Computer Systems Science & Engineering
DOI: 10.32604/csse.2023.030513
Article

Tech Science Press

Effective and Efficient Video Compression by the Deep Learning Techniques

Karthick Panneerselvam[1,2,*], K. Mahesh[1], V. L. Helen Josephine[3] and A. Ranjith Kumar[2]

[1]Department of Computer Applications, Alagappa University, Karaikudi, India
[2]Department of Computer Science and Engineering, lovely professional University, Phagwara, Punjab, India
[3]School of Business and Management, Christ University, Bengaluru, Karnataka, India
*Corresponding Author: Karthick Panneerselvam. Email: karthickpanneerselvam14@gmail.com
Received: 28 March 2022; Accepted: 10 May 2022

Abstract: Deep learning has reached many successes in Video Processing. Video has become a growing important part of our daily digital interactions. The advancement of better resolution content and the large volume offers serious challenges to the goal of receiving, distributing, compressing and revealing high-quality video content. In this paper we propose a novel Effective and Efficient video compression by the Deep Learning framework based on the flask, which creatively combines the Deep Learning Techniques on Convolutional Neural Networks (CNN) and Generative Adversarial Networks (GAN). The video compression method involves the layers are divided into different groups for data processing, using CNN to remove the duplicate frames, repeating the single image instead of the duplicate images by recognizing and detecting minute changes using GAN and recorded with Long Short-Term Memory (LSTM). Instead of the complete image, the small changes generated using GAN are substituted, which helps with frame-level compression. Pixel wise comparison is performed using K-nearest Neighbours (KNN) over the frame, clustered with K-means and Singular Value Decomposition (SVD) is applied for every frame in the video for all three colour channels [Red, Green, Blue] to decrease the dimension of the utility matrix [R, G, B] by extracting its latent factors. Video frames are packed with parameters with the aid of a codec and converted to video format and the results are compared with the original video. Repeated experiments on several videos with different sizes, duration, Frames per second (FPS), and quality results demonstrated a significant resampling rate. On normal, the outcome delivered had around a 10% deviation in quality and over half in size when contrasted, and the original video.

Keywords: Convolutional neural networks (CNN); generative adversarial network (GAN); singular value decomposition (SVD); K-nearest neighbours (KNN); stochastic gradient descent (SGD); long short-term memory (LSTM)

This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1 Introduction

Video compression plays an essential role in saving a large quantity of space needed to store data and also providing high-quality image and video services [1]. Over the past decade, machine learning methods based on deep neural networks have provided revolutionary advances across various computer vision applications, in particular for image/video processing [2].

Image compression has long been thought to be one of the tasks for which neural networks are excellent, but there has been no evidence that it is feasible to train a single neural network that is competitive across compression rates and image sizes [3].The Image and Video Process has received many achievements through rapid development of deep learning and neural networks. Deep learning is a type of machine learning technology that is distinct from standard machine learning techniques by its computational model, known as the deep artificial neural network, or deep network [4].

The compression is compressing the images by removing redundancy in the image. The compression will cut back the information needed for data transmission in a network and therefore the space for storing of a device, which is of nice worth in improvising network efficiency [5]. It can decrease the blurring outcomes and the artifacts and consequently produces excessive exceptional decoded video frames. The CNN based in-loop filtering and immense freedoms in the field of learning-based video compression. These CNN characteristics are then compared across images to identify near-duplicate variants of the query. We use the sum-pooling operation on convolutional feature maps (CFMs) to produce global CNN models, and then match these global CNN models between a given query picture and database images to conveniently filter out some of the query's irrelevant images [6]. Applying deep learning to realize high data compression with low procedure quality is comparatively new and currently seen as a rising trend within the field of multimedia [7].

Generative adversarial network is employed to get rid of the visual object of compressed video. The output image will effectively guide the differentiator of GAN to approximate the mapping between the encoded frame and also the original frame [8]. GAN have been broadly investigated in single-picture super-resolution and have acquired amazing performance. In that case, GAN has been demonstrated to have a great ability to reduce artifacts when improving image/video quality [9].

Takes in an ideal minimized portrayal from an info picture, which safeguards the primary data and is then encoded utilizing a picture codec [10]. All our designs comprise of a recurrent neural network (RNN)-based encoder and decoder, a binarize, and a neural network for entropy coding [11]. They applied a recurrent neural network (RNN) architecture and outperformed JPEG significantly [12].

The hybrid coding architecture with motion-compensated prediction and residual transform coding is followed by all these algorithms. These algorithms, however, rely on hand-made modules to reduce spatial and temporal redundancies, e.g., block-based motion estimation and discrete cosine transformation (DCT), sequences in the video and which provide scalability [13]. Transform coding primarily overcomes the problem of correlation redundancy. However, this results in a loss in image resolution, an increase in block effect, and image noise, all of which have an impact on human visual perception [14].

Recently, auto—based deep neural network (DNN), picture compression encoders [15] performed equally or even better than conventional image codecs, such as JPEG [16] and JPEG2000 [17]. A semantic analysis-based end-to-end deep learning image compression framework is proposed, consisting of a semantic analysis network based on a CNN and an image compression network based on a RNN.

As an initial input, a JPEG image that has to be compressed is provided. This input image is saved as an integer array. SVD is used to compress the RGB components of the input JPEG image and its operations on matrices [18]. After approximating S matrix, the resultant decomposed matrix is regenerated [19]. SVD is an orthogonal matrix decomposition method that is both strong and efficient. Singular Value Decomposition

represents image data regarding the number of eigen vectors that depend on the image's dimension. Compression is achieved by exploiting an image's psycho visual redundancies [20].

Deep neural networks (DNN), one of the most engaging machine learning approaches, have proven great performance in a variety of intelligence tasks such as image compression. Also, existing methods often attempt to use DNN for video compression. The Video Codec divides the videos into frames as macroblocks [21].

Section 2 reviews some literatures which presented video compression techniques. Section 3 presents the methodologies. Section 4 explains the materials and methods. Results of the methodologies are discussed in Section 5. The conclusion of the manuscript is explained in Section 6.

2 Related Works

With substantial improvements in user numbers and the adoption of modern immersive formats and higher formats video content is now by far the biggest consumer of global internet bandwidth, with user efficiency standards. Increased demand is placing a strain on all distribution technology, especially video compression, which plays a key role in balancing network bandwidth and consumer video quality demands.

The video file is composed of a series of images that need to be framed in a single entity. Image compression is an efficient method for reducing storage capacity and increasing transmission speeds. The point of picture compression is to lessen the immateriality and excess of a picture to store or communicate the image at low piece rates. Image compression has generally been one of the tasks for which neural Networks were suspected to be acceptable, but there was little proof that it was feasible to train a single neural network that would be serious across compression rates and picture size [22]. Transform-based image compression methods can efficiently reduce spatial redundancy [22]. Those techniques (for example, JPEG [23], and JPEG2000 [24] are still the most commonly used image compression algorithms today. Deep learning-based image compression methods [24] were recently proposed and demonstrated to be latest.

Deep image compression works by converting input images into quantized bit streams that can then be compressed further using lossless coding algorithms. Any models [24] used RNNs to compact the images in a progressive manner to achieve this goal.

First introduced a simple RNN-based approach to compress the image and further proposed a method, which enhances the performance by progressively compressing reconstructed residual information [25] Introduced a continuous and differentiable proxy for the rate-distortion loss and further proposed a variational auto-encoder-based compression algorithm [26]. Pixels are used to represent every image. Pixels represent the image's intensity. These pixel values are organized in the form of a matrix [27].

Current picture coding algorithms based on learning can be divided into two key groups. The first concerns end-to-end training and optimization using architectures of the auto-encoder sort While the solutions in this category do not yet complete with the new standardized codecs, such as VVC (Versatile Video Coding) and AV1, they display considerable promise for the future. A second class includes algorithms that are designed to develop individual coding tools within a standard codec configuration. The semantic analysis networks are in control of extracting the image's relevant semantic areas using CNN and estimating the compression level based on the semantic relevance of each image block [28]. There is one category of methods among these CNN-based coding tools that stand out by providing greater coding benefits compared to others. To improve the quality of the reconstructed video frames, they conducted CNN operations on the decoder. While most PP approaches focus on single-frame enhancement, multi-frame enhancement methods have been reported in some recent contributions [29]. Because these are associated with much higher computational values, however, complexity, here we only address single-frame enhancement.

Srinivasan et al. [30] had proposed a speedy (quick) inter-coding unit decision algorithm based on deep learning for video compression using HEVC. In this approach, estimated motion vectors were clustered using fuzzy c-means clustering algorithm. By presenting this algorithm, the authors enhanced the coding performance of HEVC. Soulef Bouaafia et al. [31] had presented a wide-activated squeeze-and-excitation deep CNN to enhance the quality for versatile video coding. The authors had enhanced the performance in terms of RD cost. Amna et al. [32] had proposed a deep CNN based fast Quard-Tree partitioning for predicting coding unit partition. Because of the proposed scheme, the authors had enhanced the performance of quad-tree binary-tree (QTBT) for intra-mode coding. Zaki et al. [33] had presented CtuNet to partition the coding tree unit by approximating its functionality using deep learning methods. For predicting the coding tree unit partition of the HEVC standard, ResNet18-CNN model was adopted. Because of the proposed scheme, the authors had reduced the computational complexity.

By reviewing above articles, we inferred that computational complexity during video compression is further to be reduced. Also they didn't focus to remove duplicate frames from the video frames. Besides we focus to introduce hybrid deep learning techniques to increase the compression ratio.

3 Methodology

We offer low-complexity strategies in this research, which involve the following phases. Initially, the video was split into frames. A video file has to be segmented into pictures and scenes. The quality of the video Spilt-up has a direct impact on the quality of the products and can improve these Compression outcomes. The first stage in processing video is to divide them into shots and extract representative frames, often known as key frames.

The fast development of deep learning networks for detecting tasks has significantly improved the performance of object detectors [13]. The images are compared and the duplicates and the repeated frames per second are removed using CNN. CNNs utilize parameterized, sparsely connected kernels which preserve the spatial characteristics of images. Because of the impressive growth of CNNs in the field of computer vision, present approaches for detecting near-duplicate pictures compare the global or local CNN characteristics between images.

The next step is to identify the minute changes and repeat the single image in place of the duplicate image. Slight changes in the frames were identified by GAN and recorded with LSTM. For calculating the similarity between frames, the Nasnet model output vector of the current frame is used as input to an LSTM model. GAN stabilizes the training process and further minimizes the difference between generator output and training target. Using GAN and LSTM, small changes are replaced in the place of the complete image. The original Image has a semantic label map, our technique may fully synthesize irrelevant areas in the decoded image such as light and trees from the label map, lowering storage costs accordingly. The prepared data in secret layer units will, in general, be dispersed in a more extensive territory.

The pixel-wise comparison is done by using KNN to check the pixel comparison of the original and compressed frames. The KNN algorithm is a popular non-parametric classification approach. Finally, with the help of codec video frames are packed with criteria and converted to video format and the results in compressed video size are compared to the original video.

4 Materials and Methods

4.1 Architecture

The architecture of the proposed framework is shown in Fig. 1, of this article. In this work, we propose the deep-learning-based most efficient video compression technique. This is achieved by the modified usage

of CNN to identify similar images and store coordinates of unique portions of those images in a single frame. Traits of classified data are stored in LSTM by CNN in the form of a dictionary.

$$\{\boldsymbol{image}1:\ \boldsymbol{spec}1,\ \boldsymbol{image}2:\boldsymbol{spec}2,..\}\ \boldsymbol{spec}\ -> [\boldsymbol{Im\_no},[(\boldsymbol{x}1,\boldsymbol{y}1),(\boldsymbol{x}2,\boldsymbol{y}2)]]$$

where spec is a list of specifications which contains image number in that frame (Im_no), top-left (x1,y1) and bottom-right (x2,y2) rectangle coordinates of unique portion in that image. GAN is used to generate a unique portion of the image so that the whole image can be replaced by the unique portion and appended with binary addition during video creation. K-means clustering is performed on RGB pixels of every image in the list, which helps in fast computation and frame-level reduction of size. Further size reduction is done by eliminating the unwanted regions of colors and keeping the first few horizontal and vertical rows of color pixels. This is achieved by performing SVD on every image. The above process is followed for all the images in the List and a dictionary of data is carried forward to the video codec which processes the data and produces a similar video of reduced size. This process is carried over the web through Amazon Web Services (AWS). A detailed explanation is given below.

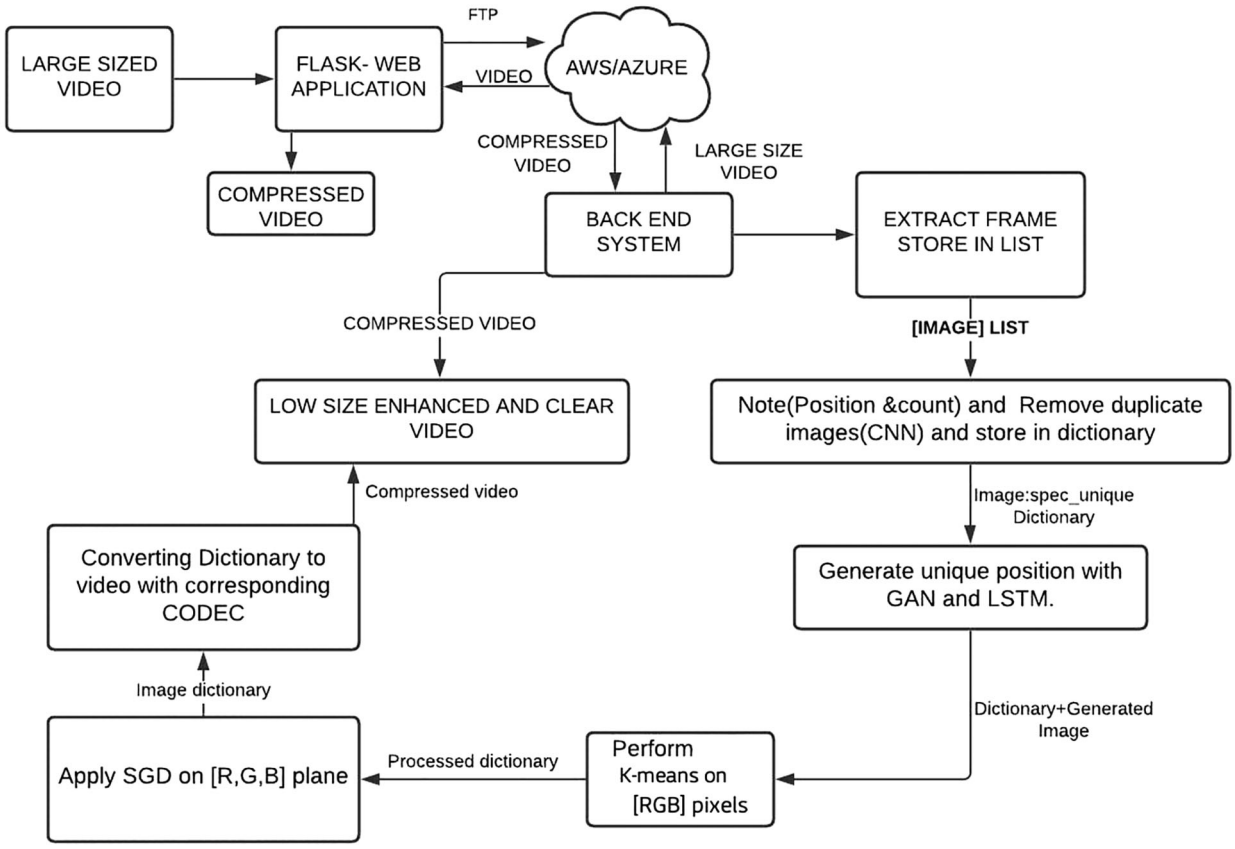
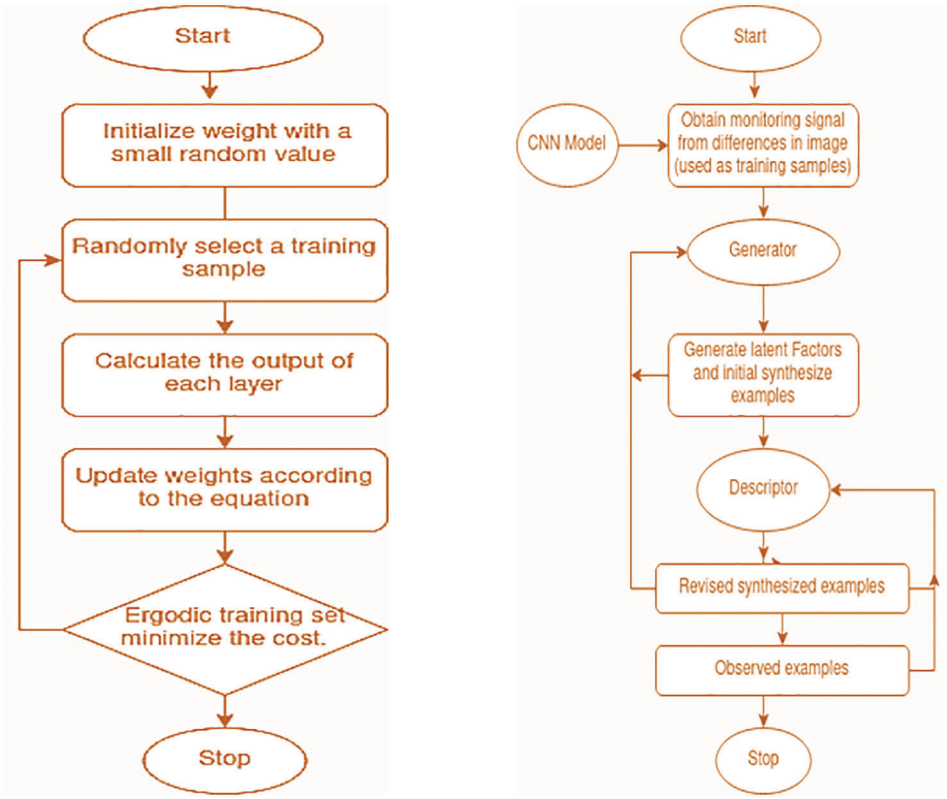


Figure 1: Architecture of the methodology

4.2 Flow Chart

Fig. 2 shows thatWeights are initialized with a random value by randomly selecting training samples from the extracted frames from the video total duration. The output of each layer in CNN is calculated and weights are updated according to the equation. Ergodic training sets are used to minimize cost if cost is not minimized. In the case of non-minimal and more deviating costs, the process is continued until a

minimized cost is produced after training. The trained model is stored in the form of md5 or h5 models for further training and prediction.



(a) Convolutional Neural Networks Flow Chart (b) Generative Adversarial Network

Figure 2: Deep learning techniques flow chart

A trained CNN model is fetched as an input for generating a GAN-based image. Using the trained samples monitoring signals are obtained in images. This image is passed to the generator model which uses generating equations to generate latent factors and initial synthesize examples which produce a sample image. This image is passed to the generator and descriptor model to validate with the obtained monitoring signal. In the passage of the descriptor, a revised synthesis is performed for further comparison with observed examples. This generates the unique portion of the image by eliminating unwanted portions.

4.3 Identify Video FPS and Extract Images

A Flask-based Web Application is developed to receive the input video file and transfer the file to Backend running on AWS. The video is split into frames using Codec, which has all information about the video to decode [21]. According to the Ubuntu documentation team, "A **codec** is a device or computer program which encodes or decodes a digital data stream or signal. **Codec** is a portmanteau of coder-decoder.".The OpenCV library is used to get the total video Frame per second (FPS), Image width, height and video formats like MP4, AVI, etc…

Fig. 3 shows how RGB colors are scattered throughout the video. This shows there are more unwanted color arrays that can be removed. Even after removing 50% of the color array, the quality will not get decreased as the change is negligible for the human eye.

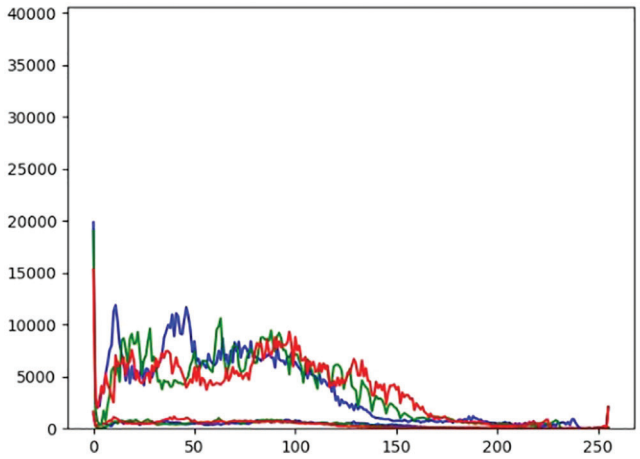


Figure 3: Visualizing video frame pixels as data

4.4 Identify and Generate Unique Portion from Image List With (CNN, GAN, LSTM)

Duplicate images from each frame are identified using CNN, before removing the similar image count and position is noted so the same image can iterate in place of the duplicate. Small changes in the frames are identified and generated using GAN. To restore video chrominance information, a GAN model was adopted. The slight changes in the frames are recorded by LSTM. LSTM network models are a type of recurrent neural network that can learn and remember over long sequences of input data.

With GAN and LSTM, unwanted sections of the image are removed and replaced with black or zeroes. During video creation, an image that is selected to replace the duplicate is added with the newly generated image using GAN to form the new image. Fig. 4 shows the generate image using GAN. This is achieved by storing generated images in LSTM and providing the trigger period with the count which will perform this operation.

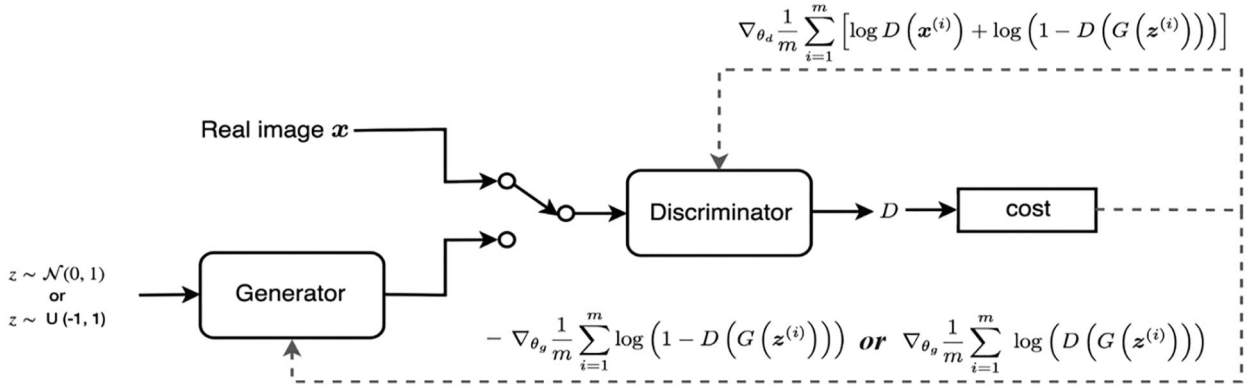


Figure 4: Generate image using GAN [14]

In Fig. 5 the picture at left shows the selected image, generated image using GAN is at the centre and produced new image is at right. This shows how CNN, GAN and LSTM work.

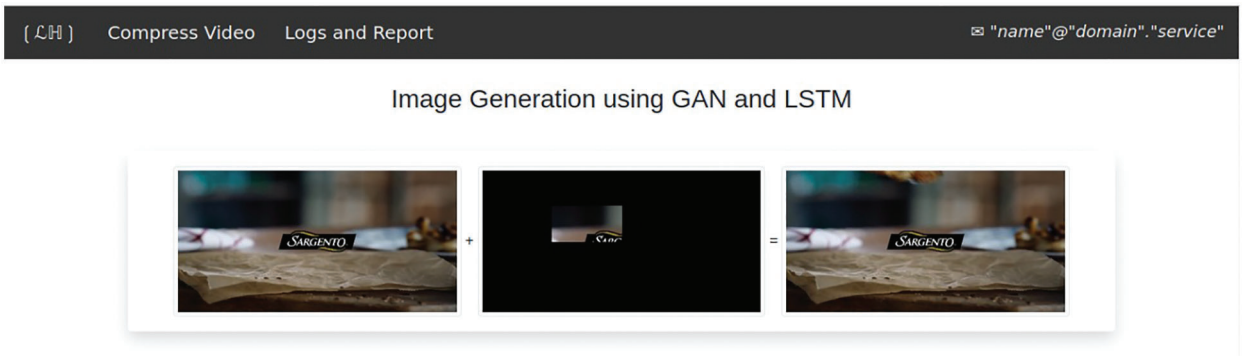


Figure 5: Adding generated image with selected image to form new image

4.5 K-Means Clustering On Image Pixels

Pixel-wise comparison is done using K-means clustering. In this case, the number of clusters is three (*i.e.*, R-Red, G-Green, B-Blue). A pixel-level traversal is performed so that different colored pixels are changed to their nearest neighbours. Doing this increase the time it takes for the Image frame to render, which internally reduces the size of the Image. This helps in pixel-level reduction in size.

Algorithm 1: The proposed method for pixel level k-means

Input: Image List

Output: Manipulated image list

Step 1: + Image_K means = [] # Empty list

Step 2: for image in Image_list

Step 3: colour,_ = find_Kmeans (image, K, max_iter)

Step 4: # K = desired number of colour

Step 5: xid = find_closest_centroid (image, colours)

Step 6: convert xid to array of datatype “numpy unit8”

Step 7: Reconstructed_image = colours [idx, :] * 255

Step 8: Compressed_image = image (Reconstructed_image)

Step 9: Image_Kmeans.append (compressed_image)

In the above algorithm input is the Image list which comes after performing CNN, LSTM and GAN, output will be an image list after K-means clustering. In step 1 we created an empty list named Image_Kmeans and used it to store manipulated images. A loop iterates over the image list to color code is found in step 3 where k is the desired number of colours. The closest centroid is found in step 5 using image and colors which are in turn converted to a NumPy array of type “uint8”. Reconstructed array values are converted to images to form compressed images. This process is repeated for all available frames and appended to the Image_Kmeans list to process Singular Value Decomposition (SVD).

4.6 Apply SVD on RGB Plane

SVD is a way to divide an image into three components. The first component is U-> unitary matrix ∑-> Rectangular diagonal V-> unitary matrix (A = U∑V). ∑ is the diagonal matrix which is returned by SVD where σ -> diagonal values which will be in below format are σi = i^{th} singular value and σ1 > σ2 > σ3>…

σ1	0	0
0	σ2	0
0	0	σ3

In our work, we are going to perform SVD on RGB-colored images where the image is split into red, green and blue colours from step 2 to step 4 and the singular value is set (K) in step 5. The left channel is calculated by multiplying the v-channel array with N-X-K rows and the diagonal of the S-channel. A-channel is the outcome of the cross-product of the left side and vertical height channel in step 9. Similarly, this operation is performed on all Red, Green and Blue colored images and by compressing the channel to "UINT8". The final image is produced by merging the compressed image of the three colored channels. This operation is performed on all images in the list to get a compressed image list and stored in the SGD Image list.

Algorithm 2: The proposed method for SDG image compression

Input: Image List

Output: Manipulated image list

Step 1: while image in Image_list

Step 2: image_R = Image [:, :, 0]

Step 3: image_G = Image [:, :, 1]

Step 4: image_B = Image [:, :, 2]

Step 5: set singular value K;

Step 6: RCB_list = [] # Empty list

Step 7: for col_image in [image_R, image_G, image_B]:

Step 8: leftside = multiply (vchanel [:, 0, k], diagonal of schanel)

Step 9: achanel = multiply (leftside, vhchanel [:, 0, k])

Step 10: compress achanel type to ('UNIT8')

Step 11: RGB_list.append(achanel);

Step 12: SGD_Image = merge (RGB_list)

Stochastic gradient descent on RGB plane as Singular Value Decomposition

4.7 Convert Processed Image Dictionary to Video

Finally, with the help of codec video frames are packed with the criteria and are converted to video format. In this stage codec is an encoder that compresses all the processed frames and converts them to video format. By using CNNs identified unique portions in duplicates are generated by GAN and LSTM Help to Process. This combined work reduces computation work and increases the rate of compression by decreasing the loss of quality which is negligible to the naked eye.

5 Result and Discussion

This process is carried out over the web and a downloadable result is produced with an analysed report on Fig. 6.

Figure 6: Web based video result and analysis

The performance of the application is calculated with the table system specification under isolated conditions. The Tab. 1 supports the Proof of Concept.

According to the results in Tab. 1, Some size videos were used in the experiments to find the compression size, FPS and time duration by our Compression method. The average completion time is 12.22 s for an average duration of 6.5 min.

We compare the colour pixels on Fig. 7 shows how red, green and blue colour pixel values are scattered before and after compression. This shows even after compression Image Quality is not reduced in a visible manner.

Table 1: Performance evaluation table

S.NO	Original size	Compressed size	FPS	Completion time	Video duration
1	5638.51 kB	478.5 KB	30	0 m 5.578 s	19 s
2	10240 KB	5120 KB	45	0 m 7.18 s	33 s
3	46899.2 KB	17821.69 KB	32	0 m 20.51 s	5 min
4	65324.13 KB	32761.20 KB	20	0 m 15.621 s	20 min
5	12610.72 KB	6141.52 KB	27	13424	162 s
6	14652.85 KB	4027.52 KB	27	8240	180 s
7	3872.67 KB	1247.92 KB	24	7290	14 s
8	662.45 KB	93.41 KB	21	2430	6 s
9	761.12 KB	204.08 KB	24	3520	10 s
10	5504.3 KB	975.45 KB	24	9040	11 s

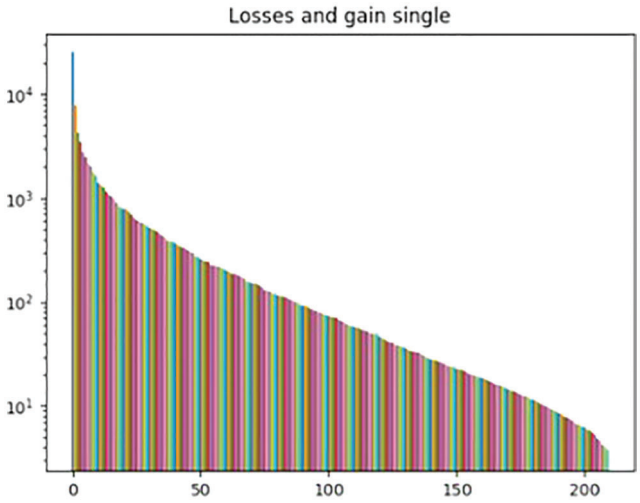


Figure 7: Plot losses and gain for single sample

Fig. 8 shows analysed report for maximum compression. Analysed report of the compressed videos shows video of 30 Fps & 24 Fps having 5.63 MB & 147619 MB can be reduced maximum up to 91.5% & 91.45% lesser than the older file, that is new file will have 478.5 KB & 1500 MB in size with same 30 FPS & 24 FPS (Frame per second).

Figs. 9J–9L shows how data is populated, for reference five random YouTube videos are taken and analysed. The video is taken in such a way that it has various qualities from lowest to highest, with various sizes and duration. The result came out to have at least a few percent of reduction in file size if the quality is too low and with high compression if quality is too high. This is because the system is capable of processing the frame pixel wise achieved using Python. Jinja plays a vital role as an intermediate between flask python code and front-end scripts like HTML, JavaScript and CSS along with Bootstrap. Fig. 9J shows the compilation time against the video size. This is clear that compilation time is always less when compared to video duration which is clear that there will be no longer wait that half of duration. Fig. 9K shows the performance Chart where there is always an increase in performance with increase in video size. In some cases, there is reduced performance because the input video quality is too low that the system is capable of identifying that further compression may lead to Quality Loss. This

shows the intelenge of the system in case of Low Quality, this is possible only when pixel wise compression is done. Fig. 9L shows the difference between original and compressed video size. Whatever the size is there is at least a bit of compression visible to see, for some cases the compression is low and for some it's high. This deviation is because of the intelligence of the system in finding the quality of video and overcoming the problem of over compression.

Figure 8: Analysed Report for maximum compression

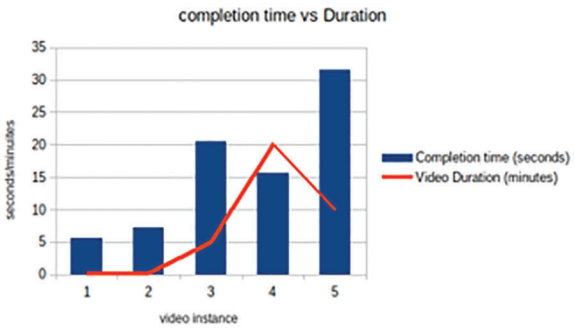


Fig. J. Completion time vs Duration

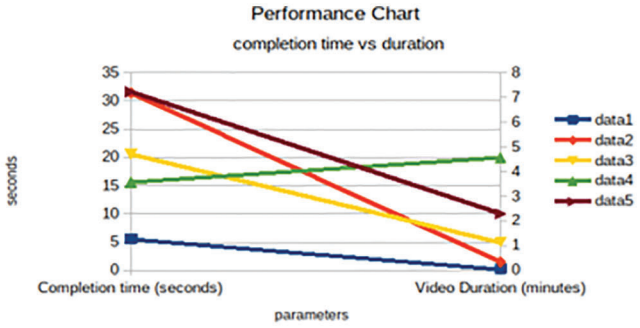


Fig. K. Performance Chart (Completion Vs Duration)

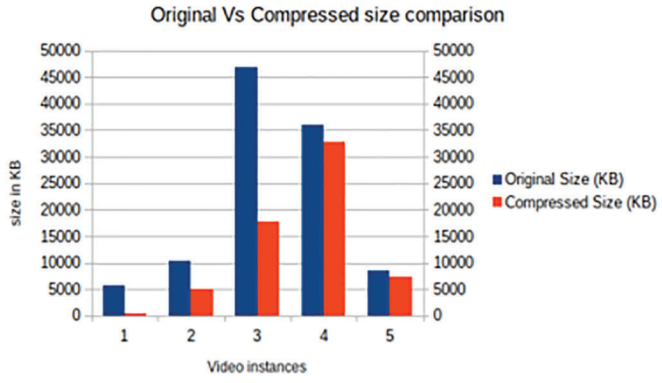


Fig. L. comparison chart between original and compressed video

Figure 9: Compression chart

Fig. 10a depicts the comparison of PSNR of proposed method and existing versatile video coding (VVC) method. As depicted in the figure, the PSNR of the proposed method is increased to 3% than that of VVC method. The comparison of SSIM of different compression methods is illustrated in Fig. 10b. Compared to existing VVC method, SSIM of the proposed method is increased to 4.5%. Fig. 10c shows the comparative analysis of the compression ratio of different compression methods. As shown in the figure, compression ratio of the proposed scheme is achieved 50% than that of existing method. As depicted in Fig. 10d, RD cost of the proposed method is reduced than the existing method.

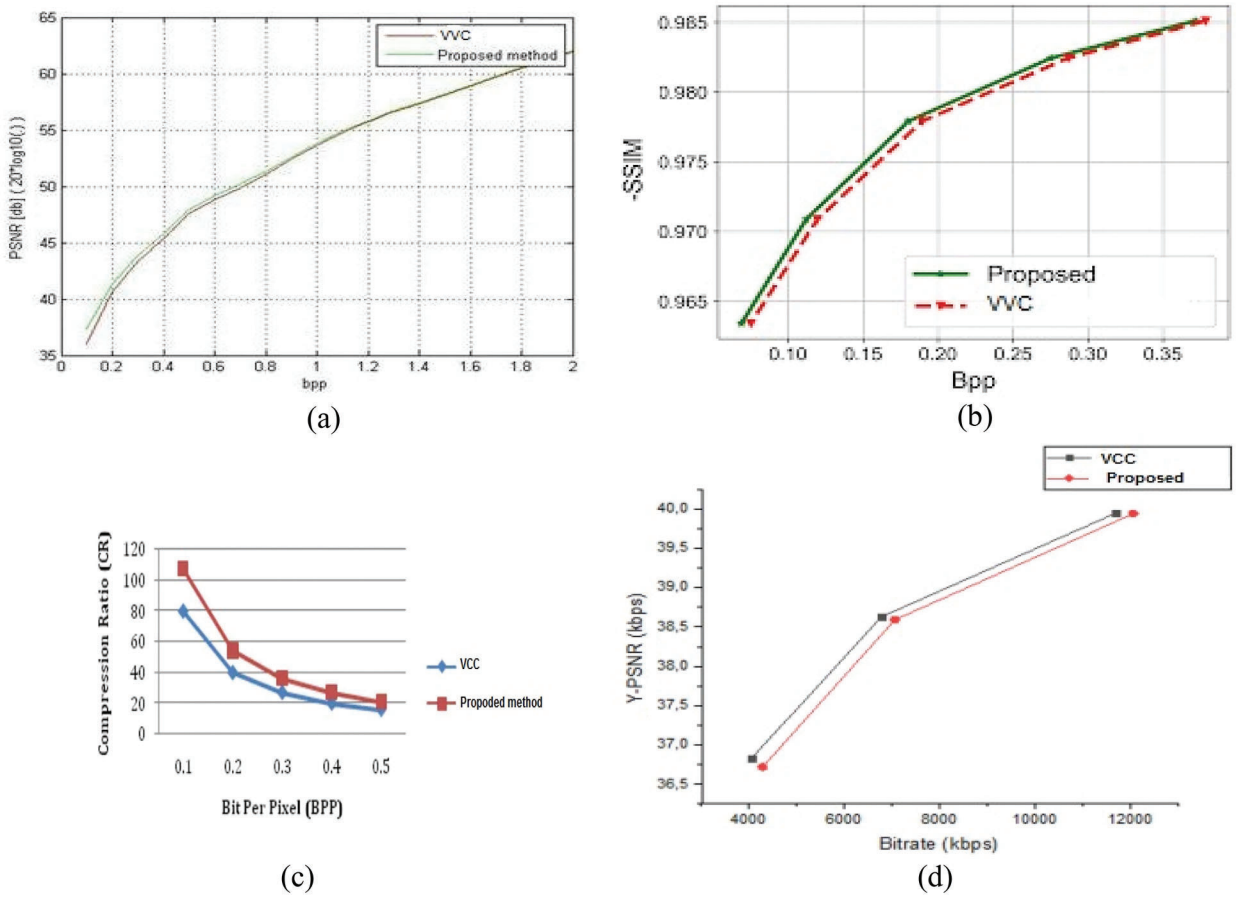


Figure 10: (a) PSNR Vs bpp, (b) SSIM Vs bpp, (c) compression ratio Vs bpp, (d) RD cost

6 Conclusion

In this paper, an effective and efficient video compression technique based deep learning methods has been presented. CNN has been used to remove duplicate frames. The minute changes and repeat the single image in place of the duplicate image have been identified using GAN and recorded with LSTM. For calculating the similarity between frames, the Nasnet model output vector of the current frame has been used as input to an LSTM model. GAN stabilized the training process and further minimized the difference between generator output and training target. Using GAN and LSTM, small changes have been replaced in the place of the complete image. Pixel wise comparison has been performed using KNN over the frame, clustered with K-means and Singular SVD has been applied for every frame in the video.

Simulation results showed that the proposed method attained better compression ratio and RD cost than the existing VVC method.

Acknowledgement: The authors with a deep sense of gratitude would thank the supervisor for his guidance and constant support rendered during this research.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] S. Ma, X. Zhang, C. Jia, Z. Zhao, S. Wang *et al.,* "Image and video compression with neural networks: A review," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 6, pp. 1683–1698, 2020.

[2] A. Voulodimos, N. Doulamis, A. Doulamis and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational Intelligence and Neuroscience*, vol. 2018, pp. 1–13, 2018.

[3] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen *et al.,* "Variable rate image compression with recurrent neural networks," *arXiv preprint*, arXiv:1511.06085, 2015.

[4] D. Liu, Z. Chen, S. Liu and F. Wu, "Deep learning-based technology in responses to the joint call for proposals on video compression with capability beyond HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 5, pp. 1267–1280, 2020.

[5] C. Wang, Y. Han and W. Wang, "An end-to-end deep learning image compression framework based on semantic analysis," *Applied Sciences*, vol. 9, no. 17, pp. 3580, 2019.

[6] Z. Zhou, K. Lin, Y. Cao, C. Yang and Y. Liu, "Near-duplicate image detection system using coarse-to-fine matching scheme based on global and local CNN features," *Mathematics*, vol. 8, no. 4, pp. 644, 2020.

[7] A. K. Sinha and D. Mishra, "T3D-Y Codec: A video compression framework using temporal 3-D CNN encoder and Y-style CNN decoder," in *2020 11th Int. Conf. on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–7, 2020.

[8] Z. Zhao, Q. Sun, H. Yang, H. Qiao, Z. Wang *et al.,* "Compression artifacts reduction by improved generative adversarial networks," *EURASIP Journal on Image and Video Processing*, vol. 2019, no. 1, pp. 2341, 2019.

[9] T. Wang, J. He, S. Xiong, P. Karn and X. He, "Visual perception enhancement for hevc compressed video using a generative adversarial network," in *2020 Int. Conf. on UK-China Emerging Technologies (UCET)*, pp. 1–4, 2020.

[10] F. Jiang, W. Tao, S. Liu, J. Ren, X. Guo *et al.,* "An end-to-end compression framework based on convolutional neural networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 3007–3018, 2018.

[11] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen *et al.,* "Full resolution image compression with recurrent neural networks," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 5306–5314, 2017.

[12] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li *et al.,* "A survey of deep learning-based object detection," *IEEE Access*, vol. 7, pp. 128837–128868, 2019.

[13] L. Tan, Y. Zeng and W. Zhang, "Research on image compression coding technology," *Journal of Physics: Conference Series*, vol. 1284, no. 1, pp. 12069, 2019.

[14] V. Verma, N. Agarwal and N. Khanna, "DCT-domain deep convolutional neural networks for multiple JPEG compression classification," *Signal Processing: Image Communication*, vol. 67, pp. 22–33, 2018.

[15] Z. Liu, T. Liu, W. Wen, L. Jiang, J. Xu *et al.,* "DeepN-JPEG: A deep neural network favorable JPEG-based image compression framework," in *Proc. of the 55th Annual Design Automation Conf.*, pp. 1–6, 2018.

[16] K. Mounika, D. S. N. Lakshmi and K. Alekya, "SVD based image compression," *International Journal of Engineering Research and General Science*, vol. 3, no. 2, pp. 1271–1278, 2015.

[17] S. Kahu and R. Rahate, "Image compression using singular value decomposition," *International Journal of Advancements in Research & Technology*, vol. 2, no. 8, pp. 244–248, 2013.

[18] A. Chadha, A. Abbas and Y. Andreopoulos, "Video classification with CNNs: Using the codec as a spatio-temporal activity Sensor," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 2, pp. 475–485, 2019.

[19] S. Thayammal and D. Selvathi, "A review on transform based image compression techniques," *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 10, pp. 1589–1596, 2013.

[20] V. Y. Varma, T. N. Prasad and N. V. P. S. Kumar, "Image compression methods based on transform coding and fractal coding," *International Journal of Engineering Sciences and Research Technology*, vol. 6, no. 10, pp. 481–487, 2017.

[21] G. Sullivan, J. Ohm, W. Han and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[22] A. Habibian, T. V. Rozendaal, J. M. Tomczak and T. S. Cohen, "Video compression with rate-distortion autoencoders," in *Proc. of the IEEE/CVF Int. Conf. on Computer Vision*, pp. 7033–7042, 2019.

[23] L. Zhou, C. Cai, Y. Gao, S. Su and J. Wu, "Variational autoencoder for low bit-rate image compression," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, pp. 2617–2620, 2018.

[24] S. Wenger, "H.264/AVC over IP," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 645–656, 2003.

[25] L. Theis, W. Shi, A. Cunningham and F. Huszár, "Lossy image compression with compressive autoencoders," *arXiv preprint*, arXiv:1703.00395, 2017.

[26] M. Bouyahi and Y. Ayed, "Video scenes segmentation based on multimodal genre prediction," *Procedia Computer Science*, vol. 176, no. 5, pp. 10–21, 2020.

[27] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte and L. V. Gool, "Generative adversarial networks for extreme learned image compression," in *Proc. of the IEEE/CVF Int. Conf. on Computer Vision*, pp. 221–231, 2019.

[28] H. Ahn and C. Yim, "Convolutional neural networks using skip connections with layer groups for super-resolution image reconstruction based on deep learning," *Applied Sciences*, vol. 10, no. 6, pp. 1959, 2020.

[29] J. Salvador-Meneses, Z. Ruiz-Chavez and J. Garcia-Rodriguez, "Compressed kNN: K-Nearest neighbors with data compression," *Entropy*, vol. 21, no. 3, pp. 234, 2019.

[30] A. Srinivasan and G. Rohini, "An Improvised video coding algorithm for deep learning-based video transmission using HEVC," *Soft Computing*, vol. 23, no. 18, pp. 8503–8514, 2019.

[31] S. Bouaafia, R. Khemiri, S. Messaoud, O. B. Ahmed and F. E. Sayadi, "Deep learning-based video quality enhancement for the new versatile video coding," *Neural Computing and Applications*, vol. 3, no. 6, pp. 1–15, 2021.

[32] M. Amna, W. Imen, S. F. Ezahra and A. Mohamed, "Fast intra-coding unit partition decision in H. 266/FVC based on deep learning," *Journal of Real-Time Image Processing*, vol. 17, no. 6, pp. 1971–1981, 2020.

[33] F. Zaki, A. E. Mohamed and S. G. Sayed, "CtuNet: A deep learning-based framework for fast CTU partitioning of H265/HEVC intra-coding," *Ain Shams Engineering Journal*, vol. 12, no. 2, pp. 1859–1866, 2021.