Efficient Video Compression System for Resource Constrained Devices Using Block Skipping

Pradyumn Vikram
240759

Arya Achal Mehta
240202

Akshat Sathiya Narayanan
240087

Contents

Abstract

Efficient video compression is a key challenge in modern multimedia systems, where minimizing computation power and bandwidth requirements is crucial. This work presents a trainable framework for efficient video compression for further transmission using an encoder-decoder based architecture. Generated skip masks identify and encode redundant blocks for each frame with reference to the previous one, which are then passed to the convolutional encoder network - a thresholding step for the interpolated (down-sampled) skip-masks in latent-space, using element-wise summation to determine if it's corresponding frame should be brought down to the latent space for further processing or the previous frame latent is a good substitute. This sequence is passed down to a ConvLSTM for temporal modeling, predicting the frame from the previous one, the difference of which generates the residuals vector. The residuals are subsequently quantized and entropy coded using a learned probability model, assuming Gaussian distribution for each individual latent element, for achieving the desired rate-distortion performance. Only the first latent representation and the residuals between the predicted and actual latents are transmitted along with the decoder parameters, at the receiver end, the ConvLSTM reconstructs the latent sequence by adding these residuals to the predicted latent vector, ensuring accurate recovery. On a sample video, the proposed model achieved an MSE of 0.001558, PSNR of 28.07 dB, and a rate metric of -5.487817 bpp.

1. Introduction

In today's world, video has become the primary medium for entertainment, data exchange, communication, and are increasingly being accessed through IoT - based systems such as CCTV cameras, dashcams, smart home devices, etc. Traditional compression methods, while effective, pose additional challenges on these edge devices and bandwidth-limited networks, where computational power is constrained.

The primary use case is on such devices where continuous recording creates large amounts of redundant data, where every frame does not carry 'useful' visual information. Therefore, a higher compression ratio is prioritized over extract reconstruction quality, to ensure minimum computational power has been used.

Traditionally ConvLSTM's in compression models operate on the entire frame's latent vectors without spatial selectivity therefore, they redundantly reuse full-frame latents, leading to a waste in computation power. We propose a novel method to skip blocks of pixels, which consist of some similarity with their counterparts in the previous frame, and have come up with a block skip model using bit masks (referred to as skip masks) to skip frames, in the latent vector space, with redundant information for efficient encoding and transmission. To ensure the applicability of the model on resource constrained devices, we also propose to replace the entropy model with an appropriate estimate by assuming a Gaussian distribution for the latent vector element values.

2. Proposed Method

2.1. Frame Preprocessing

2.1.1. Resizing and Normalizing

All video frames are first resized to a fixed spatial resolution of 128×128×3. This standardization is essential so that the encoder is not overloaded as it accepts inputs only of a fixed dimension, which simplifies the processing pipeline and avoids inconsistencies due to varying video resolutions. Additionally, uniform scaling ensures that spatial features are learned more effectively as the encoder encounters patterns at a consistent scale across all samples.

2.1.2. Color Space Conversion to YUV

Once the frames are resized, they are converted from RGB to YUV. This separates the brightness information (Y) from the color information (U and V), helping the model pay more attention to luminance, a key factor in how humans perceive video quality. Working in YUV also reduces unnecessary redundancy in color data, making compression more efficient and helping the network learn more useful features.

2.1.3. Block-Skip Preprocessing

This step is done to remove temporal redundancy in a frame sequence. Each frame is divided into 8x8 blocks, and the difference between the corresponding blocks in consecutive frames is measured. If the corresponding block difference is below a set threshold value, we set our corresponding mask pixels to zero; otherwise, we set it to one. This is done to reduce the computation cost for transmission and will help decide on selecting latent vectors for representation.

2.2. Skip Aware Encoding

2.2.1. Latent-level Mask Check

Before performing any computation, the skip mask generated for each frame is sampled down via 2 convolution blocks each consisting - a convolution filter followed by a batch normalization succeeded by a residual block with skip connections for efficient feature extraction, taking the input frame from a shape of (3, 128, 128) to (32, 64, 64) to (48, 32, 32) and finally to the dimension corresponding to the latent space - (24, 32, 32) via a final convolution layer. The arithmetic sum of the latent mask produced is performed to see if any latent features in the current frame are significantly affected by changes(if the sum is greater than 0). If no latent-level features have changed, the encoder directly reuses the latent representation of the previous frame, and thus avoids any unnecessary computation while retaining video integrity.

2.2.2. Latent Channel Generation

For frames requiring processing, the encoder applies a sequence of convolutions and residual layers specified in 2.2.1 to extract the most relevant features in the latent space. This latent representation captures the essential structure and appearance of each frame while discarding redundant details.

2.3. Temporal Prediction

2.3.1. ConvLSTM Based Modeling

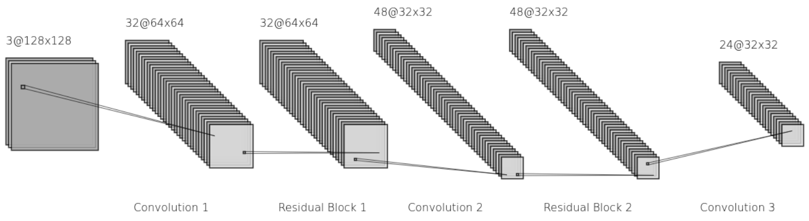
To predict the next frame, the latent channel is passed through a ConvLSTM. It maintains a hidden state (which acts as long term memory), allowing it to model temporal variations between consecutive frames. The model predicts each new frame based on a combination of the updated hidden state and the previous latent representation. For all subsequent frames, the ConvLSTM updates its hidden and cell states using the previous latent representation, effectively learning a temporal pattern across frames, and predicts using a combination of the hidden state and previous latent frame.

2.3.2. Refinement Network

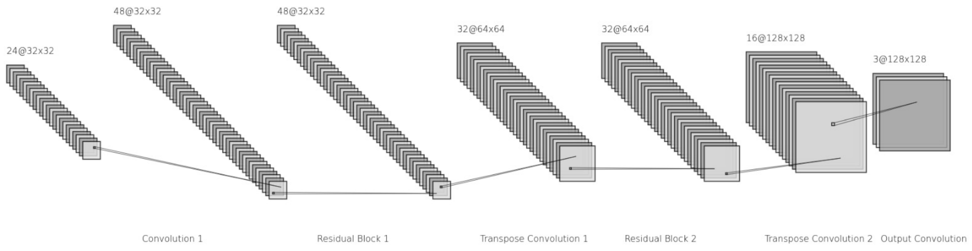
The ConvLSTM produces a rough estimate of the next latent frame, and this resultant output is passed through a refinement network made of 2 convolutions with a ReLU layer sandwiched in between. This is used to clean the ConvLSTM's prediction, i.e., to reduce the noise of the latent channel, and hence give us better representations for the input frame.

2.4. Computing Residuals

The residual is calculated by subtracting the prediction of the ConvLSTM with the actual latent features generated by the encoder. As mentioned in the abstract, we propose to transmit the first latent channel and the calculated residual



(a) Encoder architecture with convolution layers and residual blocks.



(b) Decoder architecture with convolution layers and residual blocks.

Figure 1. Encoder and Decoder models

matrix. On the receiver end, using the exact same temporal prediction model as was used during transmission, the latent space is rebuilt. The residuals are then added back to the reconstructed latent space matrix, to recover the exact latent space representation produced by the encoder. This is further passed on to the decoder model, to reconstruct the original sequence of frames, i.e, the final representation of the transmitted video.

$$\Delta L_{residual}^{(t)} = L_{original}^{(t)} - \Delta L_{predicted}^{(t)} \tag{1}$$

2.5. Adaptive Quantizing

To prepare the residual matrix for entropy modeling and enable encoding for final data transmission, the latent values must be, at evenly spaced intervals ($\frac{1}{N-1}$). Instead of using a fixed quantization scale, the model learns an adaptive scaling factor during training, allowing it to adjust the degree to which values are compressed. The standard rounding operator is non-differentiable and thus unsuitable for gradient based optimization. Instead, a differential approximation is used by adding Gaussian Noise to simulate the uncertainty as shown in Equation (2). At inference time, the operation is replaced with a hard quantization step, as shown in Equation (3) that maps them to the nearest quantization level, i.e., approximating it to the closest value.

$$\bar{x} = sx + u, \quad u \sim \mathcal{U}(-0.5, 0.5) \tag{2}$$

$$Q(x) = \frac{\text{round}((N-1)\,sx)}{N-1} \tag{3}$$

2.6. Entropy Modeling

Most natural images exhibit strong local correlations, i.e., the pixels or features are predictable from their temporal and spatial neighbours. The model receives the quantized residual latent map and for each element, it learns two parameters, Mean μ_i and Scale (standard deviation) σ_i, which represent the spatial and temporal context of each latent coefficient and are used to estimate the probability distribution of that element. A Gaussian Distribution is used to model these residuals, as prediction errors in image and videos tend to be symmetric in the sense that there is usually balanced over and underestimation of values. Thus this formulation provides a realistic statistical modeling, while the differentiable nature makes it a good choice for computation of likelihoods for end-end training which relies on gradient based optimization.

$$p(x_i|\mu_i, \sigma_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(x_i - \mu_i)^2}{2\sigma_i^2}\right) \tag{4}$$

Based on Shannon's information theory, higher predicted probabilities correspond to more predictable latent elements, which require fewer bits to represent, making them more efficient to encode during transmission.

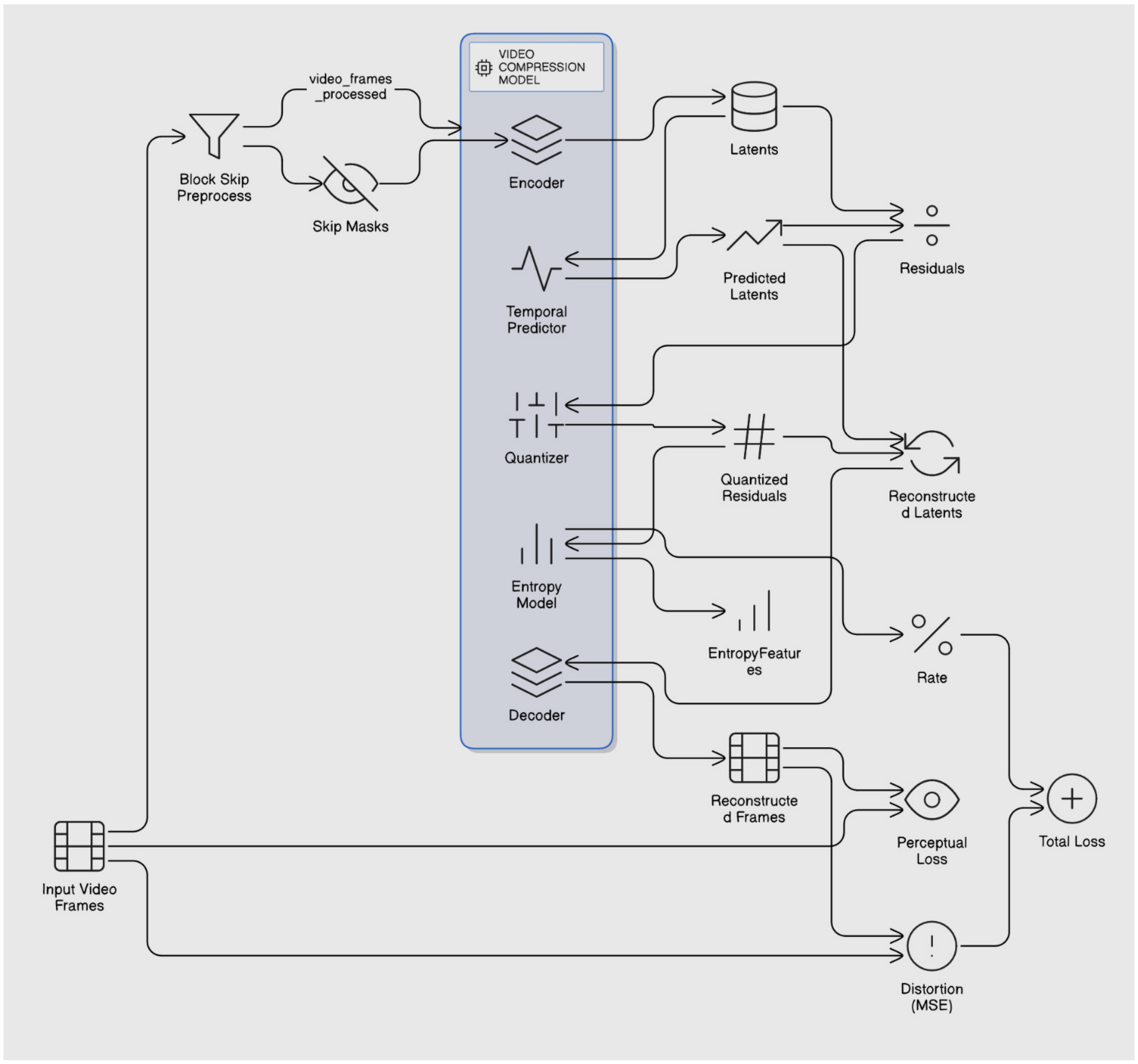


Figure 2. Model Architecture

However, in practice, implementing Shannon's discrete formulation is computationally expensive and requires the knowledge of the true probability distribution, which is not directly available. Therefore, we approximate it using a continuous Gaussian model for the latent residuals, which follows similar information-theoretic properties.

$$H^{(t)} = -\int_{-\infty}^{\infty} p(x^{(t)})\,\log_2 p(x^{(t)})\,dx^{(t)}. \tag{5}$$

$$H_i^{(t)} = \frac{1}{2}\log(2\pi e\sigma_i^2)$$

2.7. Decoding Frames

2.7.1. Rebuilding Latent Space

After receiving the first frame post transmission, the exact same temporal model used during encoding, is run to predict the remaining sequence of latent channels. To this we add the residual matrix, received during transmission. This new matrix of predicted frames, with residuals added, mirrors the one originally produced during encoding.

$$L_{final}^{(t)} = \hat{L}_{reconstructed}^{(t)} + \Delta L_{residuals}^{(t)}. \tag{6}$$

2.7.2. Refining Latent Features

Once the latent representation is recovered, it is passed through a convolutional layer, followed by batch normalization, and finally a residual block, taking the original latent dimension from (24, 32, 32) to (48, 32, 32). This is used to strengthen the base structure for decoding, by attempting to suppress noise generated during quantization and residual transmission. The new feature map provides a more stable foundation for upsampling, that we perform next.

2.7.3. Progressive Upsampling

After the latent structure is enhanced, the decoder reconstructs the high-resolution frame through a sequence of upsampling steps. In the first step, the matrix is passed through a transposed convolution layer that expands the spatial resolution from (48, 32, 32) to (32, 64, 64), followed by batch normalization and a residual block. Next, a second transposed convolution layer further upsamples the feature matrix from (32, 64, 64) to (16, 128, 128), followed by batch normalization. We don't directly upsample to (3, 128, 128) as such a high resolution change with only 3 channels would lead to a significant amount of information loss, and noticeably lower accuracy in video reconstructions. Finally, a 3×3 convolutional layer maps the features from (16, 128, 128) back to the original RGB space of (3, 128, 128). This architecture as observed is almost the mirror of the architecture proposed in the encoding stage, this aligns the end goal idea of accurate video reconstruction.

3. Training and Evaluation

3.1. Evaluation Metrics

3.1.1. Distortion

Distortion is calculated using mean squared error (MSE) between reconstructed frame and its corresponding input frame, it measures the deviation of predicted pixels from original pixels.

$$\text{MSE} = \frac{1}{T\,C\,H\,W}\sum_{t=1}^{T}\sum_{c=1}^{C}\sum_{i=1}^{H}\sum_{j=1}^{W}\left(x_{c,i,j}^{(t)} - \hat{x}_{c,i,j}^{(t)}\right)^2 \quad (7)$$

3.1.2. Rate Metric (bpp)

Bits per pixel (bpp) represents the average number of bits required to encode a single pixel, is a normalized measure of compression efficiency as this metric, it is independent of image size and resolution.

For a video sequence of T frames, and N denoting the number of latent elements in one frame. The overall rate metric is then approximated by averaging these entropy estimates across all elements and frames:

$$R \approx \frac{1}{N \cdot T}\sum_{t=1}^{T}\sum_{i} H_i^{(t)}.$$

3.1.3. Peak Signal-to-Noise Ratio (PSNR)

PSNR gives a measure of quality of reconstruction in decibels (dB), derived directly from the MSE:

$$\text{PSNR} = 10\log_{10}\left(\frac{1}{\text{MSE}}\right) \quad (8)$$

Higher PSNR corresponds to more accurate reconstruction and lower distortion.

3.1.4. Total loss

The total loss is defined as a weighted sum of the distortion error, and the rate metric term, which estimates the number of bits required to encode the latent representation.

$$\mathcal{L}_{\text{total}} = \text{MSE} + \alpha R \quad (9)$$

The total loss is what is used to train the whole architecture using gradient descent to obtain best fit values of all parameters.

3.2. Training

- The encoder and decoder models are trained together with end to end loss calculation in order to learn the video features as efficiently as possible, thereby ensuring a good compression.
- The model was trained for 25 epochs with the training being done in chunks of 10 frames, in order to utilize memory efficiently.
- The graphs of the evolution of the corresponding training parameters over the course of training can be seen as a function of the number of epochs in Figure 3.
- The best model achieved a PSNR score of $28.07dB$, a distortion margin of 0.001558 and an optimized rate of -5.487817 resulting in a net loss of -0.025881.

4. Experiments Conducted

4.1. Shannon entropy Vs Parametric entropy

OBJECTIVE: The traditional Shannon loss entropy model proved to be highly computational in nature and was not at all feasible for fast processing, An alternative for the same is proposed in this subsection

OBSERVATIONS: The latent elements can be safely assumed to belong to a Gaussian distribution function with corresponding μ and σ, this approximation can be used to simplify the Continuous Shannon Entropy function ($\int f(x)\log f(x)dx$), on further simplification resulting in the estimate formula used in 2.6. On comparing the two entropy models on a T4 GPU with 8GB RAM, the formula without the Gaussian assumption, took an exponentially larger amount of time to train, eventually not giving a result at all due to resource constraints whereas the Gaussian

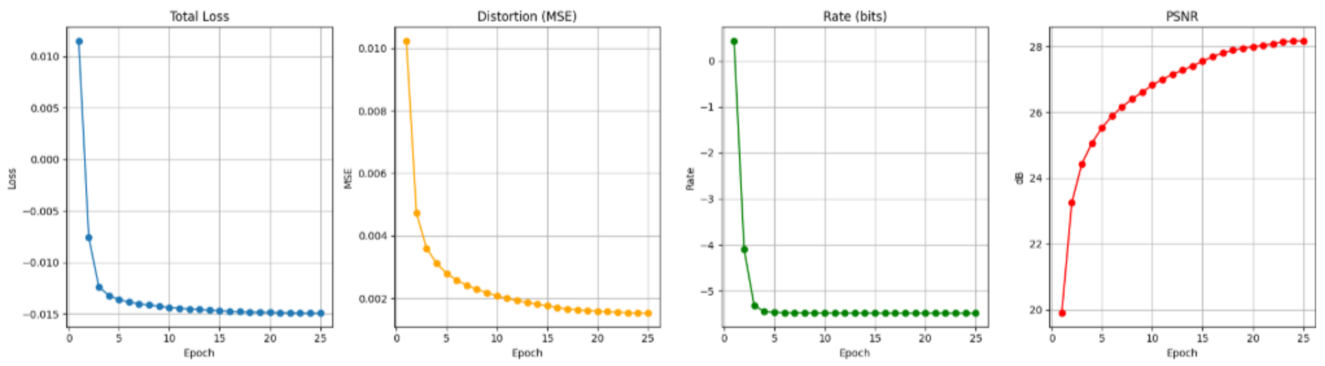


Figure 3. Evolution of training parameters as functions of epochs

assumed parametric model was able to train completely for 25 epochs successfully

Inference: This shift to the new parametric entropy model resulted in a faster training time, solving the computation intensive problem

4.2. Including vs Excluding Perceptual Loss

Objective: To study the effect of including Perpetual loss, which quantizes the error in the predicted latent space as compared to the actual latent space, in efforts to reduce the residuals and thereby making transmission load even lighter.

Observations: The PSNR is 26.01 when perceptual cost is included in total loss which drives our gradient descent, and 28.06 when it is not. That is we are getting a better reconstructed video quality when perceptual loss is not included.

Inference: The extra amount of computation, results in worse outcomes and hence perceptual loss has been excluded as a training metric.

4.3. Block Skipping Vs No Skipping

Objective: To study the effect on the video of block skipping, compared to no skipping at all. Block skipping is much more efficient for the system to handle, but might come at the cost of image distortion.

Observations: PSNR calculated when no frame/sub-block was skipped was 28.10, whereas with block skipping came out to be 28.06. As expected the reconstructed quality depriciated.

Inference: Block skipping was implemented as the resource efficiency vs quality tradeoff was very skewed in favour of block skipping, as observed there is barely a change in the reconstruction quality.

4.4. YUV vs RGB for Operational Color space

Objective: To understand which color space would be more efficient to work in as YUV works in an illumination basis which is better percieved by human eye for major changes, but is not as widely used.

Observations: PSNR value obtained with RGB is 24.11, whereas that with YUV is 28.06, indicating a large difference between the two.

Inference: YUV colorspace was used during model inference and training.

5. Appendix

1. **Project Repository**: Source Code and Demonstration

References

[1] *Efficient Video Neural Network Processing Based on Motion Estimation*, https://arxiv.org/html/2501.15119v1

[2] *Enhancing Cloud-Based Video Streaming Efficiency using Neural Networks*, https://ieeexplore.ieee.org/document/10189314/.

[3] *Review of Deep Learning Methods for Enhanced Video Compression*, https://ieeexplore.ieee.org/document/10649029/.

[4] *Enhanced Neural Video Compression for Cloud Gaming Videos with Aligned Frame Generation*, https://www.sciencedirect.com/science/article/pii/S0957417424024023.