Video compression using unsupervised learning techniques

Yash Agrawal MT23205
Vinayak Katoch MT23105
Aman Ranjan 2021376
Bikramjyot Singh Sethi 2021139
Indraprastha Institute of Information Technology
Delhi

Abstract

In this project, we have tried to do Video Compression using unsupervised Machine learning. This report gives stepwise details, like the research papers we read, various techniques used from preprocessing to EDA to algorithm creation, and finally, evaluation, are given. After doing this, we hope to compress the video using the techniques we learn.

1 Motivation

As people have started consuming video content, a lot of content is being produced and stored, which consumes a lot of storage space. Transferring this content from one place to another takes a lot of time, and transmission costs increase for more extensive data/videos. Also, much hardware must be manufactured to store that much content, and the need is increasing daily.
So, to save storage space and reduce the cost of transmission, video compression and decompression are needed. Also, various projects will look good on our resumes, so this project will add value to the collection of projects done in our college life and benefit us in the future.

2 Introduction

The demand for video compression is increasing daily in today's world, which is digitalized at a breakneck pace. So, video compression will add a boost to many things, like enhancement in the video sharing quality of social media platforms. It will also solve storage and cloud problems. This project is an effort to understand how video compression and decompression can be done.

In this project, we will devise a video compression technique using Machine Learning. We will be using various techniques, starting from gathering the datasets, which will include videos from Kaggle, and then applying Exploratory data analysis (EDA) and then applying some preprocessing techniques and followed by devising an algorithm that uses machine learning to compress data and then evaluating the final results.

3 Literature Review

3.1 Research Paper1

In a research paper titled "Adaptive Key Frame Extraction using Unsupervised Learning" published at Zhejiang University, China, by Yueting Zhuang, Yong Rui, Thomas S. Huang, and Sharad Mehrotra discuss the importance of key-frame extraction in video information retrieval.
Existing techniques are either computationally demanding or ineffective at capturing the content

Submitted to IIIT Delhi ML Project . Do not distribute.

properly. So, the authors suggested a new algorithm based on unsupervised clustering. The paper discusses techniques like Short Boundary, visual content-based methods, motion-based analysis, and other techniques and their benefits, strengths, and failures.

Finally, the authors propose their algorithm, which employs unsupervised clustering. It darts the assignment of the first frame as cluster center, and now new frames are compared based on their histograms. If similarity exceeds a certain threshold, they join the cluster; otherwise, a new cluster is formed, and the cluster centers are updated in each iteration. Keyframes are selected from sizeable clusters based on proximity to the center. The number of keyframes depends on the complexity of the video. For example, in a romantic vs. action film video with the same length, the keyframes are more in the action film video as the movement and switches between frames are more often.
The algorithm discussed in the paper offers a balance of efficiency and adaptability. The approach discussed in the paper offers a balance of efficiency and adaptability.

`https://drive.google.com/file/d/16X5h54FXxFYz0moM1nyPTZnX8R6w2Nrj/view?usp=sharing`

3.2 Research Paper 2

Another paper we read is titled “Shot Boundary Detection: Fundamentals Concepts and Survey ”.This discusses the process of segmenting video into smaller temporal units known as shots, and the process is known as Shot Boundary Detection(SBD). The paper reviews various SBD algorithms, focusing on recent advancements in machine learning . It also highlights the basic concepts, video hierarchy, and shot transition types. In short, the paper aims to explain and enhance automated shot boundary detection for efficient video content management.

`https://drive.google.com/file/d/17It8ybZEQQTqtCUIE4akoa-8Ij8tSrHU/view?usp=drive_link`

4 Dataset

We have used the dataset from Kaggle which contained 2 videos in the mkv format.The properties of each dataset are:

```
Total number of frames: 14406
Frame resolution: 3840 x 1608
Frames per second: 23.98
Video duration: 600.85 seconds
Already used Codec: hevc
Bitrate: 2340 bps
Audio Duration: 600.85 seconds
Audio Sample Rate: 44100 Hz
```

(a) Video 1 Properties

```
Total number of frames: 14291
Frame resolution: 1920 x 800
Frames per second: 23.81
Video duration: 600.22 seconds
Already used Codec: hevc
Bitrate: 3623 bps
Audio Duration: 600.24 seconds
Audio Sample Rate: 44100 Hz
```

(b) Video 2 Properties

Figure 1: Dataset Properties

5 EDA

5.0.1 Basic steps

In this, we first counted the frames and other properties of the video, like video duration, fps, bitrate, codec used, and frame resolution. Also, we have found audio properties, like duration and sample rate, for further use.
We have made splits of frames like converted the whole video into around 14000 frames. Futher on to reduce the computation cost we have taken one frame per second for further use.

5.0.2 Feature Extraction

We have used Feature extraction, by which we get properties of individual frames using histograms and other representation techniques and calculating the difference in each frame and taking the mean of the image difference matrix.

(a) Video 1 Brightness Plot

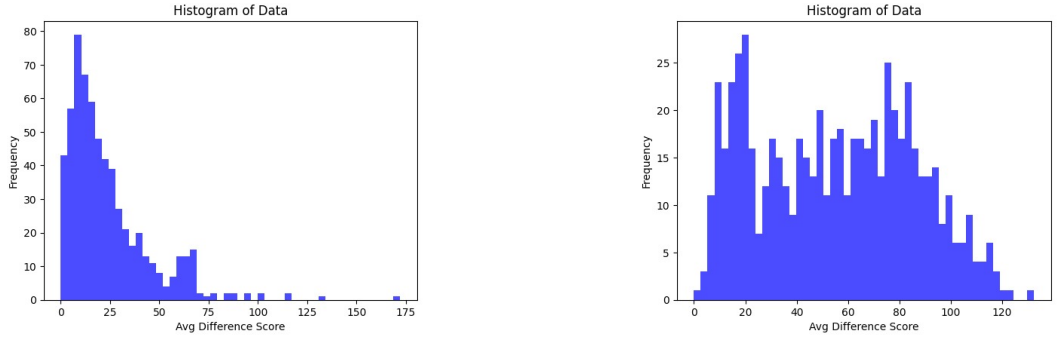
(b) Video 2 BrightnessPlot

Figure 2: Dataset Properties

Inference from the graph:
We are making temporal plot using brightness as a feature for every 23rd frame to get a overview of whole video.

5.0.3 Shot Boundary Detection

Using Shot Boundary detection,we determined shots, and then found keyframes based on a variation of frames and the complexity of the video. We also tried to use different plots of different shots.



(a) Video 1 Plot

(b) Video 2 Plot

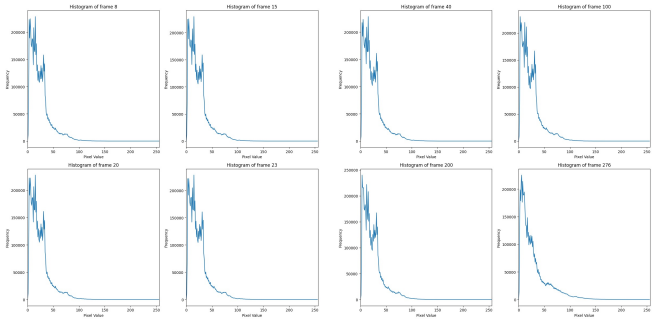
Figure 3: Average difference score vs frequency plot

The graph for video 1 shows that the number of scene changes is less or each scene runs for a longer time .So we have more number of redundant frames. The graph for video 2 shows that the number of scene changes are quite high or each scene runs for a shorter time .So we have less number of redundant frames .

Explanation for Fig4 and Fig6:
In the above result,we are changing threshold due to which number of shots is changing.In the graphs we are verifying whether redundant frames are captured in a particular shot or not.

Time(sec)	Avg_diff_score	original_frame	start_point	end_point
1	1.63	23	0	0
12	39.16	276	1	11
18	39.74	414	12	17
27	34.51	621	18	26
44	59.41	1012	27	43
47	63.88	1081	44	46
49	65.59	1127	47	48
52	49.91	1196	49	51
118	87.44	2714	52	117
120	95.69	2760	118	119
132	100.45	3036	120	131
223	133.39	5129	132	222
226	116.61	5198	223	225
230	59.96	5290	226	229
335	103.25	7705	230	334
346	172.24	7958	335	345
456	66.41	10488	346	455
464	61.13	10672	456	463
466	61.56	10718	464	465
468	62.60	10764	466	467
470	34.13	10810	468	469
618	33.28	14214	470	618

(a) Detection of shots

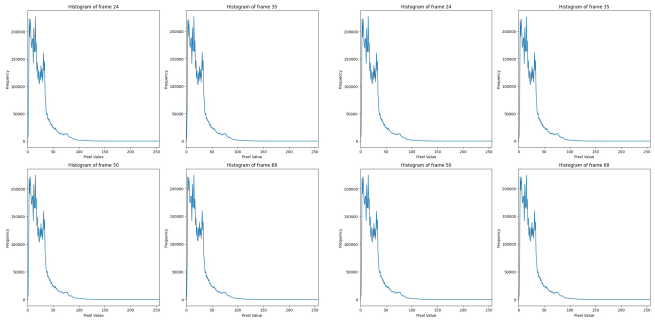


(b) Verification of above table using histogram plot

(c) Verification of above table using histogram plot

Time(sec)	Avg_diff_score	original_frame	start_point	end_point
1	30.90	23	0	0
5	74.65	115	1	4
9	82.03	207	5	8
15	77.88	345	9	14
21	80.58	483	15	20
28	75.39	644	21	27
32	67.39	736	28	31
44	89.86	1012	32	43
50	116.50	1150	44	49
177	99.00	4071	50	176
179	85.21	4117	177	178
183	74.83	4209	179	182
187	76.37	4301	183	186
192	76.43	4416	187	191
195	75.52	4485	192	194
197	87.36	4531	195	196
206	93.52	4738	197	205
208	75.95	4784	206	207
230	77.30	5290	208	229
236	76.81	5428	230	235
241	81.32	5543	236	240
273	78.78	6279	241	272
275	87.97	6325	273	274
277	79.27	6371	275	276

(d) Detection of shots



(e) Verification of above table using histogram plot

(f) Verification of above table using histogram plot

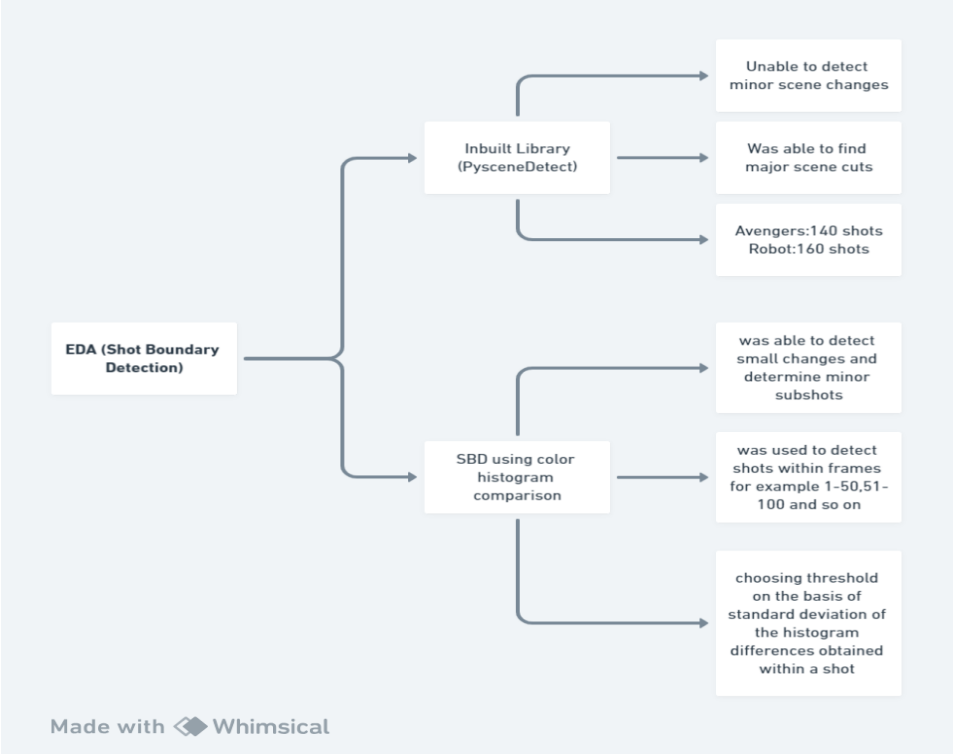
Figure 4: a,b,c: Graphs for window size: 2, threshold: 3 for video 1
d,e,f: Graphs for window size: 2, threshold: 1.35 for video 1

SHOT_INDEX
1
13
114
140
142
204
206
255
258
409
411
604
606
613

SHOT_INDEX
1
42
45
108
110
192
194
335
337
474
476
543
545
639

(a) Video Properties

Figure 5: Shot Boundary indexes for Avenger and Robot respectively



(a) EDA

Figure 6: EDA finally applied

6 Methodology

We have used the following techniques:

6.1 Video Compression

Using the K-means clustering algorithm, we first applied color quantization to begin the process of image compression. The process of color quantization lowers the number of colors needed to depict an image. In this instance, we chose K=32, which means that we reduced the color palette by using 32 representative colors to represent our image.
We used shot boundary detection to identify the first frame of each shot, which we then applied K-means clustering to compress our image. The end product was a compressed image, and every video shot's first frame was compressed using the same procedure. After compression, these frames were saved as metadata.
The generated metadata acts as a storehouse for the frames that have shot transitions. By preserving these frames, we guarantee the availability of vital information required to reconstruct the video. To help with the effective organization and retrieval of shot-related data, the metadata also includes a CSV file with the shot boundary indices.
To summarise, our methodology entails the deliberate compression of frames linked to shot transitions, with the preserved compressed representations retained in the metadata. This novel approach uses the compressed frames contained in the metadata to optimize storage while allowing for a smooth reconstruction of the video.

6.2 Video Decompression

We have the metadata where all the compressed frame and shot indexes are present. We will recreate the video from the frames and shot indexes. We have the indexes where shots are changing say

1,9,47 etc and i have the compressed frame corresponding to 1,9,47 indexes.For creating video we need all the frames so we will run the compressed frame 1 till 8 , then we will run the compressed frame at indexe 9 till 47.We are doing this because within a shot we are getting similar frames. The compressed file is not playable as it contains all the frames.We have written an algorithm so that the compressed file can be converted back to playable video during the time of decompression.

6.3 Audio Compression

Principal Component Analysis, or PCA, was used for the audio compression stage.Reducing the size of the audio data while maintaining its essential qualities is the main goal of audio compression.

6.3.1 Libraries used

numpy,sklearn. decomposition.PCA,Librosa,soundfile

6.3.2 Implementation

Audio Data Loading:the audio data was extracted from the video file in the mp3 format.
Data Reshaping: converted to a 2D array .
Principal Component Analysis (PCA):PCA is a dimensionality reduction technique widely used in signal processing and data analysis.
Saving Compressed Representation:The resulting compressed representation was saved to a file in npy format.

6.4 Audio Decompression

Reconstructing Audio Data:The compressed audio representation obtained from the compression phase is used to reconstruct the audio data using PCA inverse transformation only.
Flattening the Data:The reconstructed audio data is initially in a 2D array. The data is flattened using the method to obtain the final audio signal.
Writing to File: The reconstructed audio data is then written to a new audio file using the SF. write function with the original sampling rate .

6.5 Merging Video and Audio

Load Clips:Load video and audio clips from specified paths. Combine Audio with Video:Set the audio from the loaded audio clip to the video clip.
Write Merged Video:Write the merged video with combined audio to a new file. Use specified codecs for video and audio.
Execution:Main block specifies input video, reconstructed audio, and output paths.Call the function to perform the audio-video margin
In summary, this methodology uses MoviePy to load video and audio clips, combines the audio with the video, and writes the resulting merged video to a new file, ensuring proper codec specifications for both video and audio components.

7 Results

DATASET	ORGINAL VIDEO SIZE	COMPRESSED VIDEO SIZE	DECOMPRESSED VIDEO SIZE
VIDEO 1	167 MB	101 MB	381 MB
VIDEO 2	259 MB	61.6 MB	315 MB
AUDIO 1	9.16 MB	132 B	4.54 MB
AUDIO 2	9.15 MB	132 B	5.31 MB

(a) Table for data

Figure 7: changes in size

The metadata generated contains compressed frames,compressed audio and csv file containing shot indexes.

We were able to compress video first by storing its shots and metadata and then compressed the audio to npy file .
Further decompressed the video by using the shots and meta data.And we also decompressed the audio by using npy file to get audio.
Finally we were able to get the decompressed video by adding video and audio together .

8 Inferences

- Utilized shot boundary detection for creating shots and representing the entire video with a reduced number of frames.
- Applied k-Means for color quantization, where 'k' defines the size of compressed frames.
  - Increasing 'k' enhances image quality but also enlarges the compressed frame, leading to a trade-off between image quality and size.
- Understood the basic working of a video codec, comprising an encoder and decoder for video compression and decompression.
- Determined the size of metadata by choosing the number of clusters and threshold in color quantization.
- The threshold chosen for shot boundary detection directly influences the number of shots identified in the shot boundary detection process.
- Acquired knowledge about different compression types, including lossy and lossless compression.

9 Existing Analysis

We were unable to find any algorithm-related to unsupervised learning for video compression.But the same process is being done by deep learning and neural network. Neural networks learn complex patterns for more efficient compression, achieving higher ratios while maintaining quality.
Neural networks adapt to diverse content, offering versatility beyond traditional algorithms.
Neural network-based compression improves video quality at lower bitrates, crucial for streaming and limited bandwidth.
Learning from Data on vast video datasets, neural networks discover efficient compression strategies.

10 Citations

Research Paper 1:Adaptive Key Frame Extraction using Unsupervised Learning
Research Paper2:Shot Boundary Detection: Fundamentals Concepts and Survey