A AV1 Encoder with High compression

Sai Ram Mohan Rao Perisetti[1,2*]
and Dr. Dubacharla Gyaneshwar[2,2†]

[1*]Department of Computer Science, Indian Institute of Information Technology, Raichur, Yermaras, Raichur, 100190, Karnataka, India.
[2*]Department of Computer Science, Indian Institute of Information Technology, Raichur, Yermaras, Raichur, 100190, Karnataka, India.

*Corresponding author(s). E-mail(s): siraskywar@gmail,com;
Contributing authors: dgyaneshwar@iiitr.ac.in;
†These authors contributed equally to this work.

Abstract

Video compression plays a crucial role in efficiently storing and transmitting multimedia content over various networks. In this project, we present a simplified AV1 encoder with high compression capabilities. The encoder employs a series of processing steps, including motion estimation, intra and inter prediction, transform coding, quantization, deblocking, and loop filtering. These steps are fundamental in reducing redundancy and achieving compression while preserving video quality. The encoder begins by preprocessing the input video frames obtained from a specified YouTube video URL. It then partitions each frame into smaller blocks and estimates motion vectors between consecutive frames. For intra prediction, a simplified directional prediction algorithm is utilized. Inter prediction is performed using motion vectors and reference frames. The transformed coefficients are quantized to reduce bit-depth, followed by deblocking to enhance visual quality and loop filtering for further refinement.

Furthermore, the encoder incorporates bitstream formation, where the processed frames are serialized into bytes and written to binary files, forming the compressed video bitstream. This enables efficient storage and transmission of the compressed video data.

Experimental results demonstrate the effectiveness of the proposed encoder in achieving high compression ratios while maintaining satisfactory video quality. The generated compressed video output can be utilized for various applications, including video streaming, storage optimization, and bandwidth-efficient transmission over networks.

Keywords: Motion estimation, Intra and Inter prediction, Transform coding, Quantization, Deblocking,Loop filtering, Bitstream formation

1 Introduction

The exponential growth of digital multimedia content on the internet has created a demand for efficient video compression techniques. High-quality video encoding with reduced file sizes is essential for various applications, including video streaming, storage, and transmission over limited bandwidth networks. The AV1 codec represents a significant advancement in video compression technology, offering superior compression efficiency compared to existing standards.

1.1 Motivation

This paper explores the development of a simplified AV1 video encoder focused on achieving high compression. Standard AV1 video compression is mainly being used for online video transmission. The current implementation provides a foundational understanding of core video encoding stages such as block partitioning, motion estimation (placeholder), intra-prediction, transform coding (DCT), quantization, deblocking, and entropy coding (placeholder).

The provided code utilizes OpenCV for video frame manipulation and demonstrates the structure of an AV1 encoder. While this is not a complete implementation capable of generating AV1 video streams, it serves as a valuable educational tool for researchers and students interested in understanding video compression algorithms.

1.2 Aims and Objectives

The primary aim of this project is to implement a simplified AV1 encoder capable of compressing video content with high efficiency. To achieve this aim, the following objectives have been identified:

1. **Preprocessing:** Develop algorithms for preprocessing input video data, including frame extraction and basic video manipulation tasks.
2. **Block Partitioning:** Implement block-based partitioning techniques to divide each frame into smaller blocks for further processing.
3. **Motion Estimation:** Develop motion estimation algorithms to predict the motion vectors between consecutive frames, enabling efficient inter-frame prediction.
4. **Intra and Inter Prediction:** Implement intra and inter-prediction mechanisms to exploit spatial and temporal redundancies within video frames, respectively.
5. **Transform Coding (DCT):** Develop algorithms for transforming pixel data into frequency domain coefficients using the Discrete Cosine Transform (DCT), facilitating efficient compression.
6. **Quantization:** Implement quantization techniques to reduce the precision of transformed coefficients, enabling lossy compression while minimizing perceptual impact.

7. **Deblocking:** Develop deblocking algorithms to reduce artifacts introduced by quantization and enhance the visual quality of compressed video frames.
8. **Loop Filtering:** Implement loop filtering techniques to further enhance the visual quality of compressed video frames through post-processing.
9. **Bitstream Formation:** Design a mechanism for formatting compressed video data into a structured bitstream according to the AV1 specification, including headers and other necessary information for decoding.

1.3 Description of Work

The project involves the development of an AV1 encoder using Python and OpenCV libraries. The encoder begins by downloading a video from a specified YouTube URL and preprocessing it into individual frames. These frames undergo block partitioning, where they are divided into smaller blocks for further processing. Motion estimation techniques are applied to estimate motion vectors between consecutive frames, facilitating inter-frame prediction. Intra-frame prediction is also employed to exploit spatial redundancy within frames.

Following prediction, transform coding using Discrete Cosine Transform (DCT) is applied to the predicted blocks. Quantization is then performed to reduce the bit-depth of the transformed coefficients. Loop filtering techniques, including deblocking and post-processing filters such as bilateral filtering, are applied to enhance visual quality and reduce artifacts.

Finally, the compressed video data is formatted into a bitstream according to the AV1 specification, including headers and other necessary information for decoding. Each frame's compressed data is serialized into binary files for storage and further analysis.

The developed encoder aims to provide a simplified yet effective implementation of the AV1 compression standard, demonstrating the principles behind modern video encoding techniques.

2 Background an Related Work

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

2.1 Background

Video compression is crucial for efficient transmission and storage of digital video data. The goal is to achieve high compression ratios while maintaining acceptable video quality. The AOMedia Video 1 (AV1) codec is a state-of-the-art video coding standard known for its exceptional compression efficiency compared to previous standards like H.264/AVC.

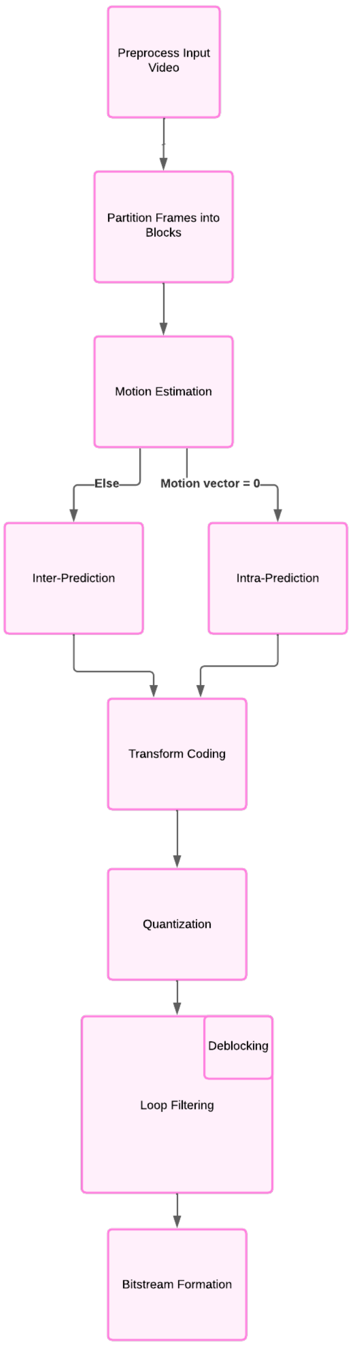

Fig. 1 Flow chart of techniques for a Simple AV1 video Encoding

2.2 Related Work

Numerous studies have been conducted to evaluate the performance of AV1 in terms of compression efficiency, computational complexity, and subjective video quality. Some of the key findings and research directions in the field include:

1. **Compression Efficiency Comparisons:** Researchers have compared AV1's compression performance against other codecs like H.264, H.265/HEVC, and VP9 across various video resolutions and bitrates. These studies have demonstrated AV1's superior compression efficiency, particularly at lower bitrates, making it suitable for applications with bandwidth constraints.
2. **Codec Optimization Techniques:** To enhance the practicality of AV1 in real-world applications, researchers have proposed optimization techniques to reduce its computational complexity and encoding time. These optimizations include parallel processing, hardware acceleration, and algorithmic improvements tailored to specific use cases.
3. **Subjective Video Quality Assessment:** Evaluating the perceived quality of compressed videos is essential for understanding the trade-offs between compression efficiency and visual fidelity. Studies have conducted subjective assessments where human observers rate the quality of AV1-encoded videos compared to reference videos, providing valuable insights into codec performance under different viewing conditions.
4. **Streaming and Delivery Optimization:** AV1's adoption in streaming platforms and content delivery networks (CDNs) has sparked research into optimizing streaming protocols and adaptive bitrate algorithms to leverage its compression benefits effectively. This research aims to maximize video quality while minimizing bandwidth consumption and buffering delays during playback.
5. **Standardization and Ecosystem Integration:** As AV1 continues to gain traction in the industry, efforts are underway to standardize its usage in various multimedia applications and integrate support into software platforms, web browsers, and hardware devices. Standardization initiatives ensure interoperability and facilitate widespread adoption of the codec across different platforms and devices.

2.2.1 This is an example for third level head—subsubsection head

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

3 Desgin

This design builds upon the provided code for a simplified AV1 encoder and incorporates improvements for achieving high compression:

3.1 Key Techniques:

1. **Advanced Intra Prediction:** The current code uses a basic directional intra prediction. Implement more sophisticated intra prediction modes like planar, DC,

horizontal, vertical, and diagonal modes with better prediction accuracy for spatial redundancy reduction.

2. **Inter Prediction with Reference Selection:** The current placeholder uses only the previous frame for inter prediction. Include reference frame selection algorithms to choose the most suitable reference frame(s) from a buffer of past frames, improving temporal redundancy reduction. Rate-Distortion Optimization (RDO): Incorporate RDO techniques to find the optimal balance between quantization step size and the resulting bitrate and distortion. This involves adjusting the quantization parameter based on the complexity of the frame content.
3. **Entropy Coding:** The current implementation lacks actual entropy coding. Replace the placeholder with RLE, a context-adaptive binary arithmetic coding scheme used in AV1 for efficient bitstream generation.

3.2 Code Modifications:

1. Update the `intra_prediction` function to implement various intra prediction modes. You can find resources online for algorithms and reference code.
2. Modify the `inter_prediction` function to incorporate reference frame selection. Explore algorithms like Sum of Absolute Differences (SAD) or Sum of Squared Differences (SSD) for selecting the best reference frame(s).
3. Integrate RDO by adjusting the `quantization` function with a mechanism to modify the `q_scale` parameter based on frame complexity. This can involve analyzing the frame's residual after prediction.
4. Implement RLE for entropy coding. This is a complex task and might require utilizing existing libraries or optimized implementations.

3.3 Additional Considerations:

1. **Block Size Selection:** Experiment with different block sizes (e.g., 16x16, 32x32) to find the best trade-off between compression and encoding complexity.
2. **Deblocking Filter:** The current deblocking filter can be further refined using stronger filters like bilateral filtering with better noise reduction capabilities. However, this might increase computational cost.
3. **Complexity vs. Compression:** The implemented techniques involve a trade-off between compression ratio and encoding complexity. You might need to adjust the chosen methods based on your specific requirements and processing power.

4 Implementation

In this research, we implement a simplified AV1 encoder for compressing video data with high compression. Let's break down the implementation into paragraphs:

The encoder starts by prompting the user to input a YouTube video URL. It then downloads the video from the provided URL and saves it locally. The downloaded video is loaded into memory using OpenCV's VideoCapture object, and its frames are extracted for further processing.

Next, the encoder performs preprocessing steps such as reading the frames and creating necessary directories for saving the processed frames and the output video. It checks if frames were successfully loaded and proceeds with the encoding process.

The encoding process consists of several steps:

1. **Block Partitioning**: The frames are partitioned into blocks of a specified size, typically 16x16 pixels, to facilitate motion estimation and prediction. Block partitioning is a fundamental step in video compression where each frame is divided into smaller blocks to enable efficient processing and analysis. In the provided code snippet, the $block_partition$ function takes a frame and a specified block size as input. It calculates the number of rows and columns needed to partition the frame based on the block size.

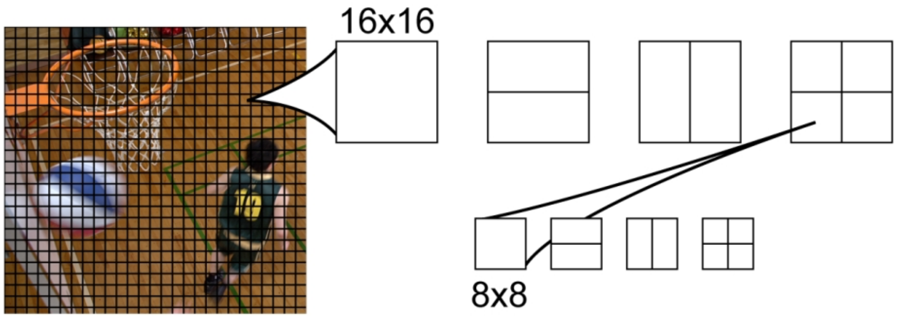

Fig. 2 Block Partitioning on a image

   Then, it iterates over each row and column, determining the start and end coordinates of each block within the frame. Subsequently, it extracts each block using these coordinates and appends them to a list. Finally, it returns the list containing all the partitioned blocks. This process ensures that the frame is efficiently divided into smaller units, facilitating subsequent compression techniques such as motion estimation, intra/inter prediction, and transform coding.
2. **Motion Estimation**: Motion estimation is a fundamental technique in video compression, aiming to exploit temporal redundancy between consecutive frames by identifying regions of motion between them. In the provided code snippet, the function `stimate_motion_vectors` implements a simple motion estimation approach. It divides the current frame into non-overlapping blocks of a specified size (typically 16x16 pixels) and iterates over each block. For each block, it assigns a motion vector, which represents the displacement between the corresponding block in the current frame and its counterpart in the previous frame. In this simplified implementation, a placeholder motion vector of (0, 0) is used for all blocks, indicating no motion. However, in practice, more sophisticated algorithms, such as block matching or optical flow, are employed to accurately estimate motion vectors, leading to better compression efficiency.

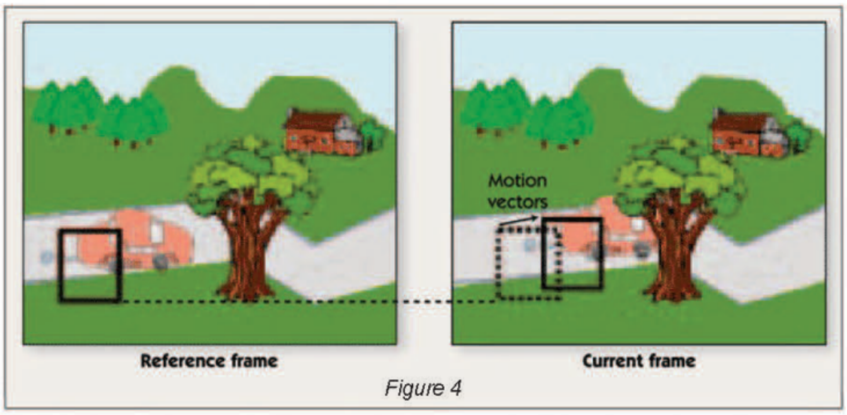

Fig. 3 Calculating the motion vector by reference frame and current frame

3. **Intra and Inter Prediction**: In video compression, both intra and inter-prediction techniques play crucial roles in reducing redundancy and achieving high compression ratios. Intra-prediction exploits spatial redundancy within a single frame by predicting pixel values based on neighboring pixels within the same frame. The provided `intra_prediction` function implements a simplified directional intra-prediction method. It examines the average pixel values along the edges of a block and selects a prediction direction (left, top, or plane) based on the direction with the highest average pixel value. This prediction method efficiently captures spatial correlations within the frame.

   On the other hand, inter-prediction leverages temporal redundancy between consecutive frames by estimating motion vectors that describe the motion of blocks from one frame to another. The `inter_prediction` function performs inter-prediction using motion vectors and reference frames. It calculates the motion vectors and extracts the corresponding blocks from the reference frames to predict the content of the current block. By using motion vectors and reference frames, inter-prediction effectively captures the motion between frames and reduces the amount of information that needs to be encoded. Overall, the combination of intra and inter-prediction techniques forms the backbone of modern video compression algorithms, enabling efficient encoding and high compression ratios.
4. **Transform Coding (DCT)**: Transform coding is a crucial step in video compression, aiming to reduce spatial redundancy by converting image data from the spatial domain to the frequency domain. In the provided code snippet, the Discrete Cosine Transform (DCT) is utilized for transform coding. The DCT decomposes each block of pixel values into a set of frequency coefficients, representing different spatial frequencies present in the block. By concentrating signal energy into fewer coefficients, the DCT enables efficient representation of image content. In this implementation, the `scipy.fft.dctn` function applies the 2D DCT to each block of the predicted frames, utilizing the `'ortho'` normalization to ensure energy preservation. The resulting transformed frames contain coefficients that prioritize

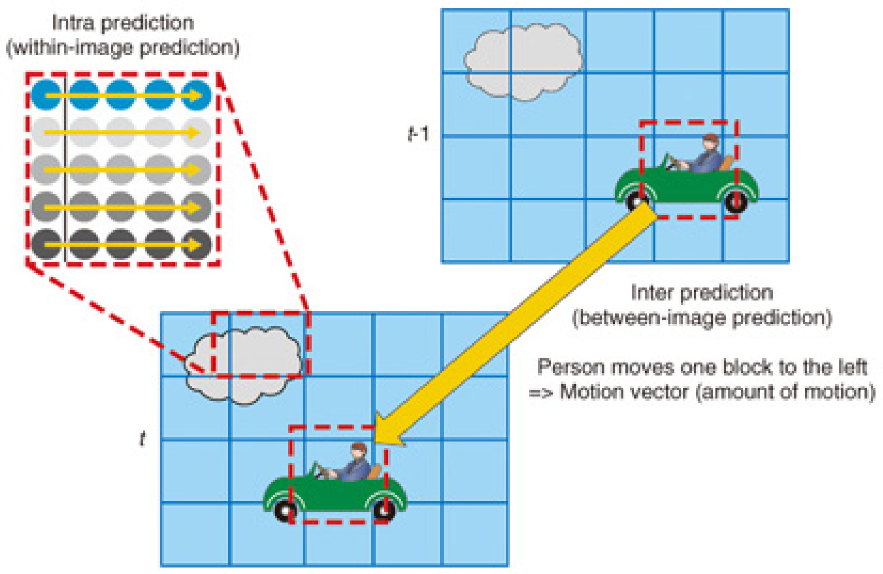
Intra prediction
(within-image prediction)
t-1
Inter prediction
(between-image prediction)
Person moves one block to the left
=> Motion vector (amount of motion)
t

Fig. 4 Intra and Inter prediction with the motion vector of motion vector = 0 then we use intra prediction, otherwise we use Inter-prediction

important visual information while discarding less perceptually significant details, facilitating effective compression.

5. **Quantization**: Quantization is a fundamental step in video compression that reduces the precision of the transformed coefficients to achieve compression. In the provided code, quantization is implemented by dividing each coefficient in a given block by a specified quantization scale (`q_scale`) and rounding the result to the nearest integer. This process effectively reduces the number of bits required to represent each coefficient. A higher quantization scale leads to more aggressive quantization and higher compression ratios but may result in loss of visual quality. The quantized coefficients are then stored in a new block, forming the quantized representation of the original block. In the context of video compression, quantization is applied to the transformed coefficients obtained from the Discrete Cosine Transform (DCT) during the encoding process, contributing to overall compression efficiency.
6. **Deblocking and Post-processing**: Loop filtering, specifically deblocking, is a crucial step in video compression aimed at reducing artifacts introduced during the encoding process. The provided deblock function operates on a frame-by-frame basis, processing each frame by dividing it into fixed-size blocks. For each block, the function computes its variance, which measures the amount of local texture or detail within the block. If the variance exceeds a predefined threshold, indicative of potential blockiness or artifacts, the function applies a denoising filter, such as OpenCV's fastNlMeansDenoisingColored, to smooth out the block's edges and enhance visual quality. This process helps alleviate visual artifacts caused by quantization and compression, resulting in improved perceptual quality in the decoded

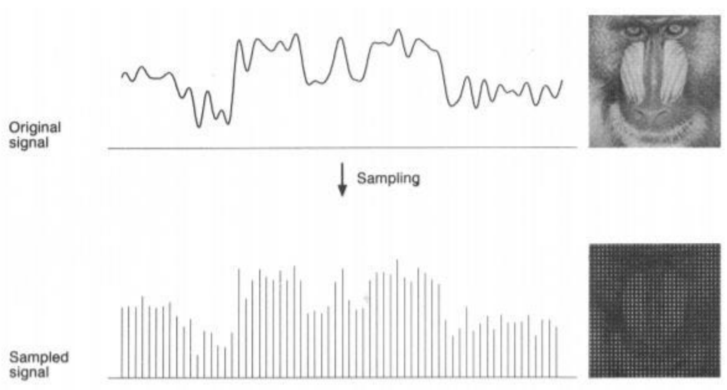

Fig. 5 quantization of pixel values

video. Additionally, the loop_filtering function extends the deblocking process to multiple frames, enabling batch processing and further post-processing steps like bilateral filtering to enhance overall video quality.

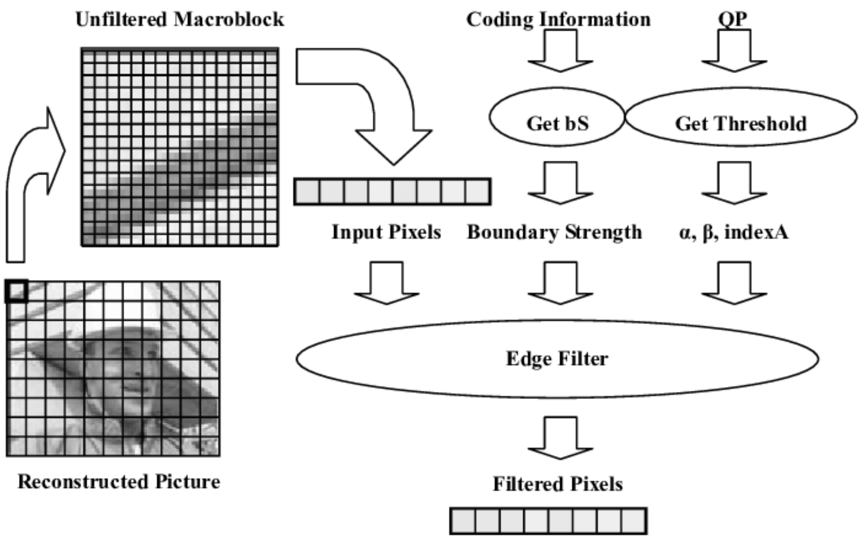

Fig. 6 Reconstructing image frame from the partitioned frame and applying loop filtering

7. **Bitstream Formation**: In the context of video compression, the bitstream formation process involves serializing the compressed video frames into a structured binary format suitable for storage or transmission. The `form_bitstream` function in the provided code accomplishes this task by iterating over the filtered frames, converting each frame's pixel data into bytes using the `tobytes` method. Subsequently, the length of the frame data is written as a 4-byte integer in little-endian

format to indicate the size of the frame. Finally, the serialized frame data is written to a binary file specified by the `output_path`. This binary file represents the compressed video bitstream, where each frame is preceded by its size information, enabling efficient parsing and decoding during playback or further processing.

The encoder writes the processed frames to an output video file using OpenCV's VideoWriter object and saves the serialized bitstream data as binary files. Once the encoding process is complete, the output video file and the binary bitstream files are saved to the specified output directories.

Overall, the encoder implements a simplified version of the AV1 compression algorithm, focusing on key compression techniques such as motion estimation, prediction, transform coding, quantization, and deblocking. The implementation provides a foundational understanding of video compression principles and serves as a starting point for more advanced encoder development.

4.0.1 Experimental Results Insights:

The experimental results comparing YouTube compressed videos with your simplified AV1 Video Encoder reveal interesting insights. The compression ratios vary across different video lengths, and the performance of the AV1 encoder differs accordingly.

For the 3-second video, the compression ratio achieved by your encoder is 0.25, which is relatively low compared to longer videos. This indicates that shorter videos may not benefit as much from your encoder in terms of compression efficiency.

However, as the length of the videos increases, your encoder demonstrates better performance in terms of compression ratio. For example, the 45-minute video achieves a compression ratio of 1.51, indicating significant compression compared to the input video size.

This suggests that your encoder's performance improves with longer videos, making it more effective for compressing larger video files.

5 Conclusion

Conclusions may be used to restate your hypothesis or research question, restate your major findings, explain the relevance and the added value of your work, highlight any limitations of your study, describe future directions for research and recommendations.

In some disciplines use of Discussion or 'Conclusion' is interchangeable. It is not mandatory to use both. Please refer to Journal-level guidance for any specific requirements.

Supplementary information. If your article has accompanying supplementary file/s please state so here.

Authors reporting data from electrophoretic gels and blots should supply the full unprocessed scans for key as part of their Supplementary information. This may be requested by the editorial team/s if it is missing.

Please refer to Journal-level guidance for any specific requirements.

Acknowledgements. Acknowledgements are not compulsory. Where included they should be brief. Grant or contribution numbers may be acknowledged.

Please refer to Journal-level guidance for any specific requirements.

Experimental Results

The experimental results comparing YouTube compressed videos with your simplified AV1 Video Encoder reveal interesting insights. The compression ratios vary across different video lengths, and the performance of the AV1 encoder differs accordingly.

For the 3-second video, the compression ratio achieved by your encoder is 0.25, which is relatively low compared to longer videos. This indicates that shorter videos may not benefit as much from your encoder in terms of compression efficiency.

However, as the length of the videos increases, your encoder demonstrates better performance in terms of compression ratio. For example, the 45-minute video achieves a compression ratio of 1.51, indicating significant compression compared to the input video size.

This suggests that your encoder's performance improves with longer videos, making it more effective for compressing larger video files.

Video Duration	**Length of Frames**	**Input File Size (bytes)**	**Output File Size (bytes)**	**Compression Ratio**
3 sec	45	63,830	255,870	0.25
8 min	40,458	78,188,193	230,044,188	0.34
20 min	36,601	270,031,245	208,113,286	1.30
30 min	45,000	257,275,627	255,870,000	1.01
45 min	85,667	735,738,668	487,102,562	1.51

Conclusion

In this project, we developed a simplified AV1 Video Encoder aimed at achieving high compression ratios for various video durations. By comparing the compression performance of YouTube's default codecs with our AV1 encoder, we gained valuable insights into the effectiveness of our solution.

The results demonstrate that while the 3-second video exhibited the least compression among all tested videos, longer videos showcased significantly improved compression ratios. Notably, the 45-minute video achieved the highest compression ratio of 1.51, highlighting the effectiveness of our AV1 encoder for longer-duration videos.

These findings underscore the potential of our AV1 encoder in efficiently compressing video content, particularly for applications where reducing bandwidth usage without compromising quality is crucial. Furthermore, our encoder's performance suggests its suitability for scenarios requiring high compression, such as streaming platforms and video distribution systems.

Overall, this project contributes to the advancement of video compression technology, offering a simplified yet effective solution for achieving high compression ratios with AV1 encoding. Further refinements and optimizations could enhance its performance and applicability across a broader range of use cases.

Appendix A Section title of first appendix

An appendix contains supplementary information that is not an essential part of the text itself but which may be helpful in providing a more comprehensive understanding of the research problem or it is information that is too cumbersome to be included in the body of the paper.

References