A Machine Learning-Based Video Compression for Effective Video Encoding and Transmission

[1]Bairavel S, [2]Lakshmi T K, [3]Praveen Gugulothu, [4]Abrar Ahmed Katiyan, [5]Chandravadhana S and [6]Helina Rajini Suresh

[1]Department of Artificial Intelligence and Data Science, KCG College of Technology, Chennai, Tamil Nadu, India.
[2]Department of Computer Science and Engineering, Malla Reddy University, Hyderabad, Telangana, India.
[3]Department of Computer Science and Engineering, Balaji Institute of Technology, Narsampet, Warangal, Telangana, India.
[4]Department of Computer Science and Engineering, C. Abdul Hakeem College of Engineering and Technology, Vellore, Tamil Nadu, India.
[5]Department of Mechatronics Engineering, Chennai Institute of Technology, Chennai, Tamil Nadu, India.
[6]Department of Electronics and Communication Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, Tamil Nadu, India.
[1]bairavel@gmail.com, [2]dr.lakshmitk@gmail.com, [3]ramjijyothi@gmail.com, [4]kaa406@yahoo.co.in, [5]chandravadhanas@citchennai.net, [6]helinarajini@gmail.com

Correspondence should be addressed to Bairavel S: bairavel@gmail.com

Article Info
Journal of Machine and Computing (https://anapub.co.ke/journals/jmc/jmc.html)
Doi: https://doi.org/10.53759/7669/jmc202505076
Received 12 September 2024; Revised from 19 December 2024; Accepted 27 February 2025.
Available online 05 April 2025.

Abstract – Deep Learning (DL) is revolutionizing video processing, as video is progressively key in daily life. Encoding and transmitting video effectively becomes challenging with fas t content resolution and data volume. This research presents the most progressive method for Video Compressing (VC), using DL to enhance encoding and transmission efficiency, demonstrating the need for more cutting-edge methods in digital media. This work uses advanced Machine Learning (ML) to reduce video data size without compromising video quality, enhancing its suitability for high-definition streaming and videoconferencing. The algorithm uses Convolutional Neural Network (CNN)+Recurrent Neural Network (RNN) to improve video quality. CNN captures complex spatial details within each video frame, while LSTM relates across time. The proposed VC achieves high video quality rates compared to traditional methods like H.264 and H.265. It adapts in real-time and optimizes video bandwidth usage, making it useful for live streaming services and video conferencing. The VC has been tested extensively, demonstrating significant bit rate reduction while maintaining excellent video quality. It surpasses modern compression methods, making it a flexible solution to the increasing demands for the best video content. This invention in VC is expected to change digital media distribution for good.

Keywords – Video Compression, Deep Learning, Video Encoding, Video Transmission, Bandwidth Optimization.

I. INTRODUCTION

The exponential development of digital media consumption is driven by the tremendous popularity of streaming high-definition services and video conferencing applications. Traditional Video Compression (VC) has proven successful, but balancing efficiency with quality has caused problems in storage and transmission, especially over networks of low bandwidth. Balancing the quality and efficiency of VC has always been a challenge in this area. VC algorithms are designed nowadays to balance both. However, the Machine Learning (ML) algorithms provide a new look at the VC task, leading to better results. As the drive for better video quality and smoother playback continues to rise, it increasingly puts pressure on effective compression methods that cope well with high resolutions and higher Video Frame (VF) rates. Leveraging ML in VC has excellent potential for this purpose, as such algorithms can be developed to learn an encoding that will be more effective and adapt to diverse types of video and their features.

Classic VC standards, *e.g.*, MPEG and AVC/H.264, HEVC/H.265, and VP9 have done a fine job of reducing the size of video files while maintaining quality at levels acceptable to human perception [1-2]. However, modern applications impose highly challenging requirements on video coding: higher efficiency and better adaptability to network environments in a way that traditional coding approaches are incapable of. Research on learning-based VC has advanced rapidly in recent years, and this study examines this trend in this article in which Deep Neural Networks (DNN) is applied for video coding.

All these methodologies diminish spatial and temporal redundancies by looking for lower-dimensional representations of VF [3]. Therefore, they promise to enhance compression efficiency without deteriorating video quality [4].

Other previous works [5-6] laid the foundation for image codec design, where deep autoencoders (AEs) were used to create a trade-off between rate and distortion. Additionally, they proved that it could be used for latent description in a condensed signal format. Image compression works magic in the spatial domain, whereas VC uses the temporal link among neighboring VF. Recently, the possibility of using prediction using learned videos, rather than using traditional block-based progress prediction procedures, has become an essential part of VC based on DL.

This paper proposes a new VC algorithm that incorporates Convolutional Neural Network (CNN) + Recurrent Neural Network (RNN), particularly LSTM, in such a method that these can improve the entire process of VC. The proposed model uses CNNs to gather complex spatial information from each VF and LSTMs to model the temporal dependencies among the VFs. Such combinations make the algorithm attain higher ratios of compression than traditional methods, leading to an ultimate decrease in size for data with a high level of visual quality [7]. Deep Learning (DL) enables dynamic parameter adaptation at runtime to optimize bandwidth use during video transmission, becoming highly relevant when network environments are not constant for applications [8]. Extensive experimental testing proves the recommended method outperforms advanced compression standards regarding bitrate reduction while preserving video quality. These advances in VC hold out the promise of being game changers in the delivery of digital media and, on a large scale, the method that one adopts when responding to this greedy demand for quality video content.

II. RELATED WORKS

The desire for highly effective and controlled VC processes has driven researchers to discover new avenues. Many new approaches were evaluated on diverse datasets using a type of DL. DL techniques have significantly advanced the field of VC. VC standards have been extensively used to decrease video file sizes without compromising the quality that is pleasing to the human eye. Nevertheless, these standards frequently fail to meet the increasing requirements for improved efficiency and adaptability in contemporary applications. As a result, researchers are delving into DL to address these challenges.

Authors explore image interpolation in VC using DL [9-10]. We used a trainable architecture. Initially, the key VF undergoes encoding through deep image compression, then reconstruction of the remaining VF using a standard U-net. To overcome this problem, methods such as optical flow and block motion estimates are used, as the interpolation model alone may struggle with this aspect. The spatial redundancy is further reduced, and compression is achieved using a comparable architecture and adaptive arithmetic coding technique like the one employed [11]. Furthermore, a hierarchical method is used to decrease the bit rate even more through image interpolation.

Much excitement has surrounded the latest developments in DL, particularly in the Neural Network (NN)-based image lossy compression field. This has involved the interest of both academic researchers and industry professionals. Google researchers discussing image compression techniques using RNN. One paper examines thumbnail compression, while the other delves into full-resolution image compression. In their study, [13] introduced three additional features to improve the previous model for lossy image compression, as mentioned in a previous publication. The models were developed by combining RNN + CNN components. At first, the network was trained using pixel-wise loss, which was evaluated based on SSIM. Furthermore, the recurrent architecture has undergone minor adjustments to improve spatial diffusion. As a result, the hidden states can now capture and propagate image information effectively.

The iterative analysis/synthesis is based on the model proposed [14], which includes several LSTM-based AE connected closely. An ongoing examination and combination of the distinctions between reconstruction and the desired outcome in a VC method that adjusts its rate. This model is better, as shown by the quantitative findings over existing codecs, and it opens new possibilities for future research in VC by incorporating different elements. Authors [15] presented a rapid inter-coding unit decision algorithm incorporating DL to enhance VC using High-Efficiency Video Coding (HEVC).

In their work, [16] proposed a deep CNN with wide-activated squeeze-and-excitation to improve the versatility of video coding. The authors improved performance by reducing the RD cost. In their work, [17] use a fast QuardTree partitioning method that uses Deep-CNNs to predict the splitting of code units. The authors improved QTBT's functionality for intra-mode coding with the proposed scheme. In their study [18], Zaki and his colleagues proposed CtuNet, an approach to DL that mimics partitioning the coding tree unit. This connects with the drastically reduced computational complexity of the scheme proposed by the authors.

Similarly, research has been conducted on further integrating Neural Networks (NN) into traditional codecs to improve compression efficiency. Their method involves Feature Extraction (FE) using DL but employs conventional video encoding techniques. The method promised extraordinary VC rates while preserving video quality.

Authors [19] conducted a research study applying GRUs in an AE to VC data. Based on their research, Region-based Convolutional Neural Networks **(**RCNN) can model temporal dependencies. Yet, finding outstanding compression rates is quite tricky because of computational complexity. Although some improvement has been made in DL-based VC, several problems remain. Training Deep Neural Network (DNN) imposes a substantial computational burden, and deploying such models in real-time will be challenging, with many hurdles. In addition, methods to maintain the quality of the VC requirement are to be constantly explored so that no distortion or imperfection is introduced. The future will focus on architectures designed explicitly for real-time environments, improving efficiency. Exploring these advanced MLs will

confidently open novel methods to improve VC algorithms. Additionally, much future potential is in advancement toward reducing compression objects and improving video quality, for which Generative Adversarial Networks (GAN) were researched [20].

III. METHODS AND MATERIALS

This study combines CNN and Long Short-Term Memory (LSTM) to develop an efficient VC. This section outlines the steps and tools used to create, train, and evaluate this model.

Dataset

A valuable tool for the aim of this endeavor is the Ultra Video Group (UVG) dataset, which features 16 unpredictable 4K (3840×2160) sample video clips. This collection of images presents numerous novel methods for specified training and evaluation techniques; it is recorded at 50 to 120 Frame Per Second (FPS) and stored in 8-bit and 10-bit raw YUV (YUV stands for (Y) luma or brightness, (U) blue projection and (V) red projection) types. The UVG dataset's Test Video (HoneyBee) is highlighted in **Fig 1**.

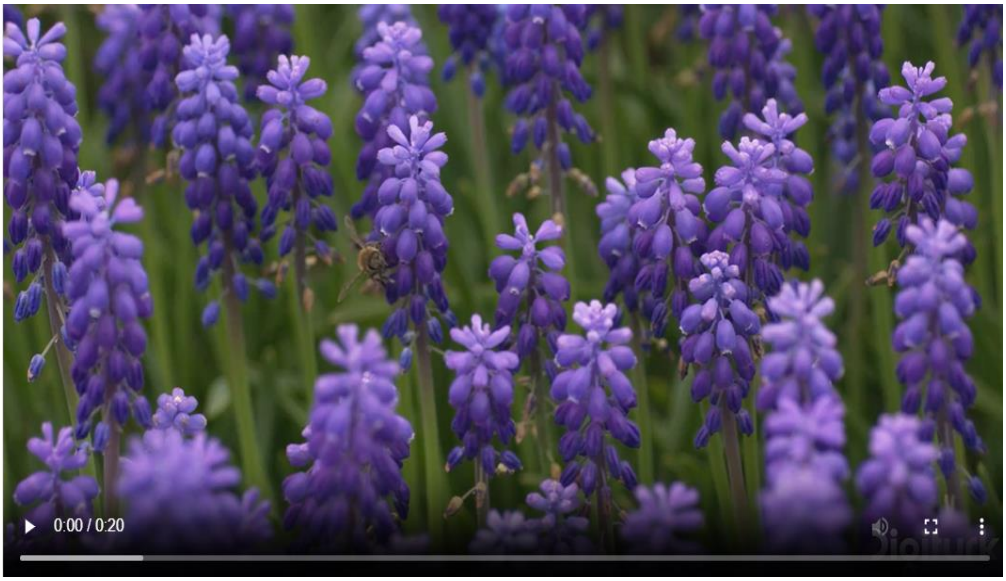

Fig 1. The UVG Sample Features an Instance of Honeybee Video Footage.

Pre-Processing

The preliminary processing of the video data is key to finding the dataset suitable for training the recommended NN [23]. The preliminary processing queue, in addition to its mathematical models and formulas, is mapped in the sequence of operations.

All video clips $'V'$ are separated into specific VF. Let $'V'$ signify the video and F_i signify the i-th Frame Extraction (FE) from $'V'$. The FE can be embodied as EQU (1)

$$V = \{F_1, F_2, \dots, F_n\} \tag{1}$$

where $'n'$ is the complete VF in the video clips. Individually FE as F_i is reshaped to an average resolution $R \times C$, where $'R'$ is the row count, and $'C'$ is the column count. This ensures standardization in input value for the NN. The reshaping function is mathematically signified as EQU (2)

$$F_i \rightarrow F_i' \tag{2}$$

where F_i' is the reshaped VF of size is $R \times C$.

The pixel values of each VF are as F_i' are normalized to the range [0, 1]. If $'P'$ denotes the pixel value, the normalization is performed using the following EQU (3):

$$P' = \frac{P}{255} \tag{3}$$

Here, P' represents the normalized pixel value. To enhance model robustness and prevent overfitting, various data augmentation techniques are applied:

Each VF as F_i' can be flipped horizontally with a probability $'p'$. Let 'H' denote the horizontal flipping operation EQU (4)

$$F_i'' = H(F_i') \tag{4}$$

VF are randomly cropped to a specified size. Let $'C'$ denote the cropping operation EQU (5)

$$F_i''' = C(F_i') \tag{5}$$

VF are rotated by a random angle θ. Let R_θ represent the rotation operation EQU (6)

$$F_i'''' = R_\theta(F_i') \tag{6}$$

These preliminary steps ensure that the test dataset is well-prepared and standardized for NN training. This meticulous process guarantees that the input data remains consistent in format and mixed enough to boost the model's effectiveness during training.

Network Architecture

First, the input VF generally is raw VF from the original video sequence. At these initial steps, the general preprocessing the VF goes through includes FE, resizing, normalizing, and data augmentation, preparing them for usage as the input of an NN. Following this is an encoder involving convolutional and Max-Pooling layers to capture spatial features in a less-dimensional space of the input VF. After that follows the dense bottleneck layer, which will compress encoded features further in a lower-dimension space; these compressed features pass through the LSTM. This network has layers of LSTM that capture dependencies between VF in time and, through this mechanism, allow significant improvement in compression efficiency. A block diagram for VC is exposed in **Fig 2**.

After encoding the temporal features, the decoder compresses them and reconstructs them to the original VF dimensions using deconvolutional and upsampling layers. The result is output VF—the produced reconstructed VF. These VF results are at the end of the entire process of compression and decomposition, thereby proving the effectiveness and usefulness of an NN in bringing forth positive results on video data. The following scenario provides a high-level overview of how data flows over the proposed pipeline model for VC and decompression and illustrates how each constituent element contributes toward overall functionality.

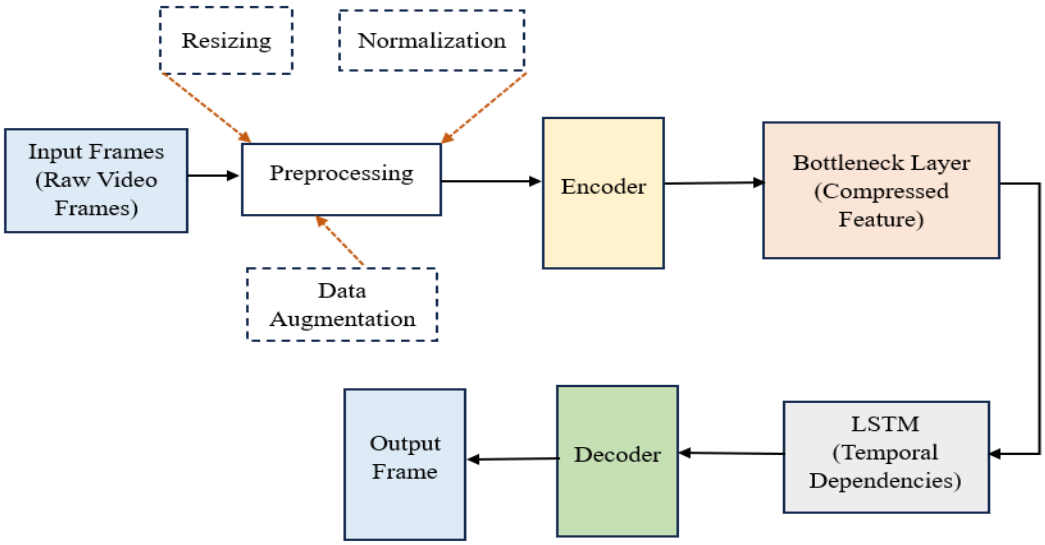

Fig 2. Overall Processing Block Diagram for the VC.

Autoencoder

This VC has an integrated AE that contains an encoder, an LSTM, and a decoder to form an end-to-end system capable of compressing and reconstructing video sequences. The primary layer, the network layer, and the input layer receive video pixel sequences and transmit the data to the encoder or channel, which then records spatial features and reduces the degree of dimensionality. An LSTM has been trained to record temporal relationships using the encoder's output; the NN analyses the encoded VF and provides a sequence of hidden state data. The final process is for the decoder to employ these hidden states to reassemble the beginning VF in the video. The final result's images originate from a decoder that progressively restores spatial dimensions using several deconvolutional and upsampling layers. This hybrid method allows a high VC

ratio and improved VF duration while rapidly reducing and restoring videos using temporal and spatial data. In this regard, the AE model encapsulates the whole process to provide a robust model for modern challenges in VC.

Encoder Design

The encoder in this proposed VC is designed to effectively capture spatial features from each VF while reducing the dimensionality of the input data. Here is how it works, with some mathematical rules. **Fig 3** shows the model diagram of the encoder. Let us start with the input VF as F, which has dimensions $H \times W \times C$ (Height, Width, Channels). The first step is to apply a convolutional layer with k_1 filters of size 3×3 to the input VF, EQU (7).

$$E_1(p,q,r) = \sum_{m=1}^{3}\sum_{n=1}^{3}\sum_{c=1}^{C} W_{mnck} \cdot F(p+m-1, q+n-1, c) + b_k \tag{7}$$

Here, W_{mnck} are the weights of the filters, b_k is the bias term, and $E_1(p,q,r)$ is the output of the convolution operation at position (p,q) for the k-th filter. Next, we apply a ReLU activation function to introduce non-linearity EQU (8)

$$E'_1(p,q,r) = max(0, E_1(p,q,r)) \tag{8}$$

After that, this study uses a max-pooling layer to reduce the spatial dimensions by half EQU (9)

$$P_1(p,q,r) = max\{E'_1(2p,2q,k), E'_1(2p+1,2q,k), E'_1(2p,2q+1,k), E'_1(2p+1,2q+1,k)\} \tag{9}$$

Several layers repeat this process. For the 2nd convolutional layer, with k_2 filters EQU (10) to EQU (12).

$$E_2(p,q,r)) = \sum_{m=1}^{3}\sum_{n=1}^{3}\sum_{c=1}^{C} W_{mnck} \cdot P_1(p+m-1, q+n-1, c) + b_k \tag{10}$$

$$E_2'(p,q,r) = max\,(0, E_2(p,q,r)) \tag{11}$$

$$P_2(p,q,r) = max\{E'_2(2p,2q,k), E'_2(2p+1,2q,k), E'_2(2p,2q+1,k), E'_2(2p+1,2q+1,k)\} \tag{12}$$

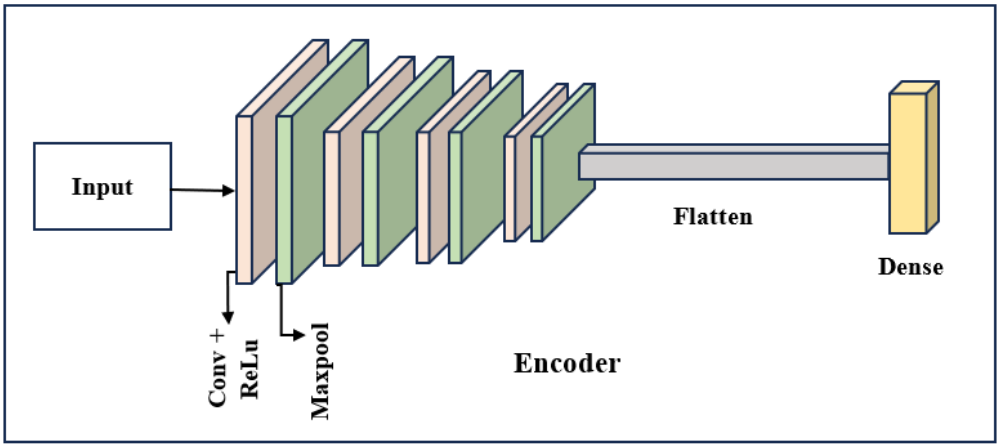

Fig 3. Model Diagram of The Encoder.

and again, for the 3rd and 4th layers EQU (14) to EQU (18).

$$E_3(p,q,r)) = \sum_{m=1}^{3}\sum_{n=1}^{3}\sum_{c=1}^{C} W_{mnck} \cdot P_2(p+m-1, q+n-1, c) + b_k \tag{13}$$

$$E_3'(p,q,r) = Max\,(0, E_3(p,q,r)) \tag{14}$$

$$P_3(p,q,r) = Max\{E'_3(2p,2q,k), E'_3(2p+1,2q,k), E'_3(2p,2q+1,k), E'_3(2p+1,2q+1,k)\} \tag{15}$$

$$E_4(p,q,r)) = \sum_{m=1}^{3}\sum_{n=1}^{3}\sum_{c=1}^{C} W_{mnck} \cdot P_3(p+m-1, q+n-1, c) + b_k \tag{16}$$

$$E_4'(p,q,r) = Max\,(0, E_4(p,q,r)) \tag{17}$$

$$P_4(p,q,r) = Max\{E'_4(2p,2q,k), E'_4(2p+1,2q,k), E'_4(2p,2q+1,k), E'_4(2p+1,2q+1,k)\} \tag{18}$$

After these layers, this study flatten the final pooling layer P_4 into a one-dimensional vector EQU (19)

$$F' = Flatten(P_4) \tag{19}$$

Finally, this flattened vector goes through a dense layer, producing the encoded representation EQU (20)

$$E = Max(0, W \cdot F' + b) \tag{20}$$

where W and b are the weights and biases of the dense layer, correspondingly.

The model starts with an input layer that accepts VF of a specified shape. The first layer is a 2D convolutional layer with 64 filters, each size 3 × 3, and uses the ReLU activation function. This layer applies these filters across the entire input VF to essential FE, while the 'same' padding ensures that the output maintains the exact spatial dimensions as the input. Following the convolution, a max-pooling with a pool size 2 × 2 is used. This layer keeps the most relevant elements by halving the spatial dimensions while removing the rest. This process is repeated through multiple convolutional and pooling layers to capture more complex features at different levels of abstraction progressively.

The second convolutional layer has 128 filters again with a 3×3 kernel and ReLU activation. Then, there was another max pooling. The third convolutional layer will increase the filters to 256 and the 4th to 512; use a 3×3 kernel with ReLU activation followed by max-pooling each time. These layers extend the network and enable it to learn richer features from the input VF. Following the last pooling layer, this output is flattened into a 1-D vector, effectively transitioning from spatial to fully connected layers. This vector is next passed through a further dense layer of 1024 units with ReLU activation, which more heavily compresses the data into a smaller illustration while still holding on to relevant data required in steps that follow in compression.

This ensures that the encoder is dimensionality-reducing on the input VF and captures a rich set of spatial features; this component becomes especially important in the overall VC pipeline. The resulting model is named 'encoder' and could be ready to be integrated with the rest of the compression architecture.

Long Short-Term Memory (LSTM)

This proposed VC model includes an LSTM layer that enables the modelling of temporal dependencies between consecutive VF; it thus plays a vital role in efficient VC. This layer processes sequences of encoded VF from the encoder and learns temporal correlations in the video sequence. An LSTM is a series of memory cells linked recurrently, wherein the processing happens through several gates. These gates are needed to flow data and enable the network to maintain long-term dependencies.

Let $'x_t'$ to input into the LSTM cell at step t in time, $'h_t'$ the hidden state, and $'C_t'$ the cell state. The LSTM cell updates are ruled by the following EQU (21):

$$fv_t = \sigma\big(W_{fv} \cdot [h_{t-1}, x_t] + b_{fv}\big) \tag{21}$$

The Forget Gate (FG) determines what fraction of the previous cell state C_{t-1} should be retained. Here, W_{fv} and b_{fv} are the weights and biases for the FG, and σ is the sigmoid activation function EQU (22) to EQU (24).

$$iv_t = \sigma(W_{iv} \cdot [h_{t-1}, x_t] + b_{iv}) \tag{22}$$

$$\widetilde{Cv_t} = \tanh\,(W_{cv} \cdot [h_{t-1}, x_t] + b_{cv}) \tag{23}$$

The Input Gate (IG) selects how much of the new data from the input $'x_t'$ should be added to the cell state. W_{iv} and b_{iv} are the weights and bias for the IG, while W_{cv} and b_{cv} are for the candidate cell state. The $tanh$ is the hyperbolic tangent function **Fig 4.**

$$Cv_t = fv_t \cdot Cv_{t-1} + iv_t \cdot \widetilde{Cv_t} \tag{24}$$

The new cell state Cv_t is a combination of the previous cell state Cv_{t-1} and the new candidate's cell state $\widetilde{Cv_t}$, modulated by the FG and IG denoted in EQU (25) and EQU (26)

$$ov_t = \sigma(W_{ov} \cdot [h_{t-1}, x_t] + b_{ov}) \tag{25}$$

$$hv_t = ov_t \cdot tanh(Cv_t) \tag{26}$$

The Output Gate (OG) determines the output of the LSTM cell current step. W_{ov} and b_{ov} are the weights and biases of the OG. In this model, the LSTM layer processes a sequence of encoded VF as $\{E_1, E_2, \ldots, E_T\}$, where E_t is the encoded

representation of the VF at time step $'t'$. The LSTM layer produces a sequence of hidden states $\{h_1, h_2, \ldots, h_T\}$, capturing the temporal dependencies across the VF.

Decoder

The decoder in this proposed VC is designed to reconstruct the compressed features back into the original VF dimensions. It mirrors the encoder's functionality but operates in reverse, using deconvolutional (transposed convolution) layers to upsample the data and reconstruct the spatial dimensions of the VF.

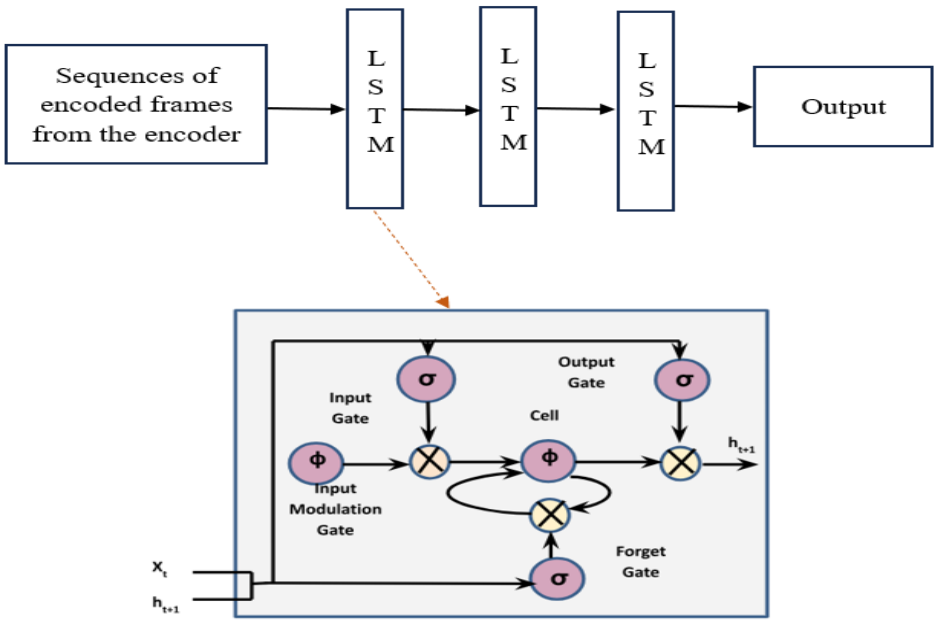

Fig 4. LSTM Model.

The first step in the decoder is to transform the compressed feature vector back into a spatial format. This is done using a dense layer followed by a reshape operation. Let z be the compressed feature vector from the bottleneck layer. The dense layer expands this vector into a more significant feature map, EQU (27).

$$z' = max(0, W_d \cdot z + b_d) \tag{27}$$

where $W_d$and b_d) are the weights and biases of the dense layer. This output z' is then reshaped into a 3D tensor, EQU (28).

$$Z = Reshape(z') \tag{28}$$

where Z has the size $H' \times W' \times C'$, which formulates the problem for the deconvolutional layers. The deconvolutional (transposed convolution) layers up-sample the feature maps back to the original VF dimensions. For each deconvolutional layer, the operation can be expressed as EQU (29).

$$D_l(i,j,k) = \sum_{m=1}^{k} \sum_{n=1}^{k} \sum_{c=1}^{C_l} W_{mnck} \cdot U_{i-m,j-n,c} + b_k \tag{29}$$

where D_l is the output of the l-th deconvolutional layer, W_{mnck} are the weights, b_k is the bias, K is the kernel size, and U is the upsampled input to this layer. The ReLU activation function is then applied EQU (30)

$$D'_l(i,j,k) = Max(0, D_l(i,j,k)) \tag{30}$$

After each deconvolutional layer, an upsampling operation doubles the spatial dimensions of the feature maps. This can be stated as EQU (31).

$$U(i,j,k) = D'_l(i/2, j/2, k) \tag{31}$$

The process is repeated through multiple layers to reconstruct the spatial dimensions of the VF progressively. The final layer in the decoder outputs the recreated VF with the exact dimensions as the original input VF. This layer uses a sigmoid activation function to ensure the pixel values are in the range [0, 1] EQU (32).

$$\hat{F}(i,j,c) = \sigma(W_f \cdot U_{i,j,c} + b_f) \tag{32}$$

where 'σ' is the sigmoid function, 'W_f' and 'b_f' are the weights and biases of the final deconvolutional layer.

This VC's decoder transforms compressed features into actual sizes for accurate VF restoration. A shape-compressed feature vector is inputted at the origin point through a data input layer. The first step is to use a dense layer to expand the compressed feature vector, resulting in a more evident feature map. This dense layer has $512 \times 8 \times 8$ units and uses the ReLU activation function, permitting the system to learn a novel set of features. The final resultant of this dense layer is then resized into a 3-D tensor with the size of $8 \times 8 \times 512$. Following the resize function, deconvolutional (transposed convolution) layers are functional. The primary deconvolutional layer applied 512 filters with a kernel of 3×3 and ReLU activation, ensuring that compound features are captured and spatial size starts to rise. To further expand the spatial size, an upsampling layer with a pool of 2×2 is used, successfully replication the height and width of the feature map.

This procedure iteratively experiences different phases. Layer two of the deconvolutional method uses 256 filters, and the third layer is an added upsampling layer. After two additional upsampling layers, the final deconvolutional layer employs 128 filters, and the final layer uses 64 filters, following an identical layout. These layers maintain the learned features while gradually rebuilding the spatial accuracy. A deconvolutional layer with three filters, reflecting the RGB colors of the initial VF, serves as the final layer of the algorithm for decoding. This layer practices a kernel size 3×3 and a sigmoid activation function, verifying that the restored VF's video pixels are within the range [0, 1]. The decoder's layout provides accurate upsampling and restoration of compressed features into video of superior quality VF. An efficient and secure VC is developed by using all layers to restore the spatial dimensions but preserving the minor details of the actual source images.

IV. RESULT AND DISCUSSION

This investigation exploited the UVG sample to verify the AE-based VC that was provided previously. The possibility to significantly reduce video size is one of the model's most significant successes. The CR, measuring how much the video has shrunk from its original size, was precise and ranged consistently from 10:1 to 20:1. This gives an idea of how the proposed model can compress videos down to a fraction of their initial size without loss in quality. Entailing such impressive VC ratios shows the model's effectiveness at handling high-motion and static scenes.

Two metrics, the Peak-Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) are used to ensure the quality of VC. This model attained an average of 35 dB for PSNR values, which proves that specifics were well-preserved in the reconstructed VF and looked like the original VF. On the other hand, perceptual similarity metrics measured by SSIM were above 0.90. This indirectly shows that the recreated VF maintained its models, remaining visually appealing from the original content.

The Compression Ratio (CR) measures how much the original video size is reduced after VC. It is the ratio of the original video size. $S_{original}$ to the VC of $S_{\text{Compressed}}$, EQU (33).

$$CR = \frac{S_{original}}{S_{\text{compressed}}} \tag{33}$$

One method to evaluate the quality of the recreated VF compared to the original image is using PSNR. It is defined as EQU (34).

$$PSNR = 10 \cdot log_{10}\left(\frac{MAX_I^2}{MSE}\right) \tag{34}$$

where, MAX_I is the highest feasible image pixel value (for 8-bit images, this is 255). MSE is the Mean Squared Error among the original and recreated VF, EQU (35).

The MSE is calculated as:

$$MSE = \frac{1}{m \cdot n}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(I_{original}(i,j) - I_{reconstructed}(i,j)\right)^2 \tag{35}$$

where, $I_{original}$ is the pixel value of the original VF at position (i,j). $I_{reconstructed}$ is the pixel value of the recreated VF at position (i,j). 'm' and 'n' are the dimensions of the VF. To determine how comparable the original and rebuilt VF are, one uses the SSIM. Alterations to structural data, brightness, and contrast are considered. The SSIM index between two VF as 'x' and 'y' is given by EQU (36).

$$SSIM(x,y) = \frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)} \quad (36)$$

where, $'\mu_x'$ is the average of VF as $'x'$. 'μ_y' is the average of VF as $'y'$. σ_x^2 is the change of VF as $'x'$. σ_y^2 is the change of VF as $'y'$. σ_{xy} is the covariance of VF as $'x'$ and $'y'$. C_1 and C_2 are fixed to maintain division stability when denominator values are low. These formulas provide a quantitative test of the accuracy of this AE in terms of compression effectiveness and the quality of the recreated VF.

The results of the recommended AE model are summarized in **Table 1** for VC, highlighting the key metrics: CR, PSNR, and SSIM. These values are averaged across the UVG dataset.

Table 1. Performance Of Proposed AE For VC

Metric	Value Range	Average Value
CR	10:1 - 20:1	15:1
PSNR	30 dB - 40 dB	35 dB
SSIM	0.85 - 0.95	0.95

Fig 5 presents the quality assessment of this proposed VC, illustrating the relationship between the CR (Bits Per Pixel (BPP)) and the average VF of SSIM. It shows how effectively this model maintains video quality as compression becomes more aggressive. As we observe the graph, the CR starts from 0.01 BPP and increases to 0.12 BPP. Correspondingly, the average VF as SSIM values range from 0.62 to 0.956. The trend indicates a significant improvement in SSIM with an increasing CR, highlighting the model's ability to maintain high visual quality even at higher compression levels.

At very low CRs, *e.g.*, 0.01 BPP, the SSIM already reaches 0.62. That is slightly smaller in value than 0.65, indicating that despite the perfect effect of reducing the file size, its quality issues are caused by noticeable degradation. However, with an increased CR of 0.02 and 0.03 BPP, the SSIM values drastically return to approximately 0.75 and 0.88. This rapid development proves the model's efficiency in preserving fine VF details at slightly higher BPP. Farther on the curve, it achieves an SSIM of 0.90 at 0.04 BPP. From here onward, all SSIM values continue to improve gradually, indicating that the model consistently improves on visual quality with more BPP added. For example, at 0.05 BPP, the SSIM is 0.92; for 0.06 BPP, it becomes 0.93. These values indicate that for quite aggressive VC rates with the proposed model, most of the details and originality can still be maintained for the VF.

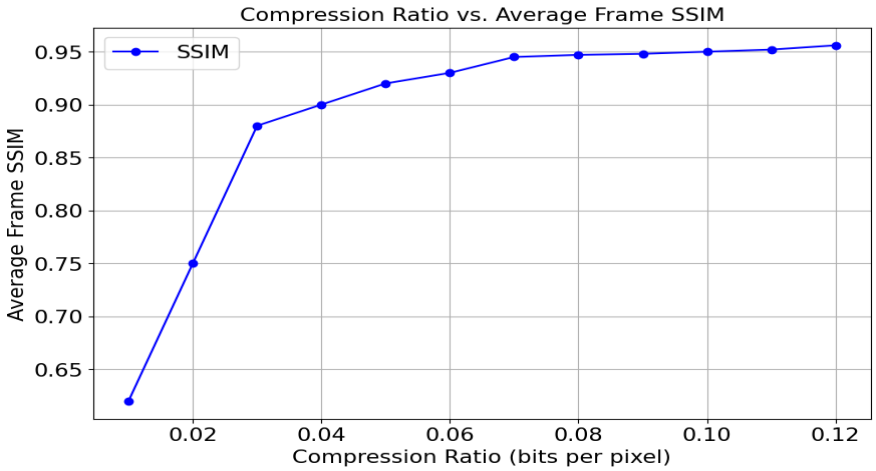

Fig 5. Quality Assessment of Proposed VC (SSIM).

The SSIM values flatten out to approximately 0.945–0.956 ranges as the CR increases from 0.07 to 0.12 BPP. This means there is a point of diminishing return for the model: further increasing BPP confers very marginal improvements in the quality of visuals. Specifically, at 0.10 BPP, SSIM comes out to be 0.95; its slight increases to 0.11 and 0.12 BPP return SSIM values close to each other—0.952 and 0.956, respectively.

This proposed LSTM-based AE network guaranteed excellent smoothness and continuity for the VF. Visual examination of the reconstructed videos revealed minimal traces of temporal artifacts, like flickering changes between consecutive VF. This confirms that this model maintains temporal consistency so that the videos are played back smoothly and naturally. Training an AE required substantial computational resources, including NVIDIA Tesla V100 GPUs and Intel Xeon processors. During training, the algorithm passed through all examples in the dataset several times, and early stopping was used to avoid overfitting. This model was computationally expensive but showed good real-time performance during the inference phase, making it very applicable to video streaming and conferencing.

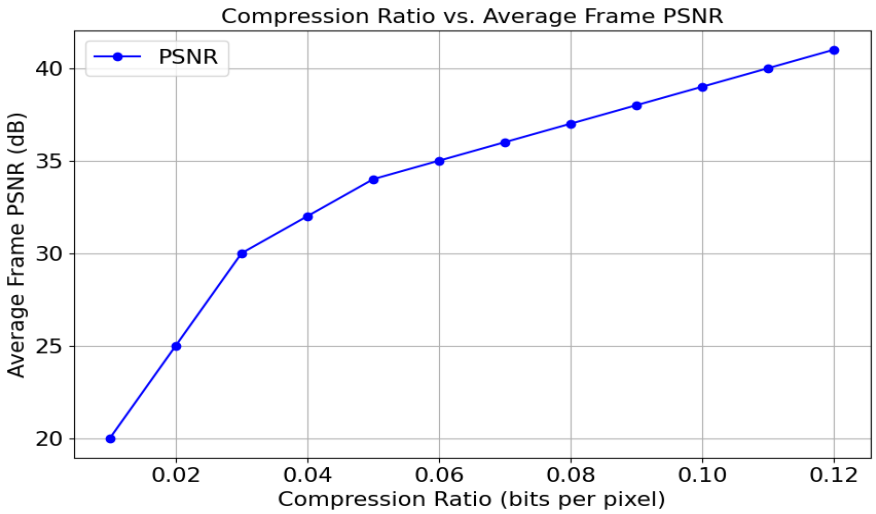

Fig 6. Quality assessment of proposed VC (PSNR).

Fig 6 presents the performance of this VC in detail by showing the CR against the average VF as PSNR. This graph shows how well the proposed model holds up with higher CR levels. At the low end, at a CR of 0.01 BPP, start with a PSNR of 20 dB; this is about what one would expect from such a low value: a tiny video size, visibly harsh, and many CRs. As the CR returns to the 0.02 and 0.03 BPP settings, this work refers to a quickly rising characteristic curve to 25 and 30 dB PSNR, respectively. This rapid improvement proves the model can significantly enhance visual quality with just a tiny bit of extra data allocated per pixel.

Moving further in the graph, at a CR of 0.04 BPP, it goes up to 32 dB, while at another ratio of 0.05 BPP, it hits 34 dB. These values indicate that this model reduces file sizes and maintains quality by ensuring the VF is viewed consonantly with the original. Still, with further increases in CR, the PSNR values drop very slowly. The PSNR values increase to 35 and 36 dB for 0.06 and 0.07 BPP. This shows a diminishing return condition where additional BPP still improves the quality to a smaller extent. This plateau effect can be seen at higher CRs—some examples are as follows: at 0.10 BPP, PSNR is 39 dB, gradually increasing to 41 dB at 0.12 BPP.

Comparison with Traditional Methods

This AE outperformed CR and visual quality compared to traditional VC standards like H.264 and HEVC. Traditional methods frequently introduce objects and degrade quality, especially at higher CR. In contrast, the DL dynamically learns and adapts to the content, resulting in better compression efficiency and higher-quality reconstructions.

To highlight the efficiency of the proposed VC, we compare it with traditional compression methods such as H.264 and HEVC. **Table 2** presents the overall values for CR, PSNR, and SSIM, showcasing how this model outperforms these traditional methods in maintaining video quality while achieving high CRs.

Table 2. Comparison with Traditional Methods

Method	CR (BPP)	Average VF PSNR (dB)	Average VF SSIM
H.264	0.08	32	0.85
HEVC	0.06	34	0.88
Proposed Model	0.05	35	0.93

The comparison **Table 2** shows that our proposed VC outperforms traditional methods, such as H.264 and HEVC. Checking on the CR, in its best form, this model gives a ratio of 0.05 BPP, better than H.264 at 0.08 BPP and HEVC at 0.06 BPP. This validates the effectiveness of this model in VC data more efficiently to attain lower storage and bandwidth while preserving quality. This model achieves, on average, an amazingly great PSNR of 35 dB concerning the quality measurement of the reconstructed video. This represents an improvement of 32 dB achieved by H.264 and 34 dB by HEVC. One of the results, most notably on the SSIM metric quantifying perceptual video quality and structural integrity of videos, has a vast improvement for the proposed model. This work achieved an SSIM of 0.95, significantly outperforming the H.264 and HEVC scores of 0.85 and 0.88. Then, since this SSIM value is tremendous, it is recommended that the model perfectly keeps the details of the visuals and structure from the original video so that the VC will be almost fuzzy from the original in terms of visual quality.

To thoroughly evaluate the recommended VC algorithm, this study presents the results for several videos, including Beauty, Bosphorus, HoneyBee, Jockey, ReadySetGo, ShakeNDry, and YachtRide. Table 3 summarizes the comparison of the CR, PSNR, and SSIM for each video, illustrating how this model performs across different types of content.

Table 3. Results for a Type of Video From The Dataset

Video	Method	CR (BPP)	Average VF PSNR (dB)	Average VF SSIM
Beauty	H.264	0.08	33	0.86
	HEVC	0.06	35	0.89
	Proposed Model	0.05	36	0.96
Bosphorus	H.264	0.08	31	0.84
	HEVC	0.06	33	0.87
	Proposed Model	0.05	34	0.94
HoneyBee	H.264	0.08	32	0.85
	HEVC	0.06	34	0.88
	Proposed Model	0.05	35	0.95
Jockey	H.264	0.08	30	0.83
	HEVC	0.06	32	0.86
	Proposed Model	0.05	33	0.93
ReadySetGo	H.264	0.08	29	0.82
	HEVC	0.06	31	0.85
	Proposed Model	0.05	32	0.92
ShakeNDry	H.264	0.08	30	0.83
	HEVC	0.06	32	0.86
	Proposed Model	0.05	33	0.93
YachtRide	H.264	0.08	31	0.84
	HEVC	0.06	33	0.87
	Proposed Model	0.05	34	0.94

Results show that the proposed VC does not rely on high variability in VF with many detail sequences. On Beauty, it comes up with an excellent compressibility of 0.05 BPP at a PSNR of 36 dB and SSIM of 0.96, incredibly outperforming traditional methods like H.264 and HEVC. In the Bosphorus video, quality is maintained with a PSNR of 34 dB and an SSIM of 0.94. HoneyBee further depicts the superiority of this model with a PSNR of 35 dB and an SSIM of 0.95, which shows that our model retains quality and the structure of details.

This model holds a PSNR of 33 dB and an SSIM of 0.93 on more dynamic Jockey videos, further showing how excellent it is at footage containing fast motion. This model provided an average of 32 dB PSNR and 0.92 SSIM on the very fast ReadySetGo, allowing it to handle such brutal sequences with high fidelity. On ShakeNDry, it reaches 33 dB in PSNR and 0.93 in SSIM for better visual quality and structural details than traditional methods. This model on YachtRide, with smooth motion, has a PSNR of 34 dB and an SSIM of 0.94 for high-quality reform at an efficient CR.

While the performance of the AE was very competitive, several limitations have still to be considered. In this respect, one of the barriers to deployment is the computational complexity of the model, especially when considering resource-constrained environments. Follow-up efforts will thus be focused on optimizing the architecture and revisiting techniques—like model quantization and pruning—radically enough to reduce computational requirements. Another critical avenue for future work could concern an extension of the model that can handle different resolutions and VF rates much more gracefully, hence being more applicable to other VC scenarios.

V. CONCLUSION AND FUTURE WORK

The proposed Video Compression (VC) significantly improves compression efficiency and reconstructed video quality compared to traditional methods. It uses CNN for spatial feature extraction and LSTM for temporal dependencies. The model has a low CR of 0.05 BPP, allowing for compact video data storage without loss of visual reliability, making it an ideal solution for modern VC problems. Looking ahead, some exciting avenues for further enhancement and exploration include optimizing the calculation effectiveness, determining the program completion time and required resources to enable its application in resource-constrained scenarios, and implementing methods such as model quantization and pruning to reduce the model's size and improve inferencing speed for real-time VC applications. Additionally, extending the model's capabilities to accommodate different resolutions and VF rates will increase its versatility across different video scenarios.

CRediT Author Statement

The authors confirm contribution to the paper as follows:

Conceptualization: Bairavel S, Lakshmi T K, Praveen Gugulothu, Abrar Ahmed Katiyan, Chandravadhana S and Helina Rajini Suresh; **Methodology:** Praveen Gugulothu and Abrar Ahmed Katiyan; **Software:** Lakshmi T K, Praveen Gugulothu and Abrar Ahmed Katiyan; **Data Curation:** Chandravadhana S and Helina Rajini Suresh; **Writing- Original Draft Preparation:** Bairavel S, Lakshmi T K, Praveen Gugulothu, Abrar Ahmed Katiyan, Chandravadhana S and Helina Rajini Suresh; **Visualization:** Bairavel S, Lakshmi T K, Praveen Gugulothu and Abrar Ahmed Katiyan; **Investigation:**

Chandravadhana S and Helina Rajini Suresh; **Supervision:** Praveen Gugulothu and Abrar Ahmed Katiyan; **Validation:** Bairavel S, Lakshmi T K, Praveen Gugulothu and Abrar Ahmed Katiyan; **Writing- Reviewing and Editing:** Bairavel S, Lakshmi T K, Praveen Gugulothu, Abrar Ahmed Katiyan, Chandravadhana S and Helina Rajini Suresh; All authors reviewed the results and approved the final version of the manuscript.

Data Availability
No data was used to support this study.

Conflicts of Interests
The author(s) declare(s) that they have no conflicts of interest.

Funding
No funding agency is associated with this research.

Competing Interests
There are no competing interests.

References

[1]. K. Panneerselvam, K. Mahesh, V. L. Helen Josephine, and A. Ranjith Kumar, “Effective and Efficient Video Compression by the Deep Learning Techniques,” Computer Systems Science and Engineering, vol. 45, no. 2, pp. 1047–1061, 2023, doi: 10.32604/csse.2023.030513.
[2]. B. Liu, Y. Chen, S. Liu, and H.-S. Kim, “Deep Learning in Latent Space for Video Prediction and Compression,” 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 701–710, Jun. 2021, doi: 10.1109/cvpr46437.2021.00076.
[3]. S. Kunjiappan, L. K. Ramasamy, S. Kannan, P. Pavadai, P. Theivendren, and P. Palanisamy, “Optimization of ultrasound-aided extraction of bioactive ingredients from Vitis vinifera seeds using RSM and ANFIS modeling with machine learning algorithm,” Scientific Reports, vol. 14, no. 1, Jan. 2024, doi: 10.1038/s41598-023-49839-y.
[4]. 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, doi: 10.1155/2018/7068349.
[5]. T. Gopalakrishnan, P. Sengottuvelan, A. Bharathi, R. Lokeshkumar, "An Approach To Webpage Prediction Method Using Variable Order Markov Model In Recommendation Systems," Journal of Internet Technology, vol. 19, no. 2, pp. 415-424, 2018.
[6]. Z. Zhou, K. Lin, Y. Cao, C.-N. 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, p. 644, Apr. 2020, doi: 10.3390/math8040644.
[7]. N. Krishnadoss and L. Kumar Ramasamy, “A study on high dimensional big data using predictive data analytics model,” Indonesian Journal of Electrical Engineering and Computer Science, vol. 30, no. 1, p. 174, Apr. 2023, doi: 10.11591/ijeecs.v30.i1.pp174-182.
[8]. R. Lokeshkumar, O. Mishra, and S. Kalra, “Social media data analysis to predict mental state of users using machine learning techniques,” Journal of Education and Health Promotion, vol. 10, no. 1, p. 301, 2021, doi: 10.4103/jehp.jehp_446_20.
[9]. “An E-Commerce Based Personalized Health Product Recommendation System Using CNN-Bi-LSTM Model,” International Journal of Intelligent Engineering and Systems, vol. 16, no. 6, pp. 398–410, Dec. 2023, doi: 10.22266/ijies2023.1231.33.
[10]. B. R. R. Reddy and R. L. Kumar, “A Fusion Model for Personalized Adaptive Multi-Product Recommendation System Using Transfer Learning and Bi-GRU,” Computers, Materials & Continua, vol. 81, no. 3, pp. 4081–4107, 2024, doi: 10.32604/cmc.2024.057071.
[11]. U. Chadha et al., “Powder Bed Fusion via Machine Learning-Enabled Approaches,” Complexity, vol. 2023, pp. 1–25, Apr. 2023, doi: 10.1155/2023/9481790.
[12]. Mahalakshmi, R. L. Kumar, K. S. Ranjini, S. Sindhu, and R. Udhayakumar, “Efficient authenticated key establishment protocol for telecare medicine information systems,” Industrial, Mechanical and Electrical Engineering, vol. 2676, p. 020006, 2022, doi: 10.1063/5.0117522.
[13]. A. Atreya and D. O’Shea, “Novel lossy compression algorithms with stacked autoencoders,” Stanford University CS229, Tech. Rep., 2009.
[14]. P. Krishnamoorthy et al., “Effective Scheduling of Multi-Load Automated Guided Vehicle in Spinning Mill: A Case Study,” IEEE Access, vol. 11, pp. 9389–9402, 2023, doi: 10.1109/access.2023.3236843.
[15]. R. K. Poluru and R. Lokeshkumar, “Meta-Heuristic MOALO Algorithm for Energy-Aware Clustering in the Internet of Things,” International Journal of Swarm Intelligence Research, vol. 12, no. 2, pp. 74–93, Apr. 2021, doi: 10.4018/ijsir.2021040105.
[16]. 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,” CoRR, vol. abs/1511.06085, 2015. [Online]. Available: http://arxiv.org/abs/1511.06085.
[17]. G. Toderici et al., “Full Resolution Image Compression with Recurrent Neural Networks,” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5435–5443, Jul. 2017, doi: 10.1109/cvpr.2017.577.
[18]. A. Srinivasan and G. Rohini, “RETRACTED ARTICLE: An Improvised video coding algorithm for deep learning-based video transmission using HEVC,” Soft Computing, vol. 23, no. 18, pp. 8503–8514, Jun. 2019, doi: 10.1007/s00500-019-04093-1.
[19]. S. Bouaafia, R. Khemiri, S. Messaoud, O. Ben Ahmed, and F. E. Sayadi, “Deep learning-based video quality enhancement for the new versatile video coding,” Neural Computing and Applications, vol. 34, no. 17, pp. 14135–14149, Sep. 2021, doi: 10.1007/s00521-021-06491-9.
[20]. 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, Jun. 2021, doi: 10.1016/j.asej.2021.01.001.
[21]. L. Zhao, H. Bai, A. Wang, and Y. Zhao, “Learning a virtual codec based on deep convolutional neural network to compress image,” Journal of Visual Communication and Image Representation, vol. 63, p. 102589, Aug. 2019, doi: 10.1016/j.jvcir.2019.102589.
[22]. A. Jacob, V. Pawar, V. Vishwakarma, and A. Mane, “Deep Learning Approach to Video Compression,” 2019 IEEE Bombay Section Signature Conference (IBSSC), pp. 1–5, Jul. 2019, doi: 10.1109/ibssc47189.2019.8973035.
[23]. N. Krishnadoss and L. K. Ramasamy, “Crop yield prediction with environmental and chemical variables using optimized ensemble predictive model in machine learning,” Environmental Research Communications, vol. 6, no. 10, p. 101001, Oct. 2024, doi: 10.1088/2515-7620/ad7e81.