

---

# Using Unsupervised Learning Techniques to perform video data compression.

---

**Aditi**  
B.Tech  
CSD

aditi20354@iiitd.ac.in

**Naman Sharma**  
B.Tech  
CSAM

naman21266@iiitd.ac.in

**Arunoday Ghorai**  
M.Tech  
CSE

arunoday23023@iiitd.ac.in

**Nyasa Panwar**  
BTech  
CSE

nyasa21076@iiitd.ac.in

**MO RASHID**

MTech  
CSE/Gen

rashid23047@iiitd.ac.in

## Abstract

In this study, we analyze a data set comprising two 10-minute video clips. The focus of our investigation is on video compression techniques, widely used to reduce file sizes while maintaining quality. We will be taking the combination of algorithm like average pooling, max pooling and vector quantization. We will be taking reference from some pre-define codec.

## 1 Introduction

The rapid development of digital multimedia has transformed the way we share information with each other. Videos have become the most dominant and useful way of conveying messages, but they arrive at substantial costs of storage and transmission. High-resolution videos with rich audio consume vast amounts of storage space and bandwidth, often making their use expensive and inefficient. Compressing the size of the video helps in keeping the balance between high-quality visual and auditory content while significantly reducing file sizes.

## 2 Existing Analysis

1. There are few existing popular techniques able to compress video while retaining quality. Video compression algorithms look for spatial and temporal redundancy which helps reduce file size by identifying duplicate data and encoding it fewer times. Video encoding Schemes use 3 concepts:-
  - (a) I-frames: Fully encoded images. [1]
  - (b) P-frames: predicted frames based on the image changes in the last I-frame.[1]
  - (c) B-frames: Predicted in both directions. Using data from the last P frames and the next I frame. [1]

The problem with this technique is that inter-frame predictive coding results in unsightly artifacts in video frames. Parts of the image might move incorrectly due to imperfect encoding. [1]

2. The image Interpolation technique of Motion Information has been identified in the past research :

- (a) Motion information here (ground truth motion information) describes how the pixels move over the period of time, which is the information needed to help generate interpolated frames. Optical flow and block estimation are two types. Optical flow gives finer details, while the block motion is easier to compress.
- (b) Now, using this motion information, the feature maps mentioned before are adjusted with respect to  $t(i)$  here,  $t(i)$  is the motion estimate at location  $i$ .
- (c) The benefit of doing this is that the decoder will just focus on creating the image and not have to think about motion estimation.[1]

3. Residual Motion Compensated Interpolation:

- (a) This model combines motion-compensated interpolation with the compressed representation of the left information. This includes motion as well as the appearance differences in these interpolated frames.
- (b) There is a progressive compression scheme that the encoder and the decoder follow that is similar to what has been described as the I-frame and P-frame approach.
- (c) This leads to bitrate encoding, which means that it uses fewer bits for close frames and more for distant frames. [1]

4. Another popular way of video data compression is image resizing by adjusting or reducing the resolution of the image. This is because in a higher video resolution, more information each frame contains.

5. In the Unified Framework of Frame Skipping and Interpolation (UF-FSI) for efficient video compression, the key idea is to selectively skip less important frames while adding interpolated frames to maintain video smoothness. This approach works in conjunction with existing video compression methods and results in a 25% reduction in video size compared to MPEG-4. [2]

6. Video compression using decision trees: The images are divided into small blocks. We have to optimize the size as well as the number of these blocks.[3]

- 1. Decision trees are supervised learning algorithms and are easier to understand as compared to more complex deep learning approaches.
- 2. According to the article, the Integration of a decision tree-based approach has led to a 40% speed up in the encoding part with minimum impact on the video quality. [3]
- 3. As decision trees are supervised learning algorithms, they are given data points with labels. There are conditions present according to which it gives binary output, and a binary tree is formed. [3]

### 3 Data Set

In the Dataset provided, we were given two video clips, "Avengers Endgame" and "Alita: Battle Angel". The Clips are 10 minute long each mkv files each and have sizes of 167 MB and 259 MB respectively. We convert mkv files to mp4 files.

## 4 EDA Analysis

1. For the dataset given to us, we had 2 Video Clips. 1st was Avengers Endgame, and the 2nd was Alita Battle Angel. Following are the inferences we have drawn from the dataset:
2. First, we extracted the basic information about the Avengers Endgame and Alita Battle Angel, like the number of frames fps, and then for each frame, we extracted the features like entropy, color distribution, pixel distribution, Luminosity, Contrast, Energy, and Correlation of Pixels in each frame Following are the graphs that we have obtained :

**Note: We did Max polling for first 1000 frames only.**

```
cp1 = cv2.VideoCapture('/kaggle/input/video-compression/AvengersEndgme.mp4')
cp1.get(cv2.CAP_PROP_FRAME_COUNT)
```

[46]: 14391.0

```
# Getting height and width for resolution and frame rate
print(cp1.get(cv2.CAP_PROP_FRAME_HEIGHT))
print(cp1.get(cv2.CAP_PROP_FRAME_WIDTH))
print(cp1.get(cv2.CAP_PROP_FPS))
```

1608.0  
3840.0  
23.976023976023978

(a) Avengers Endgame

```
cp1 = cv2.VideoCapture('/kaggle/input/video-compression/AlitaBattleAngel.mp4')
cp1.get(cv2.CAP_PROP_FRAME_COUNT)
```

[4]: 14291.0

```
print(cp1.get(cv2.CAP_PROP_FRAME_HEIGHT))
print(cp1.get(cv2.CAP_PROP_FRAME_WIDTH))
print(cp1.get(cv2.CAP_PROP_FPS))
```

800.0  
1920.0  
23.80952380952381

(b) Alita Battle Angel

Figure 1: Basic Details of videos

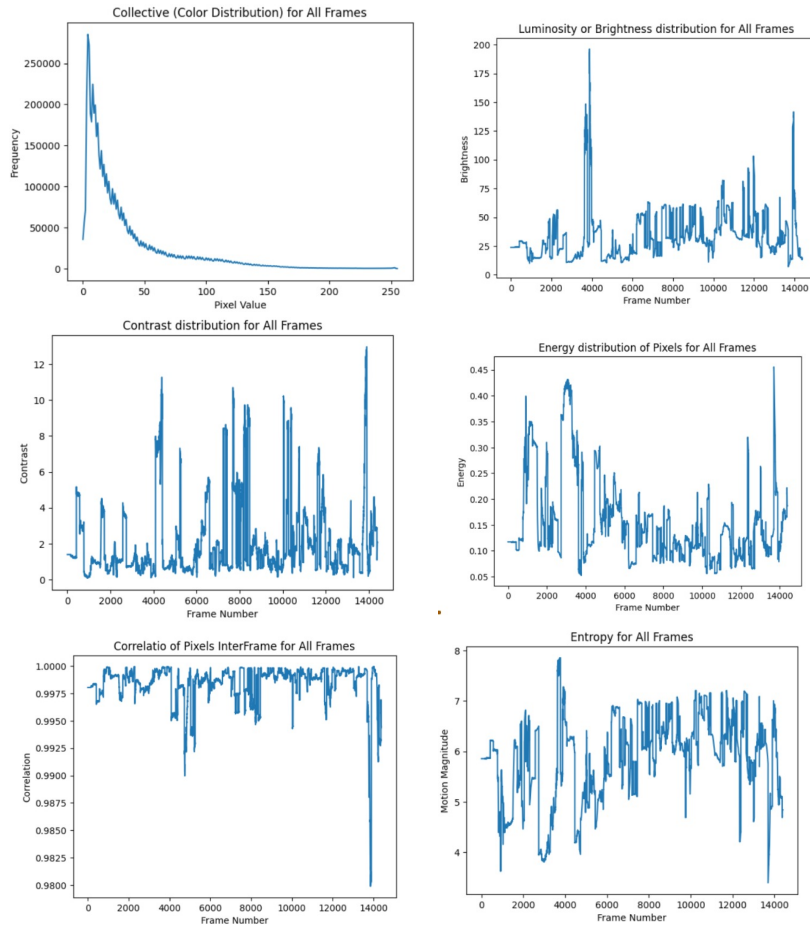


Figure 2: Avengers Endgame Plots

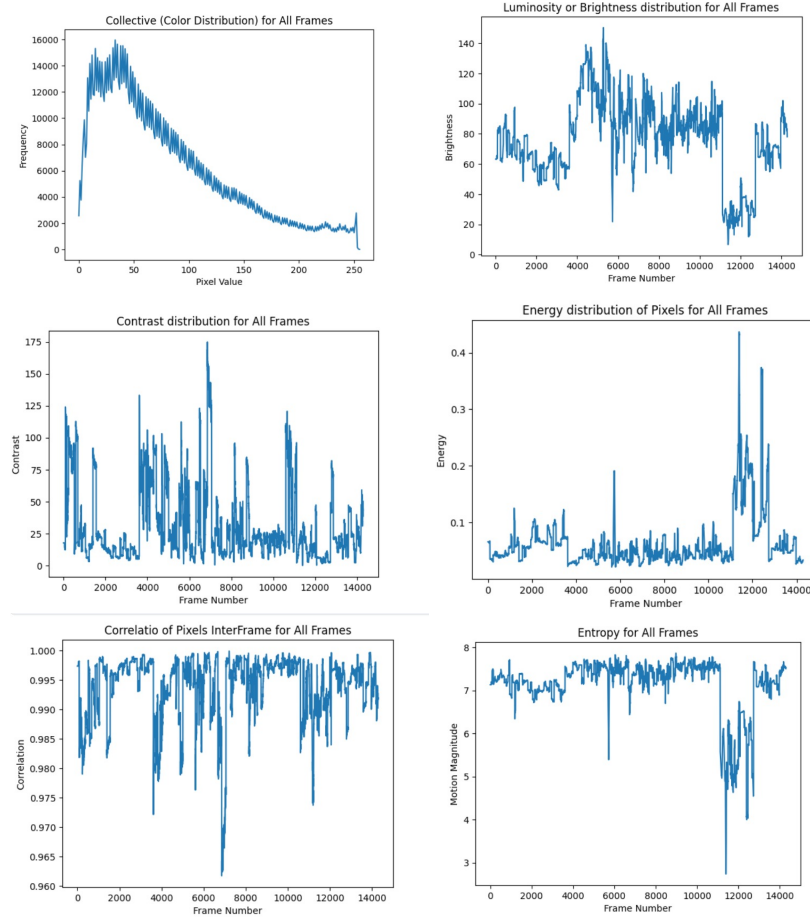


Figure 3: Alita Battle Angel Plots

### 3. Inferences Drawn from the Above Features (Avengers Endgame):

- We can see that the major part of the video has a dull brightness level as for the first 5 minutes, the video clip has a space scene, and further, after 5 minutes, the clip is composed of a single dull background. This is evident from the Luminosity curve that around 4000 frames, we have a huge peak in the curve. This is because of the entry of the spacecraft scene, and thus, for video compression, this is an important feature to check if compression is correct or not.
- From the entropy curve, it is clear that the pixel distribution is very random for each frame. Thus, entropy will not significantly validate our video's compression.
- Correlation of pixels as shown in the curve tells us that it is an important feature as around 14000 when the spacecraft scene appears again, so we have that the pixels become even less correlated due to sudden change in pixels per second, hence an important feature to validate the compression.

### 4. A similar analysis was also conducted for the 2nd clip that was for Alita Battle Angel and we got the following graphs: Conclusions drawn:

- First of all, it is a fight scene, and thus, contrast keeps on increasing, and so it is a very important factor for the validation of video compression and decompression. The color composition of pixels also contributes too much here because in fight scenes, pixel's color changes rapidly.
- Correlation is another important factor as it highlights that around the 7000 frames, there are maximum uncorrelated pixels, as that is when the fight scene ends around this time. Thus, it is again a very important feature for the validation of compression-

decompression. The fight scene starts around the 4th minute and ends at the 7th minute.

- (c) Energy and Entropy are higher in the later frames around the 10,000-14,000 frames. This is because of the Scene, and hence, they also capture very important characteristics of our video and so can validate the compression-decompression.

5. Now we will be doing max-pooling and studying the effect of max-pooling on the both video clips : Following are the graphs that we get for the 10\*10 block size chosen (also, this study was for the first 1000 frames of the video clip. For all the frames, we can do a simple modification to the code.)

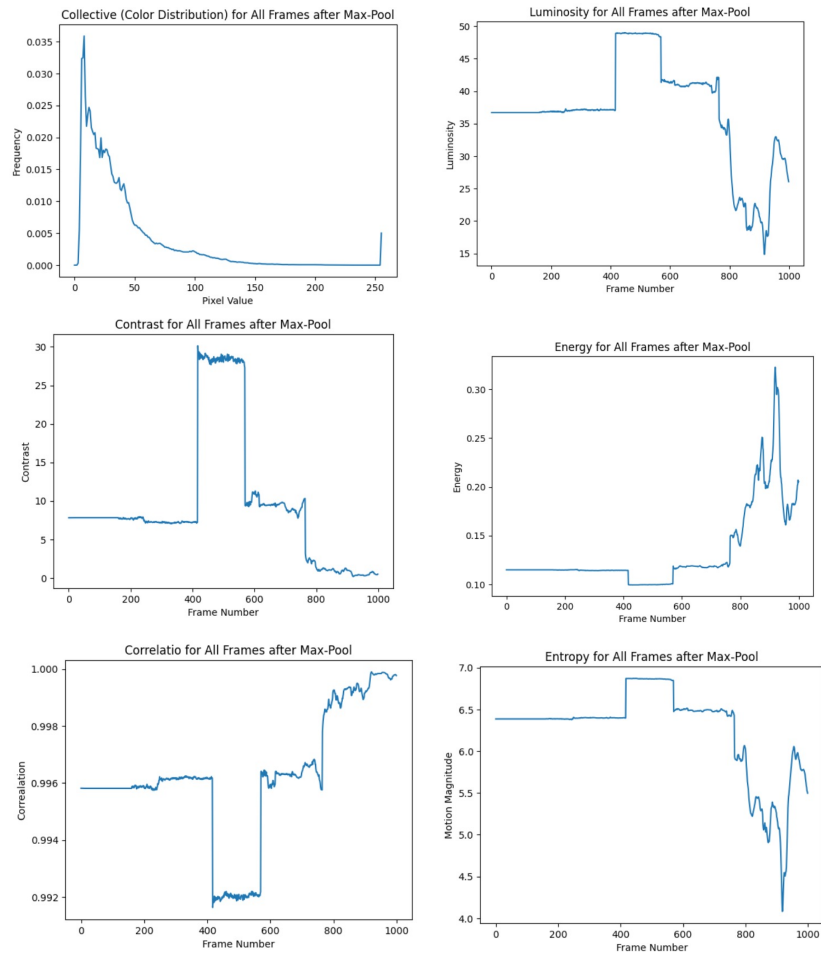


Figure 4: Avengers Endgames After Max Pooling

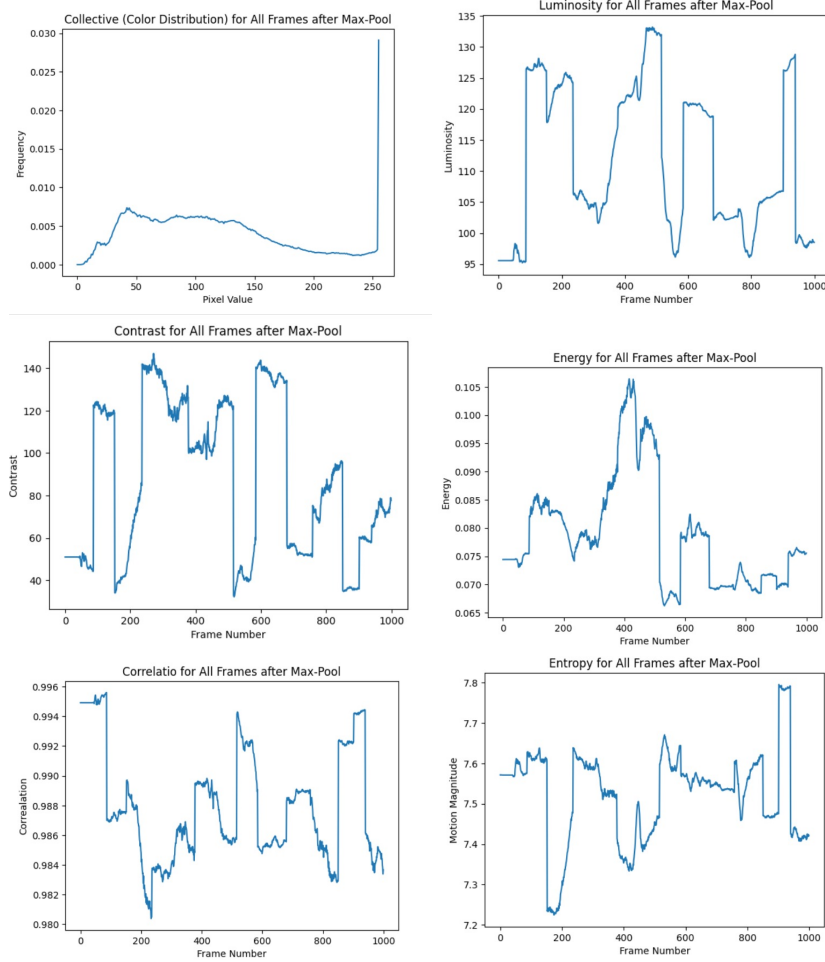


Figure 5: Alita Battle Angel After Max Pooling

- (a) Clearly, from the graphs, we have that entropy is a useless feature for the validation of compression-decompression, but correlation and contrast, energy of pixels in each frame, is a very important feature because it highlights that max-pooling is inefficient for the video compression and decompression
- (b) The observations were also taken by increasing the block size to  $20 \times 20$ , and this further showed us that, indeed, our choice of features is satisfied by Color Distribution, correlation, and contrast, the energy of pixels in each frame.

We have done an analysis for the average pooling as well, and for that, the result is even much better for feature selection for the validation of compression-decompression.

6. For the Audio part of both Video Clips, we are considering 2 features: the RMS Energy (Root Mean Square) and Spectral Centroids for each frame. Now, both for max-looping and average looping, we have that these features can validate the video compression-decompression, and their plots verify the same fact. Although graphs have not been added, it has been verified that under max looping, both RMS Energy and Spectral Centroids behave very differently and thus indicate that Max looping is inefficient. Hence, we have that these 2 features are important to validate the audio after decompression.

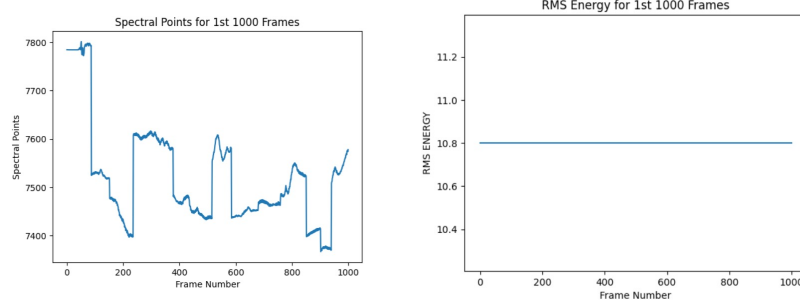


Figure 6: Alita Battle Angel plots

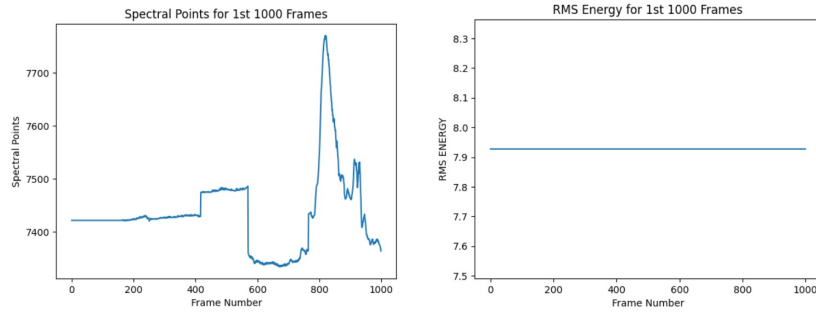


Figure 7: Avengers Endgame plots

## 5 Methodology

### Modified Vector Quantisation Algorithm

1. For image compression using vector quantization and co-  
<https://www.overleaf.com/project/6538af25e4e7f1ba7996c27clor> preservation, we have to deviate from the standard vector quantization done on 2-d image data traditionally on GrayScale Images, where 2 coordinates represent each image.
2. We take advantage of the YCbCr Scale. Y is used for capturing the Intensity or brightness, and Cb and Cr channels take care of chroma features or the color distribution. Thus, to preserve the luminosity and color contrast, our most important features during EDA, we have converted from RGB scale to YCbCr scale.
3. Standard Vector Quantization is now used to divide each image into nonoverlapping patches or boxes. We convert each patch into a vector called the code vector.
4. This code vector then goes into the encoding phase. In the encoding phase, each code vector is compared to the other code vectors using MSE (Mean Squared Error or the Weighted Mean Squared Error) to get to the closest match. After getting the closest code vector, we store the index of the code vector in the file for decompression. In the decode function, we look at the codebook and find the nearest code vector, replace the block with the values of the weights, and the codebook is read from the metadata stored from compression.

## 6 Model

### Modified Vector Quantisation Algorithm

The following model was proposed for deriving the codebook and compressing the file:

1. First, we obtain the training vector by splitting each block into the vector, and then we obtain all the training data using a similar sense



2. Calculate the initial distortion and the average of all vectors to be utilized as the initialization step. Similar to the K-Means Algorithm, we initialize the centroids randomly and then come to the updation and assignment steps to guide the flow of the algorithm.

$$\mathbf{c}_1^* = \frac{1}{M} \sum_{m=1}^M \mathbf{x}_m.$$

$$D_{ave}^* = \frac{1}{Mk} \sum_{m=1}^M \|\mathbf{x}_m - \mathbf{c}_1^*\|^2.$$

Where  $c_1$  is the Average and  $D(ave)$  is the Initial distortion.

3. Next, we split each codevector as follows:

$$\begin{aligned} \mathbf{c}_i^{(0)} &= (1 + \epsilon) \mathbf{c}_i^*, \\ \mathbf{c}_{N+i}^{(0)} &= (1 - \epsilon) \mathbf{c}_i^*. \end{aligned}$$

4. Algorithm:  
Do while error > epsilon (Acceptable Error):

- (a) Find the minimum value

$$\|\mathbf{x}_m - \mathbf{c}_n^{(i)}\|^2,$$

- (b) Update the code vector

$$\mathbf{c}_n^{(i+1)} = \frac{\sum_{Q(\mathbf{x}_m)=\mathbf{c}_n^{(i)}} \mathbf{x}_m}{\sum_{Q(\mathbf{x}_m)=\mathbf{c}_n^{(i)}} 1}$$

- (c) Calculate average distortion

$$D_{ave}^{(i)} = \frac{1}{Mk} \sum_{m=1}^M \|\mathbf{x}_m - Q(\mathbf{x}_m)\|^2.$$

$$(D_{ave}^{(i-1)} - D_{ave}^{(i)}) / D_{ave}^{(i-1)} > \epsilon$$

- (d) Set current average distortion for the next step

$$D_{ave}^* = D_{ave}^{(i)}$$

5. If the code vector exactly resembles an already saved code vector, it is not saved, saving a lot of space. In particular, if MSE is less than epsilon, it is not saved in the codebook.
6. Resemblance with the K-Mean Algorithm Update and Assignment Step:

**K-means (D, k,  $\epsilon$ ):**

```

1  $t = 0$ 
2 Randomly initialize  $k$  centroids:  $\mu_1^t, \mu_2^t, \dots, \mu_k^t \in \mathbb{R}^d$ 
3 repeat
4    $t \leftarrow t + 1$ 
5    $C_j \leftarrow \emptyset$  for all  $j = 1, \dots, k$ 
   // Cluster Assignment Step
6   foreach  $\mathbf{x}_j \in D$  do
7      $j^* \leftarrow \arg \min_i \{\|\mathbf{x}_j - \mu_i^t\|^2\}$  // Assign  $\mathbf{x}_j$  to closest
       centroid
8      $C_{j^*} \leftarrow C_{j^*} \cup \{\mathbf{x}_j\}$ 
   // Centroid Update Step
9   foreach  $i = 1$  to  $k$  do
10     $\mu_i^t \leftarrow \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j$ 
11 until  $\sum_{i=1}^k \|\mu_i^t - \mu_i^{t-1}\|^2 \leq \epsilon$ 
```

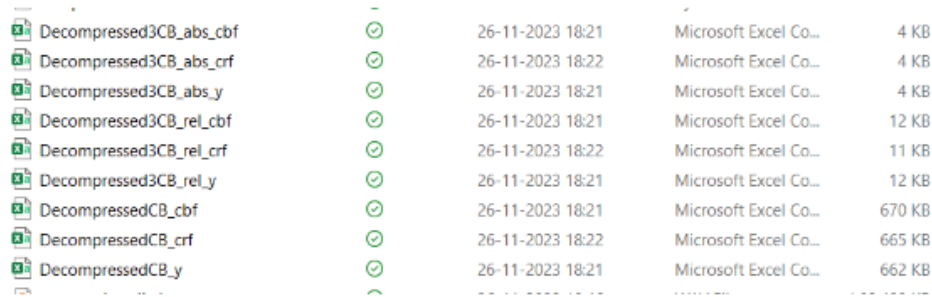
## 7 Analysis

The following observations were recorded:

1. As the codebook's size increases, the image's quality increases as the longer the code vector, the more spatial information about pixels and their intensity and color can be captured. Still, the corresponding run time is also higher.
2. As Epsilon(E) decreases, the error term has to be decreased, thus improving the image quality. Again, the time to convergence is longer here.
3. Decrease in block size from (32,32) to (2,2) increases the per pixel information that can be captured, so increased Sound To Noise Ratio but increased run times as well!
4. Using the Codebook for different images that are simultaneous in the video is helpful for an increase in variation in training data and, thus, better prediction and increased TSNR. This can be done.

## 8 Result

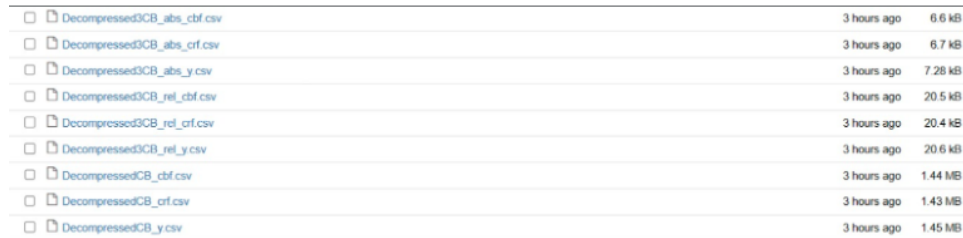
1. For Avengers Endgame, we have the following compressed files:



Decompressed3CB_abs_cbf	✓	26-11-2023 18:21	Microsoft Excel Co...	4 KB
Decompressed3CB_abs_crf	✓	26-11-2023 18:22	Microsoft Excel Co...	4 KB
Decompressed3CB_abs_y	✓	26-11-2023 18:21	Microsoft Excel Co...	4 KB
Decompressed3CB_rel_cbf	✓	26-11-2023 18:21	Microsoft Excel Co...	12 KB
Decompressed3CB_rel_crf	✓	26-11-2023 18:22	Microsoft Excel Co...	11 KB
Decompressed3CB_rel_y	✓	26-11-2023 18:21	Microsoft Excel Co...	12 KB
DecompressedCB_cbf	✓	26-11-2023 18:21	Microsoft Excel Co...	670 KB
DecompressedCB_crf	✓	26-11-2023 18:22	Microsoft Excel Co...	665 KB
DecompressedCB_y	✓	26-11-2023 18:21	Microsoft Excel Co...	662 KB

The initial video of size 270Mb has a compressed state of size 2.044 Mb, which is roughly 0.75% of the entire Size. Also, we have to add the decompressed images, roughly 140Mb; thus, overall compression is roughly 50%

2. For Alita Battle Angel, we have the following compressed files:

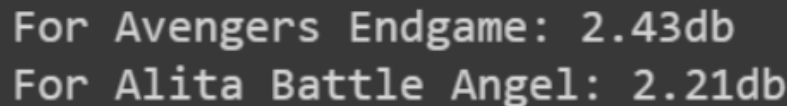


Decompressed3CB_abs_cbf.csv	3 hours ago	6.6 kB
Decompressed3CB_abs_crf.csv	3 hours ago	6.7 kB
Decompressed3CB_abs_y.csv	3 hours ago	7.28 kB
Decompressed3CB_rel_cbf.csv	3 hours ago	20.5 kB
Decompressed3CB_rel_crf.csv	3 hours ago	20.4 kB
Decompressed3CB_rel_y.csv	3 hours ago	20.6 kB
DecompressedCB_cbf.csv	3 hours ago	1.44 MB
DecompressedCB_crf.csv	3 hours ago	1.43 MB
DecompressedCB_y.csv	3 hours ago	1.45 MB

The initial video of size 220Mb has a compressed state of size 4 Mb due to intensive fight scenes, so it has a longer codebook and weight, which is roughly 2% of the entire Size. Also, we have to add the decompressed images, roughly 140Mb; thus, overall compression is roughly 50%

## 9 Conclusion

1. We determined the average value of TSNR for the 2 videos by extracting the frames and then conversion to gray frames, and these are the values we got:



For Avengers Endgame: 2.43db  
For Alita Battle Angel: 2.21db

2. Now, the values are low because of the relatively higher blocks and bigger epsilon we have taken to save time, but with effective GPU utilization, we can certainly reach 10db.

## References

- [1] Wu, Chao-Yuan, Nayan Singhal, and Philipp Krahenbuhl. "Video compression through image interpolation." *Proceedings of the European conference on computer vision (ECCV)*. 2018.
- [2] S. Lee, M. Lee, K. Choi and E. S. Jang, "Unified framework of frame skipping and interpolation for efficient video compression," *2009 IEEE International Conference on Network Infrastructure and Digital Content*, Beijing, China, 2009, pp. 670-674, doi: 10.1109/ICNIDC.2009.5360919.
- [3] <https://www.bbc.co.uk/rd/blog/2019-10-video-compression-machine-learning>