistics of non-displaced frame differences have been observed to be nicely regular. As shown in [24], the statistics of displaced frame differences are also highly regular, and more so in the direction of local motion. This makes them good video representations to learn to exploit space–time redundancies, while avoiding the computational burden of motion estimation and compensation. Although the range of motion between frames can be larger than our largest choice of displacement, larger motions can be captured by various combinations of our set of displacements.

3.3. Displacement Compression Network (DCN)

3.3.1. Framework

After a set of 13 displaced frames are generated from the Displacement Calculation Unit, they are fed into the Displacement Compression Network, where each displacement occupies three channels (RGB), hence the overall input to the DCN comprises 39 channels. The compression network comprises four parts, displacement encoder, displacement decoder, hyper encoder, and hyper decoder. Displacement encoder takes the displaced frame differences calculated from DCU and generates the latent representation y_t using several convolutional layers and convolutional LSTM layers similar to other deep learning-based compression architectures [5,20]. LSTM [50] as a special RNN structure has proven stable and powerful for modeling long-range dependencies in sequence modeling. The major innovation of LSTM is its memory cell which keeps accumulating the state information. As a result, it helps hold the spatio-temporal information provided by displaced frame differences generated by DCU. The hyper autoencoder uses y_t as input to generate side information, which is then used to better compress quantized latent representation $\hat{y}_t$. Finally, the reconstructed $\hat{d}_t$ is generated using $\hat{y}_t$. The detailed processing flow of the hyper autoencoder is explained in later sections.

3.3.2. Quantizer

Traditional quantization inevitably produces zero gradients during backpropagation (BP) which halts network training. Our network deploys BP via stochastic gradient descent, which requires differentiability of all network elements. Hence, we implemented a modified quantizer as in [6], as follows, where $\hat{y}$ is the binarization of the latent representation of displaced frame differences, which lie between -1 and 1, and ϵ represents quantization noise:

$$\hat{y} = y + \epsilon \in -1, 1 \tag{3}$$

$$\epsilon \sim \begin{cases} 1-y & \text{with probability } \frac{1+y}{2} \\ -y-1 & \text{with probability } \frac{1-y}{2}. \end{cases} \tag{4}$$

Following quantization, the size of $\hat{y}$ is $\frac{H}{16} \times \frac{W}{16} \times C$, where H and W are the height and width of the frame, and C is the number of channels of the last convolution layer in the displacement encoder. In our architecture, $C = 128$, as shown in Fig. 3.

3.3.3. Entropy coding

To estimate the entropy of the compressed codes $H(\hat{y})$, where $\hat{y}$ is the quantized latent representation of y, we adopted the hyper-prior scheme proposed by Ballé et al. [6], where they use an additional set of random variables $\hat{z}$ to capture the spatial dependencies and model the latent representations $\hat{y}$ as Gaussian distribution as follows:

$$p_{\hat{y}|\hat{z}}(\hat{y}|\hat{z}) \sim \mathcal{N}(\mu, \sigma), \tag{5}$$

where $p_{\hat{z}}(\hat{z})$ is modeled using the factorized entropy model [8].

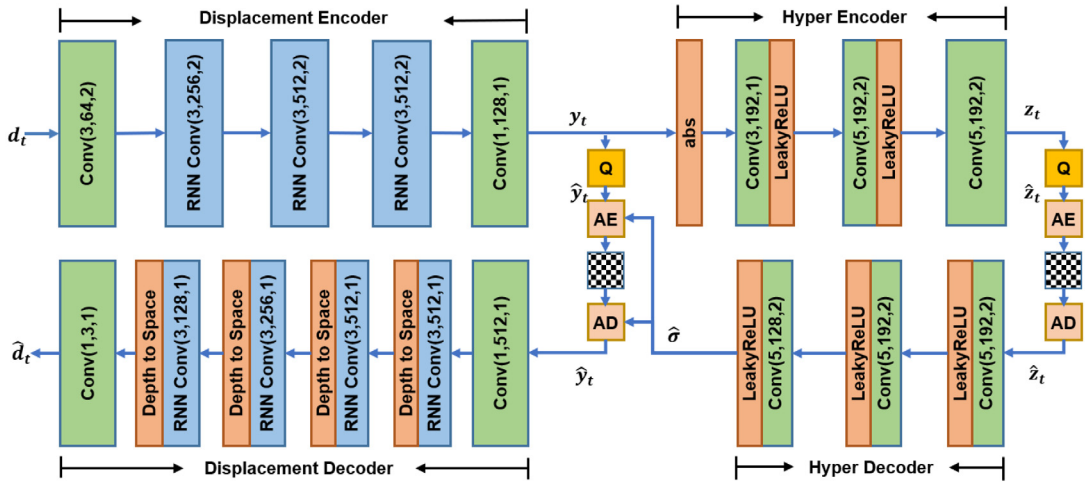


Fig. 3. Flow diagram of the Displacement Compression Network. The left side shows the displacement autoencoder architecture, and the right side corresponds to the hyperprior autoencoder architecture. Q represents quantization, and AE, AD represent arithmetic encoder and arithmetic decoder, respectively. Conv(3,64,2) represents the convolution operation with kernel size of 3 × 3, 64 output channels and a stride of 2.

The hyperprior autoencoder architecture is indicated by Hyper Encoder and Hyper Decoder in Fig. 3, which is responsible for estimating the parameters of the Gaussian model used for entropy coding. After the displacement encoder encoded the input set of displaced frame differences d_t, the resulting latent representation y_t with spatially varying standard deviations is fed into the hyper encoder, which summarizes the distribution of standard deviations in the latent representation z_t. After quantization and arithmetic coding, the quantized $\hat{z}_t$ is transmitted as side information. The hyper decoder uses the quantized $\hat{z}_t$ as input to obtain Gaussian model parameter $\hat{\sigma}$ (μ=0 in our implementation). During modeling training, the Gaussian model parameters can be used to calculate $p_{\hat{y}_t}$ and then estimate $H(\hat{y}_t)$ to guide model optimization. While during model validation and/or testing, the Gaussian model can be used to calculate the cumulative distribution function (CDF) of $\hat{y}_t$ and then guide the arithmetic encoding and decoding of $\hat{y}_t$, which could further losslessly compress $\hat{y}_t$ to bitstream.

3.4. Frame Reconstruction Network (FRN)

Fig. 4 shows the structure of the Frame Reconstruction Network (FRN). The FRN uses the reconstructed displaced frame differences $\hat{d}_t$ and the reconstructed previous frame $\hat{x}_{t-1}$ as the model input to reconstruct the current frame. The architecture of FRN incorporates Convolutional LSTM (C-LSTM) blocks into a UNet architecture. The UNet architecture, which is an encoder–decoder style network with skip connections, makes it possible to extract and represent meaningful descriptors over multiple image scales. However, without modification, the UNet architecture cannot account for temporal relationships between frames of video data, which are deeply relevant to the efficiency of video compression. The C-LSTM is a convolutional version of the original LSTM, which replaces the matrix multiplication operation of the traditional LSTM with convolutions. It is quite useful for analyzing temporal image sequences, where the C-LSTM layers act as a temporal buffer and capture the long-short dependency of previously processed displaced frame differences. By introducing C-LSTM blocks into the UNet architecture, the FRN is able to process evolving frame properties over multiple scales, by relating compact representations of them in the C-LSTM memory units, leading to better reconstructed frame quality and higher compression rates.

3.5. Training strategy

We modeled the loss function considering the rate–distortion trade-off as follows:

$$\begin{aligned} L &= D + \lambda R \\ &= [D_1(x_t, \hat{x}_t) + \beta D_2(d_t, \hat{d}_t)] + \lambda[H(\hat{y}_t) + H(\hat{z}_t)], \end{aligned} \tag{6}$$

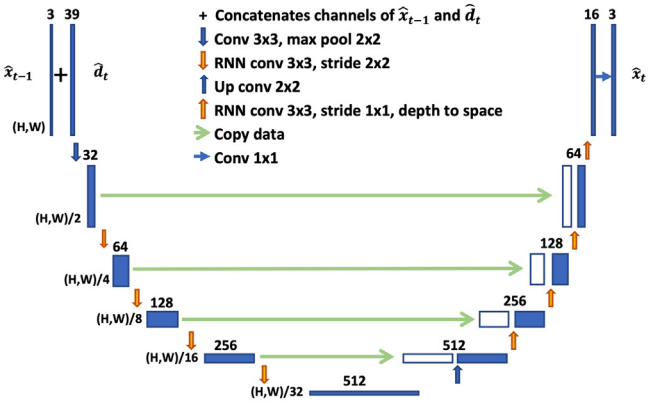


Fig. 4. LSTM-UNet architecture used in Frame Reconstruction Network.

where D and R represent the distortion and rate, respectively. λ controls the trade-off between the number of bits and distortion. D_1 denotes the distortion between the input frame x_t and reconstructed frame $\hat{x}_t$ measured by MS-SSIM or MSE, and D_2 denotes the distortion between displaced frame differences d_t and the reconstructed displaced frame differences $\hat{d}_t$ measured by MSE. β controls the trade-off between the perceptual distortion D_1 and the pixel-to-pixel distortion D_2. $H(\cdot)$ represents the bitrates for encoding the latent representations $\hat{y}$ and $\hat{z}$ estimated by the hyperprior autoencoder.

To leverage multi-frame information using our RNN-based codec structure, we update the network parameters every set of N frames during model training, using the loss function in Eq. (6) but modified as a sum of losses over the kth set of the N frames indexed $x_{t_k+1}, \ldots, x_{t_k+N}$:

$$\begin{aligned} L_k = \frac{1}{N}\sum_{n=1}^{N}&[D_1(x_{t_k+n}, \hat{x}_{t_k+n}) + \beta D_2(d_{t_k+n}, \hat{d}_{t_k+n})] \\ &+ \lambda[H(\hat{y}_{t_k+n}) + H(\hat{z}_{t_k+n})]. \end{aligned} \tag{7}$$

4. Experiments

4.1. Settings

The MOVI-Codec networks were trained end-to-end on the Kinetics-600 dataset [51,52] and the Vimeo-90K dataset [53]. The Kinetics-600

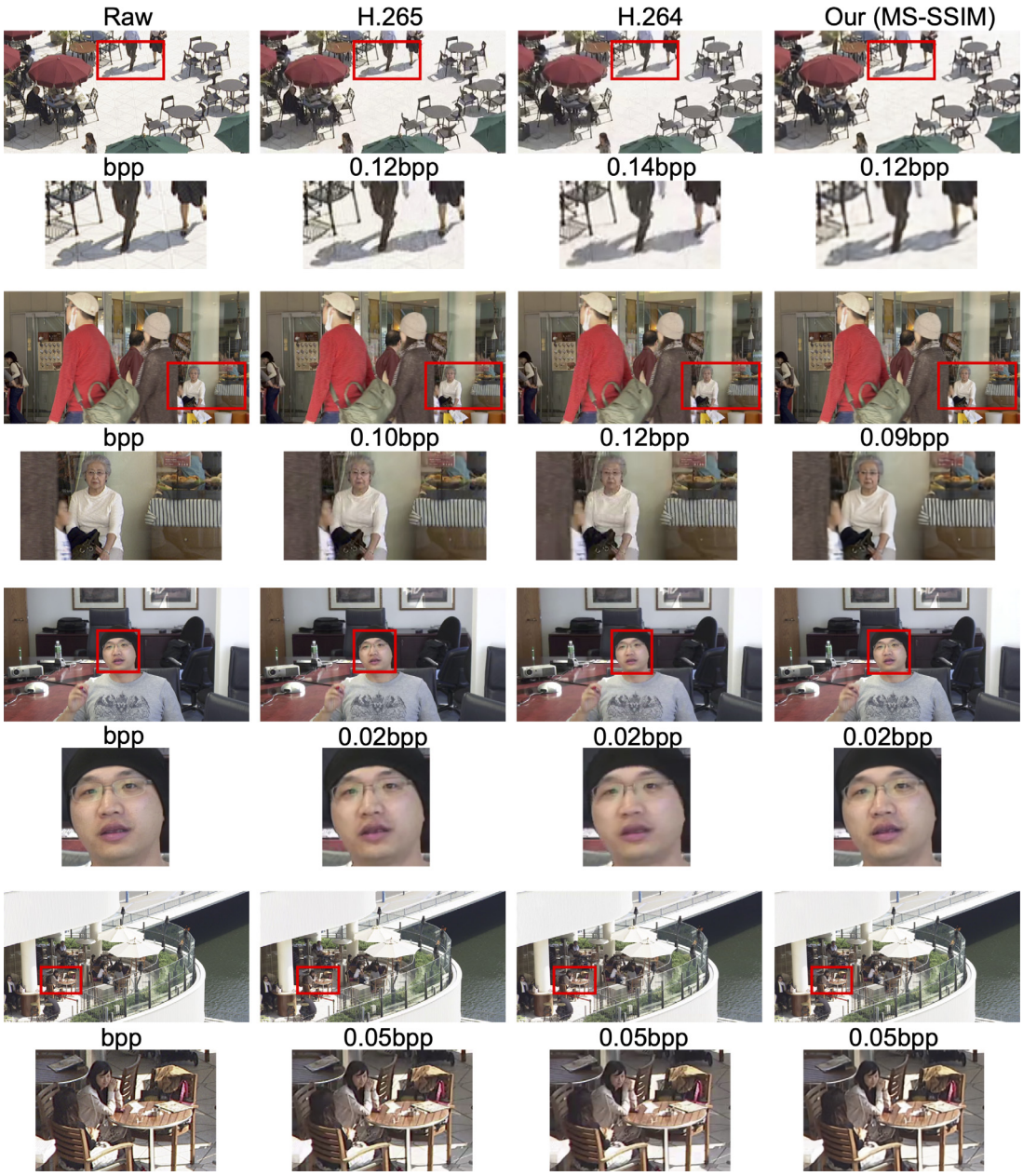


Fig. 5. Visual examples of our method as compared with H.264 and HEVC.

videos are downloaded from YouTube, each video having duration of about 10 s and various resolutions and frame rates. We used part of the testing set from Kinetics-600, which consists of around 10,000 videos, to conduct our experiments. From each video, a random 192 × 192 patch with 49 frames was randomly selected for training, and the values of each input video were normalized to [−1,1]. We also randomly downsampled the original frame to reduce any previously introduced compression artifacts. The Vimeo-90K dataset consists of 4278 videos of fixed resolution 448 × 256. Since the Vimeo-90K dataset has 7 frames per video, we randomly selected a patch of the same size as mentioned before with 7 frames for training. In the Vimeo-90K dataset, the consecutive frames are selected so that the average motion magnitude is between 1–8 pixels, whereas there is no limitation to the motion magnitude between frames in the Kinetics-600 dataset. The mini-batch size is set as 8 for training, and the step length N in our recurrent network is set as 7. By training on both the Vimeo-90K and the Kinetics-600 dataset, we are able to generalize our model to a wide range of natural motions. We tested the MOVI-Codec on the VTL dataset [54], the JCT-VC [55] (Class B, C, D and E) datasets, and the UVG datasets [56]. These datasets cover a variety of resolutions as shown in Table 1. For fair comparison with [21,22,57], we tested our framework on the JCT-VC datasets using the first 100 frames, and tested on VTL and UVG using all frames.

To evaluate the quality of the reconstructed videos, we used two quality models: the perception-based MS-SSIM [58] and the non-perceptual PSNR. Multiscale SSIM (MS-SSIM) is a widely used image quality assessment model which captures local luminance, contrast, and structural information. For each quality metric, we trained 5 models with different values of the weighting parameter λ to cover different bitrate ranges. For the MS-SSIM model, λ was set to 0.01, 0.05, 0.1, 0.5 and 1.0, respectively. For PSNR based models, λ was set to 0.0005, 0.0025, 0.005, 0.025 and 0.05. We fixed $\beta = 1$, since we did not observe any significant differences in model performance as it was varied over the range 0.1 to 10.0.

We compared our method with both traditional and recent deep learning models. H.264 [59], H.265 [60] and the most recent H.266

Table 1
Resolutions of different datasets used for evaluation.

Dataset	VTL	UVG	JCT-VC Class B	JCT-VC Class C	JCT-VC Class D	JCT-VC Class E
Resolution	352 × 288	1920 × 1080	1920 × 1080	832 × 480	416 × 240	1280 × 720

[61] were included as representatives of traditional hybrid compression codecs. We follow [21,22], and used the x264 and x265 "LDP very fast" mode. For H.266, we followed [62] to implement the "faster" mode. So that we could compare against another motion-free method, we also included the H.265 zero motion setting, using x265 with *merange* set to zero, which allows exploiting temporal redundancy using an IB prediction structure but without performing motion estimation. In this regard, this setting is most similar to our architecture [63]. Among recent deep learning models, DVC [21] and Wu et al. [20] are optimized for PSNR, Habibian et al. [40] and Cheng et al. are optimized for MS-SSIM, and HLVC [22] has both MS-SSIM optimized and PSNR optimized results.

4.2. Results

In this section, we compare our video compression engine against the standards H.264, HEVC, and H.266/VVC, and with other deep learning-based video compression architectures (Wu [20], DVC [21, 57], and Cheng [43]) on the UVG dataset, the VTL dataset, and the HEVC Standard Test Sequences (Class B, Class C, Class D, and Class E). When compressing videos using the H.264 and HEVC codecs, we followed the settings in [21] and used FFmpeg with the *very fast* mode.[3] When implementing H.266, we followed [62] using the *faster* mode. We also provide visual examples of our approach against other approaches in Fig. 5. More exemplar reconstructed videos are included on our project page with link given in the Abstract.

Figs. 6, 7, 8, and 9 show the experimental results on the VTL dataset, the UVG dataset, and the HEVC Standard Test Sequences (Class B, Class C, Class D, and Class E). These results show that our network outperformed both H.264 and the HEVC standard against MS-SSIM. On datasets with higher resolution videos (UVG dataset, HEVC Class B dataset, and HEVC Class E dataset), our network was able to outperform the latest H.266 codec at higher bitrates as assessed using the perceptually relevant MS-SSIM algorithm. We also compared our model against several deep learning-based compression models, including a frame interpolation-based model by Wu et al. [20], DVC [57], HLVC [22] and the video compression framework proposed by Cheng et al., which uses an added spatial energy compaction penalty in the loss function [43]. Among these, DVC and HLVC were trained on both PSNR and MS-SSIM, to obtain better results against each metric. In our comparison, we include the best performance for these two methods for each metric. It is worth noting that our model only uses one previous frame as input, whereas in Wu's framework, both neighboring frames are utilized when reconstructing the middle frame. Additionally, our framework replaces the classical motion estimation and compensation module by instead training the network to optimally interpolate displaced frame differences. For completeness, we also evaluated all models against the PSNR, where MOVI-Codec did not always perform as well. However, this is a problem with the PSNR, which is not perceptually relevant, and which produces significantly inferior quality predictions than perception-based quality predictors like MS-SSIM [3]. Indeed, the high quality of the reconstructions that we make

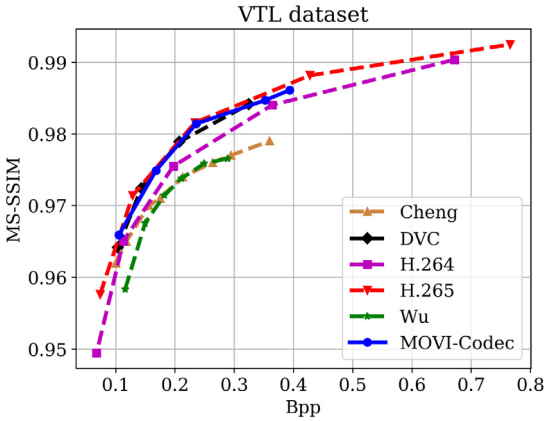


Fig. 6. MS-SSIM on the VTL dataset (352 × 288) for different compression codecs. Our method is competitive with the state of the art over varying bit rates on these low-resolution videos.

available on the model page (see link in Abstract) further attests to this. As has been observed by others [8,11,12,41], perceptual measures are better arbiters of deep compressed video quality than absolute fidelity models like the PSNR. It is worth noting that when comparing our model against the H.265 zero motion setting, while both methods do not utilize motion estimation, MOVI-Codec was able to perform better with respect to MS-SSIM than the H.265 zero motion setting, while delivering similar performance against PSNR.

4.3. Ablation studies

We conducted ablation studies to assess the choices we made in our approach, specifically with respect to the choice of displaced frame differences, and the effectiveness of the proposed LSTM-UNet. The results are shown in Figs. 10 and 11.

4.3.1. Displaced frame difference combination

Fig. 10 shows the experimental results on different combinations of displaced frame differences, where $s = 0$ refers to frame differences with no displacements, which gives the worst performance of all combinations evaluated. This shows the value of "displaced" frame differences as a way of training the network on more diverse motion induced displacements. Including displacements as large as $s = 7$ greatly increases the overall performance, by allowing interpolation of larger motions in videos. We also tried adding $s = 9$ to our choice of displacement combinations, but this new combination did not improve the overall performance, meaning that our combination of displacements was adequate to capture motions of various sizes. It is worth noting that as compared with the H.265 zero motion configuration, which also does not utilize motion estimation, our network was able to perform better as assessed by the perceptually relevant MS-SSIM, including when $s = 0$.

4.3.2. Effectiveness of the LSTM-UNet

Fig. 11 shows the experimental results on the HEVC Class B dataset when using UNet and LSTM-UNet to reconstruct frames, respectively. As shown in the example, LSTM-UNet extends the advantage of UNet for extracting and representing spatial descriptors to include spatio-temporal descriptors using C-LSTM blocks, yielding better reconstruction performance. In addition, LSTM-UNet converges faster than the UNet counterparts, shortening the training time of the network.

[3] H.264: *ffmpeg -pix_fmt yuv420p -s WxH -r FR -i Video.yuv -vframes N -c:v libx264 -preset veryfast -tune zerolatency -crf Q -g GOP -bf 2 -b_strategy 0 -sc_threshold 0 output.mkv*

H.265: *ffmpeg -pix_fmt yuv420p -s WxH -r FR -i Video.yuv -vframes N -c:v libx265 -preset veryfast -tune zerolatency -x265-params "crf=Q:keyint=GOP" output.mkv*

FR, N, Q, GOP represents the frame rate, the number of encoded frames, quality, GOP size, respectively. *N* is set to 100 for HEVC datasets.

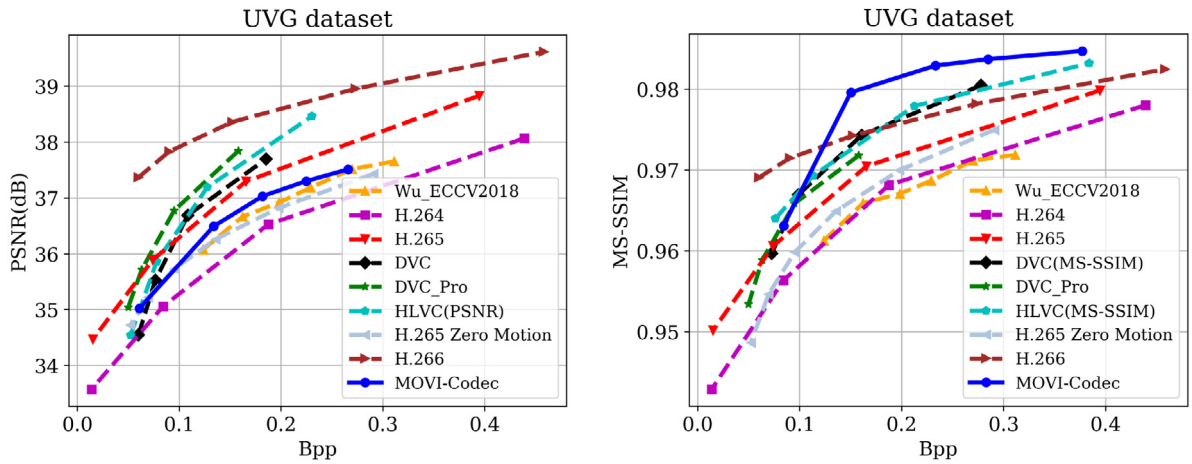


Fig. 7. PSNR and MS-SSIM on the UVG dataset (1920 × 1080) for different compression codecs. Our method outperformed the latest H.266 against the perceptually relevant MS-SSIM on high bitrates and all others on all bitrates tested, while remaining highly competitive against the non-perceptual PSNR.

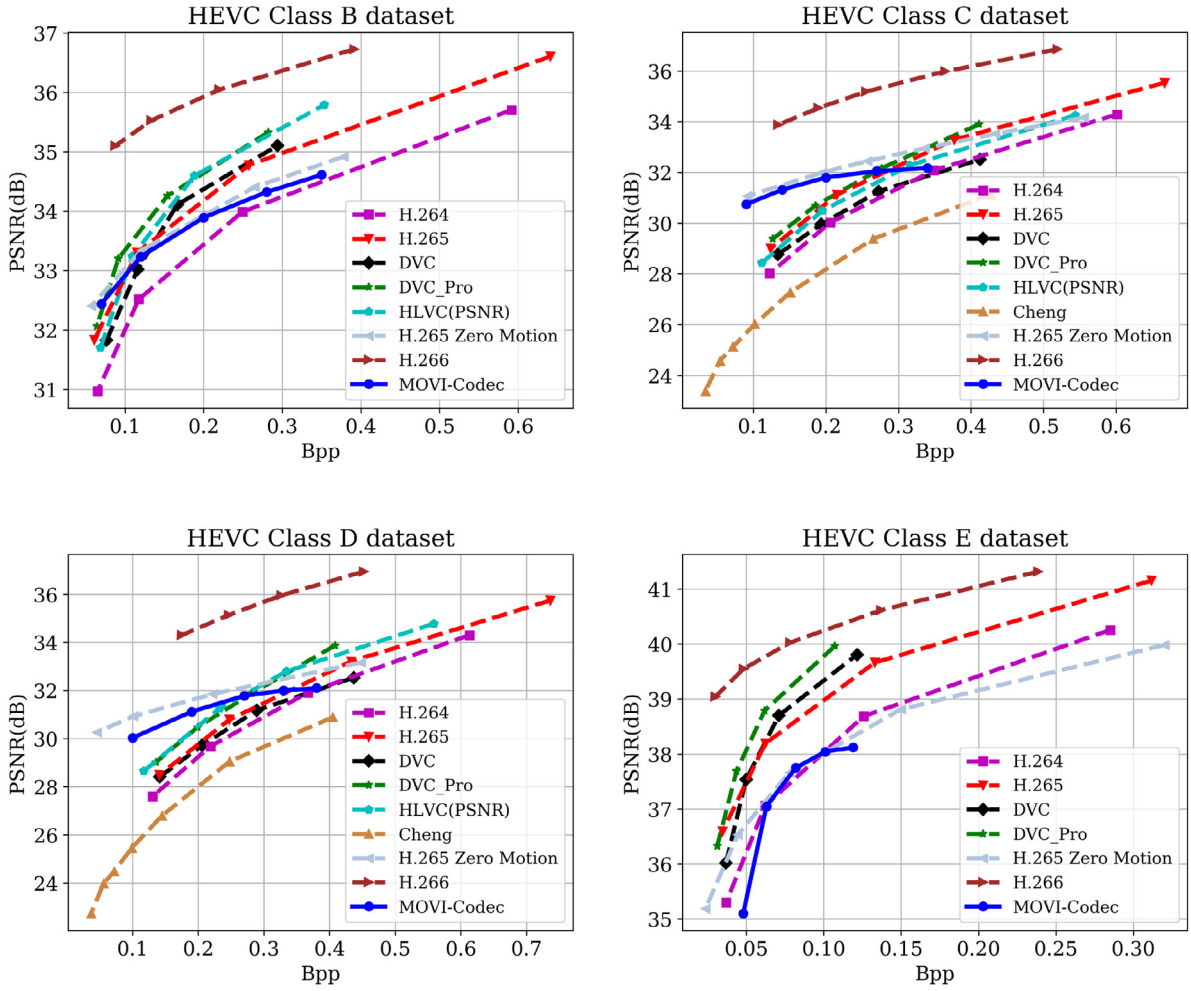


Fig. 8. PSNR of HEVC test sequences for different compression codecs. The resolution of Class B is 1920 × 1080, of Class C is 832 × 480, of Class D is 416 × 240 and of Class E is 1280 × 720. Overall, our method is competitive with H.265, and is particularity good at lower bit rates on lower resolution datasets.

4.4. Motion vector analysis

To verify that MOVI-Codec can capture large motions with the chosen set of displacement combination, we calculated the optical flow of adjacent frames in the testing datasets using a pre-trained network called SPyNet [46]. To emphasize large motions, we calculated all motion vectors against adjacent frames, and only picked the minimum and maximum motion vectors in the x and y directions. As a result, we ended up with four values of motion vectors for each adjacent frames. Fig. 12 shows the distribution of the picked motion vectors on all videos

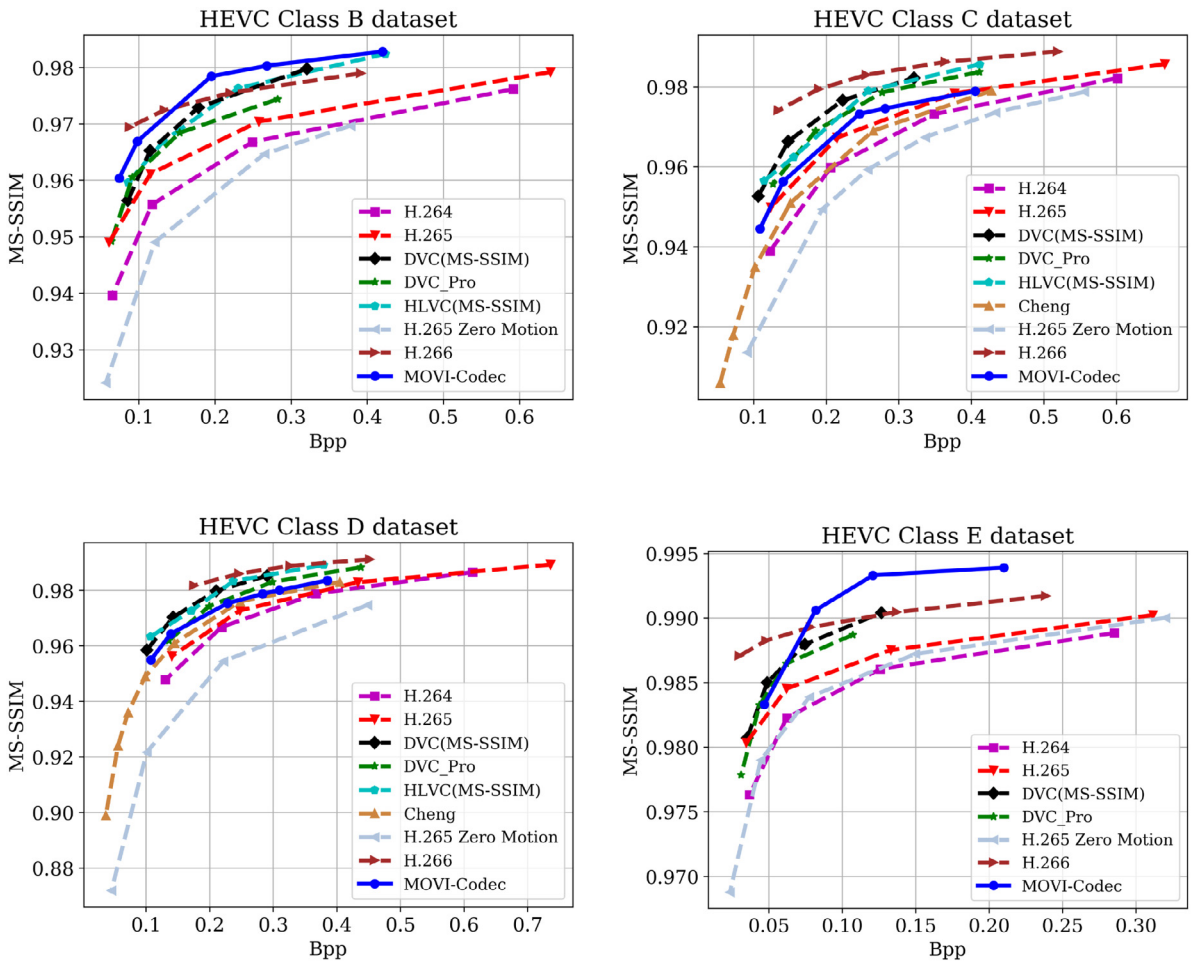


Fig. 9. MS-SSIM of HEVC test sequences for different compression codecs, where the resolution of Class B is 1920 × 1080, of Class C is 832 × 480, of Class D is 416 × 240 and of Class E is 1280 × 720. Our method outperformed H.265 and is competitive with other state of the art deep learning models.

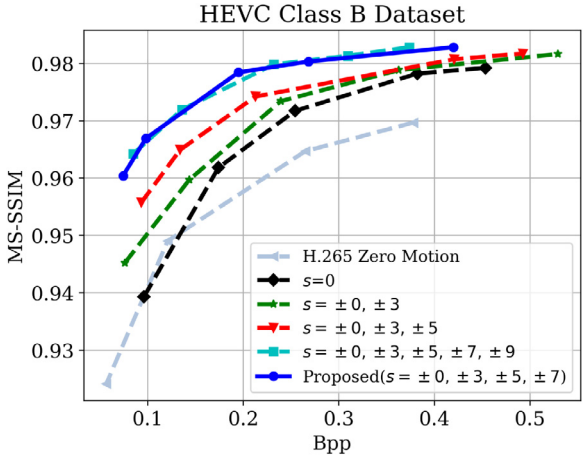


Fig. 10. Ablation study of displaced frame difference combinations.

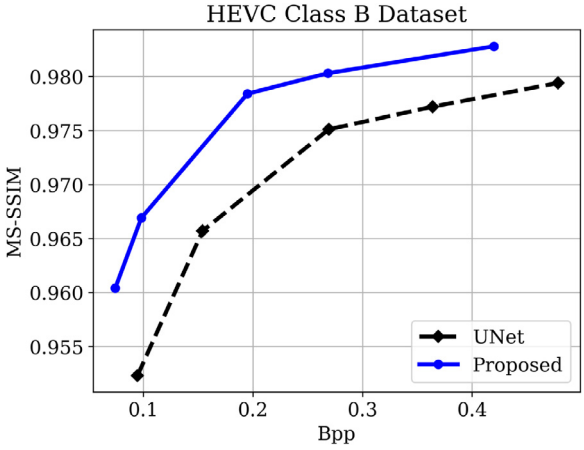


Fig. 11. Ablation study of the effectiveness of the proposed LSTM-UNet.

in the HEVC Class B dataset, which is the dataset having the highest resolution videos among our testing datasets. From the figure, we can conclude that our model produced a similar distribution as the original frame pairs, hence our model was able to capture large motions using a set of small displacements.

Figs. 13 and 14 illustrate the accuracy of our motion reconstruction. The test video in Fig. 13 shows the x axis optical flow between two adjacent frames from the Kimono video, which is a video with a moving background and slow motion, whereas Fig. 14 shows the optical flow images of two adjacent frames in Basketball Drive video, which has a

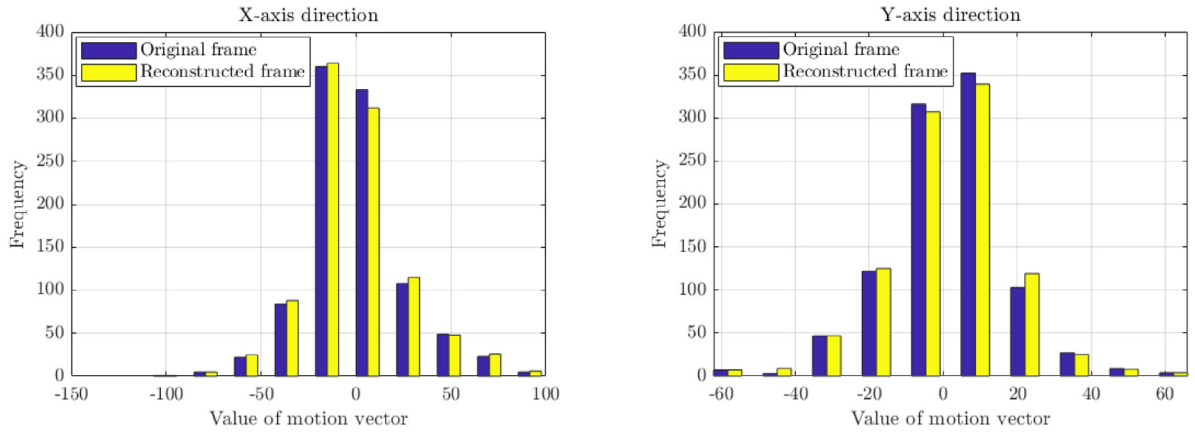
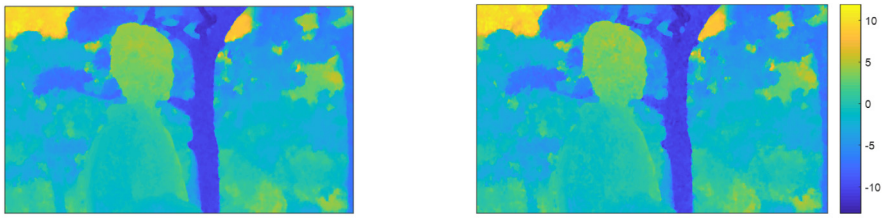
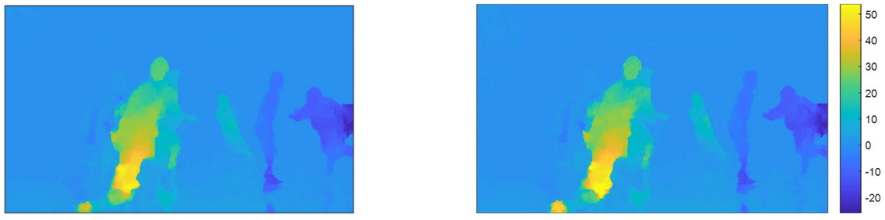


Fig. 12. Distributions of the maximum and minimum motion vector components along the horizontal (left) and vertical (right) axes of the HEVC Class B dataset.



(a) Original frame (b) Reconstructed frame

Fig. 13. Optical flow along the horizontal direction between two adjacent frames in the Kimono video.



(a) Original frame (b) Reconstructed frame

Fig. 14. Optical flow along the horizontal direction between two adjacent frames in the Basketball Drive video.

static background and large motions. In both videos, our model was able to reconstruct motion accurately.

4.5. Model analysis

To compare the computational complexity of the different codecs, we tested two deep learning models: the one proposed by Wu et al. [20], the light version of DVC called DVC Lite [57], and the commercial software x265 for H.265 compression, using a server with an Intel Core i9-9940X CPU and GTX 1080Ti on video sequences of resolution 1920×1080. The experimental results are provided in Fig. 15.

The overall encoding speed of our framework is mostly invariant of bitrate, whereas since Wu's framework adopts a progressive coding scheme, its encoding speed varies with the target bitrate. In our framework, although we adopted an RNN-based compression method on displaced frame differences, we utilized the RNN unit to store temporal dependencies and did not use a progressive coding scheme for compression. DVC Lite is a lightweight version of DVC with a more efficient motion estimation module and a lightweight motion compression network, which can be twice as fast as the original DVC model in terms of encoding speed [57]. Our framework is faster than the lightweight model, further justifying the use of learned interpolation of displaced frame differences. Since the arithmetic coding at lower bitrates is faster than at larger ones, there is a slight slope to our encoding speed curve. But overall, the complexity of our model is invariant to bitrate, which means that our model maintains a stable encoding speed regardless of video content or bitrate for a given resolution.

As shown in Fig. 15, compared with the traditional hybrid codec, our model is faster than the latest codec HEVC with *slower* setting. However, using the *very fast* setting on x264 and x265, the encoding speed can run at 110 fps and 30 fps, respectively. Of course, by applying model acceleration techniques such as model distillation, model quantization, or by decreasing the model size, it should be possible to similarly accelerate the encoding speed of our framework.

4.6. Discussion

From Figs. 6, 7, 8, 9, and 15, we can conclude that our model delivers better compression performance than LDP *veryfast* setting of

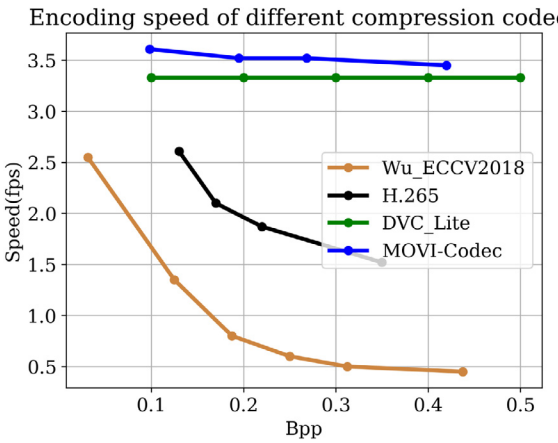


Fig. 15. Encoding speed of different compression codecs. H.265 refers to the encoding speed of the x265 codec *slower* setting.

traditional hybrid codecs like H.264 and HEVC in terms of MS-SSIM, at a low computational complexity. This justifies our use of displaced frame differences as motion information for video compression. Although our model was able to achieve competitive performances as lower settings of traditional codec having low computational complexity, and without the complicated motion estimation and compensation modules other deep learning-based models use, our model did not outperform all of the state-of-the-art models. Nonetheless, the performance achieved by our model provides a new way of motion computation that may prove quite useful for video compression. In our model, we designed the set of spatial displacements used by our network to cover a reasonable range of natural motions. A promising future direction is to automatically assign displacement combinations as a function of resolution. The encoding speed of our model is state-of-the-art among deep learning models, but has not yet been optimized to match compute-optimized traditional codecs like HEVC or VVC, e.g. by model acceleration methods.

5. Conclusion and future work

In this paper, we proposed an end-to-end deep learning video compression framework that renovated motion prediction. To be specific, we proposed the use of displaced frame differences as indicators of motion information, and fed them into a deep space–time compression network, which learns optimal between-frame interpolated representations to achieve efficiency. Additionally, we proposed a new version of UNet, called LSTM-UNet, that utilizes both spatial and temporal information to conduct frame reconstruction. Our experimental results show that our approach outperforms the LDP *veryfast* setting of the standard codecs H.264 and H.265 in terms of MS-SSIM. In addition, our network was able to outperform the latest H.266 codec at higher bitrates as assessed by the perceptual MS-SSIM algorithm, on high resolution videos. The reduced complexity of the framework and the avoidance of motion search could make it easier to implement on resource-limited devices, such as smartphones, VR headsets, and AR glasses.

CRediT authorship contribution statement

Meixu Chen: Writing – original draft, Software, Investigation, Formal analysis. **Todd Goodall:** Conceptualization. **Anjul Patney:** Methodology, Writing – review & editing. **Alan C. Bovik:** Supervision, Conceptualization, Methodology, Review and editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by a grant by Facebook, USA and by grant number 2019844 for the National Science Foundation AI Institute for Foundations of Machine Learning (IFML), USA. The authors would like to thank Fabian Mentzer and Ren Yang for insightful discussions about this work.

References

[1] C.V.N. Index, Cisco visual networking index: Forecast and methodology, 2016–2021, Complet. Vis. Netw. Index (VNI) Forecast 12 (1) (2017) 749–759.
[2] Cisco, Cisco annual internet report (2018–2023) white paper, 2020, https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.
[3] Z. Wang, A.C. Bovik, Mean squared error: Love it or leave it? A new look at signal fidelity measures, IEEE Signal Process. Mag. 26 (1) (2009) 98–117.
[4] 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, 2015, arXiv preprint arXiv:1511.06085.
[5] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, M. Covell, Full resolution image compression with recurrent neural networks, in: IEEE Conference On Computer Vision And Pattern Recognition, 2017, pp. 5306–5314.
[6] J. Ballé, D. Minnen, S. Singh, S.J. Hwang, N. Johnston, Variational image compression with a scale hyperprior, in: International Conference On Learning Representations, 2018.
[7] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, L.V. Gool, Soft-to-hard vector quantization for end-to-end learning compressible representations, in: Advances In Neural Information Processing Systems, 2017, pp. 1141–1151.
[8] J. Ballé, V. Laparra, E. Simoncelli, End-to-end optimized image compression, in: International Conference On Learning Representations, 2019.
[9] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. Chinen, S. Jin Hwang, J. Shor, G. Toderici, Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks, in: IEEE Conference On Computer Vision And Pattern Recognition, 2018, pp. 4385–4393.
[10] L. Theis, W. Shi, A. Cunningham, F. Huszár, Lossy image compression with compressive autoencoders, Int. Conf. Learn. Representations (2017).
[11] O. Rippel, L. Bourdev, Real-time adaptive image compression, in: International Conference On Machine Learning, 2017, pp. 2922–2930.
[12] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, L. Van Gool, Generative adversarial networks for extreme learned image compression, in: IEEE/CVF International Conference On Computer Vision, ICCV, IEEE, pp. 221–231.
[13] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, L. Van Gool, Conditional probability models for deep image compression, in: IEEE Conference On Computer Vision And Pattern Recognition, 2018, pp. 4394–4402.
[14] D. Minnen, J. Ballé, G.D. Toderici, Joint autoregressive and hierarchical priors for learned image compression, in: Advances In Neural Information Processing Systems, 2018, pp. 10771–10780.
[15] Y. Patel, S. Appalaraju, R. Manmatha, Deep perceptual compression, 2019, arXiv preprint arXiv:1907.08310.
[16] J. Lee, S. Cho, S.-K. Beack, Context-adaptive entropy model for end-to-end optimized image compression, 2018, arXiv preprint arXiv:1809.10452.
[17] Y. Blau, T. Michaeli, Rethinking lossy compression: The rate-distortion-perception tradeoff, in: International Conference On Machine Learning, 2019, pp. 675–685.
[18] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, L. Van Gool, Practical full resolution learned lossless image compression, in: 2019 IEEE/CVF Conference On Computer Vision And Pattern Recognition, CVPR, 2019, pp. 10621–10630.
[19] 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.
[20] 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.
[21] 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 Conference On Computer Vision And Pattern Recognition, 2019, pp. 11006–11015.
[22] R. Yang, F. Mentzer, L. Van Gool, R. Timofte, Learning for video compression with hierarchical quality and recurrent enhancement, 2020, arXiv preprint arXiv:2003.01966.

[23] R. Soundararajan, A.C. Bovik, Video quality assessment by reduced reference spatio-temporal entropic differencing, IEEE Trans. Circuits Syst. Video Technol. 23 (4) (2012) 684–694.
[24] D.Y. Lee, H. Ko, J. Kim, A.C. Bovik, On the space-time statistics of motion pictures, J. Opt. Soc. Amer. A 38 (7) (2021) 908–923.
[25] J.J. Atick, A.N. Redlich, Towards a theory of early visual processing, Neural Comput. 2 (3) (1990) 308–320.
[26] F. Attneave, Some informational aspects of visual perception, Psychol. Rev. 61 (3) (1954) 183.
[27] D.W. Dong, J.J. Atick, Temporal decorrelation: a theory of lagged and nonlagged responses in the lateral geniculate nucleus, Netw.: Comput. Neural Syst. 6 (2) (1995) 159–178.
[28] M. Rucci, J.D. Victor, The unsteady eye: an information-processing stage, not a bug, Trends Neurosci. 38 (4) (2015) 195–206.
[29] E. Chichilnisky, R.S. Kalmar, Functional asymmetries in ON and OFF ganglion cells of primate retina, J. Neurosci. 22 (7) (2002) 2737–2747.
[30] R. Engbert, Microsaccades: A microcosm for research on oculomotor control, attention, and visual perception, Progress Brain Res. 154 (2006) 177–192.
[31] M. Poletti, M. Rucci, A compact field guide to the study of microsaccades: Challenges and functions, Vis. Res. 118 (2016) 83–97.
[32] B.A. Olshausen, D.J. Field, Emergence of simple-cell receptive field properties by learning a sparse code for natural images, Nature 381 (6583) (1996) 607–609.
[33] G.K. Wallace, The JPEG still picture compression standard, IEEE Trans. Consumer Electron. 38 (1) (1992) xviii–xxxiv.
[34] A. Skodras, C. Christopoulos, T. Ebrahimi, The JPEG 2000 still image compression standard, IEEE Signal Process. Mag. 18 (5) (2001) 36–58.
[35] F. Bellard, BPG image format, 2015, https://bellard.org/bpg.
[36] D. Mukherjee, J. Bankoski, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, R. Bultje, The latest open-source video codec VP9-an overview and preliminary results, in: Picture Coding Symposium, PCS, 2013, pp. 390–393.
[37] Y. LeCun, Y. Bengio, et al., Convolutional networks for images, speech, and time series, in: The Handbook Of Brain Theory And Neural Networks, Vol. 3361, no. 10, 1995, p. 1995.
[38] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[39] K. Cho, B. van Merriënboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: Encoder–decoder approaches, Syntax, Semant. Struct. Stat. Transl. (2014) 103.
[40] A. Habibian, T.v. Rozendaal, J.M. Tomczak, T.S. Cohen, Video compression with rate-distortion autoencoders, IEEE Int. Conf. Comput. Vis. (2019) 7033–7042.
[41] O. Rippel, S. Nair, C. Lew, S. Branson, A.G. Anderson, L. Bourdev, Learned video compression, in: IEEE International Conference On Computer Vision, 2019, pp. 3454–3463.
[42] Z. Chen, T. He, X. Jin, F. Wu, Learning for video compression, IEEE Trans. Circ. Syst. Video Technol. 30 (2) (2019) 566–576.
[43] Z. Cheng, H. Sun, M. Takeuchi, J. Katto, Learning image and video compression through spatial-temporal energy compaction, in: IEEE Conference On Computer Vision And Pattern Recognition, 2019, pp. 10071–10080.
[44] G. Choi, P. Heo, S.R. Oh, H. Park, A new motion estimation method for motion-compensated frame interpolation using a convolutional neural network, in: IEEE International Conference On Image Processing, ICIP, 2017, pp. 800–804.
[45] H. Choi, I.V. Bajić, Deep frame prediction for video coding, IEEE Trans. Circ. Syst. Video Technol. (2019).
[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: IEEE International Conference On Computer Vision, 2015, pp. 2758–2766.
[48] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, T. Brox, Flownet 2.0: Evolution of optical flow estimation with deep networks, in: IEEE Conference On Computer Vision And Pattern Recognition, 2017, pp. 2462–2470.
[49] T.-W. Hui, X. Tang, C. Change Loy, Liteflownet: A lightweight convolutional neural network for optical flow estimation, in: IEEE Conference On Computer Vision And Pattern Recognition, 2018, pp. 8981–8989.
[50] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, W.-c. Woo, Convolutional LSTM network: A machine learning approach for precipitation nowcasting, in: Advances In Neural Information Processing Systems, 2015, pp. 802–810.
[51] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al., The kinetics human action video dataset, 2017, arXiv preprint arXiv:1705.06950.
[52] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier, A. Zisserman, A short note about kinetics-600, 2018, arXiv preprint arXiv:1808.01340.
[53] T. Xue, B. Chen, J. Wu, D. Wei, W.T. Freeman, Video enhancement with task-oriented flow, Int. J. Comput. Vis. 127 (8) (2019) 1106–1125.
[54] V.T. Library, VTL test sequences, 2020, http://trace.eas.asu.edu/index.html.
[55] F. Bossen, et al., Common test conditions and software reference configurations, JCTVC-L1100 12 (2013), 7.
[56] Ultra Video Group, UVG test sequences, 2020, http://ultravideo.cs.tut.fi/#testsequences.
[57] G. Lu, X. Zhang, W. Ouyang, L. Chen, Z. Gao, D. Xu, An end-to-end learning framework for video compression, IEEE Trans. Pattern Anal. Mach. Intell. (2020).
[58] Z. Wang, E.P. Simoncelli, A.C. Bovik, Multiscale structural similarity for image quality assessment, in: Asilomar Conf. Signals Syst. Comput. Vol. 2, Nov. 2003, pp. 1398–1402.
[59] T. Wiegand, G.J. Sullivan, G. Bjontegaard, A. Luthra, Overview of the H.264/AVC video coding standard, IEEE Trans. Circ. Syst. Video Technol. 13 (7) (2003) 560–576.
[60] G.J. Sullivan, J.-R. Ohm, W.-J. Han, T. Wiegand, Overview of the high efficiency video coding (HEVC) standard, IEEE Trans. Circ. Syst. Video Technol. 22 (12) (2012) 1649–1668.
[61] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G.J. Sullivan, J.-R. Ohm, Overview of the versatile video coding (VVC) standard and its applications, IEEE Trans. Circ. Syst. Video Technol. 31 (10) (2021) 3736–3764.
[62] Fraunhofer Heinrich Hertz Institute, Fraunhofer versatile video encoder (vvenc), 2021, https://www.hhi.fraunhofer.de/en/departments/vca/technologies-and-solutions/h266-vvc.html.
[63] C. Brites, F. Pereira, Distributed video coding: Assessing the HEVC upgrade, Signal Process., Image Commun. 32 (2015) 81–105.