International Journal of Research

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 01
January 2017

Spatial and Temporal Uncertainty based Context Aware Adaptive Video Compression Using Deep learning

B. Shravan Kumar[1] Dr. V. Usha Shree[2] Dr.P. Chandrasekhar Reddy[3]

[1]Research Scholar, JNTUH, Hyderabad, India.

[2]Professor and Principal, JBREC, R.R Dist, Telangana, India.

[3]BOS Chairman & Professor, Dept.of ECE, JNTUH, Hyderabad, India.

Abstract

Nowadays, the increase in the power of the processors becomes faster than that of the storage capacities, thus surpassing the speed of the bandwidth of the networks, indeed it requires a lot of changes in the telecommunications infrastructures. To overcome this problem, it is recommended toward lessen size of data through using the power of the processors rather than increasing the storage and transmission capacities of the data. It is thus important that an effective video compression system is established and high quality frames are produced at certain bandwidth budgets. The proposed method is an adaptive video compression technique using deep learning. During the training phase, visual spatial and temporal uncertainty features are extracted from input video frames. These features are given as input to 1D convolution neural network. 1D CNN is trained to identify context aware visual features. This data is used to perform context aware adaptive video compression. The DCT block based compression is used to perform adaptive quantization on the DCT coefficients. By preserving the rate of compression, this adaptive quantization improves the quality of compressed videos.

Keywords: video compression, 1D convolution neural network, context aware visual features

I. INTRODUCTION

Video compression methods have also been helpful to identify action[1] and compression models[2].However, over the last few decades, video compression algorithms have relied, for example in block based motion estimation and discrete cosine (DCT), on hand-crafted modules in order to minimise the redundancies of video sequence. A recent auto encoder based on the DNN (deep neural network) has achieved a similar or better performance compressed image compression [3, 4, 5] to that of a typical JPEG [6], JPEG2000 [7] or BpG [8] image codec. The reality that DNN compression with a large end-to-end training and extremely non-linear transformation not seen in conventional approaches is a plausible explanation. But it is not easy to apply these techniques directly for the construction of a video compression end-to-end learning method. First of all, there is an open problem how the motion information suited to video compression is created and compressed.

Video compression processes rely heavily on motion data to prevent video sequence temporary redundancy. The optical flow is an easy solution for viewing motion information. Current learning approaches to optical flow seek to build as accurate as possible flow scales. However, the accurate optical flow for a specific video task is not always optimal[9]. Also, optical flow data volume is increasing significantly compared to motion in conventional compression systems, and the amount of bits needed for storing motion information was increased significantly by applying directly the current compression methods [10, 11] to lessen the optical flow values. Second, it is not clear how a video compression device based on DNN can be developed by minimising a residual as well as motion information rate- distortion based objective. Rate-distortion optimization (RDO) is intended to achieve a better reconstructing frame quality (i.e., less distortion), if the number of bits (or bit rate) of compression was specified. For the performance of video compression, RDO is essential. In a compressing learning regime, the RDO technique has to be optimised to enhance the whole system to improve end-to-end instruction.

II. Related Work

2.1. Image Compression

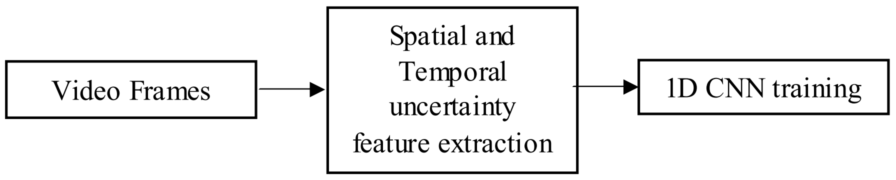
In the last few decades there have been several ideas for image compression algorithms[6, 7, 8]. These methods are largely based on techniques produced by hand. For example, pixels to another DCT display in the standard JPEG map take into consideration the necessary coefficients before entropy encoding [6]. One drawback is that these modules were optimised individually and cannot achieve maximum compression. Recent DNN- based image compression methods were used to construct a progressive image compression scheme with more and more attention to recurring Neural Networks (RNNs).The auto-encoder network for image compression [4, 12, 13] was designed by other methods using the CNNs. The aims of the neural network optimization sequence are to remove the distortions of the original frames (e.g., a mean square error) from reconstructed frames and not to take into consideration on the number of bits used for compression. By adding number of bits into the optimization process in [4, 12, 13, and 14] to increase compression performance, the rate-distortion optimization technique was introduced. Context models for adaptive arithmetic encoding method are learned in [15] to approximate the bit rates, whereas in [12, 13] non-adaptive arithmetic encoding is used. Additionally, other technology is being proposed to boost the efficiency of image compression, Take generalised DS[4], multi-scale image decomposition[16], Training on the contrary[16], and a map[14] of interest, etc. and intra-prediction[17]. These works are key components for our network of video compression.

2.2. Video Compression

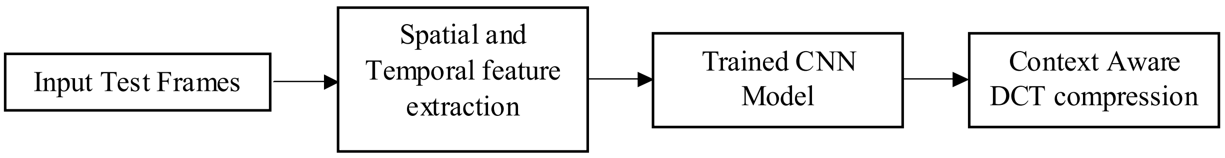
A varied, standard video compression algorithms (eg, H.264[10] and H.265], etc), have been proposed in the last few decades. The predictive architecture is followed by primarily these algorithms. Although the compression output is very efficient, it is built manually and cannot be combined end-to-end. A number of intrapredictive and residual DNN-based video compression methods[18], decision-making mode[19], coding entropy[20] and post-processing have been also put forward. These methods are used instead of constructing an end-to- end compression system to boost performance of a single module on the standards video compression algorithm. Chen et al. indicated that video compression learning should be a block based method in [21]. But in the frontier between blocks it will eventually create blockage artefact. In addition, they used the movement information spread through the conventional block movement estimation from previously reconstructed frames, which degrades compression efficiency. For special domain images, compress the residual H.264 residual, Tsai et.al. Proposed an auto encoder network[22]. This study does not use an in-depth movement model. In [23], the RNN method formulates video compression as the interpolation of a frame. However, the motions are also produced with the current methods of non-deep learning image compression by means of a traditional block movement estimate. In other words, a deep model does not conduct a motion assessment and compression and is configured along with other components. In[23], only distortions of the trainers can attempt to be eliminated (e.g. mean square error) between the initial frame and its reconstructed frame irrespective of the potential distortions throughout a training process. By contrast, the DNN, which is optimised along with the other components, performs a compression and motion evaluation by looking at the rated distortion compensation for the compression process.

III. Proposed Adaptive Video Compression Scheme

The proposed method is an adaptive video compression technique using deep learning. During the training phase, visual spatial and temporal uncertainty features are extracted from input video frames. These features are given as input to 1D convolution neural network. 1D CNN is trained to identify context aware visual features. This data is used to perform context aware adaptive video compression. The DCT block based compression is used to perform adaptive quantization on the DCT coefficients. By conserving the compression ratio, the versatile quantization improves the performance of the compressed video. A blue diagram of the proposed shape, illustrates Figure 1 below.



(a) Training Phase


Input Test Frames
Spatial and Temporal feature extraction
Trained CNN Model
Context Aware DCT compression

(b) Testing Phase

Fig 1: Proposed Adaptive Video Compression scheme

A. Spatial and Temporal Uncertainty feature extraction

Uncertainty in a video frame is the confidence in each pixel calculated from the saliency map. The uncertainty map directly corresponds to the textural and context aware features present in the image. Uncertainty map is formed form the saliency features extracted from the input frame. Let S denote the saliency map of the input video frame, the temporal uncertainty map U be demarcated as

$$U[x,y] = \gamma|\alpha S[x,y] * W^{L_t}|$$

Where,

x, y indicate the pixel positions in a frame

$U[x,y]$ is the uncertainty map

γ is the scaling factor of the uncertainty map

α is the scaling factor of the saliency map

W^{L_t} is the temporal filter

The spatial uncertainty is calculated using the K frames present in the input video. The spatial uncertainty is denoted by $\boldsymbol{S_U}$

$$\boldsymbol{S_U} = [U_1 U_2 \ \dots \ U_K]$$

U_K is calculated by

$$U_k = \gamma|\alpha S_k * W^{L_{s1} \times L_{s2}}|$$

Where

k is the frame index.

$W^{L_{s1} \times L_{s2}}$ is the spatial filter.

B. Convolution Neural Network

Convolutional neural networks (CNN) were a special type of multi-layered neural networks feed-forward often used in image processing, for a pattern recognition task whose architecture of the network is inspired by the visual cortex of animals for the character recognition task.Today, CNNs have become popular in several in artificial intelligence in automatic language processing: text classification and classification of music and environments sound. The difference in implementation of convolutional neural networks in dealing with image processing and automatic language processing mainly resides in level of extraction of lines by filters. In image processing, the extraction characteristics are performed on small regions of the image, each filter moves in two directions (horizontally and vertically) on the entrance to the network. But, in automatic language processing, the filter covers a series of words or a series of time series, and it only moves in one direction (horizontally) for feature extraction.

Unlike standard learning approaches which are based on predefined characteristics, convolutional neural networks are capable of detect, extract and learn specific traits suited to the task at hand at the time of learning without the need for resources, implementation algorithms or tools to extract "pre-defined features" (engineered features).

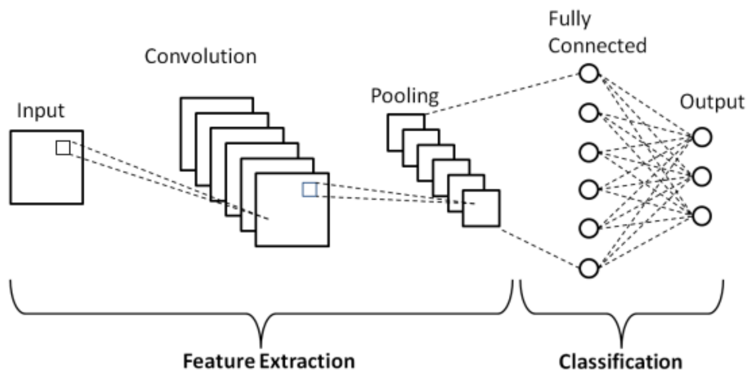


Fig 2: The main building blocks of an architecture using a simple convolutional neural network

As shown in figure 2, the architecture of a convened neural network if simple is characterized mainly by two blocks:

- Extraction of characteristics: this block is composed of an entry of dimension n × k which represents an instance (a sequence of words or a signal speech in our use case), convolution operations allowing extract a large number of characteristics which will subsequently be pressed by pooling operations (also called sub-sampling).
- Modelling and prediction: this block is characterized by a series of fully connected hidden layers to predict a unit that can be a category or value, depending on the task.

The layer structure of the proposed 1D CNN is given below:

- Input Layer

- convolution2dLayer
- batch Normalization Layer
- relu Layer
- maxPooling2dLayer
- fully Connected Layer
- softmax Layer
- classification Layer

These layers are discussed in detail in this section

Convolution2dLayer

The first stage in the CNN model is the feature extraction form the input data. The feature extraction is carried out using convolution. Convolutional is the fundamental operation of the first layers of the network which allow to extract specific information characterizing the input $A \in \mathbb{R}^{n \times k}$. This is a mathematical exercise, involving the use of a sequence of W filters in a series of areas of input uncertainty phase$A \in \mathbb{R}^{n \times k}$ With the concept of a sliding box to generate a series of function maps as output. Convolution filters correspond to randomly initialized weights then updated by the gradient back propagation algorithm at the time of learning.

Batch Normalization Layer;

Batch normalization layer was usually used to stabilize the distribution of data during learning. For the group of observations constituting the batch, the values of the matrices representing the images are centered and reduced.

Algorithm: Standardization of Batch transformation applied to activating x in baths

Input: Values of x over a mini-batch:$\mathcal{B} = \{x_{1....m}\}$;

Parameters to be learned: γ, β

Output: $\{y_i = BN_{\gamma,\beta}(x_i)\}$

Mini-batch mean $\mu\mathcal{B}$was given by

$$\mu\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

The mini-batch varience $\sigma_{\mathcal{B}}^2$was give by

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu\mathcal{B})^2$$

The normalized value $\hat{x}_i$ is given by

$$\hat{x}_i \leftarrow \frac{x_i - \mu\mathcal{B}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

The scale an shifted output is expressed as

$$y_i \leftarrow \gamma\hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i)$$

RELU Layer:

The back-propagation algorithm has the disadvantage that the gradient generated by them goes towards zero at a high activation. Since the gradient in a deep network over several layers back over must be propagated, can with several consecutive, high activations of the gradient to disappear. This phenomenon is called vanishing gradients in English-language literature. Therefore can without further adjustments, only very flat networks can be trained with the back propagation algorithm and the sigmoid function as activation. To avoid the problem, instead of the sigmoid function, a ReLU layer used. The abbreviation ReLU stands for Rectified Linear ¨ Unit.

Nonlinear threshold is controlled when a recovered linear unit activation (ReLU), all the value lower than zero inputs are set to zero.

This provider is equivalent to

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

Pooling

The pooling operation (also called sub-sampling) is a key element for convolutional neural networks, because it improves the process network learning and avoid over-learning Gradually diminishing the map layers of functions to the right through the stripes of convolution and holding the most comprehensive details. As for convolution operations, 3 pooling dimensions (1D, 2D and 3D) have been proposed in the literature depending on the target task. The dimension pooling dimension often corresponds to the dimension of the convolution operation performed. Since we apply 1D convolutions on our inputs textual and acoustic, we are therefore interested in pooling operations 1D.The 1D pooling operation is characterized by a height pooling window h p (user-defined hyper-parameter) that moves in one direction (from top to bottom) with s p no movement on each characteristic card c ∈ R T o × 1 produced by the convolution layer. This is mostly implemented with two key solutions:

- **Max-pooling**: This is to revert a local limit value to the level of each pool window;

MAX POOLING 2D LAYER takes the maximum value for each block whose size is set to (2, 2) (Pool size = 2, 2). The Strides is (2, 2). In fact, we reduce by 2 again the dimension of the matrices, that is to say (25, 10). For a function map which has nh x nw x nc dimensions, the results dimensions received after a layer pooling is calculated as

$(n_h - f + 1) / s \times (n_w - f + 1)/s \times n_c$

where,

->**n_h -** height of feature map

->**n_w -** width of feature map

->**n_c -** number of channels in the feature map

->**f -** size of filter

->**s -** stride length

Fully connected Layer:

Along with each and every neuron the volume corresponding to one's point of input, are linked neurons of the absolutely coupling layer meaning each neuron of a layer the before. This layer generates a K aspect value vector, where K means the number of grades for a predictable layout to network c. Rank probabilities for each class of an image shall be included in this vector. The FC layer consists in the multiplication of the input vector by the weight array (C). Each input value, by being linked to any output value, describes how the word totally interlaced is reflected.

Softmax Layer:

Softmax is the function to transform the vector K to the true value 1 in the form of K to the actual value 1. The product called softmax will be used as a form:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}}$$

$\vec{z}$ = The input vector to the softmax function, made up of (z0, ... zK)

Classification layer:

To calculate the function value, all Softmax activations of the classification layer determined.

The table 1 shows the layer parameters of the CNN model

Table 1: CNN layer parameters

S. No.	Layer Name	Description
1	Image Input	images with 'zerocenter' normalization
	Convolution	8 3x3 convolutions with stride [1 1] and padding 'same'
3	Batch Normalization	Batch normalization

4	ReLu	ReLu
5	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
6	Convolution	16 3x3 convolutions with stride [1 1] and padding 'same'
7	Batch Normalization	Batch normalization
8 9	ReLU Max Pooling	ReLU 2x2 max pooling with stride [2 2] and padding [0 0 0 0]
1	Convolution	32 3x3 convolutions with stride [1 1] and padding 'same'
11	Batch Normalization	Batch normalization
1	ReLU	ReLU
13	Fully Connected	3 fully connected layer
14	Softmax	softmax

C. Adaptive encoding of key frames

In this case of *adaptive quantization*. The goal is to adapt the quantification parameters at the source considered. Two approaches are generally considered, the *online* approach leading to an adaptation of the *retrograde* quantization, that is to say carried out at the same time than the quantification itself and an *offline* adaptation, where the parameters of quantization are calculated *directly*, i.e., by fixing *a priori* the parameters of the quantification.

Direct adaptive quantification

In this approach the transmitted video frames are divided into time blocks and each block is processed before quantification. The Spatial and temporal features are extracted and given as input to the deep learning algorithm. The parameters of the quantification are calculated based on this analysis. In this case, therefore, one needs to add to each coded block an additional information block to indicate to the decoder the quantization parameters which were finally retained. It is necessary then find a compromise between large blocks, which will then be more roughly described by the quantification parameters retained, and small blocks, which will lead to many information blocks.

To quantify the considered block, a possible strategy is then to consider a trained deep learning model. This paper presented a 1D CNN model to learn the spatial and temporal uncertainty features. The Uncertainty features of the test frame are extracted and given to the CNN model. The adaptive quantization model is implemented using the uncertainty features and the levels of quantization are decided based on the CNN output

Non-uniform adaptive quantification

In this section a non-uniform adaptive quantization is presented. This method was originally developed for quantification problems but now has domains very varied applications, in particular to compression. The quantization block is present after the DCT block in the MPEG video compression standard. The sequence of steps involved in the proposed algorithm are listed below:

Algorithm 1

Step 1: Read the input frame.

Step 2: Extract spatial temporal uncertainty features from the frame and Classify them using trained 1D CNN model.

Step 3: Divide the frame into blocks.

Step 4: Apply DCT on the image blocks.

Step 5: Based on the Result of CNN, apply adaptive quantization the CNN.

The following section presents the experimental results.

IV. Experimental Results

This chapter presents the findings of research conducted to review the proposed procedure.The experiment is performed on widows core i3 processor with 4 GB RAM. The first phase consists of spatial and temporal features based NN training. Figure 3 shows the video frames used for training. The statistics of the videos chosen are shown in table 2.

Table 2: Video Specifications

	Video Name	**Video type**	**Original video size**	**Video Dimensions**	**Frame rate of the video**	**Video Duration**	**Total frames**
1.	bmx-bumps	avi	105 MB	480*854	12 f/s	7 sec	86
2.	car-turn	avi	93.9 MB	480*854	12 f/s	6 sec	74
3.	dog-gooses	avi	101 MB	480*854	12 f/s	7 sec	86

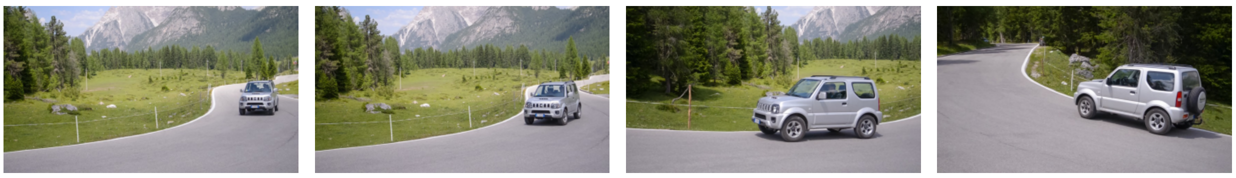
(a) 4.tif

(b) 15.tif

(c) 27.tif

(d) 32.tif

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 08 Issue 02
February 2021



(e) 7.tif (f) 21.tif (g) 50.tif (h)78.tif

Fig 3. Training video frames (a) - (d) bmx-bumps video frames, (e) - (f) car-turn video frames

These spatial and temporal uncertainty features extracted from these frames are given to the CNN model to extract the context aware saliency features. The training accuracy and error rate are shown in figures 4 and 5. The resultant frames are shown in figure 6.

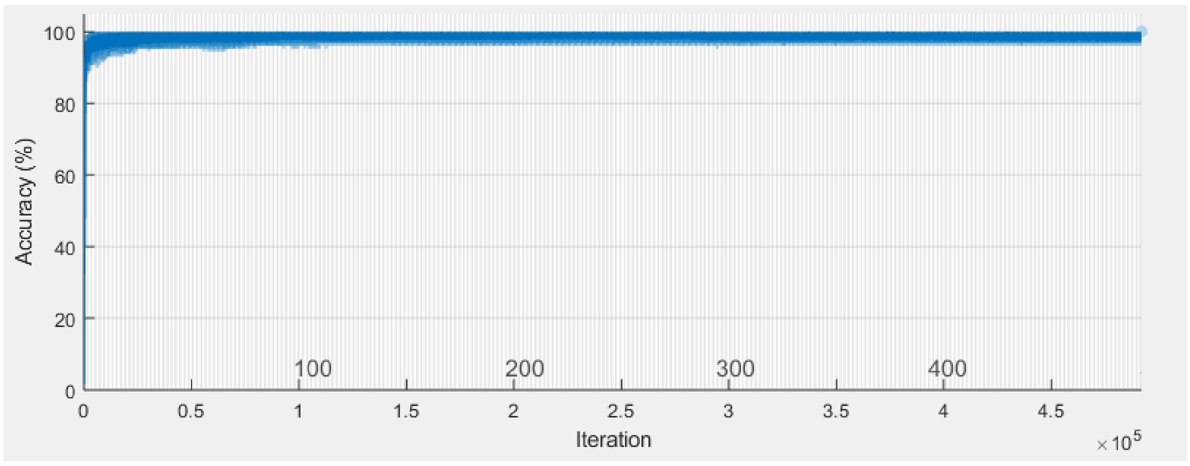


Figure 4. Training Accuracy

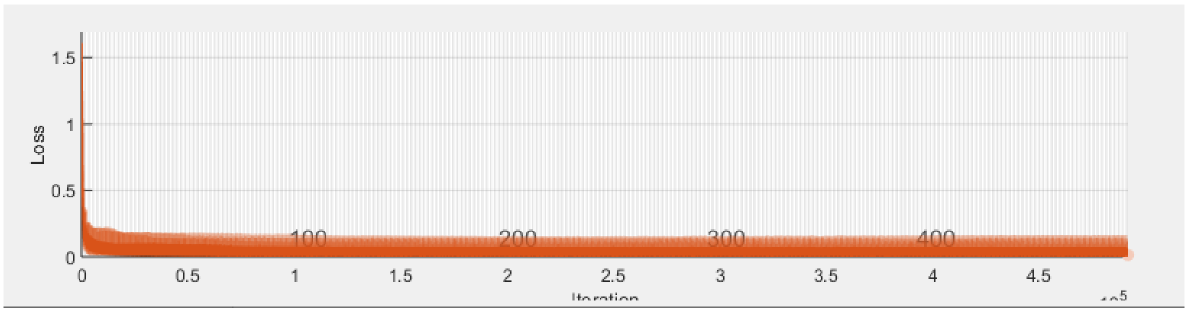


Figure 5: Training Loss

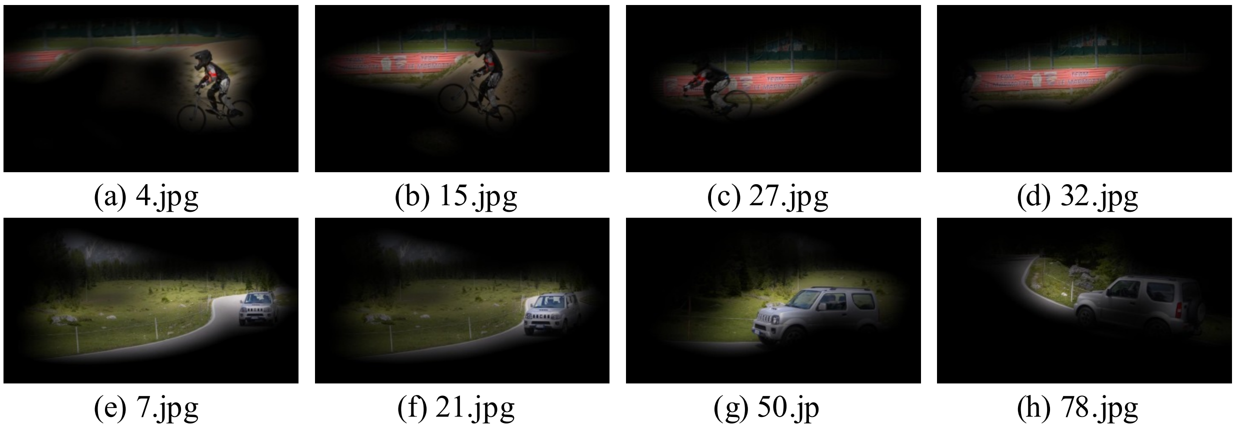


Fig 6. Spatio-temporal uncertainty feature extraction using CNN (a) - (d) bmx-bumps video feature extraction frames (e) - (h) car-turn video feature extraction frames

In the testing phase, a new video is taken as input for compression. The Spatial and temporal features are extracted and given to the trained CNN model for classification. The input video frames are shown in figure 7 and the CNN results are shown in figure 8.

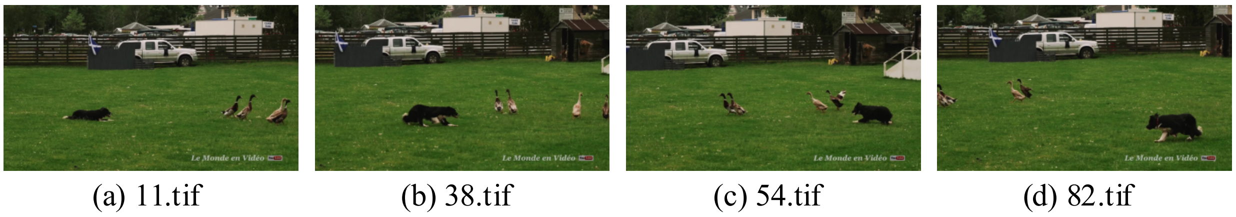
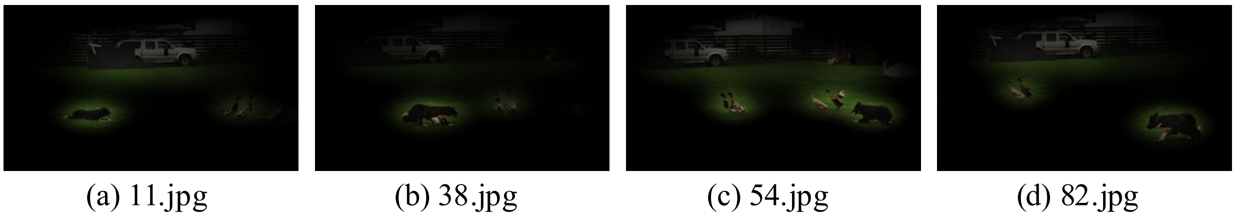


Fig 7. Testing video frames - (a) - (d) dog-gooses video frames



(a) - (d) car-turn video feature extraction frames

Fig 8. Spatio-temporal uncertainty feature extraction using CNN

The obtained results are evaluated in terms of:

i) **Mean Square Error (MSE):** The MSE is shown in the formula

$$\text{MSE} = \frac{1}{M\times N}\sum_{i=1}^{N}\sum_{j=1}^{M}[I(i,j) - I'(i,j)]^2$$

Where I and I' are the input and output images respectively,

i, j represent the image pixel coordinates

M, N are the dimension of the image.

ii) **Peak Signal to Noise Ratio (PSNR):** The expression of PSNR is given by

$$\text{PSNR} = 10.log_{10}(\frac{MAX^2}{MSE})$$

Where MAX represents the maximum value of pixel in the image

iii) **Structural Similarity Index (SSIM):** The SSIM is calculated as shown below

$$\text{SSIM} = \frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)}$$

μ_x, μ_y are the mean of x and y respectively

σ_x^2, σ_y^2 are the variance of x and y

σ_{xy} is the covariance

c_1, c_2 are variables

Figures 9, 10 and 11 represent the MSE, PSNR and SSIM of the proposed method.

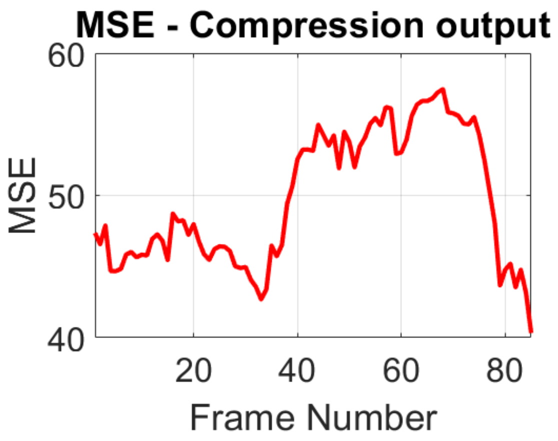


Fig9. Proposed Method MSE

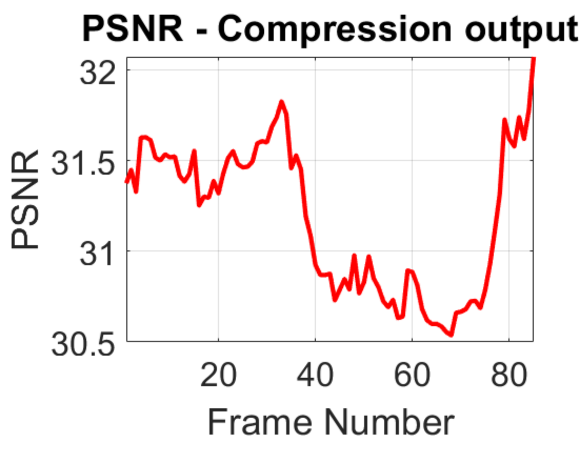


Fig 10. Proposed method PSNR

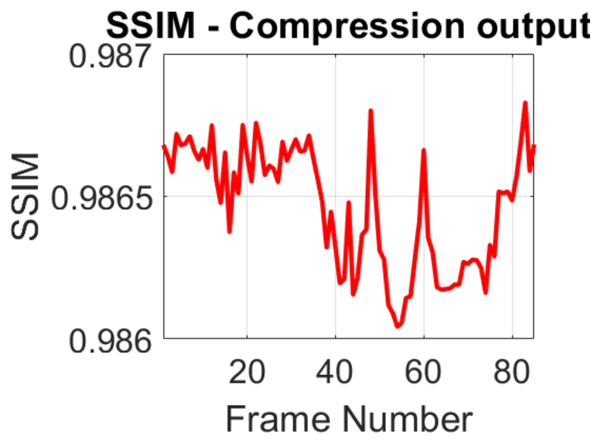


Fig 11. Proposed method SSIM

Table 3shows compression ratios of the proposed method and the existing techniques. Proposed method is compared with normal DCT compression [24] and DWT compression [25] in terms of compression ratio, MSE, PSNR and SSIM.Change in compression ratio is approximately 2.

Table 3. Compression ratio comparative results

Video Name	Original size	Normal Compression		DWT compression		Proposed Adaptive compression	
		Compression video Size	Compression Ration	Compression video Size	Compression Ration	Compression video Size	Compression Ration
bmx-bumps	105.69 MB	4.32 MB	24.47	4.54	23.28	4.66 MB	22.67

car-turn	93.96 MB	5.01 MB	18.75	5.18	18.14	5.32 MB	17.67
dog-gooses	101.00 MB	5.64 MB	17.91	5.82	17.35	5.91 MB	17.10

Table 4 shows the comparison of the proposed method with the existing methods with respect to PSNR, MSE and SSIM. The quality of the output frame is enhanced by approximately 50 percentage.

Table 4. Image Quality parameters

Video Name	Normal DCT Compression			DWT compression			Adaptive compression		
	PSNR	MSE	SSIM	PSNR	MSE	SSIM	PSNR	MSE	SSIM
bmx-bumps	14.67	2466.54	0.61	18.19	986.38	0.79	23.59	315.79	0.93
car-turn	17.67	1274.99	0.78	19.32	760.86	0.82	23.88	266.26	0.96
dog-gooses	24.46	233.41	0.95	26.80	135.75	0.87	31.18	49.75	0.98

V. Conclusion

Most vision tasks related to video, such as identification of video objects and video objects, are sensitive to compressed video quality and can benefit from effective video compression for other visual tasks. A program needs to maintain its integrity in order to function, so it is not conceivable to rebuild a program by omitting bits here and there and adding where they don't. The lossy compression allows to eliminate some information to have an optimized compression as much as possible, while keeping a result which is as close as possible to the original data. This paper presented a spatial and temporal feature based uncertainty extraction for the extraction of contextual information. This information is fed to a 1D CNN model for training. Once the model is ready, the features are extracted and classified on the test frames. After block division, an adaptive quantization model is applied on the blocks based on the results of the CNN model. The blocks with important information is quantized less when compared to the others. The proposed approach has been compared with conventional DCT and DWT approaches. The Quality of the output increased by 60% when compared to DCT and 27% when compared with DWT based on the PSR values. The reduction in compression ratio is far less when compared to the percentage increase in the quality of the compressed video.

References

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 08 Issue 02
February 2021

[1] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Kr¨ahenb¨uhl. Compressed video action recognition. In*CVPR*, pages 6026–6035, 2018.

[2] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning,trained quantization and huffman coding. arXiv preprintarXiv:1510.00149, 2015.

[3] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar. Variable rate image compression with recurrent neural networks. arXiv preprint arXiv:1511.06085, 2015.

[4] J. Ball´e, V. Laparra, and E. P. Simoncelli. Endto-end optimized image compression. arXiv preprint arXiv:1611.01704, 2016

[5] T.-W. Hui, X. Tang, and C. Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 8981–8989, 2018.

[6] G. K. Wallace. The jpeg still picture compression standard. IEEE Transactions on Consumer Electronics, 38(1):xviii–xxxiv, 1992

[7] A. Skodras, C. Christopoulos, and T. Ebrahimi. The jpeg 2000 still image compression standard. IEEE Signal Processing Magazine, 18(5):36–58, 2001

[8] F. bellard, bpg image format. http://bellard.org/bpg/. Accessed: 2021-01-03.

[9] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman. Video enhancement with task-oriented flow. *arXiv preprint arXiv:1711.09078*, 2017

[10] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h. 264/avc video coding standard. TCSVT, 13(7):560–576, 2003.

[11] G. J. Sullivan, J.-R. Ohm, W.-J. Han, T. Wiegand, et al. Overview of the high efficiency video coding(hevc) standard. TCSVT, 22(12):1649–1668, 2012.

[12] J. Ball´e, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston. Variational image compression with a scale hyperprior. arXiv preprint arXiv:1802.01436, 2018

[13] L. Theis, W. Shi, A. Cunningham, and F. Husz´ar. Lossy image compression with compressive autoencoders. arXiv preprint arXiv:1703.00395, 2017

[14] M. Li,W. Zuo, S. Gu, D. Zhao, and D. Zhang. Learning convolutional networks for content-weighted image compression. In CVPR, June 2018.

[15] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. In CVPR, volume 2, page 2. IEEE, 2017.

[16] O. Rippel and L. Bourdev. Real-time adaptive image compression. In ICML, 2017

[17] D. Minnen, G. Toderici, M. Covell, T. Chinen, N. Johnston, J. Shor, S. J. Hwang, D. Vincent, and S. Singh. Spatially adaptive image compression using a tiled deep network. In ICIP, pages 2796–2800. IEEE, 2017

[18] T. Chen, H. Liu, Q. Shen, T. Yue, X. Cao, and Z. Ma. Deepcoder: A deep neural network based video compression. In VCIP, pages 1–4. IEEE, 2017

[19] Z. Liu, X. Yu, Y. Gao, S. Chen, X. Ji, and D. Wang. Cu partition mode decision for hevc hardwired intra encoder using convolution neural network. TIP, 25(11):5088–5103, 2016.

[20] R. Song, D. Liu, H. Li, and F. Wu. Neural network-basedarithmetic coding of intra prediction modes in hevc. In *VCIP*, pages 1–4. IEEE, 2017.

[21] Z. Chen, T. He, X. Jin, and F. Wu. Learning for video compression. arXiv preprint arXiv:1804.09869, 2018

[22] Y.-H. Tsai, M.-Y. Liu, D. Sun, M.-H. Yang, and J. Kautz. Learning binary residual representations for domain-specific video streaming. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[23] C.-Y. Wu, N. Singhal, and P. Krahenbuhl. Video compression through image interpolation. In ECCV, September 2018.

[24] Sandy T, Theint K, “Analysis of video compression algorithms using DCT, DWT and FFT”, Nat J Parallel Soft Comp 1(1):22–27, 2019

[25] Yamnenko, Iuliia, and Vitalii Levchenko. "Video-Data Compression Using Wavelet Analysis." In 2019 IEEE 20th International Conference on Computational Problems of Electrical Engineering (CPEE), pp. 1-4. IEEE, 2019.