Abstract

A video is the most efficient way of representing various types of information and the video traffic on internet along with the size of these videos is increasing day by day. The primary focus of this project is to optimize compression of video by extracting features from it and compressing it using the x265 encoder with the set of parameters obtained from the machine learning model, in accordance to the extracted features. Set of features from a video are obtained by a tool which processes the video considering object motion, picture clarity, level of detail, camera jitter, dynamic background and generate a set of features. Based on a set of videos, a custom dataset is prepared for training the model, comprising of encoded videos under various parameters used in x265 encoder that optimizes the trade-off between encoding speed and compression efficiency. This dataset will include the set of parameters used for encoding, size of the output video, time required for compression, metrics used to identify quality loss after compression. Hence by the end of this project, a Machine Learning model will be prepared, that will provide the precise set of parameters to be used by x265 encoder for efficient compression of any given video.

Keywords- Video Compression - Machine Learning - Video encoding

Index

List of Figures

List of Tables

List of Acronyms

AVI Audio Video Interleaved.

GMM Gaussian Mixture Model.

HD High Definition.

MKV Matroska Video.

PCC Pearson Correlation Coefficient.

PSNR Peak Signal to Noise Ratio.

SSIM Structural Similarity Index.

UHD Ultra High Definition.

VIFP Visual Information Fidelity in Pixel Domain.

VQMT Video Quality Measurement Tool.

Video Compression Optimization

December 2018

Chapter 1 Introduction

In today's world of fast-growing technology, Video is becoming the primary form of media among all sorts of users, accounting to the amount of knowledge that a user gets easily from a video, as compared to that from audio, photo or textual media. A report by Forbes provides some interesting figures about the exploding growth of the Video market [1]. Video traffic on the internet will account for 80% of the whole internet traffic worldwide by 2019. YouTube generates a 2 fold increase in mobile video consumption every year. Facebook registers 8 billion views on videos per day approximately.

With that enormous increase in data flow of video, the subsequent costs that follow are the storage of millions of videos along with the highest retainable quality. This creates a bottleneck for the growth of video media industries, as they look towards 'Video Compression' for reducing the storage size while making compromises with video quality, that may lead to customer dissatisfaction, as the users often wish to view their videos in the highest possible quality. Here's where video compression optimization comes into the picture, as the project exploits various compression tools and algorithms in order to achieve maximum compression while minimizing quality loss and Information loss in videos.

1.1 Applications

Video sharing platforms like YouTube, Facebook, Instagram process and store millions of videos every day. This requires exabytes and petabytes of storage on servers [2]. On a similar ground, Online Streaming Services like Netflix, Amazon Prime, Hotstar aim to provide best video quality to users for streaming. But the download rates on user's network fails to cope with the bitrate of higher quality videos because of slow internet speed. In such a situation users have no other option but to watch videos in low quality for uninterrupted streaming.

Video Compression Optimization can help to reduce the size of videos while maintaining maximum quality. This leads to a number of benefits for storage as well as streaming. Even the slightest of size reductions on each video online can save exabytes of storage. Whereas a compressed video having a subsequently lower bitrate can help to improve user experience on

the same network speed available.

1.2 Motivation

The amount of data flow on the internet is growing exponentially every year. This creates a demand for exponential increase in storage space. Adding to it the fact that rapid developments in the field of video recording equipment are leading to extremely high qualities, but large-in-size videos. Therefore, the overall result increases the costs of buying and maintaining storage space.

Streaming such high-quality videos becomes an even bigger havoc, given the condition of network speed in most of the countries. User has to wait for high-quality videos to get buffered on slow internet speed, creating discomfort. Or else they have to watch videos on low quality to keep the video uninterrupted from buffering while ruining the user experience.

The need of the hour is to realize that growing video data and improving quality are creating a strain on the current architecture and must be tackled in the best ways by employing video compression tools to the fullest.

1.3 Objectives

There are a number of video compression tools and algorithms available that can compress videos, resulting in appreciable loss in size without significant loss in visual information fidelity. This is achieved by varying the parameters used in compression algorithm. What makes this process challenging is identifying the ideal values for all the parameters in order to achieve the best possible results for a specific video. The optimal values of parameters vary from video to video depending on the features of the video. A good level of expertise and understanding of the the tool is required in order to identify these parameter values.

The objective of this project is to build a Supervised Machine Learning model based on the training data of successful compressions from a tool. The input for the model will be any video that has to be compressed and the output will be the set of parameter values to achieve maximum possible size reduction and minimal quality loss in video.

The video datasets available publicly for working are uncategorised. Thus the secondary objective of the project is video categorisation, based on movement, content, noise and other features of significance, so that the resulting labelled dataset can be used for training of video compression model.

1.4 Organization of Project Report

Chapter-1 **Introduction**, provides the essence of the entire project, the need for Video Compression Optimization, it's applications, objectives and motivation. Chapter-2 **Literature Survey**, comprises the survey from research papers about various Codecs based Analysis, Container based Analysis and existing Video Compression approaches using Machine Learning. Chapter-3 **Proposed Work**, contains insights about parameter prediction using Machine Learning and proposed framework of project. Chapter-4 **Summary and Future Work**, provides information about the progress till date and the goals to be achieved in future.

Chapter 2 Literature Survey

With the advancement of Video related technologies, lots of video compression techniques and video storage formats have become available. The primary idea that one must understand is that video compression results in loss of data from original video. So there will be either significant or negligible difference between the output and the input videos, depending on the compression method used. The main goal is to achieve maximum possible size reduction with the minimal quality loss in video.

A digital video file consists of two parts, a "codec" and a "container". The codec is used in compressing and decompressing video. At present, the widely used codecs are H.264, H.265, VP9 [3]. On the other hand, a container is a collection of files that hold information about both audio and video data. MKV, AVI and MP4 are some of the popular container types.

2.1 Codecs based analysis

2.1.1 AVC H.264

H.264 is a block-oriented, open source motion-compensation-based standard for video compression. H.264 standard compares different part of video frames and finds the areas that are redundant within the subsequent frames. Later on, these areas are replaced by shorter information. These areas are of maximum block-size of 16x16 pixels. It is a lossy compression method and has the ability to lower the bit rates compared to previous standards such as H.263, MPEG-4 and Divx. It is mostly used when fast encoding is desired.

2.1.2 High Quality Video Coding (HEVC) or H.265

H.265 is a royalty-encumbered successor of H.264 standard. It works same as H.264 but instead of blocks, H.265 uses coding tree units (CTU). CTU sizes vary from 4x4 to 64x64 pixels. So improved CTU segmentation along with better spatial prediction and motion compensation results in way better bitrate reduction than the H.264. Because of this reason hardware specifi-

cations of HEVC is high to compress the data. This allows the users having compatible devices with required processing power, view 4K videos even on low bandwidth and average network speed. This compression method drops the bandwidth and storage requirement by roughly 50% [4].

2.1.3 VP9

VP9 is an open source coding format developed by Google working principles of VP9 and H.265 are same. It is also almost similar to H.265 in terms of encoding quality but it has a significant advantage when it comes to encoding and decoding speed [5]. Reason for this is HEVC is much more complex than the VP9. So VP9 can be used with relatively lesser hardware specifications then H.265. VP9 is widely used in YouTube [6].

2.1.4 AV1

AV1 is an open source video coding format. Unlike other codecs, AV1 focuses on the encoding quality rather than the speed. So the encoding time for AV1 is much higher than that for VP9 and HEVC but it is a perfect compression standard when the time is not of concern and highest quality possible is desired [7]. It is used in the transmission of videos over the internet.

The output of video compression using various codecs like HEVC, VP9 or AV1 results in various qualities and sizes for a specific video. According to a report based on High Definition (HD) and Ultra High Definition (UHD) videos, HEVC and VP9 are compared for video quality difference using PSNR [8]. It is concluded that HEVC implementation produces higher PSNR values as compared to that using VP9 for the same targeted bitrates, under similar encoding parameters.

2.2 Container Based Analysis

1. **AVI(Audio Video Interlaced) -** It is the most universally supported container but faces compression limitations. Because of that, video having AVI containers result in large file size. It can contain video and audio tracks only.
2. **MP4(MPEG-4 Version 2)-**It is widely used with H.264 for video encoding. Subtitle tracks are also supported in this format.
3. **MKV(Matroska Video Container)-** It can hold all kinds of audio and video formats including multiple subtitle tracks, menus and chapters. Thus, it is the most versatile

container.

MP4 is a widely supported container, but MKV is the most popular one because of its flexibility and broad range of features. Containers don't affect the quality of the video, but they limit the encoding method. Therefore, choosing a flexible container such as MKV is the right choice when one wants better compression.

2.3 x265 Encoder Library

To utilize the HEVC/H.265 codec for video, the need for a suitable video encoder application library arises. The most popular x265 is available as an open source library, published under the GPLv2 license and aims to provide the most efficient, highest performance HEVC video encoder.

The various encoding parameters used in x265 are categorized in ten predefined presets that affect the trade-off between compression efficiency and encoding speed. The presets are :

- ultra fast
- super fast
- very fast
- faster
- fast
- medium (default)
- slow
- slower
- very slow
- placebo

These presets include 32 parameters which are varied differently. The most significant parameters are shown in Table 1

preset	ultra fast	super fast	very fast	faster	fast	medium	slow	slower	very slow	slow
ctu	32	32	64	64	64	64	64	64	64	64
bframes	3	3	4	4	4	4	4	8	8	8
b-adapt	0	0	0	0	0	2	2	2	2	2
rc-lookahead	5	10	15	15	15	20	25	30	40	60
ref	1	1	2	2	3	3	4	4	5	5
limit-refs	0	0	3	3	3	3	3	2	1	0
me	dia	hex	hex	hex	hex	hex	star	star	star	star
merange	57	57	57	57	57	57	57	57	57	92
subme	0	1	1	2	2	2	3	3	4	5
rect	0	0	0	0	0	0	1	1	1	1
max-merge	2	2	2	2	2	2	3	3	4	5
early-skip	1	1	1	1	0	0	0	0	0	0
fast-intra	1	1	1	1	1	0	0	0	0	0
rdLevel	2	2	2	2	2	3	4	6	6	6
rdoq-level	0	0	0	0	0	0	2	2	2	2
tu-intra	1	1	1	1	1	1	1	2	3	4
tu-inter	1	1	1	1	1	1	1	2	3	4

Table 2.1: Presets and their Parameter values

2.3.1 Experiment

A 58 seconds long video of size 128 MB, having a bit rate of 17.3 Mb/s, with the total frame count of 1759 and resolution of 1920x1080 was compressed with x265 encoder using above mentioned presets. By using the Video Quality Measurement Tool (VQMT), metrics like PSNR, SSIM and VIFP were generated by comparing original video and compressed video.

PSNR - Peak Signal to Noise Ratio. It is used to compare quality between compressed image and the original image. Higher PSNR suggests that quality loss was minimal.

SSIM - Structural Similarity Index. It is a metric which gives image quality degradation caused by compression. The value lies between -1 and 1, where the value corresponding to 1 signifies similar images.

VIFP - Visual Information Fidelity in pixel domain. It is part of the Human Visual System (HVS). It is derived from two quantities: when no distortion channel is present then the mutual information between the input and the output of the HVS channel and information between the input of the distortion channel and the output of the HVS channel for the image.

Comparison of different presets for every metric is provided as follows.

Figure 2.1: PSNR

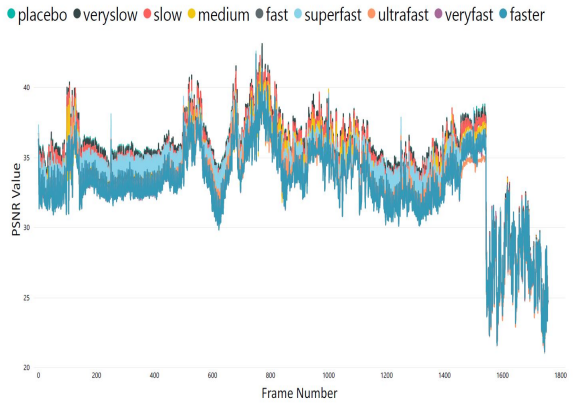


Figure 2.2: SSIM

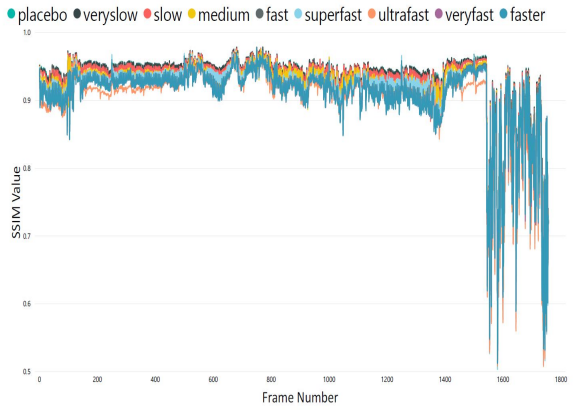


Figure 2.3: VIFP

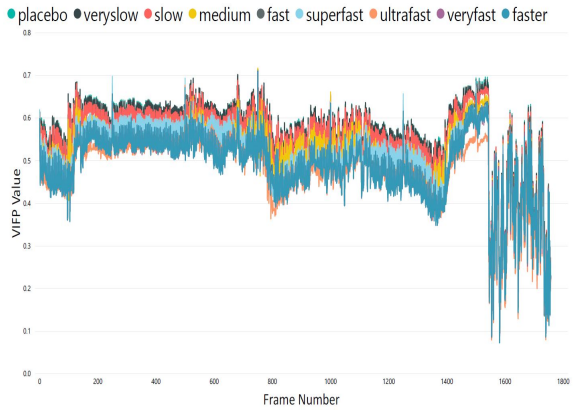


Figure 2.4: PSNR Sliced

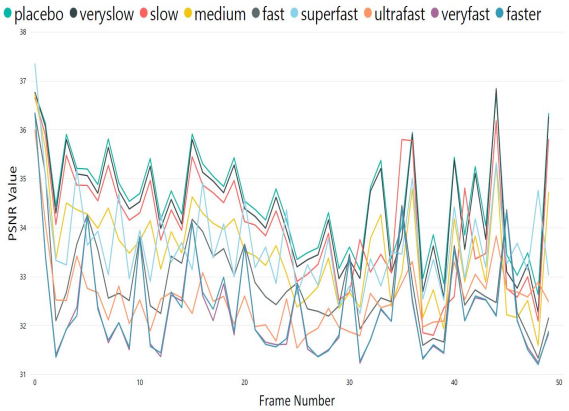


Figure 2.5: SSIM Sliced

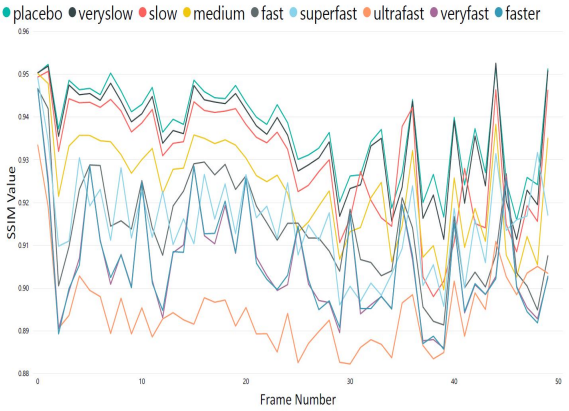
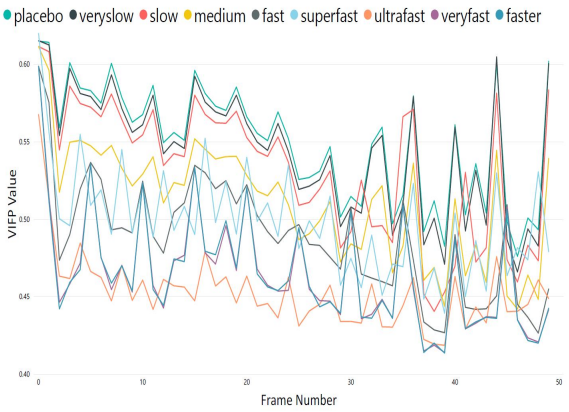


Figure 2.6: VIFP Sliced



Preset	Average PSNR	Standard Deviation of PSNR	Average of SSIM	Standard Deviation of SSIM	Average of VIFP	Standard Deviation of VIFP	Bitrate kb/s	Encoding Time
ultra fast	33.18	2.9	0.901	0.058	0.474	0.079	4608	56 secs
super fast	33.98	3.08	0.913	0.06	0.507	0.087	5202	1 min 14 secs
very fast	32.92	2.9	0.907	0.055	0.484	0.081	3824	1 min 30 secs
faster	32.92	2.9	0.907	0.055	0.484	0.081	3823	1 min 38 secs
fast	33.19	2.92	0.911	0.054	0.495	0.082	3905	3 mins 02 secs
medium	33.74	3.11	0.918	0.056	0.517	0.086	4426	3 mins 47 secs
slow	34.28	3.29	0.924	0.057	0.54	0.091	4731	10 mins 25 secs
very slow	34.47	3.36	0.926	0.058	0.547	0.094	4905	29 mins 27 secs
placebo	34.56	3.37	0.928	0.059	0.55	0.0944	5037	2 hrs 3 mins

Table 2.2: Statistical Results

The result is determined from the graphs in Figures 1 to 6 and shown in the Table 2. It includes Average PSNR, Standard Deviation of PSNR, Average SSIM, Standard Deviation of SSIM, Average VIFP, Standard Deviation of VIFP, Bitrate of compressed file and time required for compression for every preset of x265 encoder.

It is clear that preset Placebo gives the highest value of PSNR, SSIM and VIFP. It means that it retains most quality in the compressed video, as compared to other presets. Deviation of Placebo is very high, suggesting that it varies frame to frame. Faster and Very Fast presets give similar and lowest value for every metric. Also, bitrate is low for them which results in less size, as the bitrate is directly proportional to the size. Super Fast takes less time but still, average PSNR value is higher compared to other fast presets. Placebo requires the highest time for compression while Ultra Fast requires the lowest.

Hence time, size and quality are the three measures of video compression. If the requirement is to retain maximum quality, then Placebo is the suitable preset, but it trades off with longest encoding time. If a smaller size of video is required then Faster preset is useful and if faster encoding time is required then Ultra Fast preset is useful.

Another parameter that is varied in x265 encoder is CRF. It stands for Constant Rate Factor and is used to set the quality factor of the output video. Comparing two different videos with CRF values set at 25 and 28 while keeping same presets, it can be easily noticed from the graphs of PSNR, SSIM, VIFP as shown in Figures 7,8 and 9 that video encoded with CRF 25 has a significantly lower loss in quality as compared to CRF 28.

Figure 2.7: PSNR Comparison

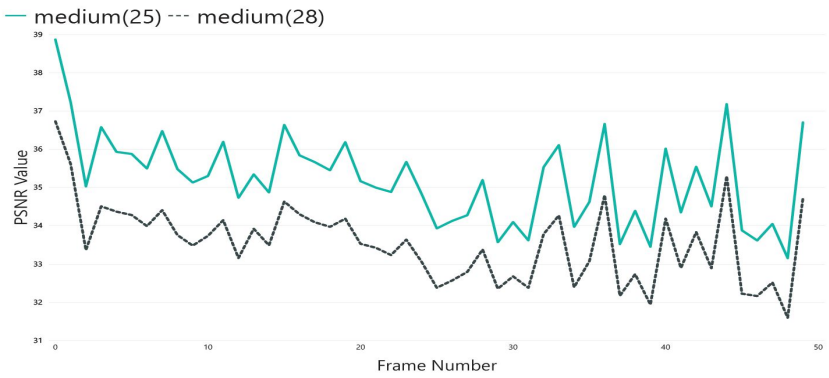


Figure 2.8: SSIM Comparison

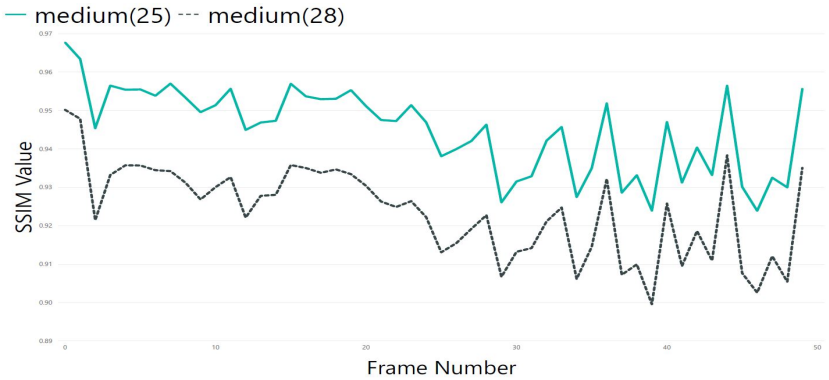
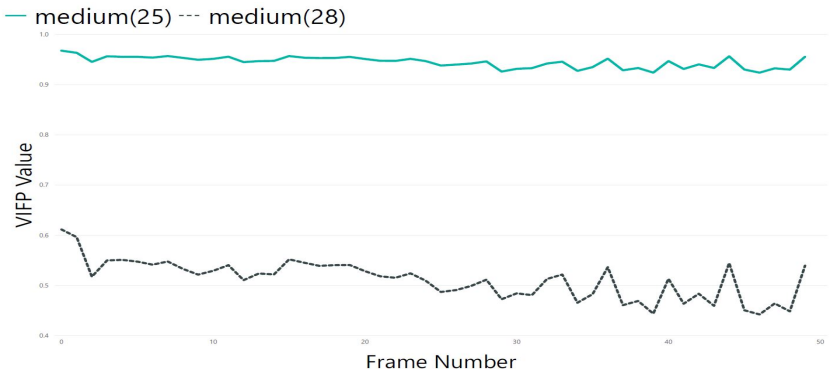


Figure 2.9: VIFP Comparison



2.4 Working of H.265/HEVC

The HEVC standard is based on a hybrid model which incorporates motion estimation and compensation, transforms and quantization, and entropy encoding[9]. The two models used are the prediction model and the spatial model.

The prediction model utilizes the redundancies present in the spatial and temporal domain. The inter prediction of the frames is done based on two factors: 1) Motion Estimation(ME): finding the best match between regions of reference and past or future frames, and 2) Motion Compensation(MC): finding the difference between matching regions. The output of the prediction model is residuals and motion vectors. Residual is the difference between a matched region and the reference region. A motion vector is a vector that indicates the direction in which block is moving and it determines the offset by which a reference block to the current block is located in the past or future picture.

The statistical model deals with transformation and quantization. Here transformation reduces the dependency between sample points, and quantization reduces the precision at which samples are represented. The most commonly used transformation is the Discrete Cosine Transform(DCT). The output from DCT is coefficients that are further quantized to reduce the number of bits required for coding it. Quantization could be scalar or vector based. Scalar quantization maps the range of values to scalars, while vector quantization maps a group of values, such as image samples, into a codeword. Further Run-Length Encoding(RLE) is used to store consecutive occurrences of a digit as a single value together with digit's count.

From the above topics, it can be said that compression revolves around the presence of motion in the video.

2.4.1 Experiment

To determine the impact of motion in compression, two 15 seconds long, different motion videos were recorded with a smartphone in the same environment. Here one video has a static background with a moving subject while another video has a moving background with moving subject. Both videos are compressed with 3 different presets: faster, medium and slower. Then the quality comparison is done using VQMT. We also compare the output sizes after compression with the presets.

As seen in Table 2.3, for all three presets, the PSNR, SSIM or VIFp values does not vary much indicating that the compressed video quality is very similar. Similarly for the dynamic video, from Table 2.4, the compressed quality remains similar regardless of the presets.

Preset	Average PSNR	Standard Deviation of PSNR	Average of SSIM	Standard Deviation of SSIM	Average of VIFP	Standard Deviation of VIFP	Bitrate kb/s	Encoding Time
faster	5.73	0.384	0.0082	0.0019	0.017	0.0052	2410	23 secs
medium	5.73	0.383	0.0081	0.0019	0.017	0.0051	2805	41 secs
slower	5.73	0.383	0.0081	0.0019	0.017	0.0052	3840	5 mins 10 secs

Table 2.3: For Static Background Video

Preset	Average PSNR	Standard Deviation of PSNR	Average of SSIM	Standard Deviation of SSIM	Average of VIFP	Standard Deviation of VIFP	Bitrate kb/s	Encoding Time
faster	6.1	0.695	0.013	0.0041	0.0159	0.0055	2410	34 secs
medium	6.1	0.695	0.013	0.0041	0.0145	0.0055	2805	1 min 2 secs
slower	6.09	0.695	0.013	0.0042	0.0145	0.0055	3840	8 mins 44 secs

Table 2.4: For Dynamic Background Video

Video/Preset	Original	Faster	Medium	Slower
Static	40.4	4.73	5.45	7.34
Dynamic	40.7	10.5	12	15.8

Table 2.5: Video Size Comparison, values in MBs

While the encoding time for the presets vary widely for both the videos. Then it can be said that rather than choosing a slower preset we can select a faster preset, and have similar quality with less time taken for the compression. Also from the Table 2.5, it can be seen that motion affects the output size after compression. Under the same preset the size of the compressed dynamic video is double to that of the compressed static video.

2.5 Existing Video Compression approaches using Machine Learning

2.5.1 Mode Decision (MD) Problem

Current compression standards mostly split video sequences into 16x16 pixels in size. Each of these blocks is called macro-blocks. Different mechanisms are then applied to code the image information. The first option is INTRA-mode which encodes the block in JPEG-like manner and the second option is INTER-mode in which a predictor block from the position of a previous frame is chosen and the difference is encoded between current block and prediction. Other modes can be splitting the blocks into sub-blocks and more than one block is used for prediction. So the availability of more modes creates a problem of mode decision (MD). MD plays a vital role in the encoding so choosing the wrong mode can directly reduce the encoding efficiency.

The main concern is fast classification because in real-time conditions MD is performed online in an encoder. Therefore various machine learning techniques can be applied for fast classification. The classifier is trained offline and resulting classifier with fixed weights is included in the encoding system. Following are the Machine Learning classifier techniques :

(i) **Linear Classifier:** Linear classifiers that work directly in the feature space are very fast to evaluate. The main reason for this is that each decision requires one scalar product of features to the weights vector only. Current encoders use heuristic mode decision methods which are typically linear.

(ii) **Decision Trees:** These are fast classifiers as well. They provide the additional advantage that they can be directly transformed into the source code of any programming language. This task is easily achieved cascading the statements.

(iii) **Neural Networks:** Artificial neural networks are universal classifier because they can learn any decision function with high precision. They lack in speed for classification than the above methods.

Other machine learning techniques are also available and might even perform better. For

example, Support Vector Machines (SVM) and Nearest Neighbour Classifiers (NNCs) [10].

Therefore in MD problem, machine learning is used in the optimal selection of mode. But overall problem still remains unsolvable that which standard is best to use under given conditions and how a trade-off between desired quality, video size and encoding time can be achieved, which is the main aim of this project.

2.5.2 Motion Estimation through Machine Learning

This approach provides a method to improve motion estimation in video encoding. Motion estimation is a method of determining predicted blocks for current frames based on the subsequent frames. Video compression is possible using this approach. Motion estimation is already used by various codes such as H.264 and H.265, but the introduction of machine learning techniques improves it. A hierarchical algorithm can be applied by identifying reference elements in one or more input frames of the video file. The estimated motion vector is determined by comparing one or more reference elements in one or more input pictures of the video file [11].

2.5.3 Motion Compensation through Machine Learning

Video can be predicted for given previous and future frames by motion compensation frame of input. So this method directly results in a reduction of data because memory required for frames will be limited after motion compensation. It is already implemented in widely used codes as in motion estimation, but motion compensation can optimize efficiency and encoding speed. It proposes a similar hierarchical algorithm. The only difference from motion estimation is that one or more blocks of video data are predicted from one or more blocks of original video data [**van2018motioncomp**].

2.6 Summary

Adding a machine learning algorithm to existing video compression techniques provides optimized compression in terms of encoding speed, quality loss, output video size, etc. but there is no existing solution available which can choose a codec that is best among all with the suitable container and can provide the desired trade-off between encoding speed, output video quality and size, that is the overall aim of this project.

Chapter 3 Proposed Work

Currently available video encoders provide various options and parameters which one can set and define how the algorithm will encode and compress the video. Determining appropriate parameters for required output characteristics like specific sized output video or keeping the quality loss in output video minimal, is a hit and miss approach. With sufficient amount of data, patterns or correlation between various video features and significant parameters can be found that will produce desired compressed output videos.

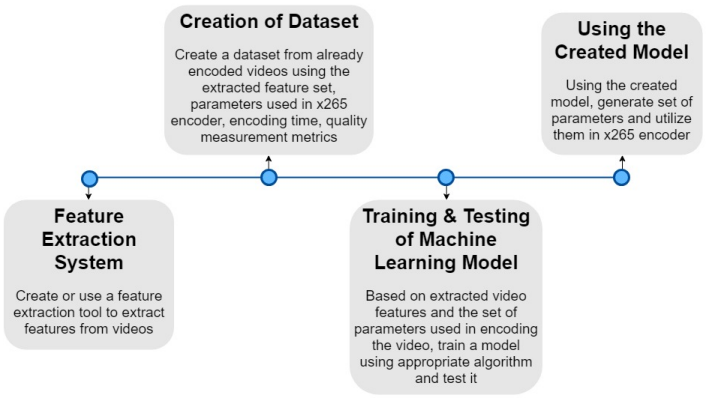
3.1 Parameter Prediction using Machine Learning Techniques

Consider the presence of a large varied dataset of videos which will include the features of the input video, set of parameters used for encoding, size of the output video, the time required for compression and metrics used to identify quality loss after compression. Various machine learning algorithms can be used based on this dataset which can associate input video features to the output video characteristics like video quality metrics determining the quality and the output size, which can further be used for prediction of various encoding parameters.

3.2 Framework for the Process

A video is provided to a feature extraction system in the first stage which will output a set of required features. Then the feature set along with the proposed quality and size is passed to the machine learning model. It will output a set of parameters based on the trained model which will be used by the x265 encoder. The output of the encoder will be a compressed video, best tuned to the proposed size and quality.

Figure 3.1: Framework



3.2.1 Feature Extraction System

This system processes a video on the basis of object motion, picture clarity, level of detail, camera jitter (if applicable), dynamic background and generates a set of features. Each feature produced can be used to identify the degree of presence of these characteristics.

3.2.2 Dataset Creation

A custom dataset needs to be created, due to the lack of required dataset. This dataset should be large enough exploring all the possible permutation of the encoding parameters used in x265 for various videos. It should include the feature set of the videos. It should also include the output characteristics of the video encoding process such as duration of encoding, size of the output video, bitrate of the output video.

3.2.3 Machine Learning Model

A Machine Learning Model will be trained using the appropriate approach on the created dataset which will co-relate the video feature set with the set of parameters used for encoding, size of the output video, the time required for compression, metrics used to identify quality loss after compression.

A model will be generated, as a result of this training that will provide a set of optimal parameters which is used to encode the video using the x265 encoder.

3.2.4 Video Encoding

The proposed set of parameters, predicted by the machine learning model are then passed on to the x265 encoder to encode the video resulting in desired quality and size.

3.3 Video Features Extracted By System

As discussed earlier in working of HEVC, it tries to estimate the current frame based on the previous set of frames and upcoming set of frames. To correctly estimate the frame various algorithm with different complexity are used. So presets which require more encoding time will use more complex algorithms and will correctly estimate the frame and will try to retain as much as fine details in the video. In short, if we will choose presets between medium to placebo then encoding time will increase and most of the video quality will be retained. This set of presets can be used in videos which consist of high amount of object motion in consecutive frames and dynamic background. Because for correctly estimating frames in these type of video, more complex algorithms will be required and these set of presets fulfill that necessity. On the other hand, presets from medium to ultra fast uses less complex algorithms and shortcuts for estimating the frame. So, videos consisting less amount of object motion and static background will need less processing for estimating frames and presets in this range fulfill that necessity. So, the main aim of feature extraction system is to find out the amount of motion in the consecutive frames. Based on that, feature extraction methods are presented in this section.

3.3.1 Background Subtraction

Background subtraction is used in various motion tracking application in computer vision. It generally consists of two steps, creating a statistical representation of the background and identifying object motion in the foreground by subtracting consecutive frames. So, in output all the background will be subtracted and only motion will be detected. Background subtraction is performed by various methods such as basic subtraction of consecutive frames, statistical method, fuzzy logic method, machine learning based method and non parametric method. In this project, the Gaussian Mixture Model (GMM) based statistical background subtraction is used with the help of the OpenCV library. In this model, the Gaussian distribution of every pixel in particular frames is generated. Initially, every pixel is represented by its intensity values in the RGB color model. Then, it is determined that whether the pixel belongs to foreground or

background by probability given in Eq. 3.1 [12].

$$p(X_t) = \sum_{i=1}^{K} \omega_{i,t} \cdot \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \tag{3.1}$$

Here, X_t is current pixel of frame t, K is number of distribution in sample, $\omega_{i,t}$ is the weight of k^{th} distribution in frame t, $\mu_{i,t}$ is the mead of the k^{th} distribution in frame t, $\Sigma_{i,t}$ the standard deviation of the k^{th} distribution in frame t. Where $\eta(X_t, \mu_{i,t}, \Sigma_{i,t})$ is probability density function. It is given by Eq. 3.2 [12].

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp^{-\frac{1}{2}(X_t - \mu)\Sigma^{-1}(X_t - \mu)} \tag{3.2}$$

Here, the difference in the value of intensity can be assumed to have uniform standard deviation [12]. So, covariance matrix can be given by Eq. 3.3 [12].

$$\Sigma_{i,t} = \sigma_{i,t}^2 I \tag{3.3}$$

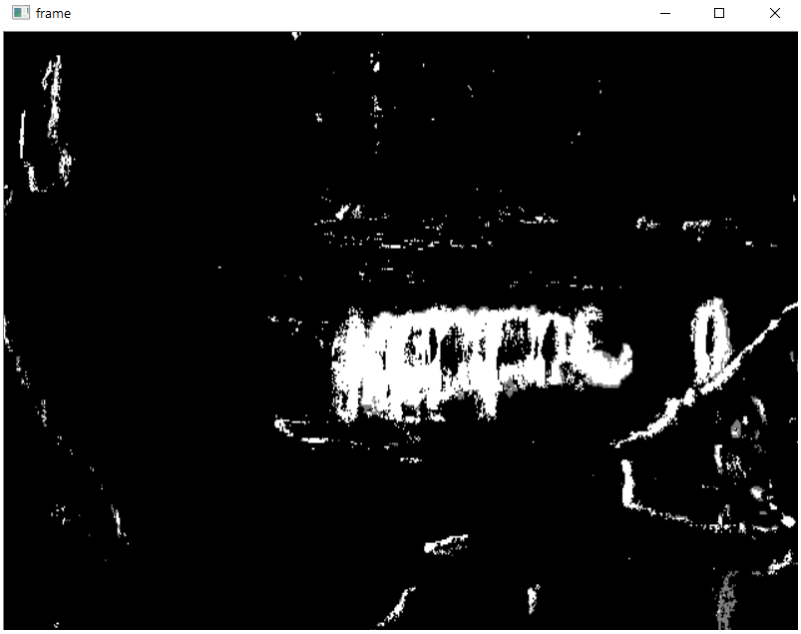
The Gaussian value which is greater than the threshold is considered as background and other values are considered as foreground. Values of ω, μ and σ is updated when pixel value matches the one of the Gaussian values [12]. To enhance the result of GMM combination of Median Filter and Hole Filling Algorithm are used [12].

Figure 3.2: Video Frame Before Background Subtraction

By using the already available functions of OpenCV, background subtraction is performed. First, video input is passed to the function which takes frames as input and performs GMM on

it.The output of this function will be black white frames for original video. Here, the white pixels represents motion in consecutive frames. Here, for finding motion frames are compared at 0.1s interval because the frame rate of all videos will be different. After that, the number of white pixels is counted for each frame and amount of white pixels in a frame is calculated. In the end, an average of all individual percentage amounts is calculated and that will be the motion feature of the video. A video frame consists of motion is shown in Fig. 3.2 and output of background subtraction is shown in Fig. 3.3. As can be seen, moving penguins are detected as white pixel in output video frame.

Figure 3.3: Video Frame After Background Subtraction



3.3.2 Pearson Correlation Coefficient

Pearson Correlation Coefficient (PCC) is a measure of the linear correlation between two variables or sets X and Y. It is also referred to as Bivariate Correlation. According to the Cauchy–Schwarz inequality it has a value between +1 and -1, where 1 is total positive linear correlation, 0 is no linear correlation, and -1 is total negative linear correlation [13]. It is given by the Eq. 3.4 [13].

$$\rho(X,Y) = \frac{cov(X,Y)}{\sigma_x \sigma_y} \tag{3.4}$$

In the context of videos, it can be used to compare the correlation between two different frames in order to check if the corresponding pixels in both frames are correlated. In order to maintain a general format for measuring all the features, PCC values will be converted

to a percentage format. For example, given two sets [1,2,3,4,5] and [2,4,6,8,10] will have a correlation of 100 %. The inbuilt function called **pearsonr** of the SciPy library can be used to return the correlation for two sets.

The images usually have pixel data in a 3-dimensional array. Hence, they are first flattened into a 1-dimensional array for calculating PCC. The resulting PCC across all frame pairs is then averaged to calculate the Average PCC for the complete video.

3.3.3 Scene Change Detection

Every video has some significant variation in pixels whenever the objects, plane or entire background changes. Such a situation can be marked as a Scene Change. Performing background subtraction on the video during the event of any scene change, causes the value of motion feature to reach around 90-100 % for roughly a second. Keeping an assumed threshold limit of 80%, whenever our motion feature exceeds this limit, a new Scene is added with the initial frame, final frame, total motion across that scene and the average motion of that scene. For compensating the error generated in the motion feature due to a scene change, the frames in the next half second are skipped before marking the beginning of the next scene.

Scene change data can contribute in identifying whether a video has similar content in longer time period or different content in shorter time period.

3.3.4 Intensity Matching

By using a Fast Library of Approximate Nearest Neighbour based matcher present in OpenCV, intensity matching is performed. It uses a K-Nearest Neighbour method in which k neighbours of a pixel as matched with the consecutive frame. If all the neighbour's pixel value are related to some factor then it is considered as an intensity change. The count of matched pixels is considered as another video feature. As can be seen in Fig. 3.4 and Fig. 3.5, intensity values are changed in 3.5 from the original image. In Fig. 3.6 intensity matching is shown.

Figure 3.4: Original Video Frame

Figure 3.5: Intensity Changed Video Frame

Figure 3.6: Intensity Matching Between Video Frames

Chapter 4 Summary and Future Work

With increasing demand for video content nowadays, it is becoming necessary to transmit and store the videos efficiently. Therefore, the need for optimal video compression system arises. To summarize, a system is being proposed that determines the set of parameters generated from Machine Learning model based on different video features. Here, features are extracted by Feature extraction system. This feature extraction system marks the pre-processing that helps in the categorization of the videos based on Motion Feature, Pearson Correlation Coefficient, Scene Change Detection and Intensity Matching.

In future, categories will be created based on the results of features extracted from the video data sets. Training data set will be made by introducing videos of more categories. Machine Learning model needs to be trained on the obtained dataset, generating its optimal preset and parameters.

References

[1] M. Templeman. (). 17 stats and facts every marketer should know about video marketing, [Online]. Available: https://www.forbes.com/sites/miketempleman/2017/09/06/17-stats-about-video-marketing/. (accessed: 29.09.2018).

[2] Wyzowl.com. (). Why video is exploding on social media in 2018, [Online]. Available: https://www.wyzowl.com/video-social-media-2018/. (accessed: 29.09.2018).

[3] J. Jachetta. (). Understanding the most common video encoding formats, [Online]. Available: https://vidovation.com/understanding-common-video-encoding-formats. (accessed: 30.09.2018).

[4] A. Rod. (). H.264 vs h.265-a technical comparison, [Online]. Available: https://medium.com/advanced-computer-vision/h-264-vs-h-265-a-technical-comparison-when-will-h-265-dominate-the-market-26659303171a. (accessed: 30.09.2018).

[5] D. Mukherjee, J. Bankoski, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, and R. Bultje, "The latest open-source video codec vp9-an overview and preliminary results", in *Picture Coding Symposium (PCS), 2013*, IEEE, 2013, pp. 390–393.

[6] R. S. Bultje. (). Vp9 encoding/decoding performance vs. hevc/h.264, [Online]. Available: https://blogs.gnome.org/rbultje/2015/09/28/vp9-encodingdecoding-performance-vs-hevch-264. (accessed: 30.09.2018).

[7] P. Akyazi and T. Ebrahimi, "Comparison of compression efficiency between hevc/h. 265, vp9 and av1 based on subjective quality assessments", in *10th International Conference on Quality of Multimedia Experience (QoMEX)*, 2018.

[8] V. Deep and T. Elarabi, "Hevc/h. 265 vs. vp9 state-of-the-art video coding comparison for hd and uhd applications", in *Electrical and Computer Engineering (CCECE), 2017 IEEE 30th Canadian Conference on*, IEEE, 2017, pp. 1–4.

[9] S. Tamanna, *Transcoding h. 265/hevc*, 2013.

[10] C. H. Lampert, “Machine learning for video compression: Macroblock mode decision”, in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, IEEE, vol. 1, 2006, pp. 936–940.

[11] S. Van Leuven, J. Caballero, Z. Wang, and R. D. Bishop, *Motion estimation through machine learning*, US Patent App. 15/856,769, May 2018.

[12] A. Nurhadiyatna, W. Jatmiko, B. Hardjono, A. Wibisono, I. Sina, and P. Mursanto, “Background subtraction using gaussian mixture model enhanced by hole filling algorithm (gmmhf)”, in *2013 IEEE International Conference on Systems, Man, and Cybernetics*, Oct. 2013, pp. 4006–4011. DOI: 10.1109/SMC.2013.684.

[13] R. S. (Britain), *Proceedings of the Royal Society of London*, v. 58. Taylor & Francis, 1895. [Online]. Available: https://books.google.co.in/books?id=60aL0zlT-90C.

Acknowledgement

We take this opportunity to express our deep sense of gratitude and indebtedness to our project guide, Dr. Rupa G. Mehta, Associate Professor in Computer Engineering Department, SVNIT Surat for her valuable guidance, useful feedback and co-operation with kind and encouraging attitude in this project.