Multimedia Tools and Applications (2022) 81:42783–42804
https://doi.org/10.1007/s11042-022-13480-0

1221: DEEP LEARNING FOR IMAGE/VIDEO COMPRESSION AND VISUAL QUALITY ASSESSMENT

Flow-MotionNet: A neural network based video compression architecture

Sangeeta Yadav[1] · Preeti Gulia[1] · Nasib Singh Gill[1]

Received: 21 February 2021 / Revised: 27 January 2022 / Accepted: 13 July 2022 /
Published online: 10 August 2022

Abstract

The growth of superfluous video content over the internet led to the emergence of highly proficient video compression techniques. These novel techniques make optimal use of the available varying bandwidths to deliver quality video content. The traditional techniques of video compression are mainly based on block designs and remove the redundancies using Discrete Cosine Transforms. Although these techniques perform well but these are not adaptive to the varying bandwidth. A number of learning based video compression schemes have been developed during previous years. Though some are performing efficiently but these are not adaptable for mobile usage because of their flexibility lack for varying reconstruction quality with varying bandwidth. In this paper, a lightweight learning-based video compression architecture has been proposed that attempts to allow variation in quality of the reconstructed video with the amount of data sent, without requiring separate low-resolution versions of the same video. The proposed model is a amalgamation of three tiny networks namely frame autoencoder, flow autoencoder and motion extension network. The performance analysis reveals a significant improvement in visual quality of the video frames but in tradeoff with frame reconstruction time. The results have also been compared to some state-of-the-art techniques including H.264 in terms of SSIM and PSNR.

Keywords Video · Compression · Deep learning · CNN · ConvGRU · SSIM · PSNR

✉ Sangeeta Yadav
sangirao5228@gmail.com

Preeti Gulia
preeti.gulia81@gmail.com

Nasib Singh Gill
nasibsgill@gmail.com

[1] Department of Computer Science & Applications, Maharshi Dayanand University, Rohtak, India

1 Introduction

The voluminous video content over the network lead to the emergence of more powerful and efficient codecs. The increase in image or video quality standards put an extra challenge before the compression standards to retain the good visual perception with better compression rate. The compression strategies reduce the size of frames by eliminating the temporal and spatial redundancies with an aim to achieve better compression rate while retaining the perceptual quality of frames. The widely used traditionally designed codecs like H.264 or HEVC primarily focuses on reducing rate distortion error. The surging demand of video streaming looks for more efficient video storage, transmission and retrieval techniques.

The impressive results of deep learning based image compression fueled the further researchers in the application of deep learning in video compression. Several DNN based image compression networks were later extended for videos as well by incorporating motion estimation and prediction strategies. End to end trainable autoencoder styled fully convolutional networks were also proposed for video compression and some are resulted in promising outputs as compared to state-of-the-art methods. Pure deep learning based compression techniques are end to end trainable and optimizable.

The proposed video compression architecture, as motivated by the deep learning based approaches, comprises of basically three small networks i.e. frame autoencoder for frame compression, flow autoencoder for compression of estimated flow and motion extension network for reconstruction of current frame. The frame is reconstructed based on previous frame, current frame and its corresponding flow value. The results have been compared to some state-of-the-art techniques including H.264 in terms of SSIM and PSNR.

In this paper, the detailed description of the work is presented in different sections. Section 2 describes the related work done in the field of video compression using differnet approaches. Section 3 presents the proposed architecure and its various components have been described in the sub sections. Section 4 deals with the experimental setup, experimental analysis, its comparison with state-of-the-art methods and ablation study respectively. Lastly Section 5 concludes the whole work.

2 Related work

The video compression techniques encompass a pair of encoder and decoder. The video frames are compressed by the encoder and reconstructed by the corresponding decoder. They both together form a codec. The codec primarily reduces the number of bits for representation but the reduction in number of bits usually results in reconstruction error. The initial video codes like motion JPEG based on individual compression of video frames, hence based on image compression.

The emerging field of deep learning has a profound effect on image compression and resulted in good quality compression [3, 13, 26, 29, 30]. These techniques mainly focus on reducing the distortion error by training the autoencoder network with a suitable training set [13, 26, 30]. Some of the autoencoder based networks made use of RNNs [2, 13, 30] to achieve variable compression rate from the same network. The same concept later extended to videos as well. Fully Convolutional networks are used for the applicability of the network to the variable frame sizes. These models made use of entropy coding to eliminate the spatially redundant information [3, 23, 26, 29, 30]. Some of the advanced models like Pixel-CNN are based on probability driven adaptive arithmetic coding [25]. Different techniques of

quantization are used to learn the binary format. Stochastic binarization has been used in the model proposed by Toderici et al. Quantization based on soft assignment also been implemented by Agustsson et al. [1]. All such models work in the same manner. Conclusively, it was found that deep learning based image compression techniques gave good compression rate with better image quality when compared to the traditional techniques like WebP [33] or JPEG. Image interpolation is a popular and powerful technique used for image compression which mainly employs a encoder-decoder network to predict the intermediate frame between two reference frames [11, 12, 19, 24]. Image extrapolation also resulted better in predicting the unseen frames [22, 31, 34]. But both these techniques are limited to small slow-motion videos.

The traditional video codes like H.264 or HEVC [15] are primarily manually designed and evolved with frequent developments. Their work strategy mainly relies on isolating the video clip into I or Image frames and referencing, P or B frames, and compressing I frames directly with prediction of referencing frames. These traditional techniques, though giving good results but as they evolved with incremental developments, their end to end optimization is not possible. A lot of machine learning based enhancements to the traditional codecs were proposed to improve the compression quality and efficiency. Some of the popular techniques are intra– prediction coding [10] and motion compensation and interpolation [21, 27, 35, 39]. Rate control and post processing refinement like approaches also lead to the improved results [4, 9, 17, 18, 28, 32, 36–38].

Encoding complexity is one of the major challenge in traditional codecs like HEVC. To address this challenge, several different techniques have been proposed [7, 8, 16]. In [8], a significant encoding complexity improvement has been achieved by proposing a new technique for non-skipped coding blocks. This technique employs entropy value for early non-Asymmetric Motion Partitioning detection and resulted in about 40% encoding time reduction with almost same video quality. Another promising technique for reducing encoder complexity has been proposed comprising of motion activity classification and early PU decision. In this technique, depth correlations and spatio temporal analyses have been used to make early PU decision in each CU level [16]. In addition to these developments in traditional codecs, a surge in development of pure deep learning based video architectures and frameworks have been observed. DeepCoder, a pure CNN based video compression architecture has been proposed. This network is based on encoding the quantized feature maps into binary stream using Scalar quantization and Huffman coding [5]. This model achieves limited efficiency but presents huge potential for further explorations. Several different deep learning based techniques for video compression has been presented in [6].

Most of the popular techniques make use of optical flow based motion estimation to extract and represent the motion information. A lot of optical flow estimation techniques including both traditional differential equations based and machine learning based are invented. Motivated by the same developments in the field of deep learning based video compression, the proposed work architecture in this paper comprises of flow driven reconstruction of frames. In this paper, Frame AutoEncoder Network module is utilized for better image generation for available current data. The previous reconstructed frame information is utilized in Motion Extension Network. This method is chosen to have better modularity in Network architecture.

3 Proposed architecture

The architecture is an amalgamation of three separate networks. A frame compression/decompression network, a flow vector compression/decompression network, and finally a

motion extension/frame reconstruction network. The frame and flow compression/decompression networks are composed of encoder and decoder networks. The relationship between the modules is as follows:

Step 1: Frame Compression
We use a recurrent ConvGRU based encoder model to encode the frame with varying degrees of compression quality.

Step 2: Flow Vector Estimation
This is done using the traditional Farneback flow estimation method. The flow vectors between every two frames are estimated.

Step 3: Flow Vector Compression
The estimated flow vectors from step 2 are compressed using a standard CNN based encoder network with Generalized Divisive Normalization (GDN) layers as the nonlinearity.

Step 4: Frame Decompression
A ConvGRU based decoder model is used to decompress the encoded videos from step 1 according to the degree of compression.

Step 5: Flow Vector Decompression
A CNN based decoder network with Inverse GDN as the nonlinearity is used to decompress the flow vectors.

Step 6: Motion based Frame Reconstruction
A three pronged CNN based network is used to estimate the current video frame from the reconstructed frame from step 4, the reconstructed flow vector from step 5, and the previous output video frame.

3.1 Network architecture

A high-level overview of the proposed network architecture is given in Fig. 1. It is mainly comprised of Frame Compression Autoencoder Network, Flow Vector Estimation Network and Motion-based Reconstruction Network. The architectures of the intermediate networks are also given below individually. The relationship between the modules is presented in Fig. 1. In the frame autoencoder, ConvGRU has been used in one of the five layers both in encoder and decoder sides. Moreover, Flow autoencoder has also been used to compress the flow values and decompressed before the next frame generation, reconstructed by motion extension network. The motion extension network reconstruct the next frame using the previous frame and taking the output from both frame and flow autoencoders.

3.1.1 Frame compression/decompression

A recurrent ConvGRU based encoder model has been used, as depicted in Fig. 2, to encode the frame with varying degrees of compression quality. A ConvGRU based decoder model is used to decompress the encoded videos from step 1 according to the degree of compression. It is inspired by the autoencoder architecture given in Fig. 2. The autoencoder network contains 5 convolutional layers with Generalized Divisive Normalization (GDN) layers as the nonlinearity to reduce the image size. This is then passed to Binarizer to convert the final output to

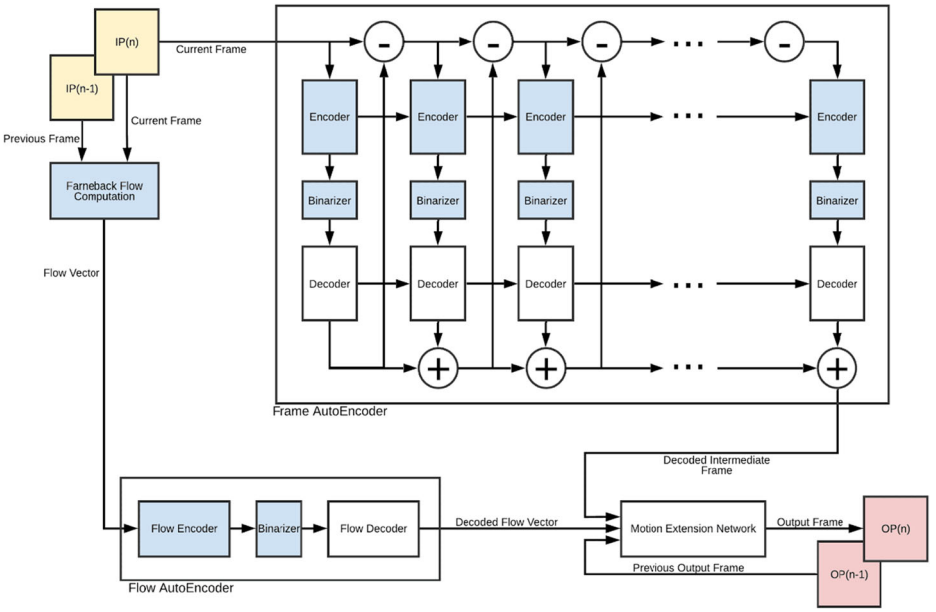

Fig. 1 High-level overview of video compression architecture

highly compressible binary values. For decompression, this binary-coded frame information is again passed through 5 deconvolutional layers to recreate the frame.

One of the convolutional layers in both encoder and decoder networks has been replaced with ConvGRU layer. Image is passed through the autoencoder network to get binary coded information and recreated image. Residual image is calculated by the difference between the original image and the recreated image. Residual image is again passed through the autoencoder network to encode into compressed binary format. This process is repeated based on the level of required compression quality.

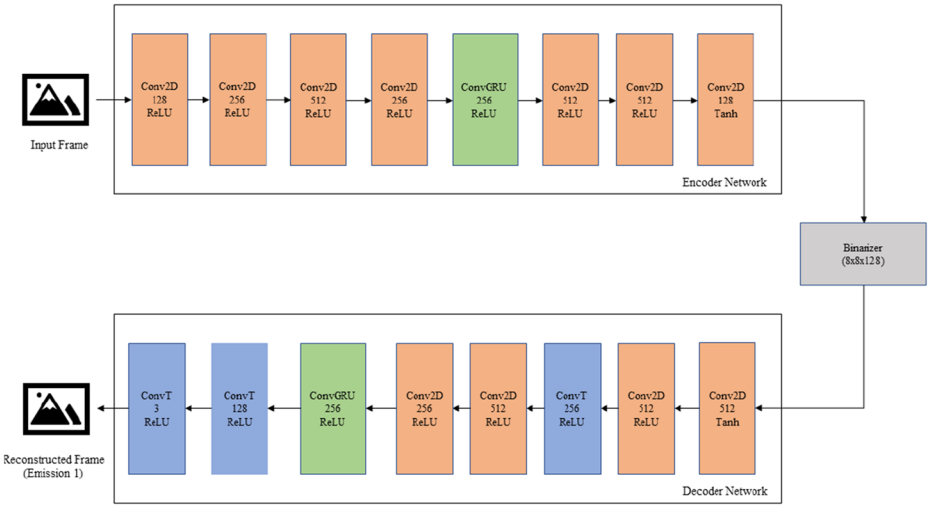

Fig. 2 Frame autoencoder network

The details of the parameters used in Frame Encoder Decoder Network are described in Table 1.

3.1.2 Flow vector estimation/compression/decompression

In the computer vision tasks, optical flow is widely used to exploit temporal relationship. Though a lot of learning based optical flow estimation methods have been recently proposed, we have calculated optical flow using the traditional Farneback flow estimation method. It is well tested method and reduced the training requirement of flow estimation network. In future, learning based flow estimation can also be integrated to result pure neural network based encoder.

The flow vectors between every two frames are estimated. The estimated flow vectors from are compressed using a standard CNN based encoder network with Generalized Divisive Normalization (GDN) layers as the nonlinearity (Fig. 3). A CNN based decoder network with Inverse GDN as the nonlinearity is used to decompress the flow vectors. The Fig. 4 represents the decoder side overview of the network.

The details of the network parameters used in Flow Autoencoder are explained in Table 2.

Table 1 Network parameters of frame autoencoder

FrameEncDec Network

Layer (type)	Output Shape	Param #
Conv2d	[128, 64, 64]	3,584
ReLU	[128, 64, 64]	0
Conv2d	[256, 32, 32]	295,168
ReLU	[256, 32, 32]	0
Conv2d	[512, 32, 32]	2,359,808
Conv2d	[256, 32, 32]	1,179,904
ConvGRUCell	[256, 32, 32]	0
Conv2d	[512, 16, 16]	1,180,160
ReLU	[512, 16, 16]	0
Conv2d	[512, 8, 8]	2,359,808
ReLU	[512, 8, 8]	0
Conv2d	[128, 8, 8]	589,952
Tanh	[128, 8, 8]	0
Binarizer	[128, 8, 8]	0
Conv2d	[512, 8, 8]	590,336
ReLU	[512, 8, 8]	0
Conv2d	[512, 8, 8]	2,359,808
ReLU	[512, 8, 8]	0
ConvTranspose2d	[256, 16, 16]	2,097,408
ReLU	[256, 16, 16]	0
Conv2d	[512, 16, 16]	2,359,808
Conv2d	[256, 16, 16]	1,179,904
ConvGRUCell	[256, 16, 16]	0
ConvTranspose2d	[128, 32, 32]	524,416
ReLU	[128, 32, 32]	0
ConvTranspose2d	[3, 64, 64]	6,147
ReLU	[3, 64, 64]	0

Total params: 17,086,211
Trainable params: 17,086,211
Non-trainable params: 0

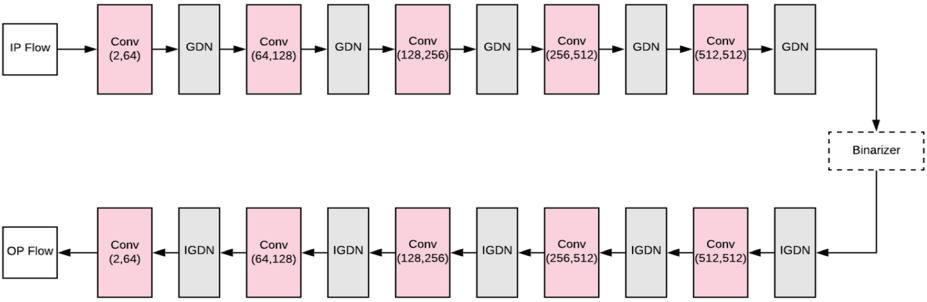

Fig. 3 Flow AutoEncoder

3.1.3 Motion based frame reconstruction

A three-pronged CNN based network is used to estimate the current video frame from the reconstructed frame from step 4, the reconstructed flow vector from step 5, and the previous output video frame. Figure 5 illustrates the architecture of the network. It uses CNN blocks to transform inputs and concat blocks to merge the two prongs.

3.2 Video encoding and decoding algorithms

Algorithm: Video Encoder

Input

1. Target Frame I_{raw}
2. Previous Frame $I_{raw_{prev}}$

Output

1. Frame bitstream $I_{encoded}$ to be transmitted
2. Flow bitstream to $F_{encoded}$be transmitted

Steps

1. Calculate flow vector F_{raw} using frames $I_{raw_{prev}}$ and I_{raw}.
2. Encode frame I_{raw} into binary bitstream $I_{encoded}$using frame encoder network.
3. Encode flow vector F_{raw} into binary bitstream $F_{encoded}$using flow encoder network.
4. Transmit binary stream $I_{encoded}$ and $F_{encoded}$.

3.2.1 Video decoding

For Video reconstruction, binary coded output of frame encoder and flow encoder are needed. The number of emission steps will control the size of frame encoder data and so the bit-rate of signal.

Intermediate frame is calculated by passing binary code frame data through frame decoder. Flow information is also decoded via flow decoder network. Finally, the Motion Extension Network takes previous image, merges this with decoded flow information to create intermediate representation of current image.. It merges this intermediate representation on already decoded Intermediate frame to result in high quality current frame as displayed in Fig. 4. The same has been represented by eq. (1).

$$I_{decoded} = f_{decoder}\left(I_{encoded}, F_{encoded}, I_{decoded_{prev}}\right) \tag{1}$$

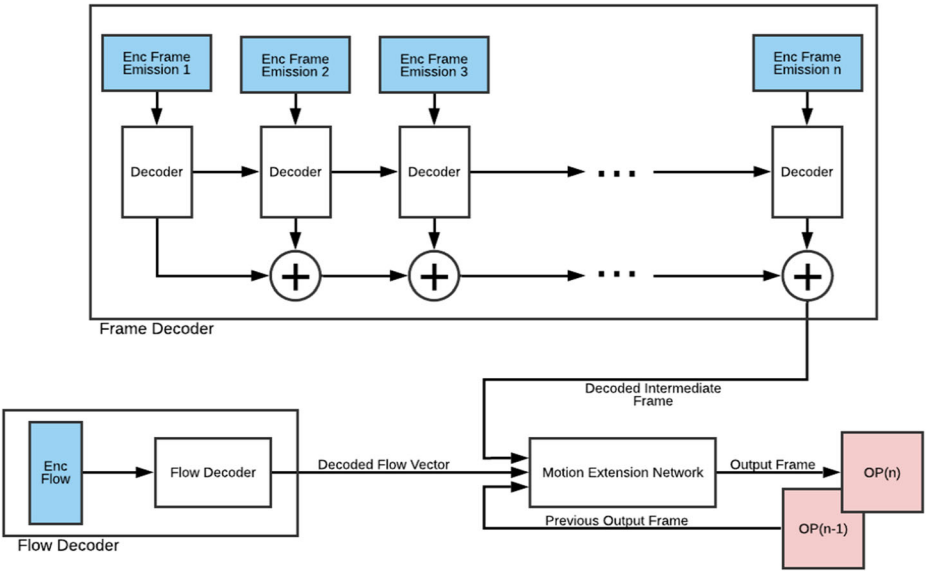

Fig. 4 Decoder side network overview

Table 2 Network parameters of flow autoencoder

FlowEncDec Network

Layer (type)	Output Shape	Param #
Conv2d	[64, 64, 64]	1,216
GDN	[64, 64, 64]	0
Conv2d	[128, 32, 32]	73,856
GDN	[128, 32, 32]	0
Conv2d	[256, 32, 32]	295,168
GDN	[256, 32, 32]	0
Conv2d	[512, 16, 16]	1,180,160
GDN	[512, 16, 16]	0
Conv2d	[512, 8, 8]	2,359,808
GDN	[512, 8, 8]	0
Conv2d	[128, 8, 8]	589,952
Tanh	[128, 8, 8]	0
Binarizer	[128, 8, 8]	0
Conv2d	[512, 8, 8]	590,336
GDN	[512, 8, 8]	0
Conv2d	[512, 8, 8]	2,359,808
GDN	[512, 8, 8]	0
ConvTranspose2d	[256, 16, 16]	2,097,408
GDN	[256, 16, 16]	0
ConvTranspose2d	[128, 32, 32]	524,416
GDN	[128, 32, 32]	0
Conv2d	[64, 32, 32]	73,792
GDN	[64, 32, 32]	0
ConvTranspose2d	[2, 64, 64]	2,050

Total params: 10,147,970
Trainable params: 10,147,970
Non-trainable params: 0

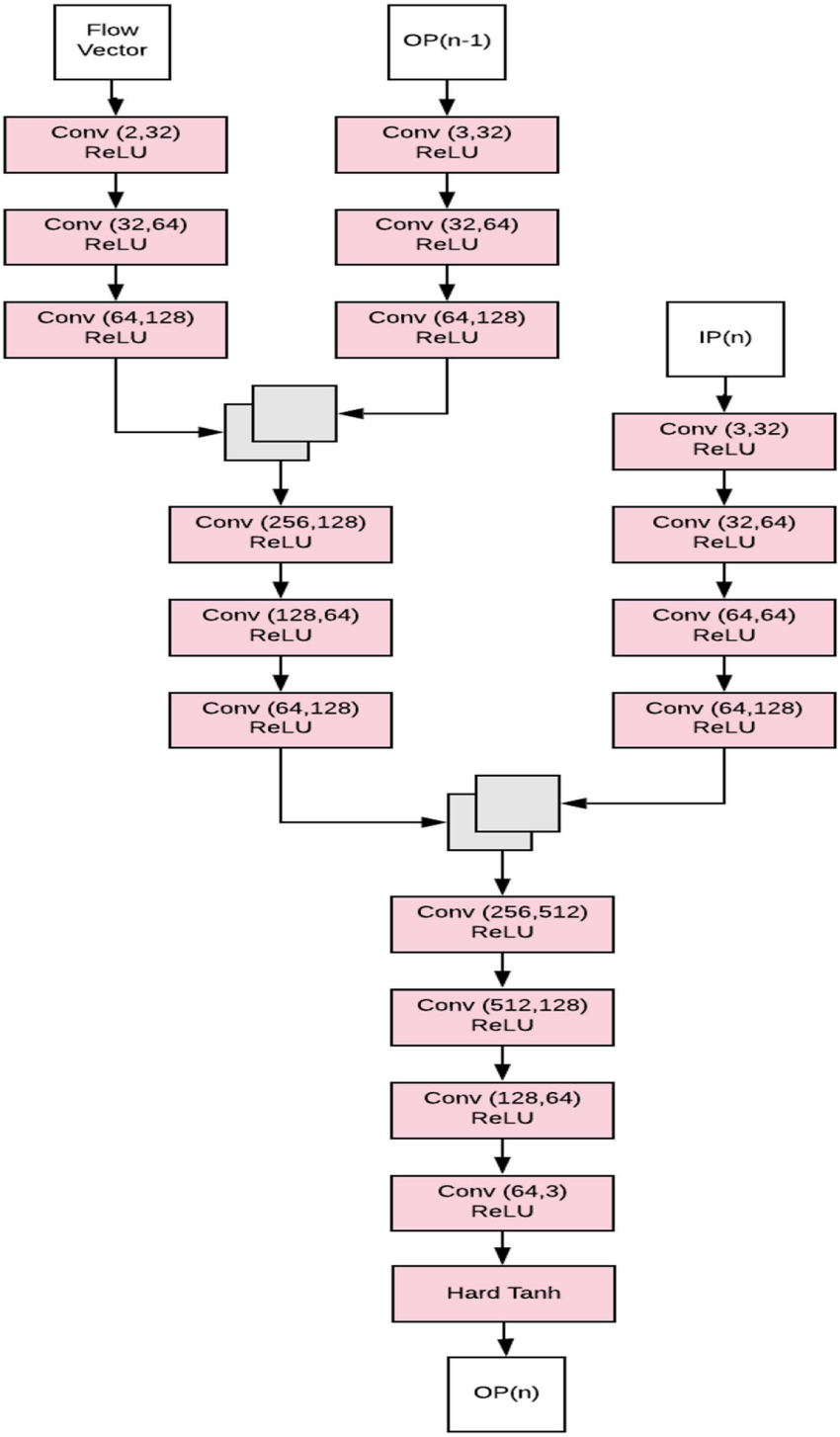

Fig. 5 Motion extension network

where $I_{encoded}$ and $F_{encoded}$ are binaried encoding of current frame and flow vectors, $I_{decoded_{prev}}$ is previously decoded frame and $f_{decoder}$ is representation of decoder neural network.

Algorithm: Video Decoder

Input

1. Frame bitstream $I_{encoded}$
2. Flow bitstream to $F_{encoded}$
3. Previously decoded target frame $I_{decoded_{prev}}$ from state.

Output

1. Decoded target frame $I_{decoded}$

Steps:

1. Decode $I_{encoded}$ using frame decoder network into $I_{intermediate}$.
2. Decode flow bitstream $F_{encoded}$ using flow decoder network into $F_{intermediate}$.
3. Take previously decoded frame $I_{decoded_{prev}}$ from state.
4. Provide $I_{intermediate}$, $I_{decoded_{prev}}$ and $F_{intermediate}$to motion extension network to yield final decode image $I_{decoded}$.
5. Store $I_{decoded}$ image into internal state for decoding of next frame.

3.3 Training strategy

Loss function The goal of the network is to reduce the structural distortion between the input video and the output. We also used mean squared error (MSE) loss function to reduce the color distortion in decompressed images.

$$L = L_{ssim} + \alpha\, L_{mse} \tag{2}$$

where MSE error is evaluated as:

$$L_{mse}\left(y, y^{'}\right) = \frac{1}{N}\sum_{0}^{n}\left(y - y_{i}^{'}\right)^{2} \tag{3}$$

and SSIM error is evaluated based upon three comparison measurements, luminance (l), contrast (c) and structure (s):

$$L_{ssim}\left(y, y^{'}\right) = \left[\, l\left(y, y^{'}\right).c\left(y, y^{'}\right).s\left(y, y^{'}\right) \,\right] \tag{4}$$

Emissions The models can be configured to emit at any number of emissions, each emission refining the output, while also decreasing the compression efficiency. For the ConvGRU model and the MotionNet model, two training strategies are utilized. Only the last emission has been trained in one strategy and choose a random emission in each epoch and train that in another strategy. For the Flow-MotionNet model, only the randomized strategy is used.

4 Experiments

4.1 Experimental setup

Dataset A dataset comprising of 20s long 571 small videos out of total 826 videos from Youtube UGC has been used to train the network, remaining clips has been for testing and validation. Videos of varying quality have been chosen i.e. 480p, 360p and 720p. The frame size has been chosen as 64 × 64, so video clips of all quality firstly rescaled to the chosen format, and then training is performed. Videos frames are taken randomly during training but while testing the clips are chosen from the starting. The model has been trained with randomized emission step training strategy with emission steps varying between 1 to 10. Addition of each emission step improves the output but have an effect on the compression efficiency.

Implementation details For the implementation purpose, a single T4, K80 or P100 GPU has been used to train the network on the Google Colaboratory platform. The frames have been kept to the size of 64 × 64. Adam Optimizer is used to train the neural network with 10e-4 be the learning rate. $\lambda 1$ is taken as one and $\lambda 2$ be 10. First frame encoder is trained for 100 epochs. Then complete network is trained end-to end for 70 epochs. During the training of frame encoder with 100 epochs; the learning rate has been divided by ten at 50th, 70th and 90th epoch. For the whole model training, only 70 epochs have been used after stacking the pretrained framer encoder first and learning rate has been altered at 35th and 55th epoch by dividing with ten.

Evaluation SSIM i.e. Structural Similarity Index and PSNR i.e. Peak Signal to Noise Ratio has been used to measure the visual quality of the reconstructed frames. The temporal distortion encountered among the frames has been evaluated by Flow EPE i.e. End Point Error. Moreover, the reconstruction time of individual frames has been measured by the TPF i.e. Time Per Frame parameter.

4.2 Experimental results and analysis

4.2.1 Experimental results

The proposed model resulted in good quality compression with a 0.963 SSIM score on the test dataset. The individual values of different performance paramters at each emission step is given is Table 3. Comparison of quality of compression achieved with different emission steps is depicted in Fig. 6.

We have shown the video quality of compressed frames for four video sequences in Fig. 7 where first row represents uncompressed sequence, second represents compressed sequence.

4.2.2 Comparison with state-of-art methods

The performance of the proposed architecture has been compared with the state-of-art conventional compression techniques like H264 and H265 and also with the deep learning based models proposed by authors of DVC [20] and Adversarial video compression [14]. Table 4 presents the comparative results of the proposed model with some state-of-art and prominent models and the corresponding graphical comparison of MS-SSIM and PSNR has been given in Figs. 8(a) and

Table 3 Performance values of proposed model

Emission Step/Metric	SSIM	PSNR	Flow EPE	Time per Frame
1	0.709	20	0.822	0.0243
2	0.819	22.5	0.577	0.0248
3	0.874	24.1	0.368	0.0254
4	0.91	25.5	0.276	0.0258
5	0.932	26.7	0.226	0.0263
6	0.948	27.8	0.253	0.0268
7	0.957	28.4	0.189	0.0274
8	0.961	28.9	0.17	0.0279
9	0.963	29.1	0.148	0.0285
10	0.963	29.2	0.17	0.029
Avg	0.9036	26.2	0.3199	0.02662

8(b). UVC dataset was utilized to measure SSIM and PSNR metrics. MS-SSIM correlates better with human perception of distortion. The proposed model outperformed in terms of MS-SSIM metrics. The MS-SSIM and PSNR values of the various architectures have been already presented above. The proposed model achieved good SSIM performance but with a drop in PSNR value.

4.3 Ablation study

The proposed Flow-MotionNet Architecture comprises of three network blocks – a ConvGRU based frame compression/decompression network, a motion-extension network and a flow

Fig. 6 Compressed frame quality at emission steps of 2,7,10

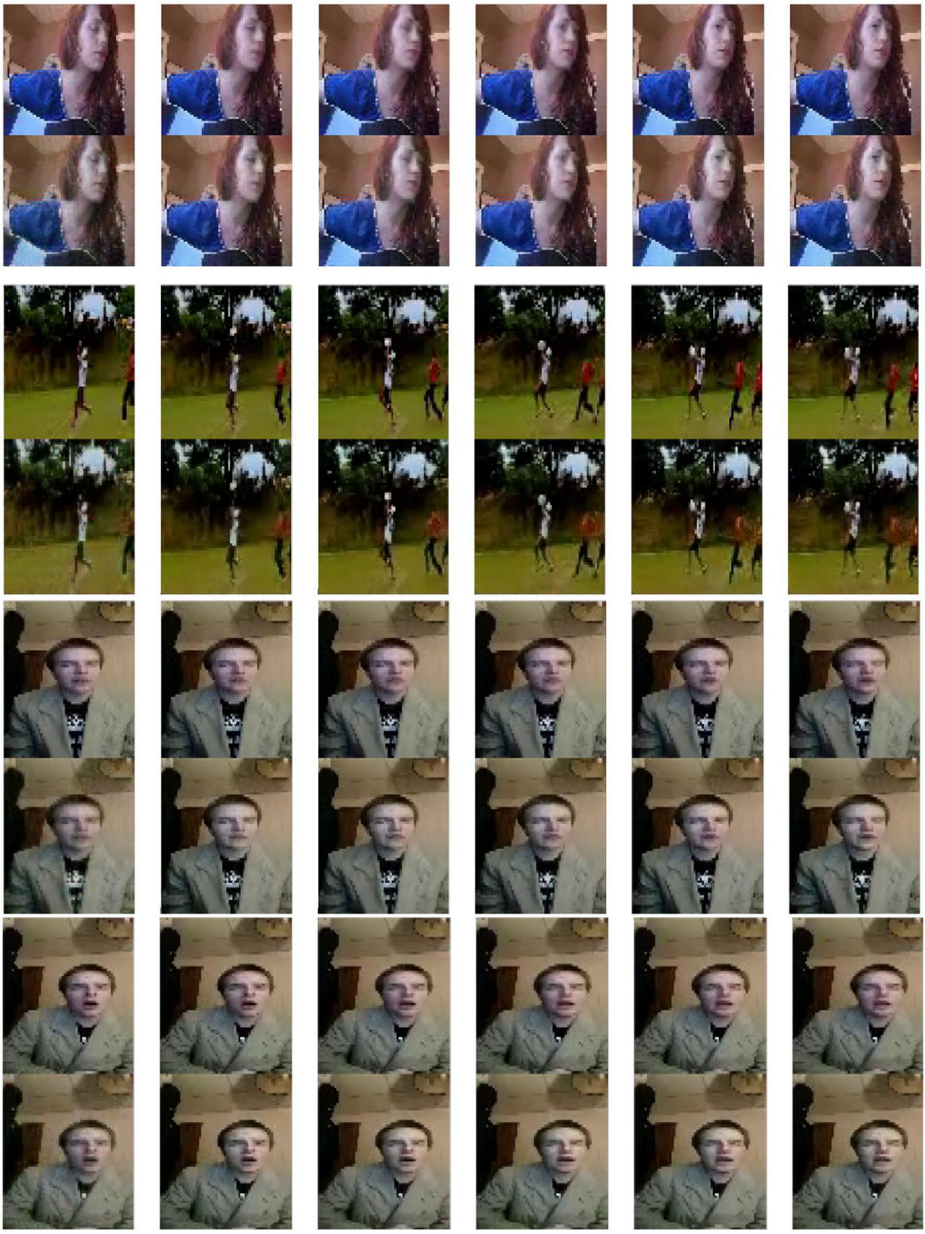

Fig. 7 Compressed frame quality compared for video sequence

vector estimation block. ConvGRU based compression/decompression network utilizes GRU based blocks for improving compressed image quality via multiple iterations. Motion extension network utilizes a previously decoded frame to help encode/decode the next frame. Flow vector estimation block calculates the optical flow of the frames and helps motion extension network to maintain this optical flow.

Table 4 MS-SSIM and PSNR values of various architectures

Architecture	MS-SSIM	PSNR
H.265	0.96	36
Flow-MotionNet (Proposed)	0.963	29.2
H.264	0.955	34
DVC [20]	0.955	35.5
Adversarial video compression [14]	0.9476	28.46

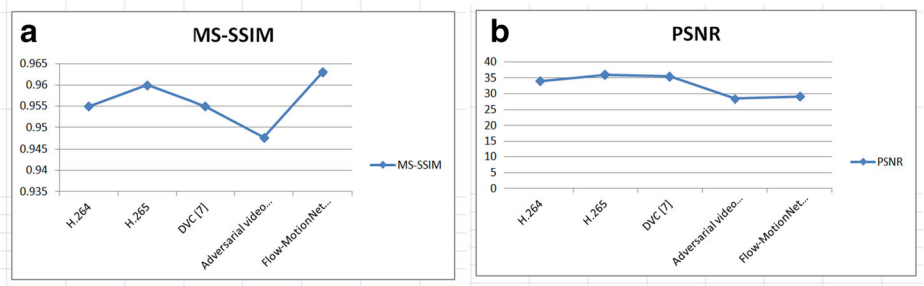

Fig. 8 **a** & **b** Graphical Comparison of MS-SSIM and PSNR values of various architectures

To train the proposed network, two two training strategies has been used. The first strategy is fixed emission step based strategy where the emission step has been fixed to the value of 10. In the second strategy, the emission step was chosen randomly between 1 to 10.

To study the incremental effect of each network block, a baseline network consisting of pure CNN network has been taken. Next, GRU blocks have been to the network to study their effect. This network has been referred to as ConvGRU network in the comparison below. Then Motion Extension Network block was added to study its effect resulting into configuration referred to as MotionNet. Finally, Flow vectors estimations have been added to MotionNet to result in the proposed configuration. These configurations were then evaluated with randomized emission step training strategy to see the effect of training strategy on each part of the network.

4.3.1 Comparison tables

The qualitative and quantitative performance of the proposed model has been measured by the four parameters namely i.e. SSIM (Structural Similarity Index Measure), EPE (End Point Error), PSNR (Peak Signal-to-Noise Ratio), and TPF (Time per Frame). As the network is trained with varying ten emission steps, the below tables presents the performance of the network with the addition of each additional emission step. Qualitative or perception quality performance parameters i.e. SSIM and PSNR values are given Tables 5 and 6 respectively. Flow EPE and TPF values are presented in corresponding Tables 7 and 8. Table 9 depicts the average performance of the network over ten emission steps.

4.3.2 Comparison charts

The below figures present the graphical representation of the the performance parameters namely SSIM, PSNR, EPE and TPF for the 10 emission steps. For the adaptive bit rate video compression, randomization emission step training strategy surpasses the fixed emission step

Table 5 SSIM values per emission

SSIM	1	2	3	4	5	6	7	8	9	10
Baseline	0.67	0.67	0.67	0.67	0.67	0.67	0.67	0.67	0.67	0.67
ConvGRU	**0.087**	0.142	0.194	0.268	0.384	0.57	0.767	0.895	0.937	**0.95**
MotionNet	**0.437**	0.569	0.648	0.771	0.811	0.829	0.921	0.915	0.957	**0.976**
ConvGRU Randomized	**0.652**	0.768	0.823	0.864	0.883	0.893	0.916	0.917	**0.92**	0.919
MotionNet Randomized	**0.706**	0.813	0.866	0.902	0.924	0.938	0.948	0.951	0.953	**0.954**
Flow-MotionNet Randomized	**0.709**	0.819	0.874	0.91	0.932	0.948	0.957	0.961	0.963	**0.963**

Table 6 PSNR values per emission

PSNR	1	2	3	4	5	6	7	8	9	10
Baseline	**18.3**	21.1	22.4	23.7	24.3	24.6	25.8	25.7	**26**	25.9
ConvGRU	**20**	22.2	23.5	24.8	25.8	26.6	27.2	27.6	27.8	**27.8**
MotionNet	**20**	22.5	24.1	25.5	26.7	27.8	28.4	28.9	29.1	**29.2**
ConvGRU Randomized	**18.3**	21.1	22.4	23.7	24.3	24.6	25.8	25.7	**26**	25.9
MotionNet Randomized	**20**	22.2	23.5	24.8	25.8	26.6	27.2	27.6	27.8	**27.8**
Flow-MotionNet Randomized	**20**	22.5	24.1	25.5	26.7	27.8	28.4	28.9	29.1	**29.2**

Table 7 Flow EPE values per emission

Flow EPE	1	2	3	4	5	6	7	8	9	10
Baseline	1.154	1.154	1.154	1.154	1.154	1.154	1.154	1.154	1.154	1.154
ConvGRU	**4.566**	3.711	3.061	2.752	2.153	1.549	0.817	0.42	0.201	**0.117**
MotionNet	**1.851**	1.03	0.799	0.536	0.433	0.433	0.233	0.242	0.137	**0.081**
ConvGRU Randomized	**1.251**	0.613	0.555	0.409	0.311	0.319	0.273	0.221	0.201	**0.173**
MotionNet Randomized	**0.826**	0.477	0.383	0.35	0.273	0.248	**0.173**	0.199	0.175	0.189
Flow-MotionNet Randomized	**0.822**	0.577	0.368	0.276	0.226	0.253	0.189	0.17	**0.148**	0.17

Table 8 TPF values per emission

Time Per Frame	1	2	3	4	5	6	7	8	9	10
Baseline	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015
ConvGRU	**0.0182**	0.0186	0.0191	0.0195	0.0201	0.0205	0.0211	0.0216	0.0221	**0.0226**
MotionNet	**0.0208**	0.0214	0.0218	0.0224	0.023	0.0234	0.0239	0.0245	0.025	**0.0255**
ConvGRU Randomized	**0.0182**	0.0186	0.0191	0.0195	0.0201	0.0205	0.0211	0.0216	0.0221	**0.0226**
MotionNet Randomized	**0.0208**	0.0214	0.0218	0.0224	0.023	0.0234	0.0239	0.0245	0.025	**0.0255**
Flow-MotionNet Randomized	**0.0243**	0.0248	0.0254	0.0258	0.0263	0.0268	0.0274	0.0279	0.0285	**0.029**

Table 9 Comparative average performance of different incremental modules of the proposed network

	Avg. SSIM	Avg. PSNR	Avg. EPE	Avg. TPF
Baseline	0.67	18.9	1.154	0.015
ConvGRU	**0.5194**	**15.08**	**1.9347**	**0.02034**
MotionNet	0.7834	21.21	0.5775	0.02317
ConvGRU Randomized	0.8555	23.78	0.4326	**0.02034**
MotionNet Randomized	0.8955	25.33	0.3293	0.02317
Flow-MotionNet Randomized	**0.9036**	**26.22**	**0.3199**	**0.02662**

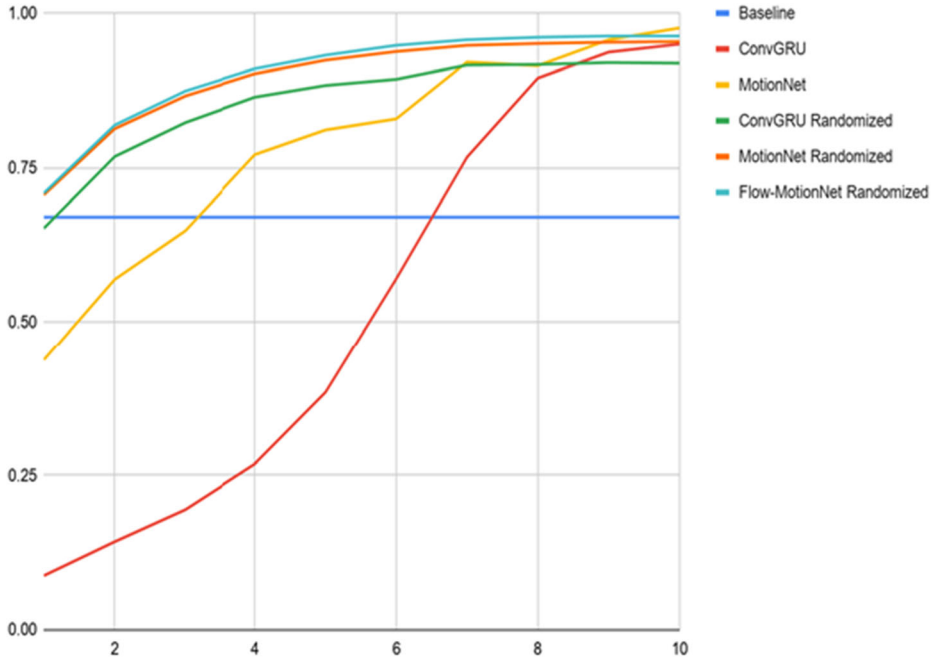

Fig. 9 Variation of MS-SSIM of different incremental modules of proposed network

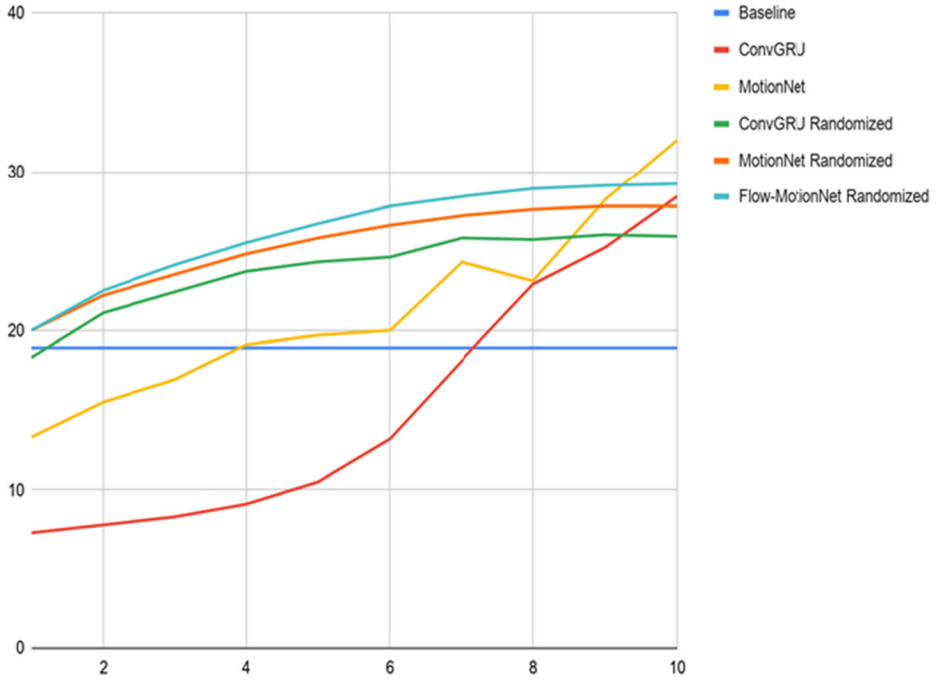

Fig. 10 Variation of PSNR of different incremental modules of proposed network

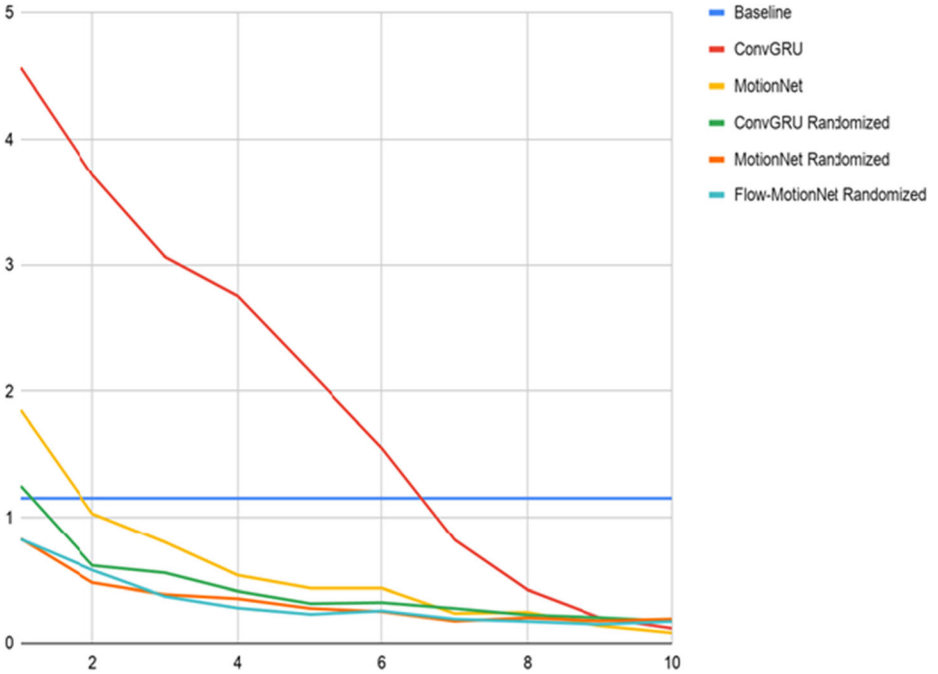

Fig. 11 Variation of Flow End Point Error of different incremental modules of proposed network

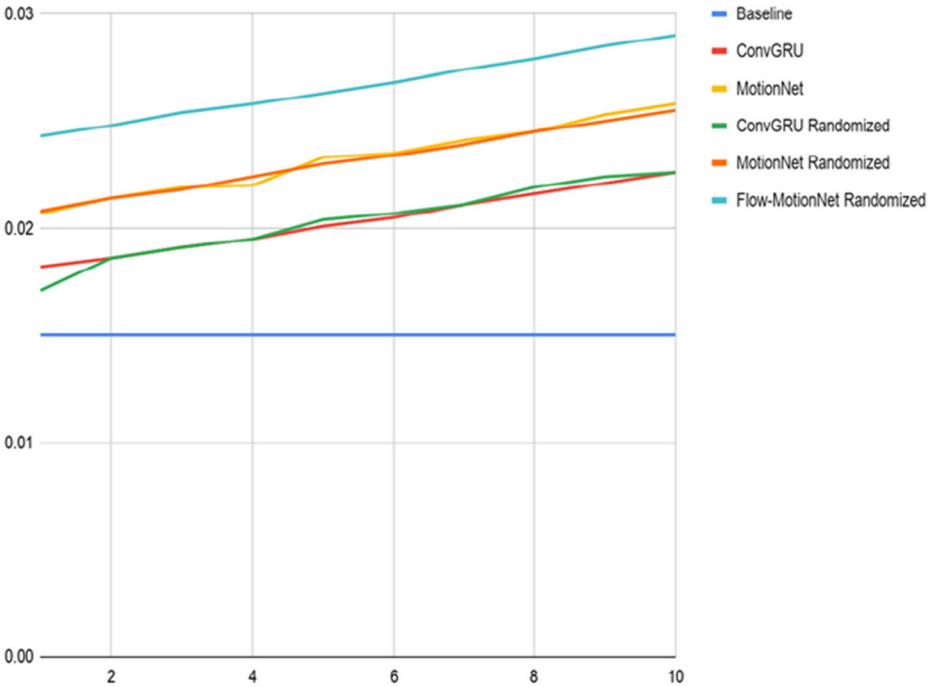

Fig. 12 Time Per Frame of different incremental modules of proposed network

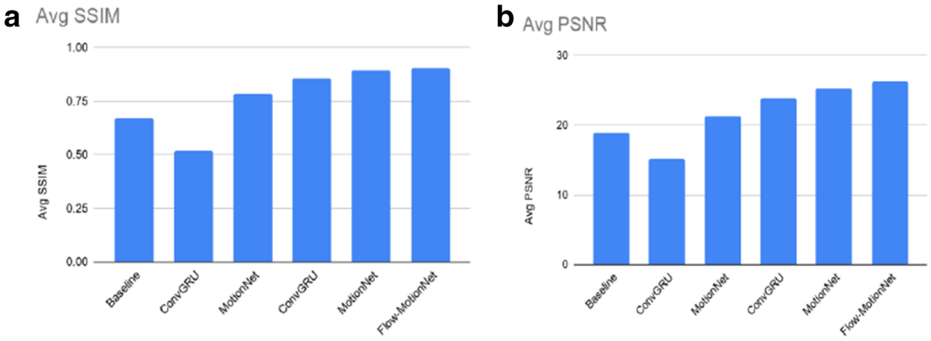

Fig. 13 **a** & **b** Relative comparartive analysis of MS-SSIM and PSNR of different incremental modules of proposed network

strategy. The relative analysis of the graphs show the growth in SSIM and PSNR values with each additional emission step. The proposed model is designed incrementally. Firstly, addition of motion extension network improves the performance of simple frame autoencoder. In further addition, flow autoencoder is added to Motion Net architectecture for motion estimation and prediction, which also outperforms the prior simple MotionNet architecture in terms of visual quality. The same thing can be inferred from the relative graphs of respective models. Moreover, addition of modules also increases the computation of the network, leading to the increased frame reconstruction time. Figures 9 to 12 presents the graphical and relative performance comparison of the different models at different emission steps ranging from 1 to 10.

4.3.3 Average comparison chart

The below charts in figures 13 and 14 represent the histogram representation of the average values of various performance parameters obatained for various compression models for better relative performance analysis. The graphs represent the comparison data between various experiments. The sample frames produced by the models are given in next sub-section. The top-left frame is input frame 1, the bottom left frame is input frame 10, the top-right frame is output frame 1, the bottom right frame is output frame 10.

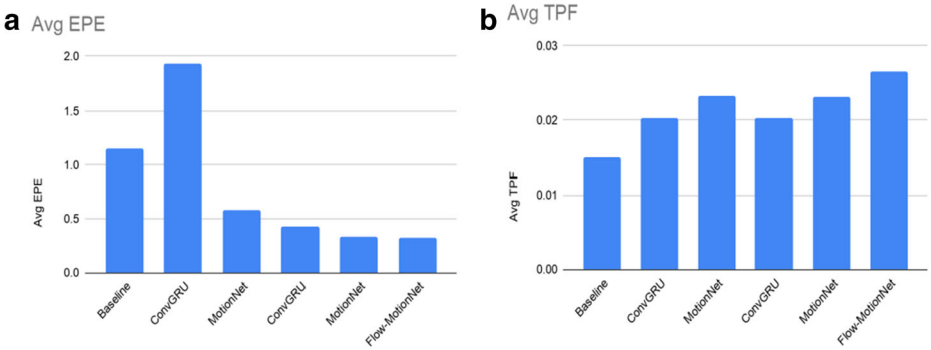

Fig. 14 **a** & **b** Relative comparative analysis of Flow-EPE and TPF of different incremental modules of proposed network

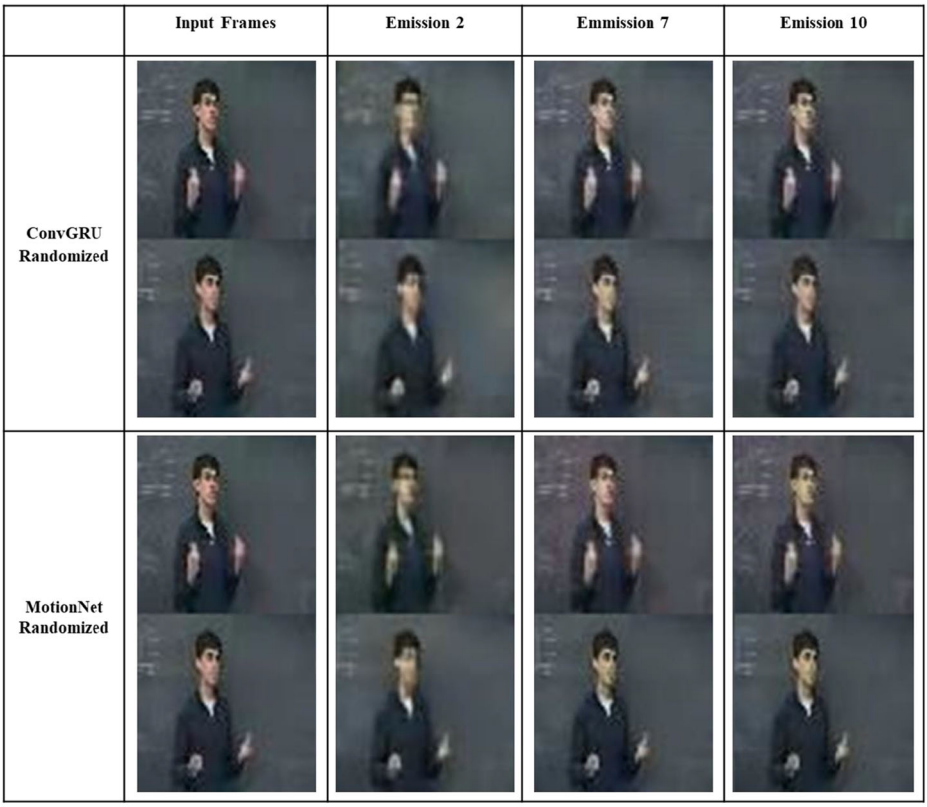

Fig. 15 Sample Frames at different emission steps

It can be inferred from the graphs that the proposed architecture achieves increased SSIM value eventually leads to improvement in visual quality of video frames. But the addition of optical flow and and motension extension network leads to rise in frame reconstruction time, so the TPF value of proposed architecture is higher than others. Further explorations and improvements may result in more efficient outcomes.

4.3.4 Sample frames

The samples of reconstructed frames after compression and decompression of different models at different emission steps are presented in this section. In the image matrix, the left column contains two original images and the right column contains the corresponding decompressed output image. Figure 15 presents the relative visual quality of some of the sample frames at different emission steps.

4.3.5 Comparison inference

Incorporation of the previously decoded frame: It is found that the flow representation capability of the model improves significantly by adding in the previous output frame while inferring the current output frame from the decoded frame. The above tables corroborate this claim, and there is also a clear visual improvement in the sample frames.

Incorporation of flow vectors: The incorporation of flow vectors during decoding of the frames, along with the previous frame and the currently decoded frame, while giving a significant boost over plain ConvGRU model, gives only marginal improvements over only incorporating previous frames. However, as the decode time required per frame is not significantly increased by incorporation of flow vectors, the improvement is welcome. The visual difference between the results, however, is pronounced.

5 Conclusion

The proposed model presents a lightweight learning-based adaptive video compression approach that allows variation in the quality of reconstructed video with the amount of data sent, without requiring separate low-resolution versions of the same video. The ConvGRU units in encoder and decoder networks and flow vector induced frame reconstruction have significantly improved the performance of the network. The whole network is designed and experimented incrementally. In comparison to some state-of-the-art techniques, a considerable improvement in visual quality and efficiency has been observed when trained the network with random emission step training strategy but with slight increment in frame reconstruction time. This architecture can be further enhanced and optimized by plugging with additional modules like entropy coding and others.

Declarations

Conflict of interest None.

References

1. Agustsson E, Mentzer F, Tschannen M, Cavigelli L, Timofte R, Benini L, Gool LV (2017) Soft-to-hard vector quantization for end-to-end learning compressible representations. In: NIPS
2. Baig MH, Koltun V, Torresani L (2017) Learning to inpaint for image compression. In: NIPS
3. Ball_e J, Laparra V, Simoncelli EP (2017) End-to-end optimized image compression. In: ICLR

4. Cavigelli L, Hager P, Benini L (2017) Cas-cnn: A deep convolutional neural network for image compression artifact suppression. In Neural Networks (IJCNN), 2017 International Joint Conference on, pages 752–759. IEEE
5. Chen T, Liu H, Shen Q, Yue T, Cao X, Ma Z (2017) DeepCoder: a deep neural network based video compression. IEEE Visual Communications and Image Processing (VCIP). https://doi.org/10.1109/VCIP.2017.8305033
6. Dong L, Li Y, Lin J, Li H, Wu F (2020) Deep Learning-Based Video Coding: A Review and a Case Study, ACM Computing SurveysFebruary
7. Goswami K, Kim BG A design of fast high-efficiency video coding scheme based on markov chain monte carlo model and bayesian classifier. IEEE Trans Ind Electron 65(11):8861–8871
8. Goswami K, Lee J, Kim B-g Fast algorithm for the high efficiency video coding (HEVC) encoder using texture analysis. Inf Sci 364:72–90
9. He X, Hu Q, Han X, Zhang X, Lin W (2018) Enhancing hevc compressed videos with a partition-masked Convolutional neural network. arXiv preprint arXiv:1805.03894
10. Huo S, Liu D, Wu F, Li H (2018) Convolutional neural network-based motion compensation refinement for video coding. In Circuits and Systems (ISCAS), 2018 IEEE International Symposium on, pages 1–4. IEEE
11. Jia X, De Brabandere B, Tuytelaars T, Gool LV (2016) Dynamic filter networks. In: NIPS
12. Jiang H, Sun D, Jampani V, Yang MH, Learned-Miller E, Kautz J (2018) Super slomo: high quality estimation of multiple intermediate frames for video interpolation. CVPR
13. Johnston N, Vincent D, Minnen D, Covell M, Singh S, Chinen T, Hwang SJ, Shor J, Toderici G (2017) Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks. arXiv preprint arXiv:1703.10114
14. Kim S, Park JS, Christos G Bampis JL, Markey MK, Dimakis AG, Bovik AC (2018) Adversarial Video Compression Guided by Soft Edge Detection. arXiv:1811.10673v1 [eess.IV] 26 Nov
15. Le Gall D (1991) MPEG: A video compression standard for multimedia applications. Commun ACM
16. Lee J-H, Park CS, Kim B-G, Jun D-S (2013) Novel fast PU decision algorithm for the HEVC video standard, IEEE International Conference on Image Processing, 1982-1985
17. Li Y, Li B, Liu D, Chen Z (2017) A convolutional neural network-based approach to rate control in hevc intra coding. In Visual Communications and Image Processing (VCIP), 2017 IEEE, pages 1–4. IEEE
18. Li C, Song L, Xie R, Zhang W, C. M. I. center (n.d.) Cnn based post-processing to improve hevc
19. Liu Z, Yeh R, Tang X, Liu Y, Agarwala A (2017) Video frame synthesis using deep voxel flow. In: ICCV
20. Lu G, Ouyang W, Xu D, Zhang X, Cai C, Gao Z (2019) DVC:An End-to-end Deep Video Compression Framework. arXiv: 1812.00101v3 [eess.IV] 7Apr
21. Mathieu M, Couprie C, LeCun Y (2015) Deep multi-scale video prediction beyond mean square error. arXiv preprint arXiv:1511.05440
22. Mathieu M, Couprie C, LeCun Y (2016) Deep multi-scale video prediction beyond mean square error. In: ICLR
23. Mentzer F, Agustsson E, Tschannen M, Timofte R, Van Gool L (2018) Conditional probability models for deep image compression. arXiv preprint arXiv:1801.04260
24. Niklaus S, Mai L, Liu F (2017) Video frame interpolation via adaptive separable convolution. In: ICCV
25. Oord AvD, Kalchbrenner N, Kavukcuoglu K (2016) Pixel recurrent neural networks. In: ICML
26. Rippel O, Bourdev L (2017) Real-time adaptive image compression. In: ICML
27. Song R, Liu D, Li H, Wu F (2017) Neural network-based arithmetic coding of intra prediction modes in hevc. In Visual Communications and Image Processing (VCIP), 2017 IEEE, pages 1–4. IEEE
28. Song X, Yao J, Zhou L, Wang L, Wu X, Xie D, Pu S (2018) A practical convolutional neural network as loop filter for intra frame. arXiv preprint arXiv:1805.06121
29. Theis L, Shi W, Cunningham A, Husz_ar F (2017) Lossy image compression with compressive autoencoders. In: ICLR
30. Toderici G, Vincent D, Johnston N, Jin Hwang S, Minnen D, Shor J, Covell M (2017) Full resolution image compression with recurrent neural networks. In: CVPR
31. Vondrick C, Pirsiavash H, Torralba A (2016) Generating videos with scene dynamics. In: NIPS
32. Wang T, Chen M, Chao H (2017) A novel deep learning based method of improving coding efficiency from the decoder-end for hevc. In Data Compression Conference (DCC), 2017, pages 410–419. IEEE
33. WebP (n.d.) https://developers.google.com/speed/webp/
34. Xue T, Wu J, Bouman K, Freeman B (2016) Visual dynamics: probabilistic future frame synthesis via cross convolutional networks. In: NIPS
35. Yan N, Liu D, Li H, Wu F (2017) A convolutional neural network approach for half-pel interpolation in video coding. In Circuits and Systems (ISCAS), 2017 IEEE International Symposium on, pages 1–4. IEEE
36. Yang R, Xu M, Wang Z (2017) Decoder-side hevc quality enhancement with scalable convolutional neural network. In 2017 IEEE International Conference on Multimedia and Expo (ICME), pages 817–822. IEEE

37. Yang R, Xu M, Wang Z, Guan Z (2017) Enhancing quality for hevc compressed videos. arXiv preprint arXiv:1709.06734
38. Yang R, Xu M, Wang Z, Li T (n.d.) Multi-frame quality enhancement for compressed video
39. Zhao Z, Wang S, Wang S, Zhang X, Ma S, Yang J (2018) Cnn-based bi-directional motion compensation for high efficiency video coding. In Circuits and Systems (ISCAS), 2018 IEEE International Symposium on, pages 1–4. IEEE