Performance Evaluation of Alternative Video Encoding Formats on Space Efficiency and Energy Consumption

Name:

Institution Affiliation:

Date:

Supervisor:

**Executive Summary**

The 21st Century is marked by rapid development in many spheres of Information and Communications Technologies (ICT). One of the most recent trends that have come with this rapid technological revolution is the mass adoption and use of hand-held computers (mobile devices). Today, mobile devices such as laptops, smartphones, and tablets have become more and more part of our daily lives. One of the most popular applications running on mobile devices today is video playback. With video playback being the norm of today's world coupled with the exponential growth of demand for watching videos, a lot of efforts have been put on the field of video encoding and decoding[1]. Recent research on video coding technology found that people's main interest is on compressing video size while delivering high definition videos able to satisfy the quality demands. The quality of a video is no longer considered a fixed parameter that can be statically provided by the video content provider. Rather, video decoder functions to select the video basing on the capabilities of the device playing the video. As a result, the video production industry and the academia have focused on coming up with a fully functional, high-performance (in terms of energy consumption and memory efficiency) and easily licensed codec. However, researchers explain that higher energy consumptions characterize higher compression efficiency and higher video quality. In a report given by global energy consumption in 2019, the streaming media witnessed an increase of 84% of energy consumption [14]. The ungoverned utilization of streaming video has continued to grow at an exponential rate, threatening sustainability through increased carbon emissions, which negatively affects the environment [2]. The main aim of this project is to emphasize the importance of sustainability in video production. Finding a balance of high performance and low energy consumption in the video industry is crucial in fighting global warming and protecting environmental resources. Therefore, the project will focus on developing a model capable of optimizing the encoding strategies. The key deliverables include:

1. A review of the literature to examine the precious research on video encoding methods, with the main focus being on examining the memory efficiency and energy consumption of the encoding methods.
2. Develop a tool that can be used to monitor and evaluate video encoding performance in different encoding formats.
3. Conduct a survey to analyze the definition of video quality from the perspectives of users.
4. Develop a machine learning based model capable of optimizing the encoding formats given the number of views and platforms.

The research proposal will then try to analyze the potentials and limitations of the previous research work on the power consumption of video encoding formats. In addition to this, the project proposes an executable optimization plan for the research question to balance performance and energy consumption of video encoding.

**Performance Evaluation of Alternative Video Encoding Formats on Space Efficiency and Energy Consumption**

1. **Introduction**

The 21st Century is marked by rapid development in many spheres of Information and Communications Technologies (ICT). One of the most recent trends that have come with this rapid technological revolution is the mass adoption and use of hand-held computers (mobile devices). Today, mobile devices like laptops, tablets, and smartphones have become part of our daily lives, with video playback being among the most popular applications that people have in their mobile devices. This has been a result of the growing development and use of platforms for sharing videos (e.g., YouTube, Courchtuner, Netflix, etc.), social networking platforms (e.g., Instagram, Facebook, Tiktok, Likee, Twitter, etc.), and mobile IPTV and video-conferencing. With this trend, video data is expected to represent up to 70% of the overall traffic used by Internet mobile by 2022 [1, 4, 14]. According to the most recent study, it was found that up to 200 million mobile users spend an average of 52 minutes per day on video [14]. With the increasing number of video-sharing platforms and applications, the main area of focus has been on compressing the size of the video as much as possible so as to promote easy sharing while at the same time pursuing high definition and satisfactory quality. This has, therefore, has been one of the main reasons why the field of video technology has witnessed remarkable innovations in video encoding and decoding.

Video encoding refers to a process that uses an image's statistical characteristics to remove redundant information in a video signal. This is basically what we refer to as compression. The compression rate of a video, which depends on different video parameters and encoding standards, can reach hundreds of times or even higher[4]. When a video is recorded, the device used for recording dictates the specific file format for the original file as well as other specifications. Consequently, if the video is to be transmitted to other more devices, there will be the need to consider the various devices that will support the video. With this approach in mind, it simply means all the videos we play in our mobile devices (smartphones, tablets, and computers) have gone through an encoding process that involves converting the original source video to enable the video to be viewed on various output formats. On the other hand, video decoding refers to the process that uses a decoding device to convert the digital signal formed by an encoder into a video [1]. Video compression standards or simply video codecs dictates the process of video encoding and decoding.

As we have already provided a little insight into video encoding, we can say that there are two main reasons for video encoding. First, encoded videos tend to be easier to transmit over the Internet because the compression reduces bandwidth, although this is usually accompanied by lower quality. Therefore, without compression, the size of a video can be too high, making it hard to transmit under normal or inadequate connection speeds [5]. This would negatively affect the streaming media because the bit rate has an impact on the fluency of the video watched by people. Secondly, encoding is very useful in ensuring compatibility. As discussed, there are situations where although the video is compressed and of the correct size, it still fails to play because it is not compatible with the device. In such a case, video encoding can ensure that the compressed video is transcoded into the right format that is compatible with the media device.

One of the most important aspects of usability [7] of mobile devices is on energy consumption. Although battery technology has been on constant improvement, it has failed to catch up with the increasing power usage of the devices as a result of the constant improvements in processors and display technologies. The video applications in mobile devices are normally executed in a heterogeneous environment. This means that different mobile devices, having varied processing and video displaying features, can access the same video content. To add to that, network technologies used in the transportation and sharing of the video content also have different bandwidths ranging from Kilobits to tens of Megabits per second. Therefore since mobile devices have different processing capabilities, it is important for video content providers (e.g., YouTube and Netflix) to support the dynamic quality adaptation of video decoding [6, 7]. Dynamic quality adaptation is a technique that allows the video decoder to automatically adjust video quality at run time [8]. The quality of a video is no longer considered as a fixed parameter that can be statically provided by the video content provider. Rather, video decoder functions to select the video basing on the capabilities of the device playing the video.

The efficiency of a video usually depends on the balance between minimizing the video's file size for both streaming and storage while maximizing the video quality. In addition to this, a lot of interests have also been on the quantity of data required to represent the bit rate of a video, the encoding and decoding algorithms used, ease of editing, and minimizing data loss. However, to achieve the high efficiency of a video, the processing capabilities of the embedded microprocessors equipping video playing device must also be enhanced. Highly efficient encoding techniques require microprocessors with clock frequency exceeding 1 GHz. The only setback associated with increasing the performance of microprocessors is high energy consumption [5]. The energy efficiency of a video codec is today considered among the most important criteria that can be used to determine the quality of a video retrieved from a network. One notable illustration is on the fact that a video decoder before selecting the quality of video playback may have to consider the remaining energy budget of a device as a way of increasing its autonomy. The ever increasing trend of video content consumption has been characterized by high energy and memory consumption issues. This can be attributed to the fact that modern video codecs have been developed using highly complex compression algorithms to meet the growing demands of small-sized high-quality videos. While the modern video codec provides possibilities of achieving a high compression ratio, the setback is that they are associated with high demand for processing resources and high energy consumption.

With the increasing calls for sustainability and conservation of the environment, this significant computational load posts a significant threat. According to Yahia and the team, the global power consumption of data centers was approximately 3% in 2018 alone [7]. Furthermore, the global energy consumption of streaming services increased by 84% in 2019 this accounting for a total of 451,000-megawatt-hours. This consumption, according to (CITE), is enough to power up to 40,000 average homes in Us for a period of one year. This clearly shows that digital technology has ushered in an age of inconspicuous energy consumption. Therefore, there is a need to have long term solutions in place to improve video encoding [14].

1. **Aims and Objectives**

There exist many published international standards on video compression algorithms. However, because these video compression algorithms were developed at different times, they have different needs, different features, and different performance and algorithmic properties. Therefore, if these algorithms are applied in mobile systems characterized by constrained resources such as memory and power, they have to be adapted to the systems. Other than high energy and memory consumption issues, other limitations include delay requirements, particularly in real-time video communications and constrained bandwidth. Although there are no codecs that can offer solutions to all the listed limitations, we can offer the direction of adjustment and development along with a number of improvements to see that one of the already established codecs meets these requirements. Therefore, while examining VP8, VP9, H.264, H.265, and AV1, the main aim of this project is to achieve the state of the most energy-efficient with the least memory space under different scenarios, using different codec technologies.

To achieve this aim, the project has the following objectives:

1. Develop a tool that can be used to test and monitor the performance (in terms of energy and memory efficiency) of video encoding formats under different platforms and scenarios.
2. Compare the energy and memory consumption of VP8, VP9, H.264, H.265, and AV1 codec technologies.
3. Quantify the energy consumption and space efficiency of VP8, VP9, H.264, H.265, and AV1 codec technologies.

The contributions of this project to the research field on video codec include: this project will help in evaluating and identifying the most energy and memory-efficient encoding method among VP8, VP9, H.264, H.265, and AV1.

1. **Literature Review**

The main focus of this project is on performance-based in memory and energy consumption in video encoding technologies. Space/memory efficiency and energy consumption, which has continued to attract the attention of many researchers, is one of the core indicators for evaluating the coding performance.

**3.1 Characterizing Video encoding Performances and energy consumption**

**3.1.1 Performance Characterization of video encoding**

Several scholars have attempted to address the performance of video encoding. Their objective has been to identify video encoding format that is most consumes most processing resources during the encoding process and as well to give an insight into performance drop. Therefore, this will help us in examining performance based on three levels: application level, system-level, and architectural level.

**3.1.2 Performance at the Application level**

Yao and the team in their study analyze the complexities of different video qualities. The authors note that each video quality has a different video sequence [9]. The results from their study indicate that the performance of an encoding format varies with the increase in video quality. The authors further add that for the scene complexity of a video may affect the decoding performance. While conducting performance analysis on per-frame, it was observed that variations in encoding time also depends on the type of the frame type (with I frames being the most complex frames to encode compared to B frame and P frames).

Kihwan and the team measured the execution time taken by H.264 and AV1 encoding formats. According to their findings, motion compensation was observed to the most time-consuming process, which consumed more than 40% of the total CPU time. On the other hand, entropy encoding consumed about 22% of the total CPU time, with the inverse quantization and transformation consuming only 7%. The authors further noted that the deblocking filter also requires a large amount of computation, consuming up to 20% of the remaining processing time [11].

**3.1.3 Performance at the system level**

In a study conducted by Horowitz and the team [12] on the impacts of video encoding technologies on the performance at the system level, the performance was mainly analyzed based on the completion of the flow of communication between a general-purpose processor and specialized processor. In their study, the authors measured the overhead of Inter-processor communication (IPC). They, therefore, propose a technique that can be adapted to estimate the performance of the IPC at run-time. Therefore, with such a technique, the IPC strategies will be dynamically adjusted depending on the environmental parameters and available system resources. (CITE) on the other hand, conducted a study analyzing the performance of Digital Signal Processors (DSP) decoding based on cache coherency and Direct Memory Access transfers. According to the result obtained by these authors, to obtain a significant performance increase, the decoder design can be modified to minimize the communication gap between the general-purpose processors and the digital signal processor.

**3.1.4 Performance at the Architecture level**

In their study to weigh the performance video decoding technologies at the architecture level, Kavvadias and the team used the Simplescaler simulator to analyze the performance of H.264 and AVC video decoding formats [15] under different architectural configurations. With their focus being on the con analyzing behaviors such as cache miss and instruction-level parallelism (ILP), the authors point out that IPL is directly related to cache performance. As noted, the video decoding process requires a significant amount of time while waiting for data to be fetched and retrieved from the main memory. As a result, there is an increase in the number of cycles per instruction, which in turn decreases the IPC.

While encoding a video, Juraj and the team in their work analyze the performance of video encoding formats based on memory access [13]. From their study, the authors gave their focus on the ratio of cache miss during the encoding process of different types of multimedia workloads. Their study noted that the increase in the number of memory instructions resulted in an increase in video quality. Similarly, Yue and the team conducted a study to analyze memory-bound instruction [14] while also considering the potential impacts of performance scaling when Dynamic voltage and frequency scaling (DVFS) are used. Basing on the statistics on a cache miss, the authors noted that when the H.264 encoding format is used, the inverse quantization is completely computation-bound. However, motion compensation consumes a lot of memory because the data obtained can be used to provide an explanation of the performance scaling. Particularly, DVFS is used; the authors explain that the relative performance scaling differs based on the encoding modules present in a video encoding format. The authors showed that the motion estimation speed-up process is slightly lower than the entropy encoding. This is because, in memory-bound instructions, a high amount of CPU time is consumed while waiting for memory access, which does not depend on the clock frequency.

Gutnik and Chandrakasan conducted a study to analyze the performance scaling of parallel video decoding on multi-core processors and looked at various parallelization possibilities [10]. Their study indicated that there are two main limitations associated with slice-level parallelism—first, the bitrate increases as the number of slices increases. Second, since the number of slices in a given frame depends on the encoder, and that not all sequences will contain a higher number of slices. The authors then proceeded to analyze frame-level parallelism. With frame-level parallelism, some frames (particularly B frames) cannot be utilized as reference frames; thus can their processing can be in parallel. The authors found this approach not to be very scalable due to the fact that the number of B frames between consecutive P frames is less than three. The authors also show that the parallelism of the MB level is very scalable and does not need requirements from the encoder side. In conclusion, the authors point that in order to avoid performance drop, there is a need to reduce communication and synchronization overhead.

**3.2 Energy consumption characterization**

**3.2.1 Energy consumption at the Application Level**

Mehul and the team analyzed the energy consumption at the application level for different video codecs [16]. They performed several experiments to evaluate the effects that codec parameters like resolution and bitrate on energy consumption. The findings from their experiments showed that increase in resolution results in a significant increase in total energy consumed. However, the authors also noted that increasing the bit rate results in better picture quality without consuming too much energy.

Jens-Rainer and the team conducted a study to analyze how video quality scalability affects energy consumption [17]. Apart from video resolution and the bitrate, the authors also analyzed the impacts of frame rate on energy consumption. The results from their study also match that from a previous study: video resolution is one of the most crucial video encoding parameters that heavily affect the consumption of energy. The authors, therefore, propose a strategy that can be used to rescale settings of video quality on encoding techniques in order to save on energy consumption.

**3.2.2 System Level**

While focusing on software and hardware-based video decoder, [4] evaluated the cost associated with the multimedia framework and the operating system overhead. According to the findings of these authors, the interfacing overheads of the operating systems and software framework hiding the implementation details can be significant irrespective of the video decoding technique implemented.

More recent studies by [19] analyze the consumption of energy in multimedia processing using a heterogeneous System on Chip (SoC). The authors noted that the energy consumption of two or more processors running concurrently is relatively lower compared to the sum of energy consumed when each processor is running independently.

**3.2.3 Architectural level**

Martin and the team evaluated the benefits relating to performance and energy consumption when integrated GPU and DSP cores were used [20]. From their studies, the authors point out that when a specialized processor is off-loaded, the execution time reduced is relatively greater than the total power consumption. Therefore, the overall consumed energy drastically reduces, as well. The authors also noted that resource-intensive algorithms were observed to have several subtasks that exhibited different characteristics.

With our focus on four video encoding technologies - VP8, VP9, H.264, H.265, and AV1, we begin by analyzing Layek's work, which focused on evaluating the performance of these video encoders. In order to balance between performance (quality), time, and size of the encoded video, the authors selected four measurable parameters, which included Absolute Bitrate, Constant Rate Factor (CRF), Constant Quality Level (CQ) and Timing Pre-set. Among these measurable parameters, the authors focused on the energy and space efficiency of the encoding method.

**3.3 Video Coding**

**3.3.1 MPEG-4AVC (Advanced Video Coding)**

MPEG-4AVC, also known as the H.264, is a video compression standard popularly known for its high-precision video recording, compression, and distribution capabilities. H.264 is the coded standard for Blu-ray discs [1]. H.264 became famous after Apple abandoned Adobe's VP6 encoding for it. H.264 has always been used in millions of iPads and iPhones. Furthermore, H.264 has also been widely applied in the field of streaming services, in high-definition broadcasting and satellite TV broadcasting. When compared to coding standards that existed previously, H.264 included new characteristics like variable block size motion compensation, multi-reference frame motion compensation, intra-frame prediction coding, etc. As a result, it maintains a high video quality and low bit rate.

**3.3.2 HEVC Format**

HEVC, also known as H.265/MPEG-H, is a high-efficiency video encoding standard formulated by ITU-TVCEG to replace H.264. It is mainly based on H.264, with major improvements being on the utilization of advanced algorithms that improves the relationships among code streams [9, 13], the quality of video encoding, delay, and algorithm complexity, thus achieving the optimal setting. The specifics research content carried out by these authors include:

1. Improving the efficiency of compression
2. Improving the robustness and error recovery capabilities
3. Decreasing the real-time delay
4. Decreasing the channel acquisition time and random access delay.
5. Decreasing complexity.

Comparing the coding architecture of HEVC to H.264, Thomas and the team point out that, to some extent, HEVC has a similar architecture to H.264, mainly in terms of intra and inter prediction, transform, quantization, deblocking filter, entropy coding and other modules [18]. The difference is, however, on the fact that the coding architecture of HEVC has three main units which are, the coding unit (CU), a prediction unit (PU), and the transform unit (TU). Therefore, unlike H.264, HEVC provides more tools that can be used for decreasing the code rate. Analyzing the CU of both HEVC and H.264, both have the smallest CU as 8\*8 and the largest as 64\*64 [2, 9]. Areas containing little information have larger macroblocks with fewer code words, while areas with more details have smaller macroblocks. The intra prediction mode of HEVC supports up to 33 directions, while H.264 supports only eight intra prediction directions. However, H.264, with its eight intra prediction directions, provides better motion compensation processing and vector prediction methods.

HEVC was mainly designed for transmitting higher-quality videos through the Internet under limited bandwidth [10, 16]. This means that users playing videos using HEVC can enjoy high quality videos with only half the original bandwidth. This also indicates that smartphones, tablets, and other mobile devices used by users will be able to play 1080p full HD videos directly online. HEVC standard also supports both 4k high-definition, and 8K ultra-high-definition. However, several studies have pointed out that HD pixel requires more complex codecs in order to minimize bandwidth requirements. Researchers also point out that increased algorithm complexity needs up to 10 times the computing power; thus, this means that the H.264 codec requires high energy consumption compared to HEVC.

**3.3.3 VP8 Format**

VP8 video is an open-source, royalty-free video compression standard developed by On2 Technologies to succeed over VP7 and is owned by Google. VP8 standard only supports progressive scan video signals. It also supports multi-core processors with up to 64 cores simultaneously. Since VP8 only needs three reference buffers, its decoder implementations require a relatively small memory. Just like H.264, VP8 also has a pure intra mode. In terms of deblocking filters, VP8 offers two different adjustable deblocking filters that can be integrated into codec loops. VP8's macroblocks can be 4x4, 8x8, or a maximum of 16x16. Motion vectors are characterized by quarter-pixel precision [15].

When comparing VP8 to H.264, H.264 offers a slightly higher quality than VP8. However, VP8 offers the highest quality real-time video delivery but uses maximum CPU resources to keep the encoding speed to be almost the same as playback speed.

**3.3.4 VP9 Format**

VP9 is a video codec technology developed by Google. It is an open-source technology and free from royalty fees. It is most commonly used video codec technology for streaming videos over the Internet due to the fact that it reduces the bit-rate of video transmissions by 50% while maintaining a high quality. VP9 codec was previously referred as "NGOV" (Next Generation Open Video). VP9 codec, which is an improvement over the VP8 codec, is also supported by Chrome, YouTube and Netflix.

The VP9 coded which supports parallel processing, functions in a similar way with the H.264 codec. However, VP9 is capable of reducing he bit rate to half of the original size without having an effect on the video quality. With this feature, using VP9 is very effective for streaming videos for low-end devices such as smartphones. VP9 can compress video files and streams at 4K resolution. The VP9 uses a 64x64 superblock. The 64x64 superblock is further subdivided into smaller blocks which facilitate effective video compression. VP9 supports four main transform sizes: 4x4, 8x8, 16x16, and 32x32. Each of the VP9 codec codes frame into three sections which are: uncompressed header, compressed header and compressed frame data.

In term of performance, Martin Rerabek and Touradj Ebrahimi in their work found that VP9's coding efficiency was inferior to H.265 with an average bit-rate overhead of 32.5% at the same objective quality [21]. The subjective scores also showed that HEVC had an average bit-rate reduction by 52.6% when compared to AVC and 49.4% when compared to VP9. This simply indicated that HEVC has better quality than VP9 [21]. In [22], VP9 and H.264 files were downloaded from YouTube. A comparative analysis of these two codecs showed that YouTube encoded VP9-based 1080p streamed at a 43% data rate lower than H.264 and for 720p, VP9 streamed about 35% lower than H.264.

**3.3.5 AV1 Format**

This is the latest and considered as the most advanced video compression standard that is open-source and copyright-free. Researchers suggest that AV1 can improve coding efficiency by up to 25% when compared with Google VP9 and H.265 [8]. The main goal for developing AV1 was to considerably improve the compression rate based on the current codec so as to ensure that the decoding complexity matched the practical feasibility of the hardware.

From a variety of traditional experimental formats, AV1 selects novel techniques to provide better resolution in low-contrast regions. Its development mainly focuses on the key five aspects:

1. Consistent delivery of high-quality videos at real time
2. Ability to scale down modem devices at various bandwidths
3. Establishing a computable footprint
4. Provide hardware optimization
5. Provide flexibility for both commercial and non-commercial content

Due to the fact that AV1 is a relatively new standardized video encoding format, there are few works of literature exploring it. This means that conducting further research in AV1 could offer more insight into video encoding research [11, 18]. Therefore, with the main aim of this project being on energy consumption and memory efficiency, we will analyze AV1 format in different platforms and then evaluate the results directly to compare the results of HEVC and VP8 formats.

**3.4 Machine Learning**

We will intrude three major machine learning based algorithms in this research, namely KNN (K-Nearest Neighbors), SVM (Support Vector Machine) and RF (Random Forest).

**3.4.1 KNN**

Proposed in 1967, KNN is one of the simplest classifiers [23]. KNN believes that the points with the same label should be clustered together in the feature space, so we can predict the new candidate point’s label by finding its K nearest neighbors at first, then counting the number of the points with the same label, and the label owns the most of votes will be the label for the new candidate point. The advantage of KNN is that there is not training procedure because it just stores all the training <label, feature> pairs and waits to vote during the prediction procedure. However, this means there will be lots of computation in prediction procedure because every new candidate should calculate the distance with all training points and then find the K nearest neighbors from the training set. But what is the exact value of K? Too large K makes it more expensive in prediction procedure while too small K makes the model sensitive to the noise from the other classes. As a result, the nearest neighbors number K should be finetuned during the training process. The other problem of KNN is that it may predict wrong answer because of the sampling imbalance [24]. For example, if class A takes account of 10% of training samples while class B takes account only 1%, which means for a new candidate, there will be 10x probability to be predicted as A class rather than B class on average.

**3.4.2 SVM**

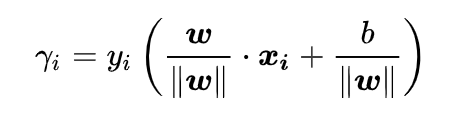
As a state-of-the-art machine learning based method, the SVM tries to find the best hyper-plane that can sperate classes in the feature space [25]. But what is the ‘best’ hyperplane? Of all the distances from the feature instances to the hyperplane, we can always find the minimum distance. The best hyperplane should maximize this minimum distance. It is the spirit of SVM. We will begin with a binary classification.



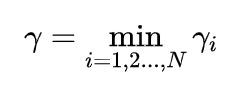
Fig.1 Visualization of binary SVM

As shown in Fig.1, x-axis represents the feature while y-axis is the label. The hyperplane can be represented as wx+b = 0, where w and b are parameters to be trained. The classification decision function can be expressed as f(x) = sign(wx + b).

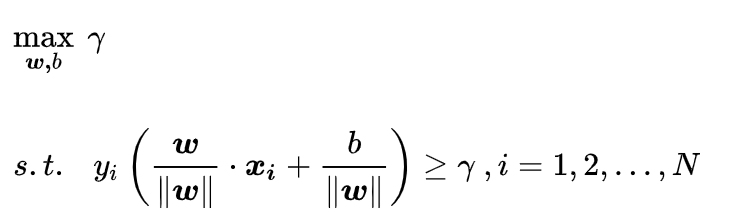
The distance from an instance (xi, yi) to the hyperplane is:



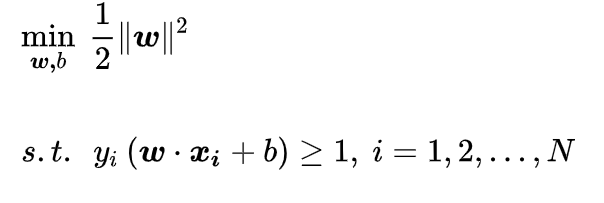
The minimum distance is:



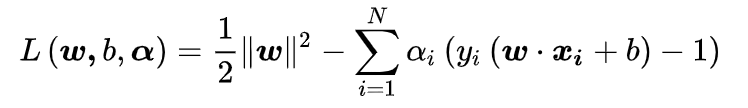
The target of the SVM model is to maximize the minimum distance γ, hence we can get the following optimization problem:



From the pre-knowledge of geometry, if we want to find the maximum value of γ, we should firstly find the maximum value of 1/||w||, which is equivalent to search to minimum value of ½\*||w||2, hence we can get the following primal problem.



As a convex problem with constrains, there is a dual problem with Lagrange multiplier method. The dual problem should be:

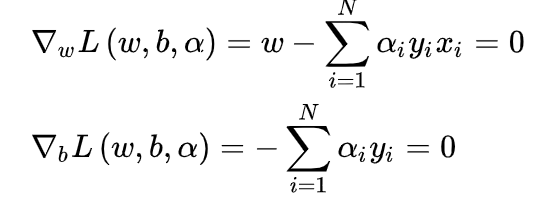


The dual problem can be solved by 2 steps:

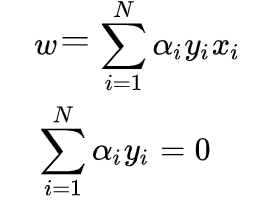
1. Solving the problem



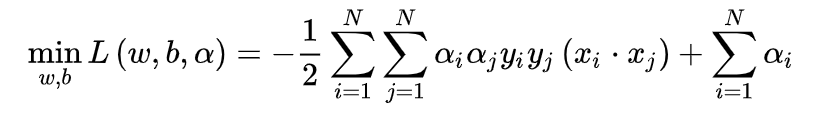
The problem can be solved by:



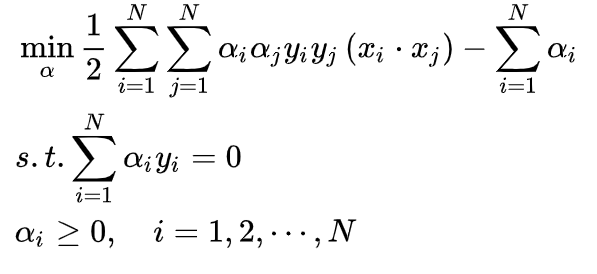
Finally, we get:



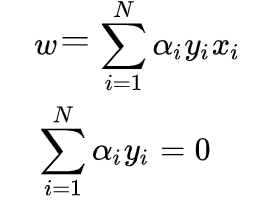
Hence, the dual problem can be simplified as:



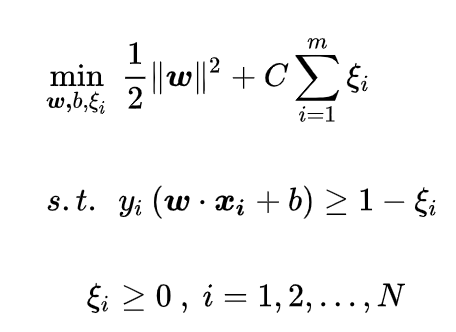
1. Solving the problem:



α is called Lagrange multiplier, which can be solved by some efficient method, such as SMO [26]. Finally, the w and b in primal problem can be calculated with α based on:

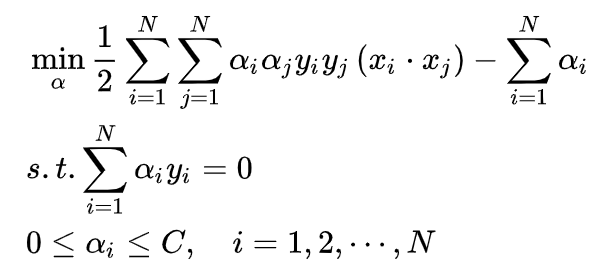


The SVM model discussed above is an ideal one. In other words, the model assumes that the training labels can be fully separated by the hyperplane, and there is not penalty for misclassification. However, in the real-world dataset, there are always outliers that are mis-classed. To process problem of this kind, there must be penalty for these outliers. Consequently, the soft margin is introduced as following [27]:

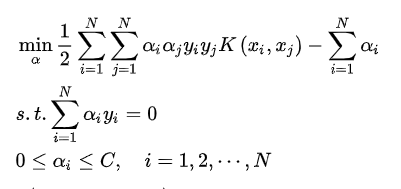


C is called penalty parameters and C > 0, while Ԑi is called soft margin variable and Ԑi > 0.

The soft margin problem can be also solved by dual problem.



Until now, all the discussion is about the SVM linear model, which largely limits the power of SVM for linear classifier is too simple to sperate the class labels. Thus, SVM has adopted so-called kernel function in the dual problem [28].



*K* is called kernel and can map the dual problem from linear space to high-dimensional space. Consequently, most of problem that cannot be classified in the low-dimensional linear space can be solved in the kernel space. There are several kernel functions, such as nonlinear, sigmoid, polynomial kernel and RBF, we will try some of these kernels in this work.

The solution of multi-classification problem is based on binary-classification. Specifically, now that the multi-classification can be divided into several binary-classification problems, we can train multi-binary SVM classifiers and ensemble them as a multi-SVM classifier.

Based on the theory of SVM, there are 3 important parameters that should be finetuned during the training process, namely the penalty factor C, the kernel type, and gamma factor.

**3.4.3 Random Forest**

Random Forest, which was proposed by Breiman in 2001 [29], is another modern machine learning algorithm. The spirit of random forest is to ensemble different decision trees together to vote for the final label. Specifically, RF will randomly choose different samples with different features to train hundreds to thousands of simple decision trees during the training procedure, respectively. Then during the prediction procedure, the input instance will be predicted by all the decision trees, and all predictions vote for the finally result. But what is the optimal number of these decision trees and what is the best depth for each tree? Too many decision trees or too large depth of each tree makes it easy to be overfitting, which means low training error but high testing error, while the learning ability is not enough if too few decision trees or too small depth of each tree [30]. As a result, the number of the decision trees and the max depth of each decision tree are two important parameters that should be finetuned during the training procedure.

Random forest has been reported to achieve better performance for different datasets and tasks [31]. One of the major reasons is that RF adopts a very powerful feature selection algorithm, which makes it can find the importance of the features. The other reason is that the random selection for samples and features makes it general to overcome the notorious overfitting problem, and the decision trees can also suppress the performance degradation caused by the imbalance training samples, which is one of the major problems for KNN and SVM.

1. **Methodology**

The main aim of this project is to develop a useful tool that can be adapted to monitor the performance of different video encoding formats. Specifically, the tool is used for testing and analyzing the highlighted video encoding formats with the goal of identifying the most memory efficient and least energy consumption.



Fig.2 The system architecture

The system Architecture is shown in Fig.1, from which the whole procedure can be split into 2 phases, namely training phase and prediction phase.

In the training phase, firstly plenty of high-quality videos are collected form some online video dataset, such as YouTube UGC dataset [32], which is a public video dataset for compression and quality assessment research. As shown in Table I, the dataset contains 1357 of videos with different categories and resolution, and each video has a 20 seconds duration.

Table I, The videos distribution of the YouTube UGC dataset

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Category | 360P | 480P | 720P | 1080P | 2160P | SUM |
| Animation | 14 | 21 | 23 | 23 | 2 | 83 |
| CoverSong | 18 | 21 | 20 | 22 | 0 | 81 |
| Gaming | 23 | 28 | 24 | 40 | 24 | 139 |
| HDR | 0 | 0 | 0 | 25 | 27 | 52 |
| HowTo | 22 | 23 | 22 | 17 | 0 | 84 |
| Lecture | 24 | 28 | 26 | 24 | 0 | 102 |
| LiveMusic | 17 | 16 | 14 | 21 | 0 | 68 |
| Lyrics | 0 | 1 | 0 | 2 | 0 | 3 |
| LyricVideo | 6 | 13 | 18 | 15 | 0 | 52 |
| MusicVideo | 13 | 19 | 17 | 26 | 0 | 75 |
| NewsClip | 24 | 27 | 28 | 17 | 0 | 96 |
| Sports | 26 | 31 | 33 | 33 | 33 | 156 |
| TelevisionClip | 6 | 17 | 11 | 18 | 0 | 52 |
| VerticalVideo | 20 | 14 | 20 | 20 | 1 | 75 |
| Vlog | 18 | 25 | 24 | 37 | 47 | 151 |
| VR | 0 | 0 | 18 | 36 | 34 | 88 |
| SUM | 231 | 284 | 298 | 376 | 168 | 1357 |

Then there will be a feature extraction module that extract features for each video. The features contain original video’s size, category, width and height, fps, etc.

On the other hand, to measure the energy consumption, we have designed a simulation flow to mimic the video encoding, transmission, and decoding & playing steps in the real Internet. Specifically, in the encoding module, the videos are encoded by one of four encoding formats using FFMPEG toolkit [33], which has a powerful video encoding tool that offers cross-platform solutions capable of recording, converting, and streaming both audio and video files. The encoding formats contain H264, HEVC, VP8 and VP9. Of note, AV1 costs too much time to be measured through the whole dataset videos, so we exclude it from the experimental list. After being encoded, the videos will be transmitted by internet cable to the requested terminal. It is hard to measure the internet transmission energy consumption because there are so many devices and cables that take part in the internet transmission. As a result, we decide to use a deduced parameter that is related to the encoded file’s size to estimate the transmission energy shown in Eq.1. The decoding and play step can be simulated by a VLC Media Player. There is a power monitor device to effectively monitor and evaluate the energy consumption during encoding and decoding & play steps.

Transmission Energy = 0.05 kwh/GB \* Encoded File Size (Eq. 1)

In the real internet environment, the videos are always encoded once but maybe requested up to billions of times, hence the encoded video will be transmitted and played multi times. Consequently, our design takes the estimation of the multi-requests’ scenario into consideration. Specifically, we will measure the energy consumption of encoding, transmission and decoding & playing once for each video, but estimate the total multi-requests energy consumption as Eq. 2.

Total Energy = Encode Energy + requests\*(Transmission Energy + Decode & Play Energy) (Eq. 2)

Now that the motivation of this research is to find the best encoding format with the least energy consumption and the most memory efficiency, we need a criterion that can measure the two metrics. In term of energy consumption, we can estimate it from Eq. 1 and Eq. 2, but how to measure the memory efficiency? Considering the mainly memory consumption in the whole simulation consists of the original video memory and encoded video memory consumption, and the original video memory consumption is the same for each encoding format, the main difference of memory consumption comes from the encoded video file, which should be taken into consideration. Now let us look back to our Eq. 1 and Eq. 2, it is the encoded file size that domains the transmission energy consumption. As a result, we can say that the criteria expressed by Eq. 2 will be suitable to choose the encoding format with the best energy and memory efficiency. In the simulation, for each specific video and requests, the total energy consumption of four encoding formats will be estimated by Eq. 1 and Eq. 2, and the encoding format with the smallest energy consumption will be chosen as the best encoding format. Once the best encoding format is found, we will ensemble the best encoding format and features from feature extraction module as a <label, features> pair and pass the pair to the next machine learning module.

In the machine learning module, three of state-of-the-art machine learning algorithms are implemented and compared, i.e., K-nearest neighbors, support vector machine and random forest. We will elaborate more on the flowing chapters.

After training phase, we will get a prediction model, which can be used to predict the best encoding format using the features that are extracted from the user input video files.

1. **Experiments**
   1. **Experiment setting up**

The project needs a software program that can allow the highlighted video encoding formats to run automatically without problems. Therefore, Python programming language appears to be the most effective programming language for developing this software. We also propose to combine FFMPEG and VLC with the automated program to finish the encoding and decoding & play steps. The newly developed software program also combines with generic tools used for monitoring and analyzing software energy consumption. This is important to evaluate the energy consumption and memory efficiency of the video encoding formats.

The software program is executed on a Linux (Ubuntu 18.04.1) server, which owns Intel (R) Core (TM) i9-7900X CPU @ 3.30 GHz with 20 NUMA nodes CPUs, 32KB L1 Cache, 1024 KB L2 Cache, 14080 KB L3 Cache. The energy monitoring device is connected to the server and measures the total energy consumption of the system every three seconds.

* 1. **Energy consumption analysis**

After several days’ execution, we have evaluated all 1357 videos from Youtube UGC dataset. The flowing chapters will describe the experiment results. We will firstly elaborate on how to measure the energy consumption for each encoding, transmission, and decoding steps, and then a thorough analysis will be presented. Finally, we will discuss the results from training and prediction phases with different machine learning methods.

* + 1. **Baseline energy consumption**

As mentioned before, the energy monitoring device will measure the power consumption of the sever every three seconds. From the perspective of the server, there are two kinds of working states, namely Idle and Busy. When the server is in Idle state, there is not any customized task being executed, thus the energy consumption comes only from the basic operations to support the system ready. But when the server is turned to Busy state, there are some other customized tasks besides the basic operations, such as video decoding, encoding & play, thus the energy consumption comes from both basic operations and customized tasks.

The Fig. 3 shows the monitored power versus time when the server is on Idle and Busy state, respectively. From Fig. 3 (a), the energy consumption is steady low for Idle state, while when the server is on the Busy state, the energy station will be boosted in Fig. 3 (b). As a result, if we want to measure the energy consumption of the customized tasks, we should sperate the basic energy consumption at first, in other words, we should substrate the basic power value from the Busy state’s measuring power. But how to measure the basic power value? In Fig.4, we have presented the frequency distribution over 24 hours measuring when server is on the Idle state. Obviously, the Idle power is distributed with a normal random style, thus its average value, 81.03 W, can be used as the basic power value. The Fig. 3 (b) shows the power consumption during encoding, decoding & play steps when the four encoding formats are implemented to a specific video file one by one. The interesting observations from Fig. 3 (b) are that

1. there are different power consumption and time duration for different encoding format, hence different energy consumption.
2. (ii) The power consumption for decoding & play is steady and small for different encoding formats.

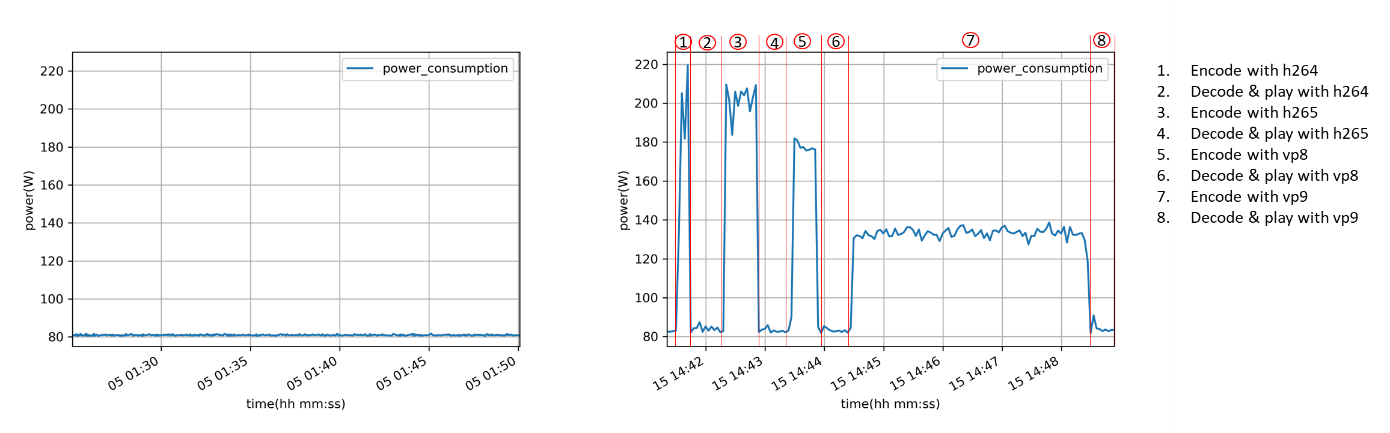


Fig.3 (a) power when server is on Idle state, (b) power when server is on Busy state

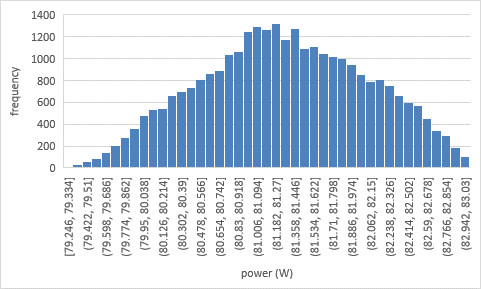
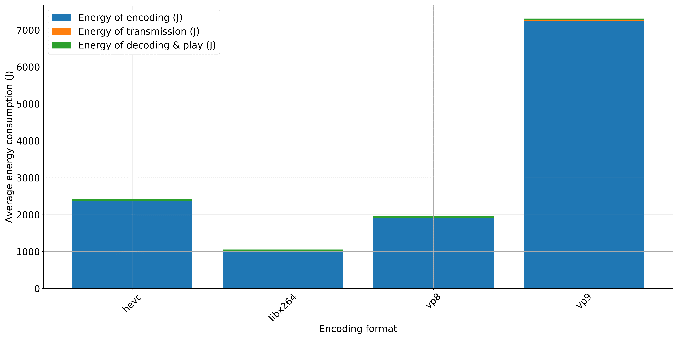
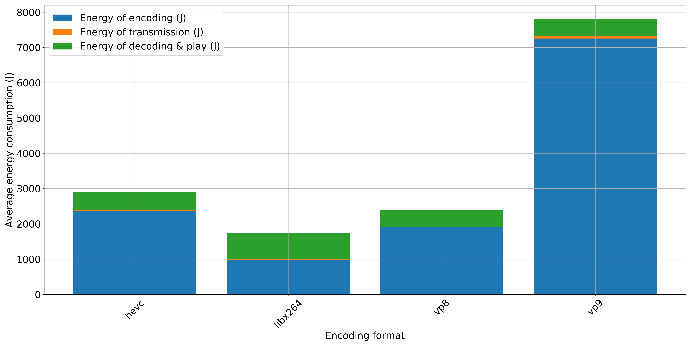


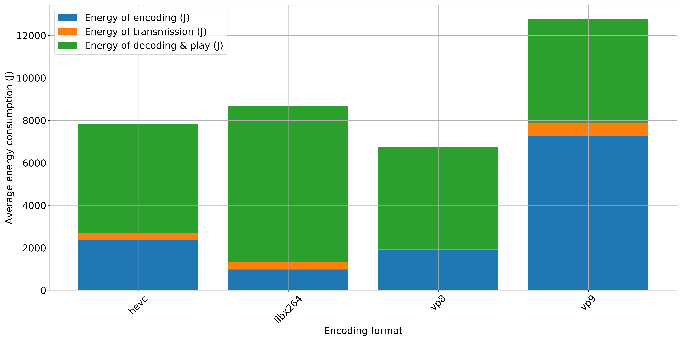
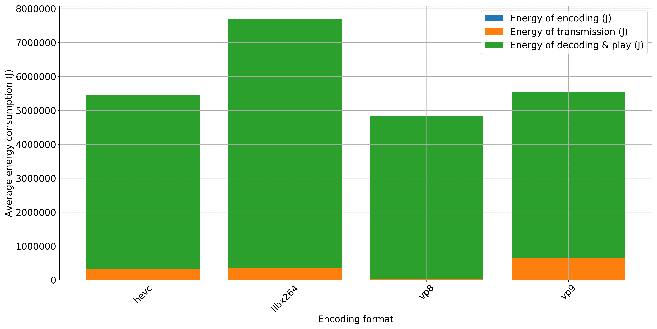
Fig.4 The power distribution when server is on the Idle state

**5.1.3 Energy consumption versus requests**

In the real internet environment, a video maybe requested from several times or billions of times. For example, the video transferred in Whatsapp from friend by friend is only requested once and correspondingly the encoding, transmission, decoding and play steps are executed only once. While for the videos in Youtube, the request pattern is one encoding, multi (or even billions) times transmission and decoding & play. To have a thorough understand of the different scenarios, we have designed the multi-requests simulation. The breakdown of average energy consumption with different requests has been shown in Fig.5.

1. requests = 1 (b) requests = 10

(c) requests = 100 (d) requests = 100000

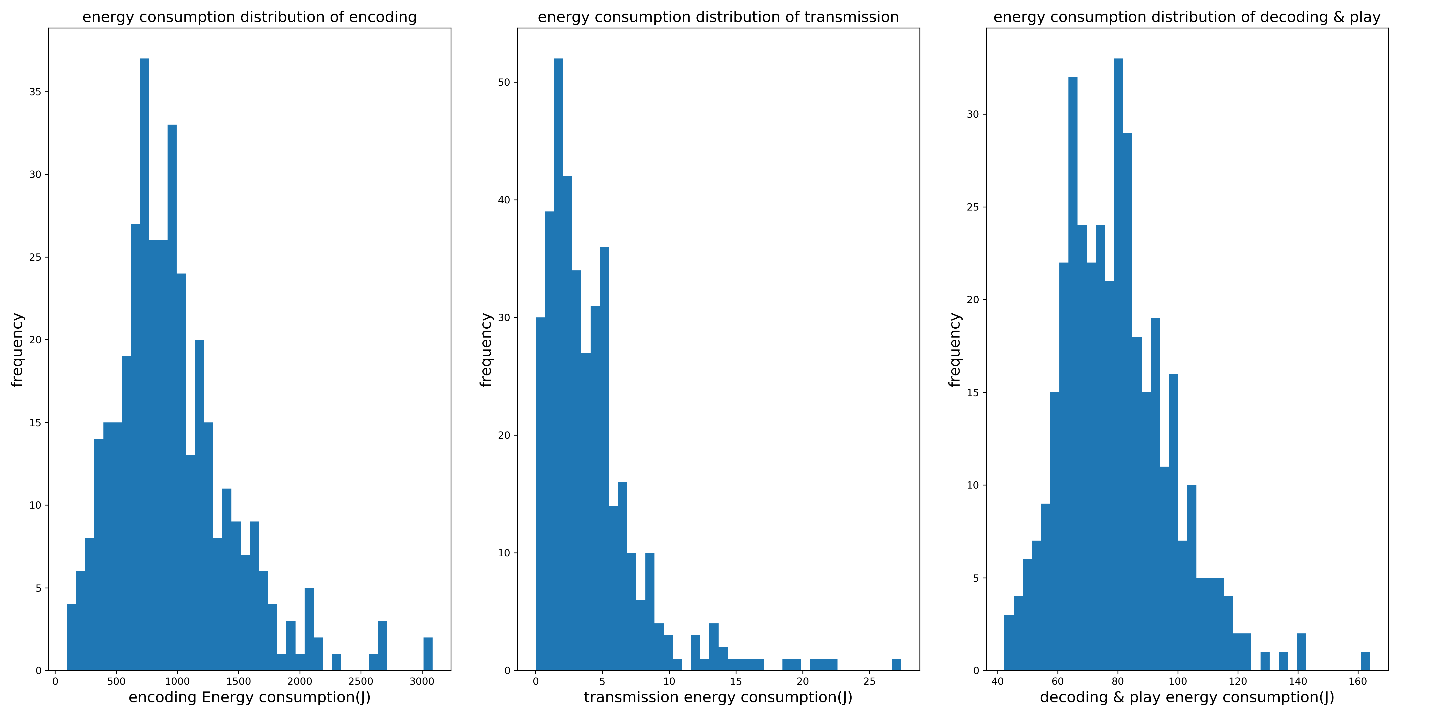
Fig.5 The breakdown of the average energy consumption with different requests

From Fig.5, we can get following interesting observations:

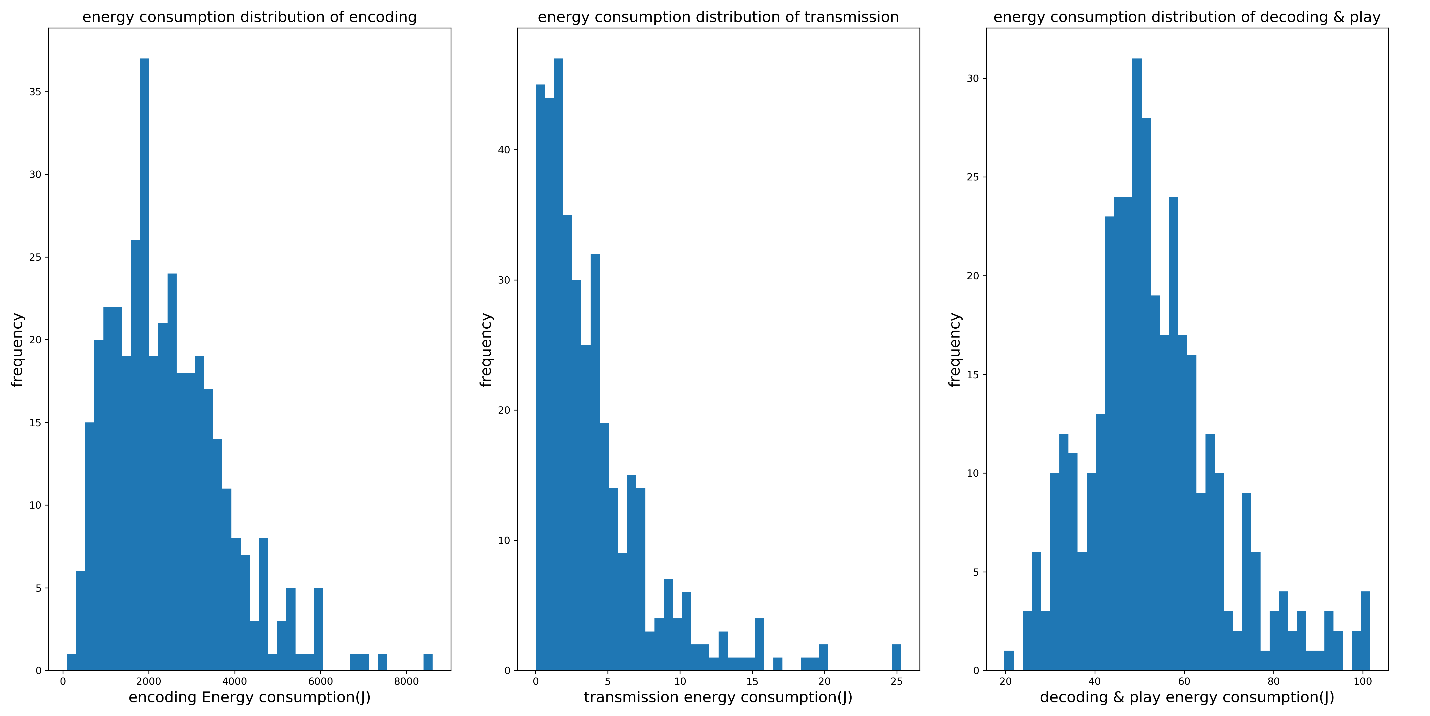
1. When the requests are less than 10, the encoding energy consumption domains the total energy consumption, and the most energy efficient encoding format is libx264 (H264) because it costs the least energy during the encoding step, then the VP8, HEVC and VP9 are the second, third and fourth most energy efficient encoding formats, respectively.
2. As the requests increase, transmission and decoding & play steps will consume more and more energy. Finally, as the requests go up to 100000, the energy consumption from encoding can be neglected compared with decoding & play energy consumption. Thus, the most energy efficient encoding format is VP8 because it consumes less during transmission and decoding & play steps, and HEVC, VP9, libx264 are second, third, fourth energy efficient encoding formats, respectively.

Fig.6 has presented the energy consumption distribution for different encoding formats when requests = 1 and video resolution = 1080P. The results show again that libx264 has the least encoding energy consumption but most decoding & play energy consumption. On the other hand, VP8 consumes least during transmission and decoding & play steps.

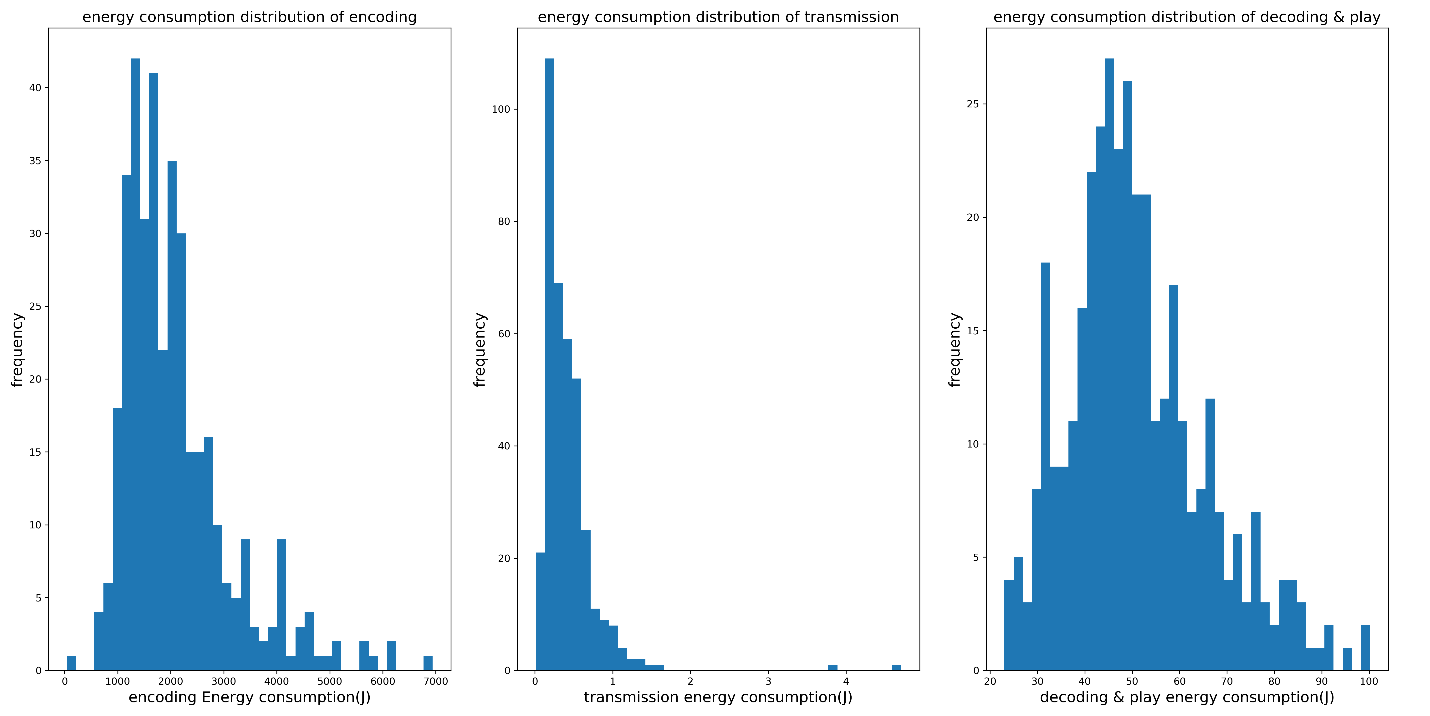
Above all, there is not a most energy efficient encoding format for all requests. In other words, we should choose different encoding formats for different applications. For example, in Whatsapp, there are usually friend-to-friend video requests, libx264 is the best of the four encoding formats because it consumes less. However, for some public social video platform, such as Youtube and Facebook, there are usually billions of requests for a hot video, vp8 will help to save more energy.



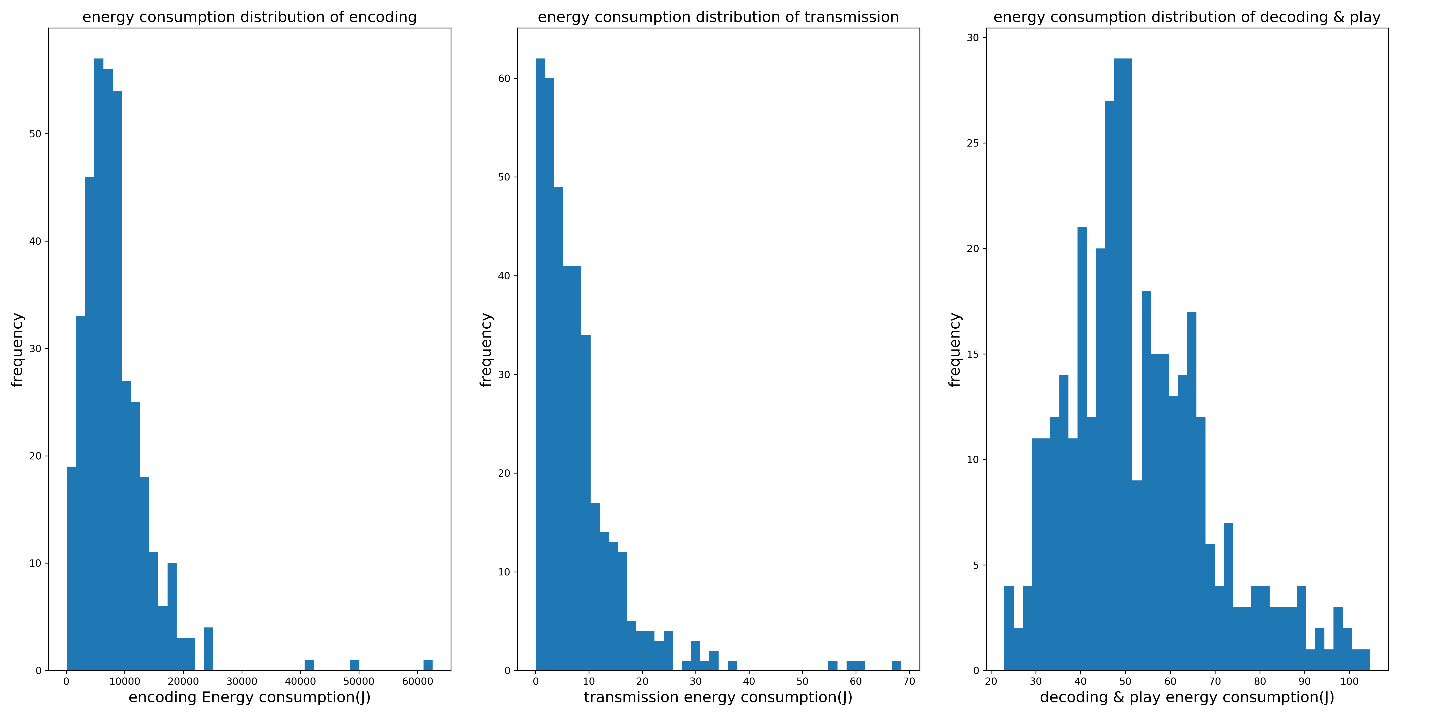
1. libx264



1. HEVC



1. VP8

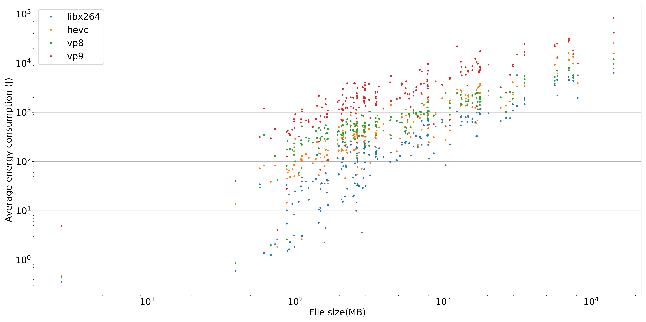
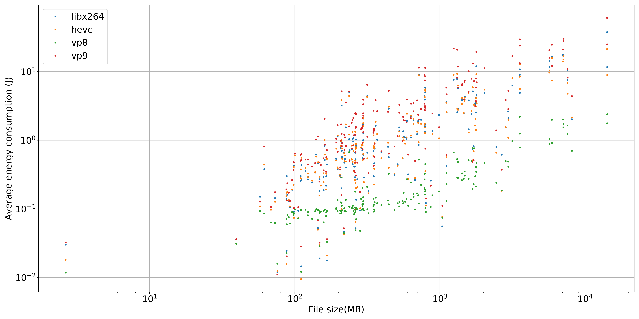


1. VP9

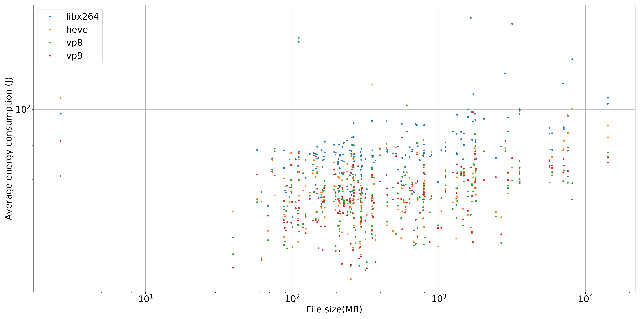
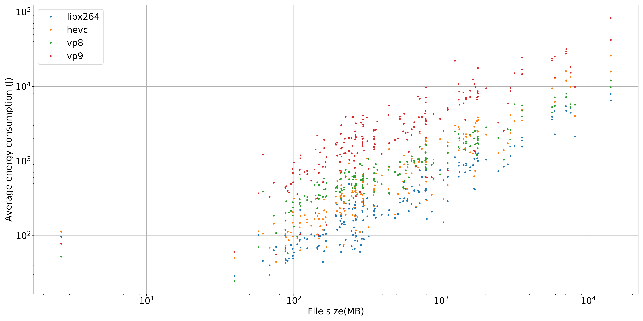
Fig. 6, the energy consumption for different encoding formats with requests = 1 and resolution = 1080P

5.1.4 **Energy consumption versus file size**

The size of the original video file is believed to be one of the major factors that have large influence on the energy consumption. It is obvious that the bigger the file size, the more time will be spent on encoding, transmission, and decoding & play, and thus more energy consumption. Fig.7 has visualized the encoding, transmission, decoding & play, and total energy consumption for different encoding formats when requests = 1

(a) encoding energy versus file size (b) transmission energy versus file size

(c) decoding & play energy versus file size (d) total energy versus file size

Fig.7 The energy consumption versus file size when requests = 1.

From Fig.7, we can find that

1. There is a linear relationship between original video file size and energy consumption for all kinds of energy consumption. On the other hand, the original video file size changes linearly with the video’s duration if frame per second is stable, so we can conduct that the energy consumption has a linear relationship with video’s duration. In other words, even we can only calculate and compare the energy behavior of different encoding formats with the same time duration videos (20s) in the experiment, the conclusions are also the same for videos with longer or shorter time duration.
2. For encoding step, the libx264 trends to consumes less energy no matter what the size of the video file, while VP9 costs the most.
3. For transmission step, VP8 trends to consumes least while VP9 consumes most. It is because that the VP8 has larger compression ratio, in other words, VP8 can compress more redundant information of the video and thus transmit less than the other encoding formats. We will discuss this in the following chapter.
4. For decoding & play step, it is obvious that libx264 will costs most, but it is hard to say which one of the other three cost least from Fig.7 (c).
5. Now that the encoding step will domain the total energy consumption, it is not strange that the libx264 has the least energy consumption when requests=1, but it will cost most when requests increases to 100000, as shown in Fig.8.

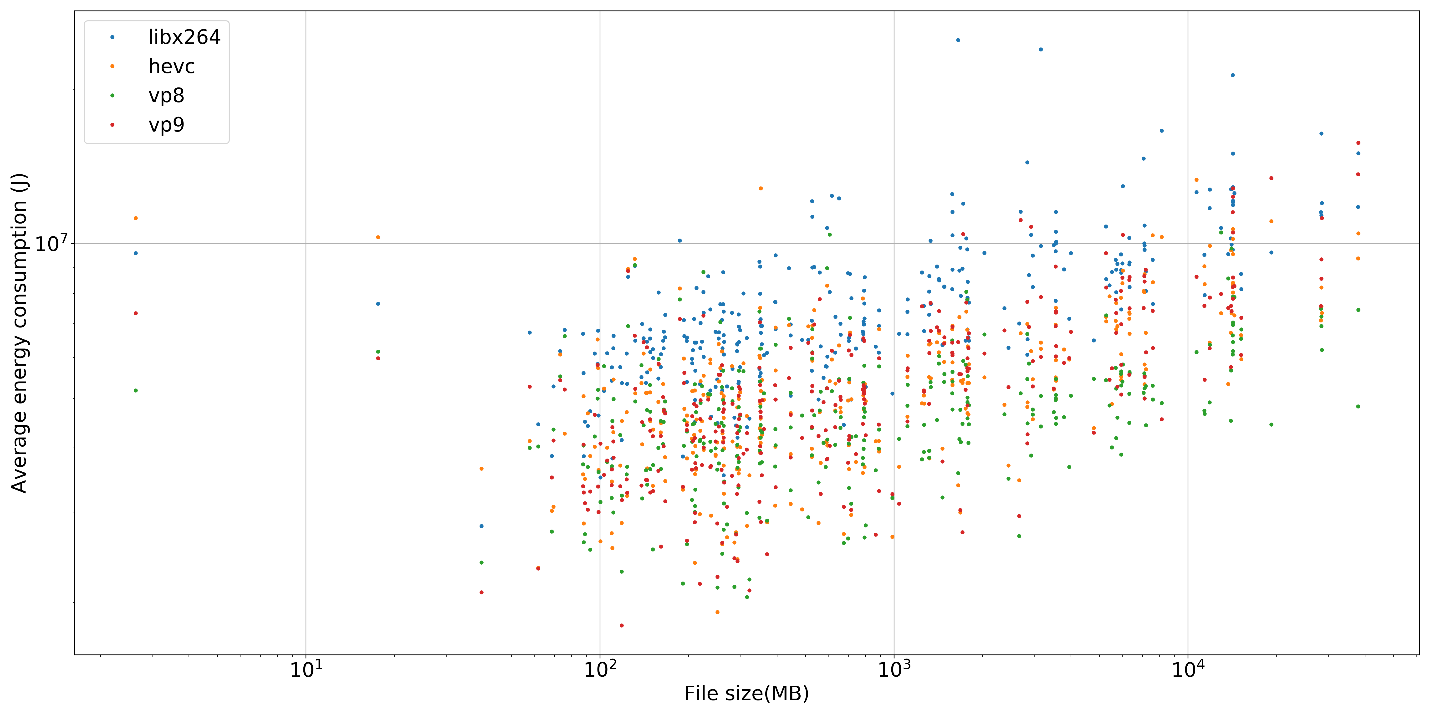
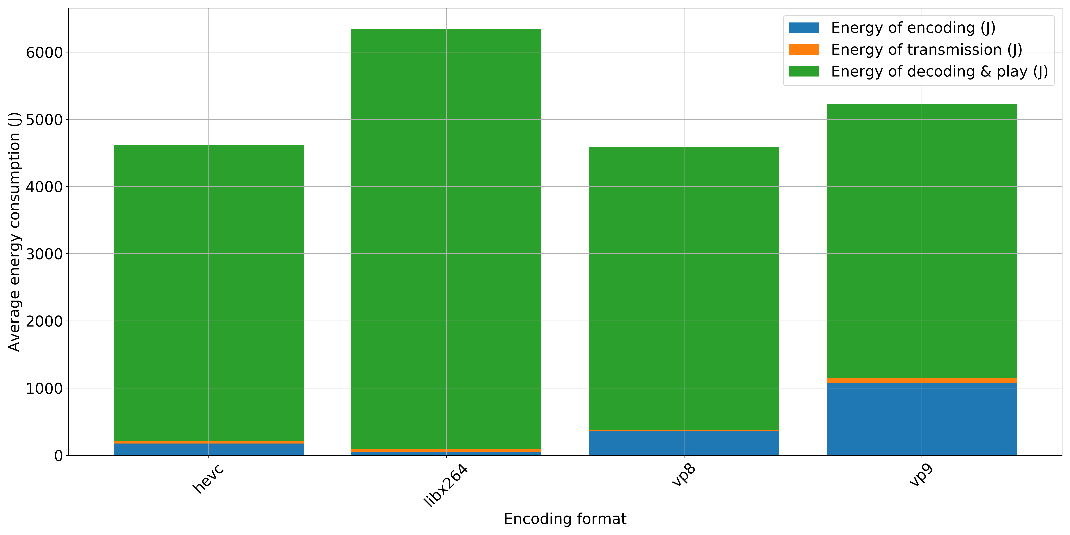


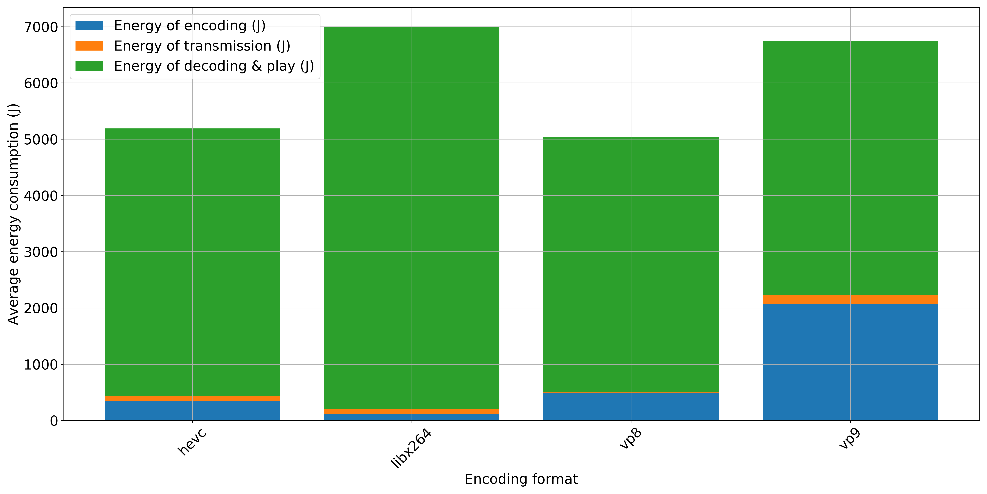
Fig.8 The total energy consumption versus file size when requests = 100000

**5.1.5 Energy consumption versus resolution**

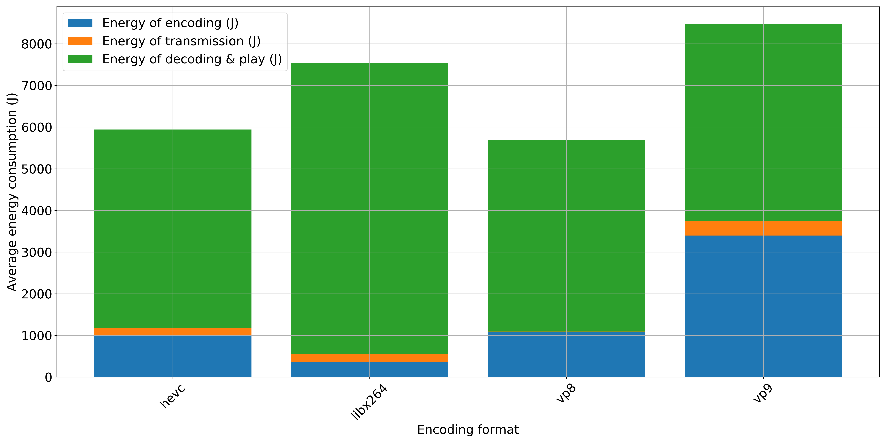
Resolution is a very import feature for video quality. The higher resolution is, the higher quality of the video is, but the original file size will be larger correspondingly, hence more energy will be consumed during encoding, transmission, and decoding & play steps. In this chapter, we want to explore the influence of the resolution for energy consumption.



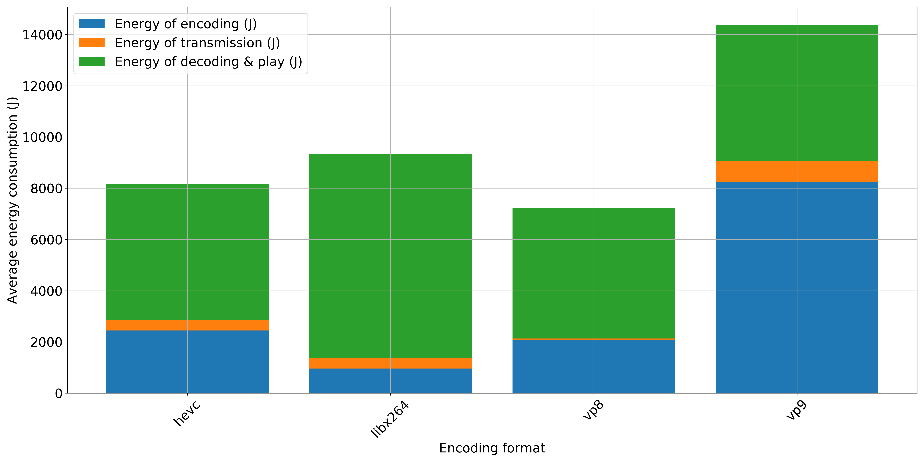
1. 360P



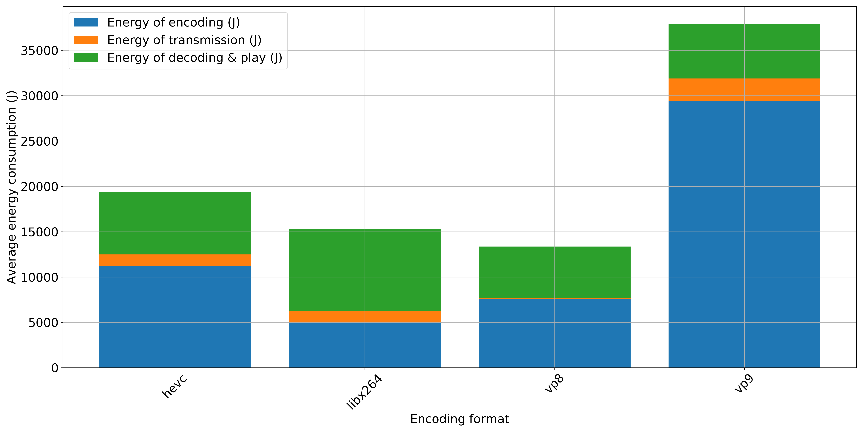
1. 480P



1. 720P



1. 1080P



1. 2160P

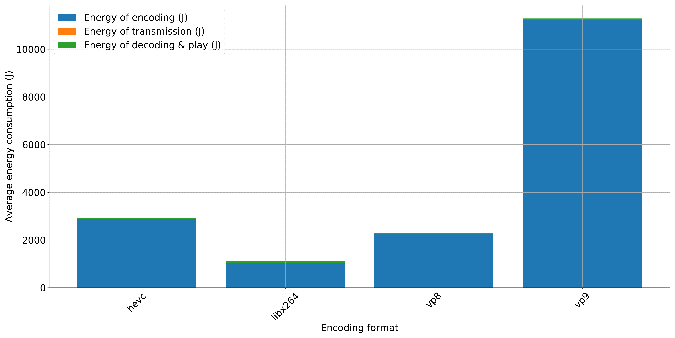
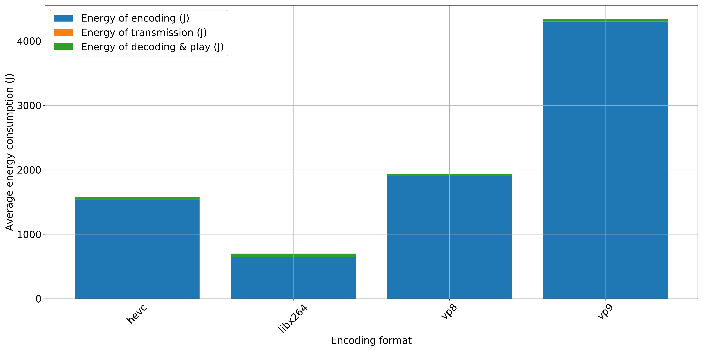
Fig. 9 Energy consumption for different resolutions when requests = 1

We can get following observations from Fig.9.

1. The total energy consumption will increase as the resolution.
2. Libx264 encoding format consumes least of encoding energy but most of decoding & play energy for all kinds of resolution.
3. VP8 encoding format consumes least of transmission energy for all resolutions.
4. VP9 encoding format consumes least of decoding & play but most of transmission energy for most of the resolutions.

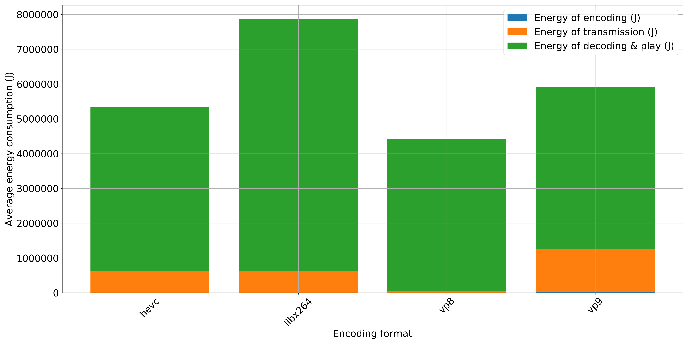
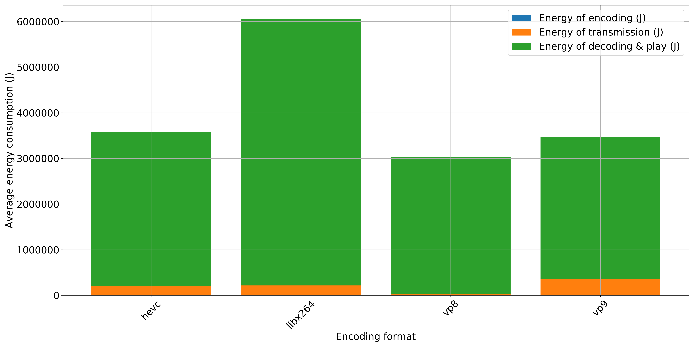
**5.1.6 Energy consumption versus category**

According to the reference [33], the category of the video may have influence on its energy consumption. We have compared the energy consumption of the videos from category ‘Sports’ and ‘Lyrics video’, of which the former is believed to have big difference among frames, while the latter changes slowly among frames. To make a fair comparison, we have chosen only the 1080P resolution videos from the two categories and the average energy consumption for different encoding formats are shown in Fig.10 and Fig.11.

(a) sports (b) Lyrics video

Fig.10 the energy comsumption of 1080P videos from different categories when requests = 1

(a) sports (b) Lyrics video

Fig.11 the energy comsumption of 1080P videos from different categories when requests = 100000.

From Fig.10 and Fig.11, we can get following observations.

1. During encoding steps in Fig. 10, libx264 is the best encoding formats for both categories, but the sports video will consume more than the lyrics video. It is because there is big difference among frames in the sports videos, so it takes more time to encode the video.
2. During the transmission step in Fig.11, it is obvious that the sports videos will consumes more than lyrics videos. It is still because the complexity of the sports video will be higher than lyrics, so more memory is needed to compress the original video, or less compression ratio for the sport videos.
3. The energy consumption in encoding & play step is also different for the two categories. For sports video, it is VP8 consumes less while VP9 consumes less for lyrics videos.

Above all, categories, which reflect the complexity of the videos and have a direct influence on encoding, transmission, and decoding & play steps, should be taken into consideration together with requests and resolution when choosing the best encoding format.

In summary, from the perspective of total energy consumption, there is not a general best encoding format. On the other hand, we should take some key factors into consideration, such as requests, file size, resolution, and video complexity. Libx264 encoding format has the best energy performance for encoding domain applications, and VP8 is suitable for transmission domain application while VP9 saves most of the energy for decoding & play domain applications.

**5.2 Memory efficiency**

The other aim of this research is to explore the memory efficiency for different encoding formats, which can be represented by compression ratio during encoding steps because the compressed video file size differs for different encoding formats. As shown in Fig. 12, VP8 has the least compression ratio, especially when the original video size is larger than hundreds of MB. While VP9 has the largest compression ratio, which means more memory space is needed to store the compressed file. Now that the transmission energy consumption is always related to the compressed file size, it is not strange the VP8 saves most of energy for transmission step while VP9 costs most.

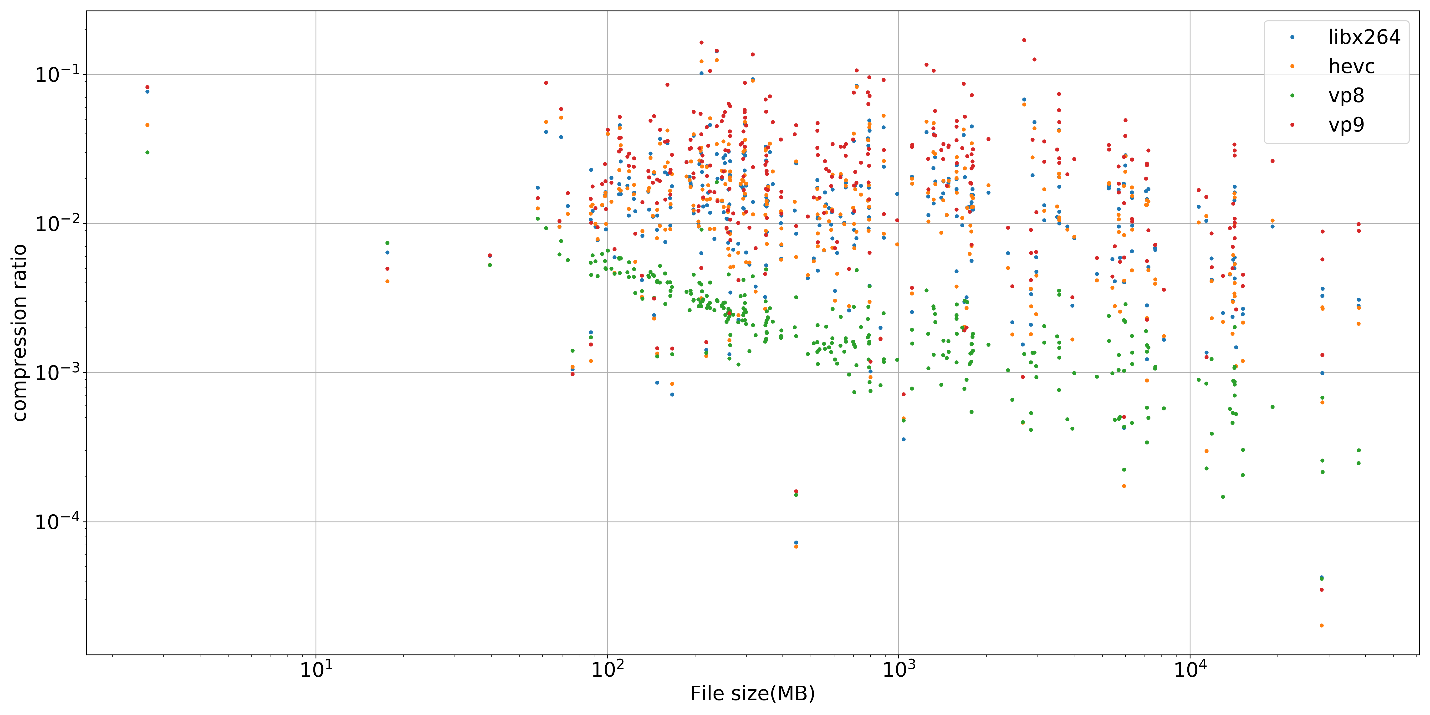


Fig. 12 the compression ratio versus original video file size

**5.3 Encoding elapsed time**

The elapsed time is another import factor that affects not only the energy consumption but also user experience. From Fig.3 (b), the elapsed time of decoding & play is fixed as video duration is always 20 s and the elapsed time of transmission is not taken into consideration in this research, so the elapsed time of encoding, which differs from different encoding formats, is necessary to be explored. Actually at the beginning of the exploration, we have tried to encode the video using AV1 encoding format, but it costs tens of hours to encode a video, which makes it impossible to encode all videos in the dataset within several months. It is a very extreme example that the encoding elapsed time will change the exploration behavior, and in the real internet environment, especially for those applications that need plenty of friend-to-friend video transmission, so long encoding time is not tolerable for either energy consumption or user experience. Fig.13 has shown the relationship between encoding elapsed time versus file size for different encoding formats. It is obvious:

1. There is a positive linear relationship between video file size and average elapsed time for all encoding formats.
2. For different original file size, libx264 has the least encoding elapsed time while VP9 consumes most of the time. VP8 and HEVC are the second and third least encoding elapsed time format, respectively.

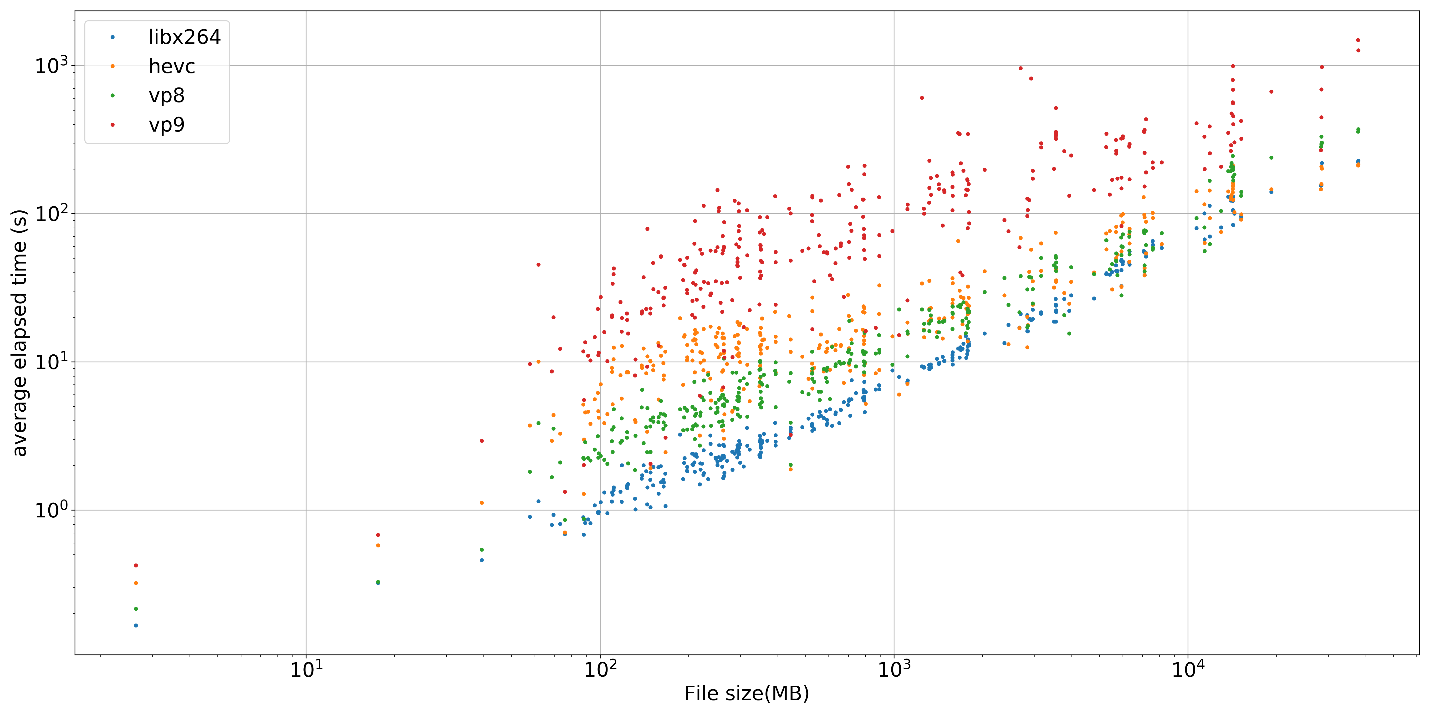
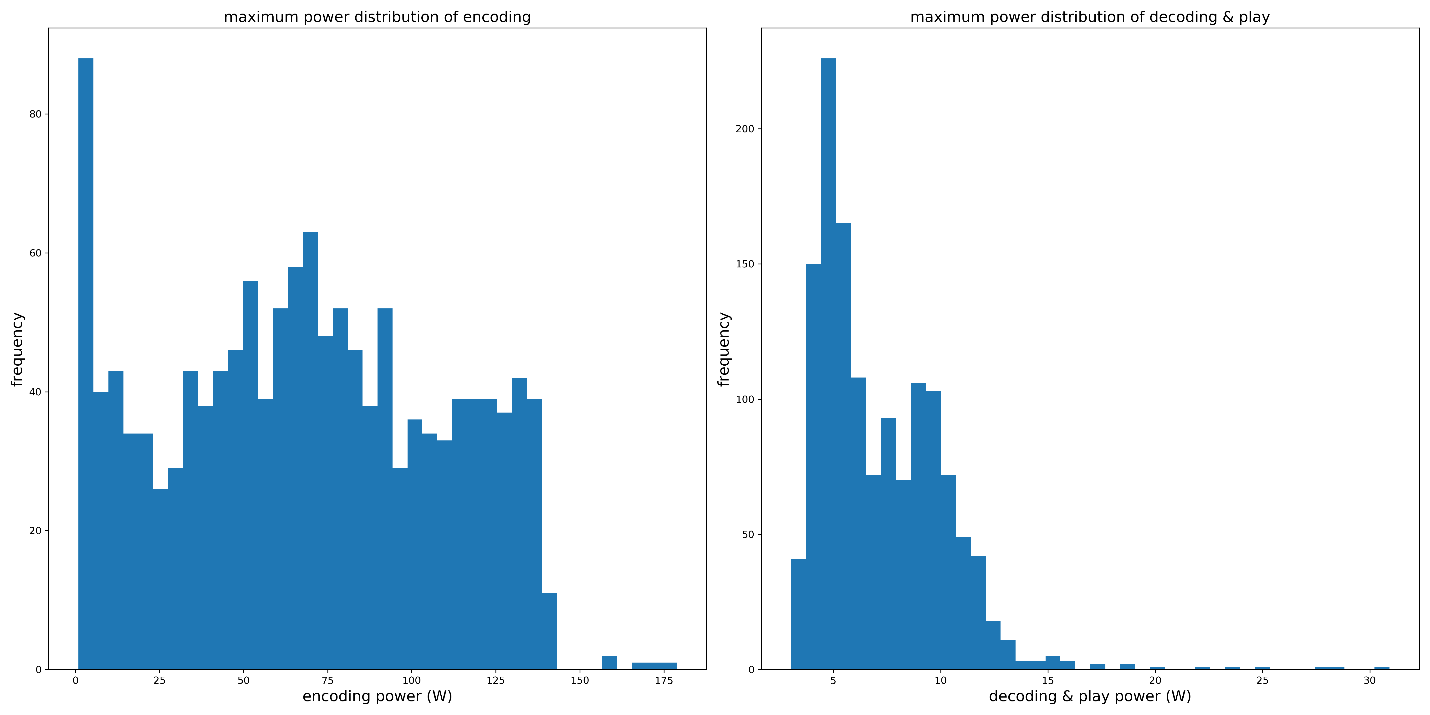


Fig.13 the average encoding elapsed time versus original video file size

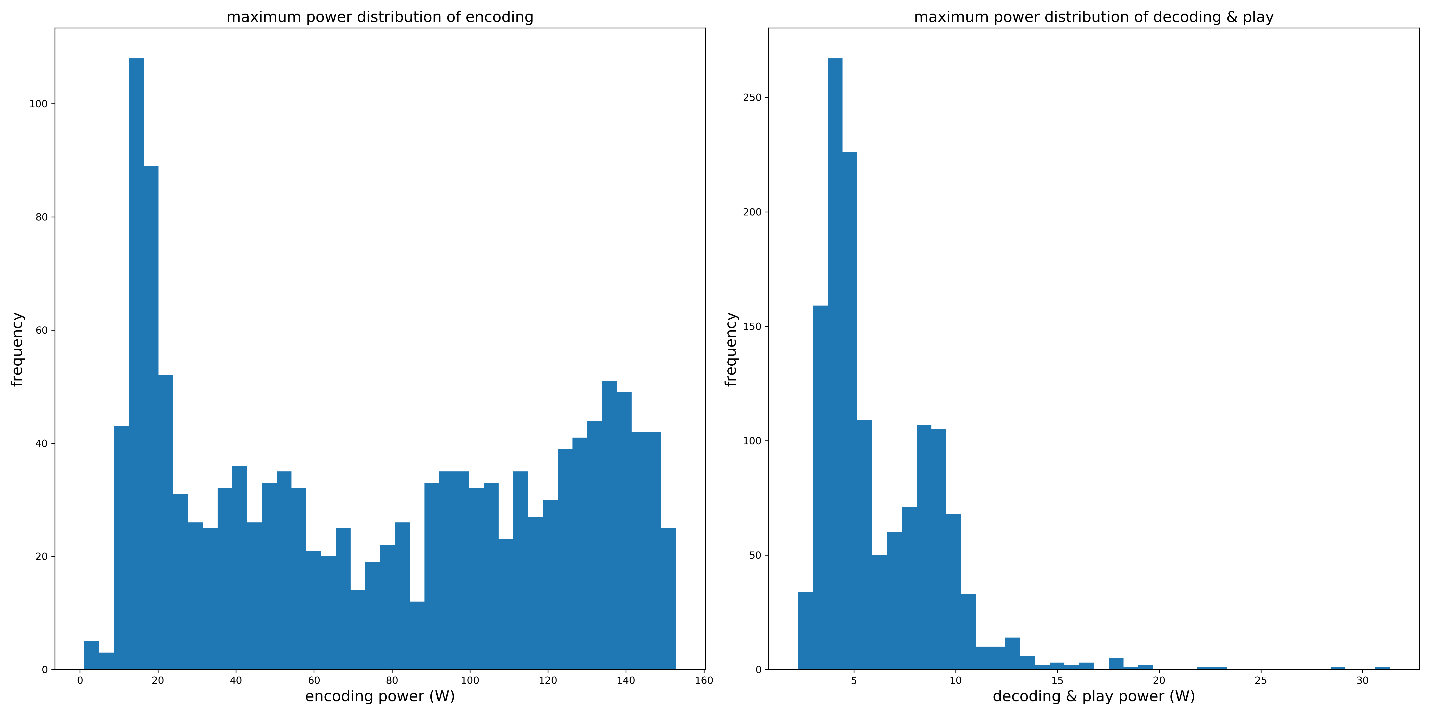
Above all, for these elapsed time sensitive applications, libx264 is more suitable than the other three formats as it consumes the least time in encoding, transmission, and decoding & play.

**5.4 Instantaneous energy consumption**

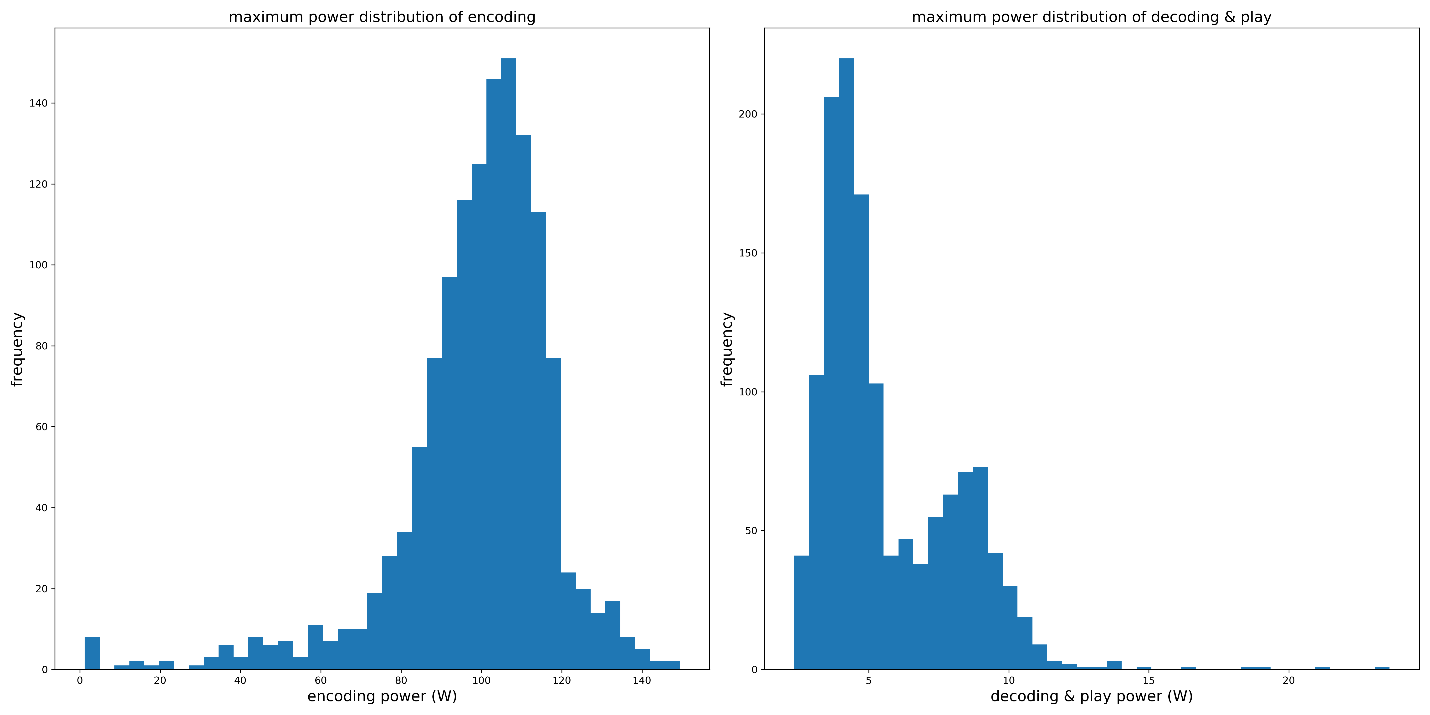
Power, which is defined as energy consumption within a very short time slice, is a metric for instantaneous energy consumption. High power means high instantaneous current flowing in the circuits when the supply voltage is constant, which may cause permanent damage for some fragile circuits in the device. As a result, most of the electrical devices have a maximum power limitation and even instantaneous power should not be over it. In this exploration, the energy monitoring device samples the instantaneous power data every three seconds as shown in Fig. 3. Thus, during each video encoding or decoding & play steps, we will collect the maximum instantaneous power value, and then record these maximum power distributions in Fig. 14 and Table II, III for different encoding formats.



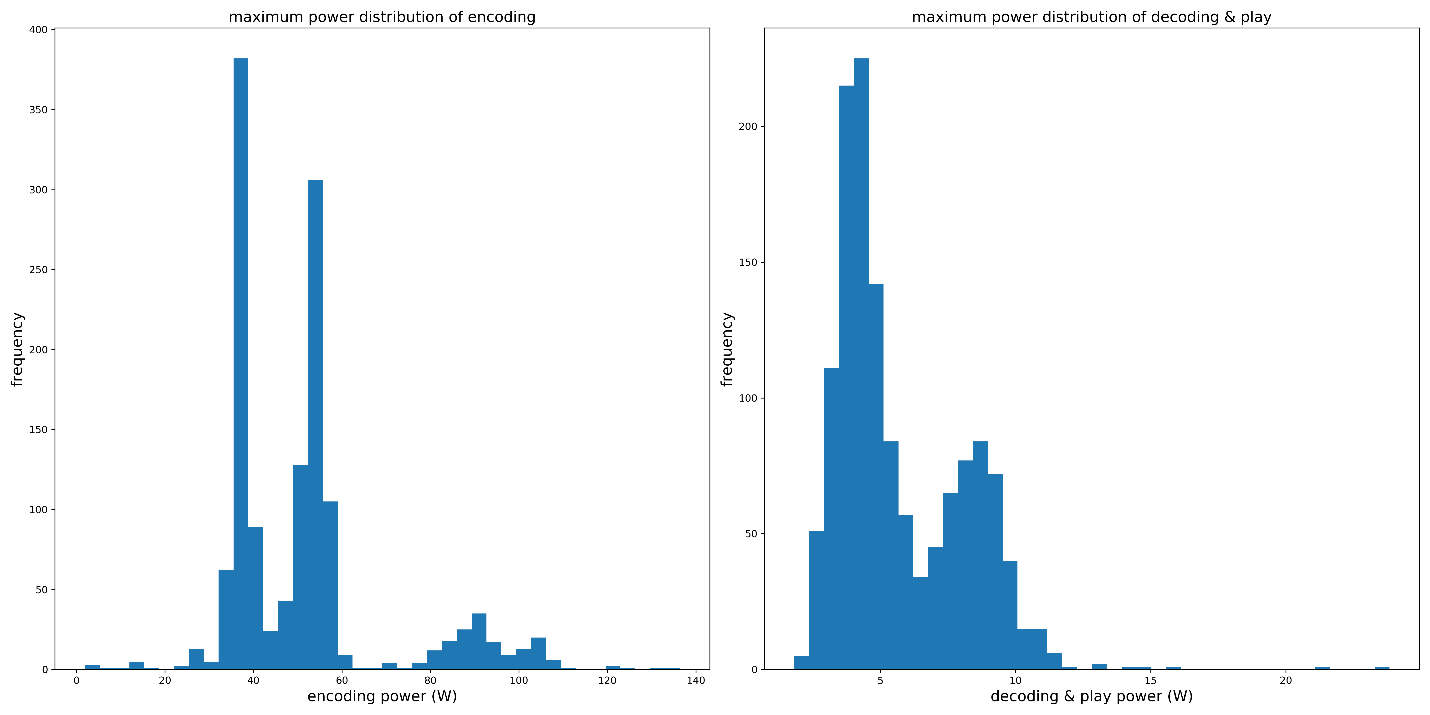
1. Maximum power distribution of encoding(left) and decoding & play (right) for encoding format libx264



1. Maximum power distribution of encoding(left) and decoding & play (right) for encoding format HEVC



1. Maximum power distribution of encoding(left) and decoding & play (right) for encoding format VP8



1. Maximum power distribution of encoding(left) and decoding & play (right) for encoding format VP9

Fig.14 Maximum power distribution for different encoding formats

Table II Maximum powers statistics metrics for different encoding formats during encoding step

|  |  |  |  |
| --- | --- | --- | --- |
| Encoding | mean(W) | std(W) | Maximum |
| Libx264 | 68.39 | 40.31 | 178.75 |
| HEVC | 76.32 | 47.00 | 152.81 |
| VP8 | 99.15 | 19.46 | 149.43 |
| VP9 | 51.23 | 18.49 | 136.39 |

Table III Maximum powers statistics metrics for different encoding formats during decoding & play step

|  |  |  |  |
| --- | --- | --- | --- |
| Decoding & play | mean(W) | std(W) | Maximum |
| Libx264 | 7.22 | 2.97 | 30.9 |
| HEVC | 6.26 | 2.96 | 31.33 |
| VP8 | 5.67 | 2.41 | 23.59 |
| VP9 | 5.72 | 2.38 | 23.93 |

From Fig.14 and Table II, III, we can find that

1. Power during encoding step is far larger than decoding & play step for all encoding formats if requests = 1.
2. Vp9 has the smallest power value during encoding and decoding & play steps, while libx264 trends to have largest power value during encoding and decoding & play steps.

Above all, if the encoding step is implemented on a low maximum power limitation machine, VP9 seems to be more suitable than the other three formats, even though it costs more time (or even more energy).

Table IV comparison of different encoding formats

|  |  |  |
| --- | --- | --- |
| Encoding format | advantages | disadvantages |
| Libx264 | 1. The least energy consumption in the encoding step 2. The shortest encoding time 3. The largest maximum instantaneous energy consumption (power) | 1. The most energy consumption in the decoding & play step |
| HEVC | 1. Medium level for all requirements |  |
| VP8 | 1. The least cost in the decoding & play step 2. The smallest file size after the encoding step, hence least energy consumption in the transmission step | 1. The largest average of maximum power |
| VP9 | 1. The smallest maximum instantaneous energy consumption (power) | 1. The most energy consumption in encoding step 2. The largest file size after encoding, hence the most energy consumption in the transmission step 3. The longest encoding time |

According to the comparison summary in Table IV, there is not a general encoding format that can be optimal for all energy, memory, time and power limitation , but we can choose the encoding formats under the specific scenario, such that

1. When the application is limited by energy consumption and the video requests is less than 100, libx264 is preferred, but as video requests increase, VP8 should be chosen.
2. When the application is limited by complex network or memory storage, VP8 should help because it has the largest compression ratio.
3. When the application needs very short request latency, libx264 should be chosen because it has the shortest encoding time.
4. When the application has a lower limitation of the maximum power, VP9 is preferred now that its instantaneous energy consumption is lower than the others.
5. For HEVC, neither one of the it metrics is the best nor the worst of the four encoding formats. It is both the advantage and disadvantage of the encoding format. Maybe it can be used in some scenarios that has implicit energy/memory/elapsed time/power requirements, but it is sub-optimal for the explicit scenarios.
6. **Machine learning based recommendation**

We have given a thorough explanation of the experiment data and tried to make some simple recommendation of the encoding format under different conditions. However, our final target is to recommend the best encoding format that has the best energy/memory efficiency, we should implement a model that can predict an explicit best encoding format for a given video file. Specifically, the problem becomes a classification problem, where we should train a model based on the experiment data, and the model should take the input features from original video file, then predict which encoding format has the best energy & memory efficiency, which can be measured by Eq. 2. Traditionally, there are lots of modelling methods, such as linear fitting or polynormal fitting [34], but these methods suffer from so called feature’s dimension disaster problems, hence less precision. On the other hand, with the boosting of the computing capability, we have witnessed fasted development of the machine learning based modeling methods in recent years, such as K-neighbor, Support Vector Machine and Random Forest. We will try these methods and choose the best one as our recommendation model.

* 1. Features exploration

Before training the classification model, we need the features vector and the corresponding label for each original video file in the dataset. The label represents one of four encoding formats, which can be determined by traversing the four encoding formats to each original video file one by one, then choosing the one with the least energy consumption. This chapter mainly discusses the exploration of features vector.

* + 1. Requests number

As we have discussed before, requests number represents the number of the original video is requested. The original video can be encoded once in the host device but can be requested multi times by different customer devices, and correspondingly multi energy consumption on transmission and decoding & play steps. Thus, the requests number determines which one domain the total energy consumption. As shown in Eq .2, when requests number is smaller than 10, it is the encoding energy consumption that domains the total energy consumption, while the decoding &play energy consumption will domain if requests number becomes larger. As a result, the requests number should be an important feature. We have calculated the total energy consumption using different request numbers (1,10,100, 100000) in the experiment. On the other hand, considering the requests number increases as an exponential style, we decide to make a logarithm transfer before adding it to our features vector. Specifically, the value (0, 1, 2, 5) will represent value (1, 10, 100, 100000) to make the feature’s value distribution more even.

* + 1. Original video file size

The original video size, which represents the total workload to be processed, is another important feature. As we have discussed before, there will be a linear relationship between the original video file size and total energy consumption for all four encoding formats. Specifically, the bigger the file size is, the more data should be encoded, transmitted, decoded, and played, and the more energy consumption, so we decide to adopt the original video file size as one feature. However, as we find the file size value distributes more even after the logarithm transfer, we decide to use this value to train our machine learning model.

* + 1. Resolution

Resolution contains the height and width information of a frame in the video file. From table V, higher resolution means larger frame height and width, which makes the video high quality, but will consume more because of larger workload. The other interesting observation from Table X is that even with the same resolution, there are different height and width for the videos in the dataset, so we decide to extract the width and height information of the video as 2 features.

Table V, the weight x height for different resolution videos from UGC dataset

|  |  |
| --- | --- |
| resolution | Weight x Height |
| 2160P | 1080 x 1920, 1080 x 2220, 1920 x 1080, 2160 x 1080, … |
| 1080P | 2160 x 3840, 3032 x 2304, 3840 x 1920, … |
| 720P | 720 x 1280, 732 x 720, 960 x 720, … |
| 480P | 640 x 480, 480 x 848, 352 x 480, … |
| 360P | 1. 360, 270 x 480, … |

6.1.4 Frame per second

Frame Per Second (FPS) is another important feature that contributes to the quality of the video, or the size of the video. As shown in Eq.3, there is an exact relationship among video size, frame per seconds, resolution, and video duration:

video size = frame per seconds x duration x W x H Eq.3

* + 1. Category

As shown in Table I, the videos come from 16 categories, such as Animation, Gaming, etc., and different categories may result in different energy consumption. Based on our previous discussion, the videos that contains more dynamic frames may cost more energy, such as Sports, while the videos that are static style trend to cost less, such as lyrics videos. The observation implies that the categories may have influence on the energy consumption, thus will be adopted into the features vector. However, the categories must be transferred into numerical style before it is feed to the training model. As shown in Fig. 15 (a), of course we can pre-define some numbers to represent different categories, but this transferring method will confuse the machine learning model because the value of the numerical representations are meaningless but the learning model will compare them. As a result, we introduce the One-Hot encoding [35] format to represents different categories as shown in Fig. 15 (b). Specifically, we will introduce the categories of all test cases as different features. For each test case, it belongs only one category, and the corresponding feature value is 1, while the other features’ value for this test case it 0 because the test case does not belong to these categories.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test case | Category | Numerical representation |  | Test case | Animation | ConverSong | Gaming |
| 1 | Animation | 0 |  | 1 | 1 | 0 | 0 |
| 2 | CoverSong | 1 |  | 2 | 0 | 1 | 0 |
| 3 | Gaming | 2 |  | 3 | 0 | 0 | 1 |
| 4 | Animation | 0 |  | 4 | 1 | 0 | 0 |

Fig. 15, numerical representation scheme for category features (a) sequential (b) One-hot

* + 1. Other features from Dataset

According to the reference [32], the YouTube UGC dataset provides the other four features for developer, namely spatial complexity, color complexity, temporal complexity, and chunk variation. The Spatial complexity is calculated by the average frame bitrate normalized by the frame area, which can be used as a metric to measure the spatial details. The color complexity is defined by the ratio between the average of mean sum of squared error in U and V channels to the mean sum of squared error in Y channel. Higher score means complex color variation while lower score means a gray image. The temporal complexity is to measure the motion among frames, and the chunk variation is introduced to detect the sudden changes.

As we have discussed before, the total energy consumption may be related to the video’s category. i.e., the dynamic sports video consumes more energy than the static lyrics videos. Now that the dynamic videos have more spatial and temporal complexity, we believe that the energy consumption should be related to these complexity features, so we decide to adopt these four features into our features vector.

* 1. Machine learning based modelling
     1. Training phase

The famous scikit-learn python library is adopted to train the KNN, SVM and RF models [36]. The training procedure has following steps:

1. The training <feature, label> sets is dived into training set and testing set with a ratio of 4:1, and the label distribution of both training set and testing set are shown in Table VI, of note, the distribution is not balanced, especially for VP9, we will discuss more in the following chapter. The training set is used to train the machine learning based models while the testing set to verify the model’s accuracy. The other major function of the testing set is to mimic the prediction procedure, so it should neither take part in the training procedure nor be used to choose the learning parameters.

Table VI, the distribution of training and testing sets

|  |  |  |
| --- | --- | --- |
| Labels | Training set | Testing set |
| Libx264 | 1712 | 462 |
| Hevc | 883 | 211 |
| Vp8 | 1378 | 322 |
| Vp9 | 350 | 86 |

1. We will implement a min-max scalar to normalize the training set as shown in Eq. 4. Specifically, we will at first to find the maximum and minimum value for each feature in training set as MAX and MIN, then the correspond feature’s value will be normalized as:

normalized feature = (original feature – MIN)/(MAX – MIN) Eq.4

For testing set, the feature’s value is normalized as the same MAX, MIN parameters learning from training set.

1. After normalization, the <label, feature> pairs from training set are fed to the machine learning models. As discussed before, there are several important parameters that should be finetuned for each model during the training procedure. For KNN, it is the number of nearest neighbors’ number, and for SVM, they are penalty factor C, kernel’s type, and gamma factor, while for RF, the parameters are maximum number and the maximum depth of the decision trees, so we choose the grid search algorithm to find the optimal parameters. On the other hand, to make the training accuracy reliable, a 5-fold cross-validation is introduced.
2. The training model will be used to predict the testing set’s label, and the model’s accuracy can be evaluated by comparing the predicted label and ground-truth label.

Fig.16 describes the training score and testing score versus different nearest neighbor’s number n\_neighbors. From the figure, we can find following observations:

1. Nether too larger nor too small nearest neighbors will make the model’s accuracy degrade.
2. The optimal nearest neighbors’ number is about 10 from the training score, even though the corresponding testing score is not the best (69.7%), it cannot be used to choose the learning parameters as we discussed before.

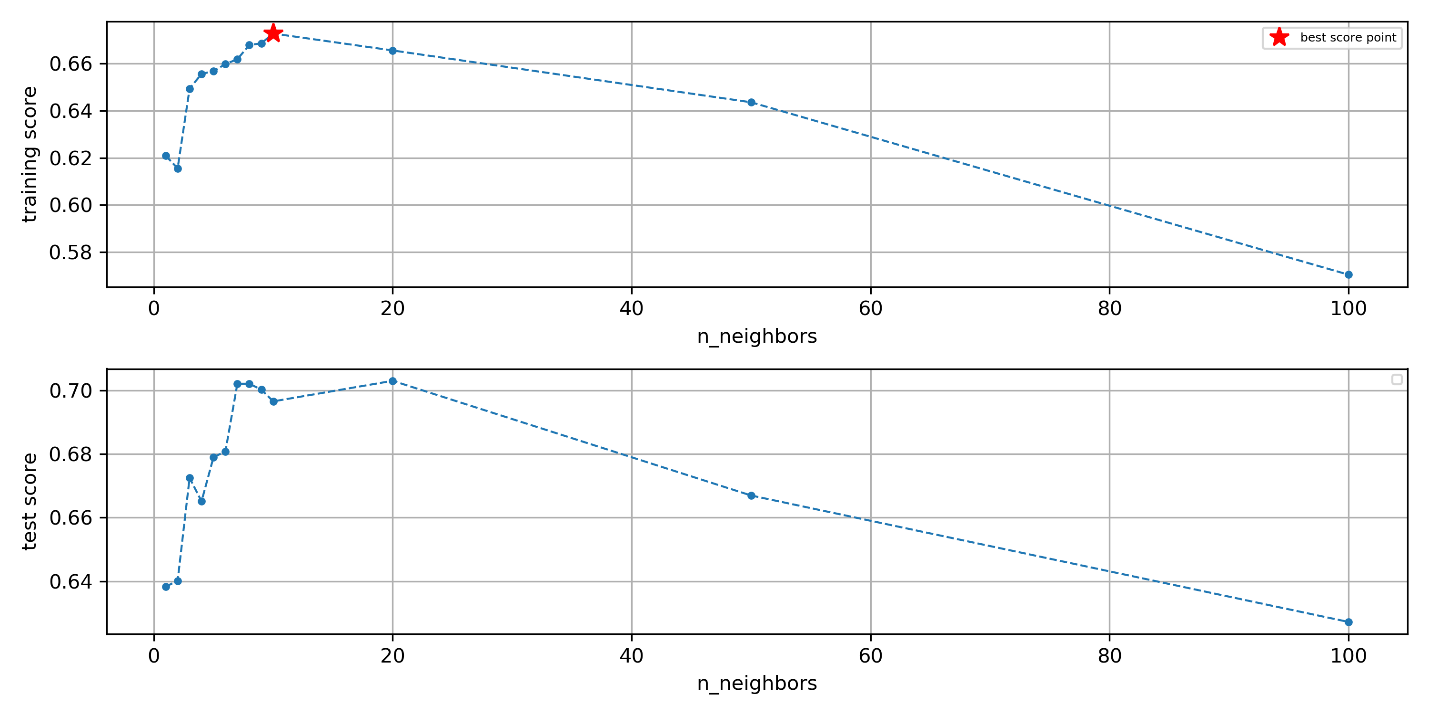


Fig.16 The KNN’s training score and testing score for different nearest neighbors’ number

With optimal nearest neighbors’ number 10, we have mapped the model to the testing set and summarized the testing accuracy for different labels in Table VII

Table VII, KNN’s testing accuracy for different encoding formats

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Prediction  Gound  Truth | Libx264 | Hevc | Vp8 | Vp9 | Accuracy (%) |
| Libx264 | 429 | 23 | 10 | 0 | 92.86 |
| Hevc | 27 | 111 | 63 | 10 | 52.61 |
| Vp8 | 39 | 66 | 202 | 15 | 62.73 |
| Vp9 | 3 | 33 | 39 | 11 | 12.79 |

From Table VII, we can find that KNN model has the best performance for libx264 while the worst performance for VP9. One of the reasons for this result is that the libx264 contributes most of training samples and hence more possible votes for this class. The other reason is there might be explicit boundary between libx264 cluster and the other three clusters in the feature space, which means less misclassification for the libx264 class while the other three classes will introduce noises for each other. Taking the worst performance VP9 as an example, only 11 of 86 testing points are rightly classified, while 39 (45.36%) are mis-classified as VP8 and 33 (38.37%) as HEVC, which proves that the VP8 and HEVC have introduced large noise for the VP9.

The training and testing scores for Support Vector Machine is shown in Fig.17. The grid searched parameters contain penalty factor C, kernel type (poly or RBF) and gamma. From the Fig. 17, we can get the following observations.

1. Within the experiment region [0.001, 10], larger penalty factor C means better accuracy.
2. RBF kernel has better performance than polynomial kernel.
3. Too small or too larger gamma makes the accuracy degrade.
4. The best parameters are kernel = RBF, C = 10 and gamma = 0.1 from the testing score plot, and the corresponding testing accuracy is about 72.24%, which has slight improvement compared to the KNN model.

Table VIII, SVM’s testing accuracy for different encoding formats

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Prediction  Gound  Truth | Libx264 | Hevc | Vp8 | Vp9 | Accuracy (%) |
| Libx264 | 425 | 18 | 19 | 0 | 91.99 |
| Hevc | 18 | 105 | 86 | 2 | 49.86 |
| Vp8 | 27 | 43 | 246 | 6 | 76.39 |
| Vp9 | 0 | 27 | 54 | 5 | 5.8 |

Table VIII summarized the testing accuracy for each class using the optimal SVM model. Compared to the KNN model, VP8 has a better prediction accuracy while the accuracy for the other three classed drops. The testing accuracy of VP9 proves that the SVM suffers more from the noises than the KNN models.

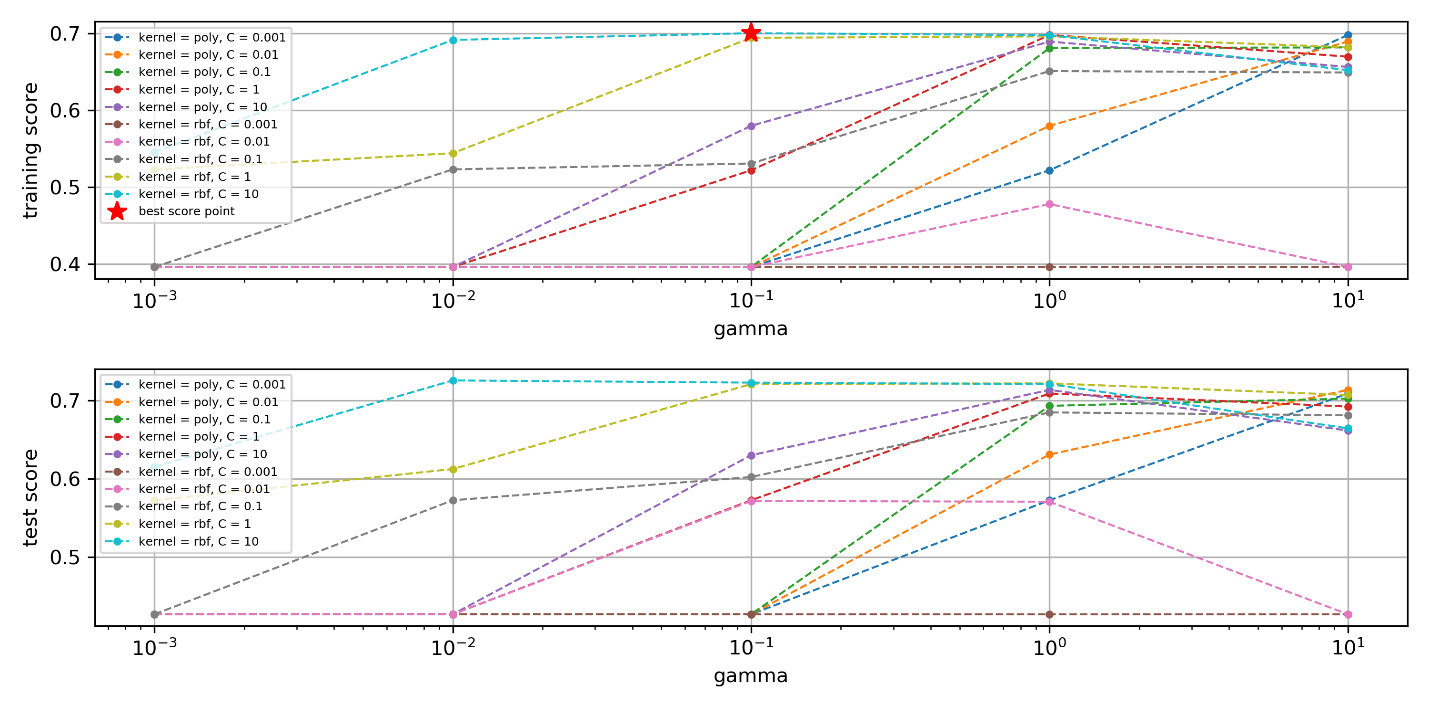


Fig.17 The SVM’s training & testing score for different penalty factor C, kernel type and gamma factor

For Random Forest, the training score and testing score under different maximum decision tree’s number (n\_estimators) and depth (max\_depth) is shown in Fig. 18, from which we can get the following observations.

1. Too few decisions tree (i.e. 10) makes model worse performance because the learning ability is limited or called sub-fitting, but too many decision trees help less for the performance.
2. Too small decision tree’s depth will be harmful for the model’s accuracy for less learning ability, while too large depth has less influence on the performance.
3. The optimal decision trees’ number is about 200 and max depth is about 150 from the training accuracy, while the corresponding testing accuracy is about 77.98%

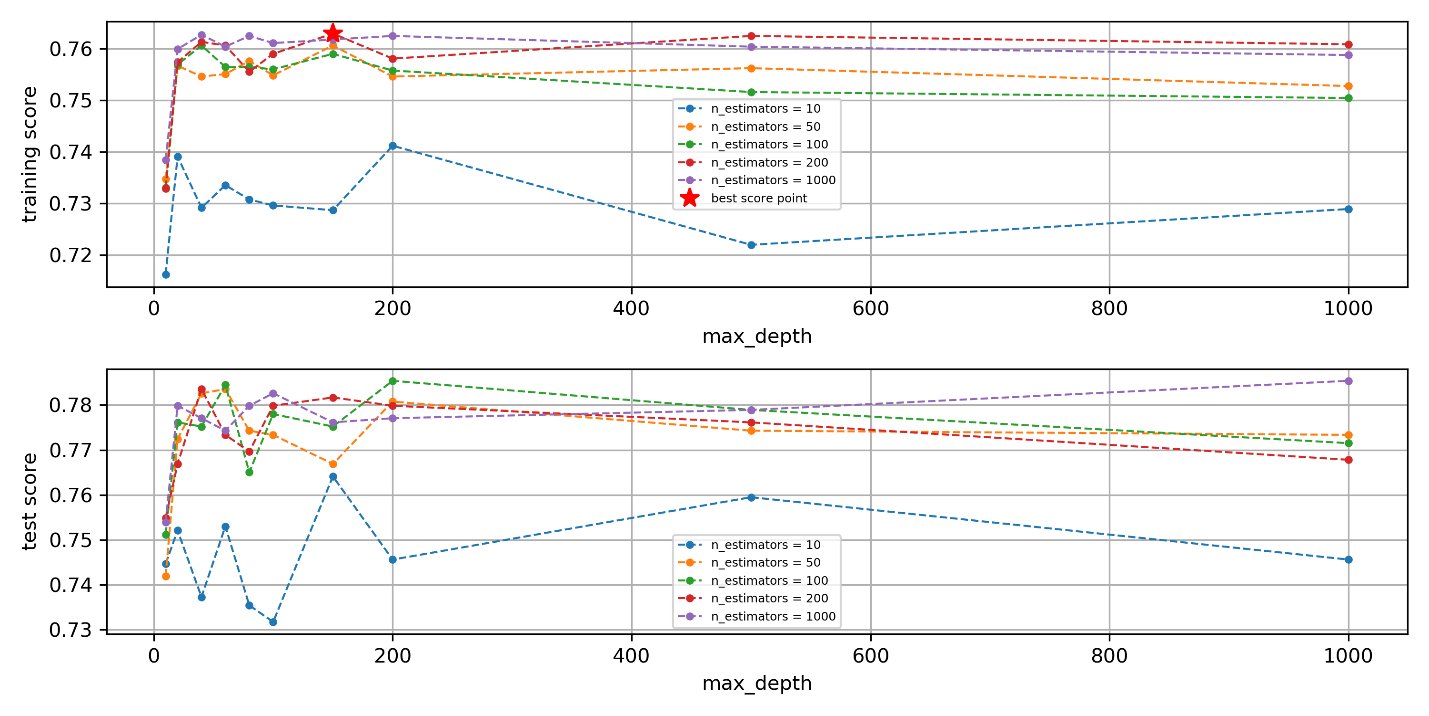


Fig.18 The RF’s training score and testing score for different maximum decision trees’ number and depth.

The testing accuracy using the optimal model is summarized on Table. IX. From the table, it is obvious that the accuracy for different labels have been improved compared to KNN and SVM. However, the accuracy for Vp9 is still very low because of the noise introduced by Vp8 and Hevc class, but compared to SVM and KNN, the improved accuracy proves that the RF method is more robust to the noise from other classes. In other words, if we continue to collect more features to reduce the influence from the other classes in feature space, we can improve the performance furthermore.

Table IX, RF’s testing accuracy for different encoding formats

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Prediction  Gound  Truth | Libx264 | Hevc | Vp8 | Vp9 | Accuracy (%) |
| Libx264 | 437 | 13 | 11 | 1 | 94.59 |
| Hevc | 21 | 129 | 50 | 11 | 61.14 |
| Vp8 | 21 | 35 | 252 | 14 | 78.26 |
| Vp9 | 1 | 27 | 33 | 25 | 29.07 |

As discussed before, the random forest can detect the importance of the features. We have explored the importance level for different features in the random forest model in Fig.19, from which we can get that:

1. The requests number is the most important feature because it determines which energy consumption domains the total as shown Eq.2.
2. The four complexities features from the dataset author are also important, then some intrinsic features of original video files, such as file size, frame per second, width and height also contribute to the classification model.
3. While the category features contribute less information for the model because they have only two values, 0 and 1.

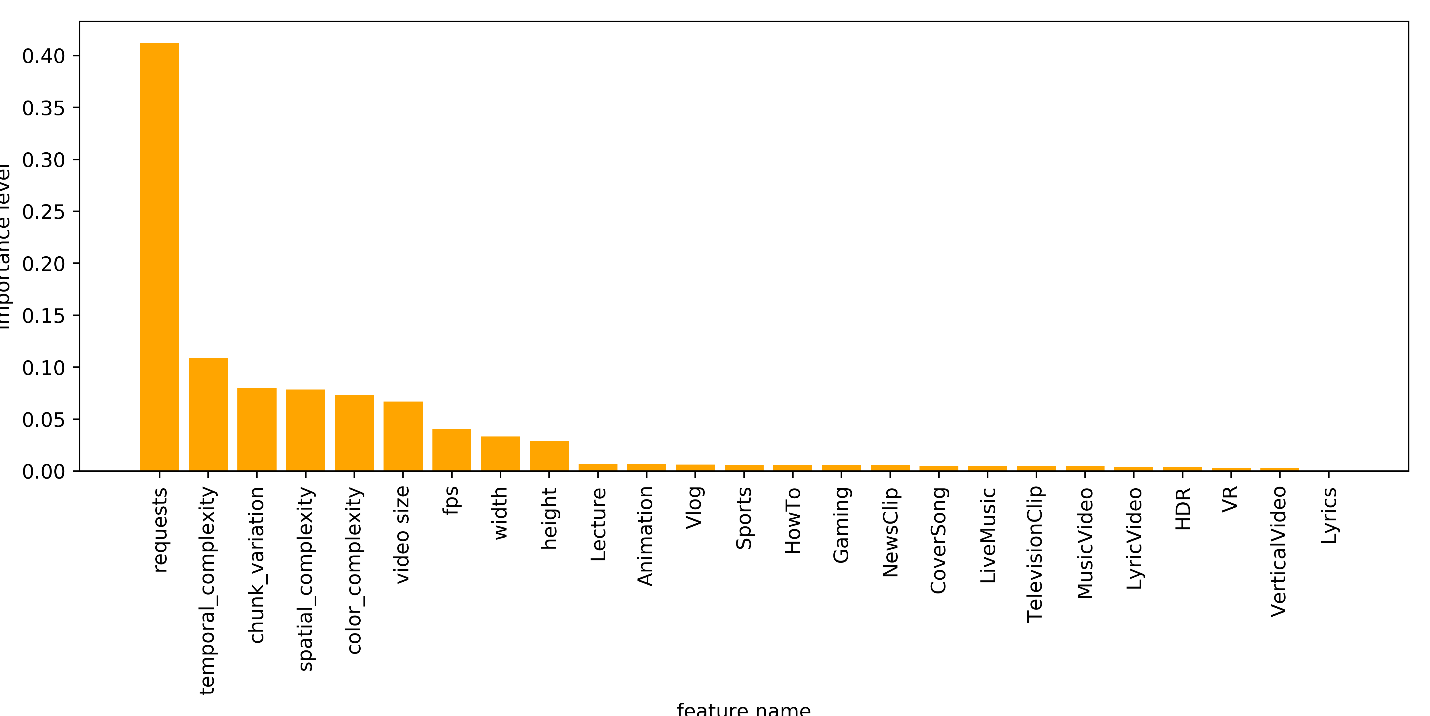


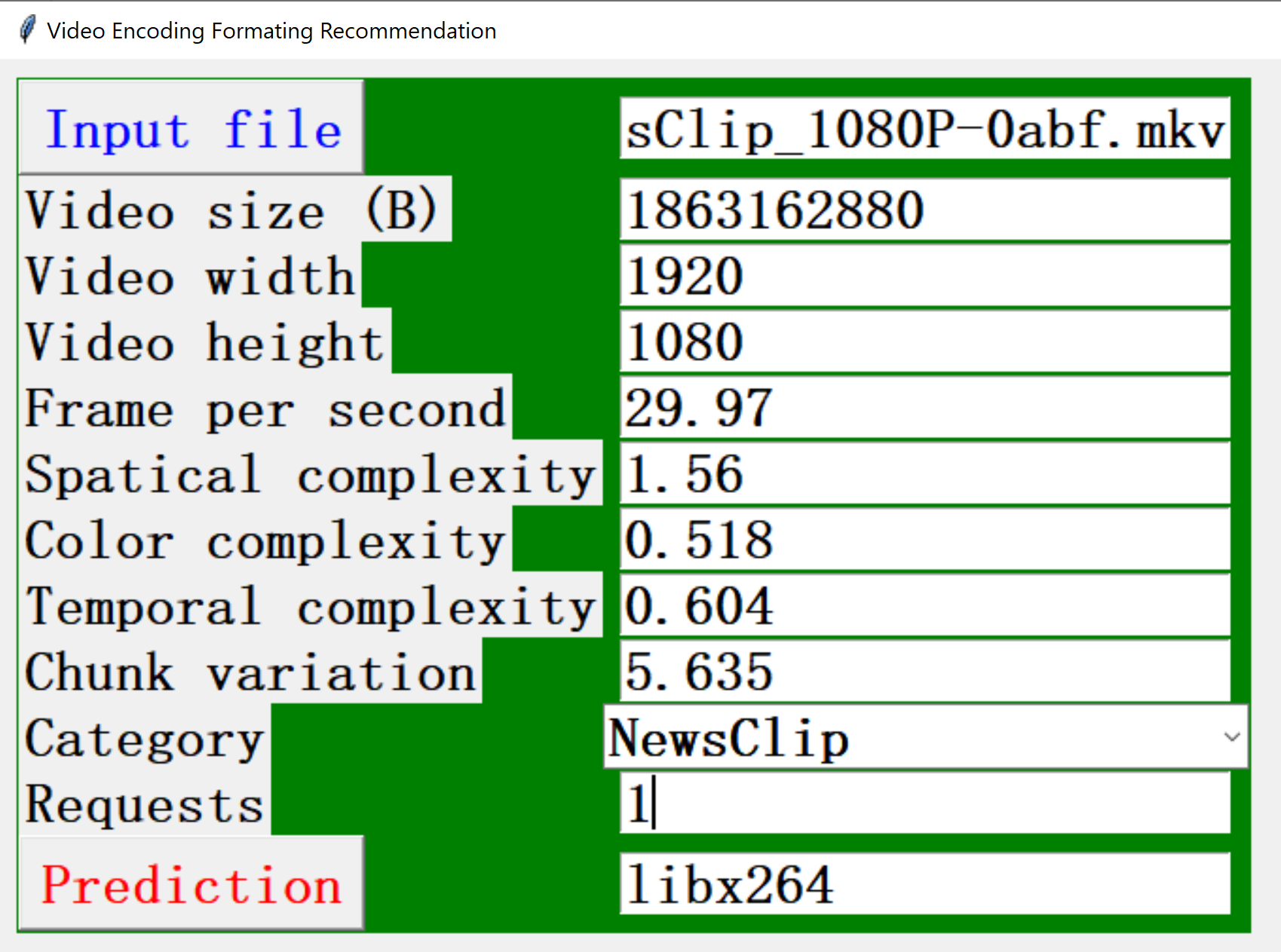
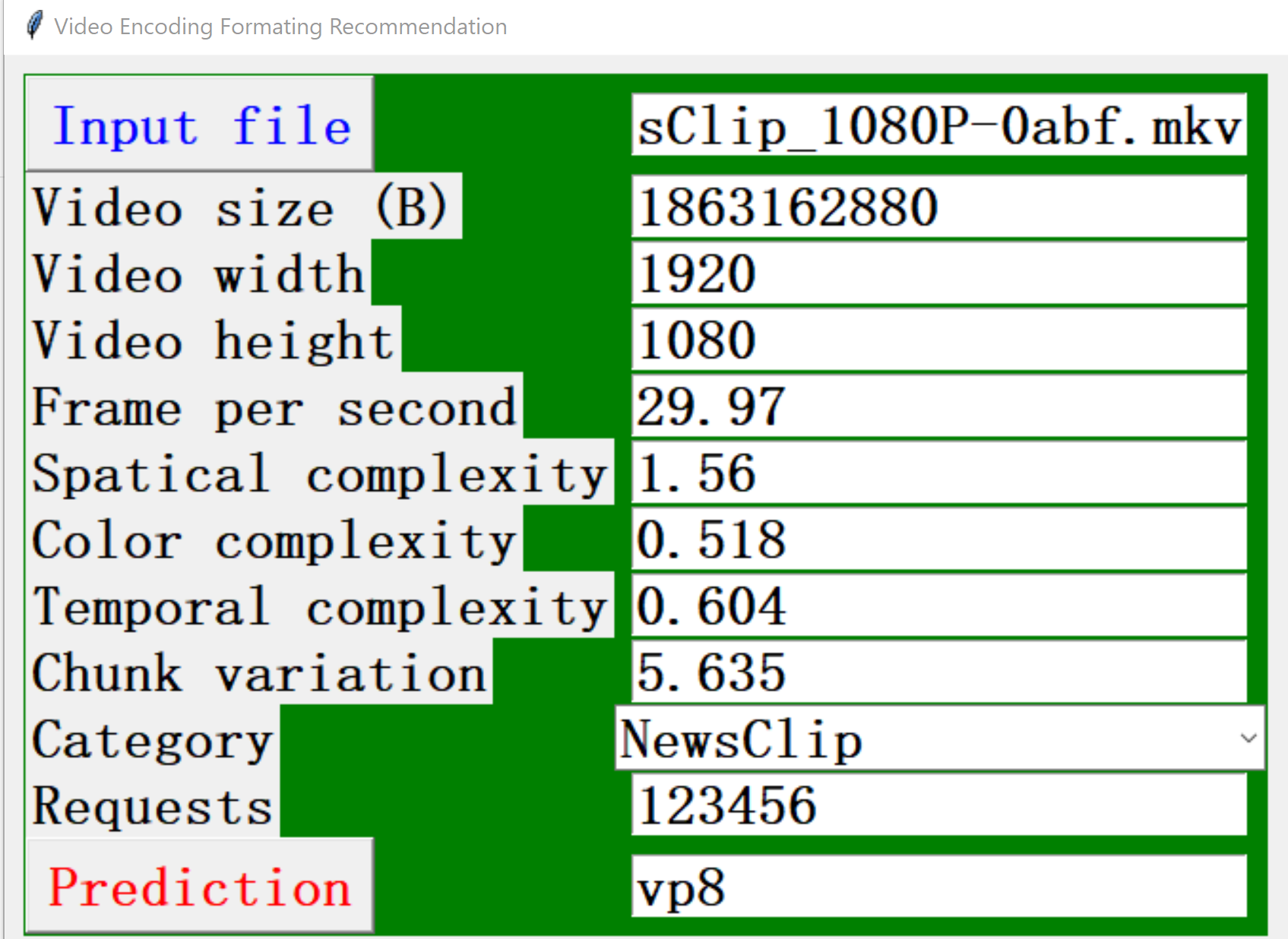
Fig.19, the features’ importance in Random Forest

Above all, we have trained three optimal models using KNN, SVM and RF, respectively. The testing accuracy implies that the RF is both more robust and accurate than the other two methods, so the RF model is used to predict the most energy/memory efficiency encoding format.

* + 1. Prediction phase

Once we have trained the optimal model, we can use it to predict the best encoding format using the input features, which is easily obtained from the original video file. Even though our prediction model has 77.98% accuracy, it still saves a lot because if we want to fully ensure which encoding format is the most energy & memory efficiency, we have to measure the energy consumption of all encoding formats for encoding, transmission, and decoding & play steps and then choose the best one, which is nearly impossible because it will cost lots of energy, time and hardware resources.

To make the prediction procedure simple, we have developed a visualization tool using Python Tkinter library [37]. As shown in Fig. 20, the visualization GUI firstly needs the user to type the input features at first. Then, after the ‘Prediction’ button is pushed, there features will be read and normalized to form the features vector, which will be used by the pre-trained mode to predict the best encoding format. Let us take a video cases from the testing set as the example. At first, we will input the features as shown in Fig. 20 (a), the tool says the libx264 is the best encoding format. Then, as we increase the requests number to 123456 in Fig. 20 (b), the prediction shows that VP8 is the best encoding format. The result totally fulfills the ground-truth measurement.

1. (b)

Fi g.20 The GUI for best encoding format prediction

1. Conclusion

Motivated to develop a tool that can find the most energy and memory efficient encoding format for different videos, we have proposal a machine learning based solution in this paper. Specifically, In the training phase, we have firstly built a flow to measure the energy consumption for the encoding, transmission, and decoding & play steps, respectively. Secondly, we have made an exhaustive analysis for the energy consumption, encoding time, compression ratio and maximum power distribution, which may be the bottleneck for different applications. From the analysis, we concluded that for different scenarios, the optimal encoding format differs. Furthermore, we have explored the relationship between total energy consumption and some features, such as video file size, requests number, resolution, frame per second, complexity, category, and we find that these features have some influence on the energy consumption, we decided to use them as features vector to train a machine learning based model. During the training phase, three popular machine learning algorithms are introduced and compared with the same training and testing datasets, the results show that the random forest, support vector machine and K-nearest neighbors models have obtained the 77.98%, 72.24%, 69.7% testing accuracy, respectively. As a result, we choose random forest as the pre-trained model in the prediction phase. Finally, we have developed a visualization GUI, which can read the user input features and predict the best encoding format using the pre-trained random forest model.

From the perspective of the energy consumption, our work has two kinds of benefits. At first, as we can easily obtain the features from the original videos and predict the most energy and memory efficient encoding format. In other words, we can save the procedure that measures all encoding formats’ energy consumption by replaying the encoding, transmission, and decoding & play steps and then chooses the best one, which will cost lots of energy and hardware resources. On the other hand, as we adopt the state-of-the-art machine learning based algorithm, we have improved the prediction accuracy form 25% (considering randomly choosing one of four encoding formats, the probability of choosing the best one is ¼, or 25%) to 77.98%, thus more than 3.1x energy consumption is saved.

**References**

1. M. Horowitz. Computing’s energy problem (and what we can do about it). In Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International, pages 10–14, Feb 2014.
2. Jae-Beom Lee, Myoung-Jin Kim, Sungroh Yoon, and Eui-Young Chung. Application-support particle filter for dynamic voltage scaling of multimedia applications. Computers, IEEE Transactions on, 61(9):1256–1269, Sept 2012.
3. R. Trestian, O. Ormond, and G.-M. Muntean. Enhanced power-friendly access network selection strategy for multimedia delivery over heterogeneous wireless networks. Broadcasting, IEEE Transactions on, 60(1):85–101, March 2014.
4. A. Wang and A. Chandrakasan. Energy-efficient DSPs for wireless sensor networks. Signal Processing Magazine, IEEE, 19(4):68–78, 2002.
5. Gilberto Contreras and Margaret Martonosi. Power prediction for intel xscale R processors using performance monitoring unit events. In Low Power Electronics and Design, 2005. ISLPED’05. Proceedings of the 2005 International Symposium on, pages 221–226. IEEE, 2005.
6. V. Spiliopoulos, A. Bagdia, A. Hansson, P. Aldworth, and S. Kaxiras. Introducing dvfs-management in a full-system simulator. In Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on, pages 535–545, Aug 2013.
7. Yahia Benmoussa, Jalil Boukhobza, Eric Senn, and Djamel Benazzouz. Gpp vs dsp: A performance/energy characterization and evaluation of video decoding.In Proceedings of the 2013 IEEE 21st International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems, MASCOTS ’13, pages 273–282. IEEE Computer Society, 2013.
8. Yahia Benmoussa, Eric Senn, Jalil Boukhobza, Mickael Lanoe, and Djamel Benazzouz. Open-PEOPLE, a collaborative platform for remote & accurate measurement and evaluation of embedded systems power consumption. in Proceedings of the IEEE 22nd International Symposium On Modeling, Analysis And Simulation Of Computer And Telecommunication Systems, 2014.
9. F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In Proceedings of the 36th Annual Symposium on Foundations of Computer Science, FOCS ’95, pages 374–, Washington, DC, USA, 1995. IEEE Computer Society.
10. V. Gutnik and A.P. Chandrakasan. Embedded power supply for low-power dsp. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 5(4):425– 435, 1997.
11. Kihwan Choi, R. Soma, and M. Pedram. Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 24(1):18–28, Jan 2005.
12. M. Horowitz, E. Alon, D. Patil, S. Naffziger, Rajesh Kumar, and K. Bernstein. Scaling, power, and the future of cmos. In Electron Devices Meeting, 2005. IEDM Technical Digest. IEEE International, pages 7 pp.–15, Dec 2005.
13. Juraj Bienik, Miroslav Uhrina, Michal Kuba, and Martin Vaculik. 2016. Performance of H. 264, H. 265, VP8 and VP9 Compression Standards for High Resolutions. 246–252.
14. Yue Chen, Debargha Murherjee, Jingning Han, Adrian Grange, Yaowu Xu, Zoe Liu, Sarah Parker, Cheng Chen, Hui Su, Urvang Joshi, Ching-Han Chiang, Yunqing Wang, Paul Wilkins, Jim Bankoski, Luc Trudeau, Nathan Egge, Jean-Marc Valin, Thomas Davies, Steinar Midtskogen, Andrey Norkin, and Peter de Rivaz. 2018. An Overview of Core Coding Tools in the AV1 Video Codec. In 2019 Picture Coding Symposium (PCS), 41–45. https://doi.org/10.1109/PCS.2018.8456249.
15. N. Kavvadias, P. Neofotistos, S. Nikolaidis, C.A. Kosmatopoulos, and T. Laopoulos. 2004. Measurements analysis of the software-related power consumption in microprocessors. IEEE Transactions on Instrumentation and Measurement 53, 4: 1106–1112. https://doi.org/10.1109/TIM.2004.830784
16. Mehul Tikekar, Chao-Tsung Huang, Chiraag Juvekar, Vivienne Sze, and Anantha P Chandrakasan. 2013. A 249-Mpixel/s HEVC video-decoder chip for 4K ultraHD applications. IEEE Journal of Solid-State Circuits 49, 1: 61–72.
17. Jens-Rainer Ohm, Gary J Sullivan, Heiko Schwarz, Thiow Keng Tan, and Thomas Wiegand. 2012. Comparison of the coding efficiency of video coding standards—including high efficiency video coding (HEVC). IEEE Transactions on circuits and systems for video technology 22, 12: 1669–1684.
18. Thomas Wiegand and Gary J Sullivan. 2007. The H. 264/AVC video coding standard [Standards in a Nutshell]. IEEE Signal Processing Magazine 24, 2:148–153.
19. MP Sharabayko. 2013. Next generation video codecs:HEVC, VP9 and DAALA. Traffic 2560, 1600: 30
20. Martin Řeřábek, Philippe Hanhart, Pavel Korshunov, and Touradj Ebrahimi. 2015. Quality evaluation of HEVC and VP9 video compression in real-time applications. 1–6.
21. Dan Grois Detlev Marpea, Tung Nguyena, and Ofer Hadarb: Comparative Assessment of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC Encoders for Low-Delay Video Applications: <http://iphome.hhi.de/marpe/download/Comp_LD_HEVC_VP9_X264_SPIE_2014-preprint.pdf>
22. <http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/YouTube-On2-and-the-Economics-of-Ultra-HD-Tallying-the-Costs-103603.aspx>
23. Cover, Thomas, and Peter Hart. "Nearest neighbor pattern classification." IEEE transactions on information theory 13, no. 1 (1967): 21-27.
24. Hassanat A B, Abbadi M A, Altarawneh G A, et al. Solving the problem of the K parameter in the KNN classifier using an ensemble learning approach[J]. arXiv preprint arXiv:1409.0919, 2014.
25. Fletcher T. Support vector machines explained[J]. Tutorial paper, 2009: 4.
26. Platt J. Sequential minimal optimization: A fast algorithm for training support vector machines[J]. 1998.
27. Wu Q, Zhou D X. SVM soft margin classifiers: linear programming versus quadratic programming[J]. Neural computation, 2005, 17(5): 1160-1187.
28. Vert J P, Tsuda K, Schölkopf B. A primer on kernel methods[J]. Kernel methods in computational biology, 2004, 47: 35-70.
29. Breiman L. Random forests[J]. Machine learning, 2001, 45(1): 5-32.
30. Oshiro T M, Perez P S, Baranauskas J A. How many trees in a random forest?[C]//International workshop on machine learning and data mining in pattern recognition. Springer, Berlin, Heidelberg, 2012: 154-168.
31. Biau G, Scornet E. A random forest guided tour[J]. Test, 2016, 25(2): 197-227.
32. Wang Y, Inguva S, Adsumilli B. Youtube ugc dataset for video compression research[C]//2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP). IEEE, 2019: 1-5.
33. Lei X, Jiang X, Wang C. Design and implementation of a real-time video stream analysis system based on FFMPEG[C]//2013 Fourth World Congress on Software Engineering. IEEE, 2013: 212-216.
34. Rencher A C. Methods of multivariate analysis[M]. John Wiley & Sons, 2003.
35. Rodríguez P, Bautista M A, Gonzalez J, et al. Beyond one-hot encoding: Lower dimensional target embedding[J]. Image and Vision Computing, 2018, 75: 21-31.
36. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine learning in Python[J]. the Journal of machine Learning research, 2011, 12: 2825-2830.
37. Lundh F. An introduction to tkinter[J]. URL: www. pythonware. com/library/tkinter/introduction/index. htm, 1999.