PREDICTING COMPRESSION MODES AND SPLIT DECISIONS FOR HEVC VIDEO CODING USING MACHINE LEARNING TECHNIQUES

by

Mahitab Alaaeldin Hassan

A Thesis presented to the Faculty of the
American University of Sharjah
College of Engineering
In Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in
Computer Engineering

Sharjah, United Arab Emirates

May 2017

Approval Signatures

We, the undersigned, approve the Master's Thesis of Mahitab Alaaeldin Hassan

Thesis Title: Predicting Compression Modes and Split Decisions for HEVC Video Coding Using Machine Learning Techniques.

Signature	**Date of Signature** (dd/mm/yyyy)
_____	_____
Dr. Tamer Shanableh Professor, Department of Computer Science and Engineering Thesis Advisor	
_____	_____
Dr. Gerassimos Barlas Professor, Department of Computer Science and Engineering Thesis Committee Member	
_____	_____
Dr. Usman Tariq Associate Professor, Department of Electrical Engineering Thesis Committee Member	
_____	_____
Dr. Fadi Aloul Head, Department of Department of Computer Science and Engineering	
_____	_____
Dr. Mohamed El-Tarhuni Associate Dean for Graduate Affairs and Research, College of Engineering	
_____	_____
Dr. Richard Schoephoerster Dean, College of Engineering	
_____	_____
Dr. Khaled Assaleh Interim Vice Provost for Research and Graduate Studies	

Acknowledgement

I would like to express my deepest gratitude to my advisor, Dr. Tamer Shanableh, for providing knowledge, guidance, support, and motivation throughout my research stages. I'm deeply beholden for his great assistance and suggestions.

Moreover, I am grateful to the American University of Sharjah for offering me a graduate assistantship opportunity that allowed me to join the Computer Engineering Master's program.

My appreciation also extends to the professors of the Computer Engineering department, who taught me numerous courses and shared their expertise over the years, and my thesis examining committee. I am grateful for their advice, guidance, and support.

I would like to express my eternal appreciation towards my family, who always displayed unconditional patience and support. I thank them for their continuous understanding and for being a source of motivation.

Finally, I would like to thank my friends and colleagues for their encouragement and kindness.

Dedication

To my beloved parents,

Mrs. Mona and Mr. Alaaeldin,

My brother,

Abdulrahman,

And my dear friends,

Who were always a source of inspiration and support.

Abstract

The High Efficiency Video Coding (HEVC) standard presents a substantial video compression efficiency improvement at the expense of increasing the computational complexity. This enhancement is primarily due to the introduction of flexible quad-based-tree partitioning structures for motion estimation (ME) and image transformation. However, finding the optimum coding structure, which is done by an exhaustive rate-distortion optimization (RDO) process, is what contributes to increasing the computational complexity. In this thesis, we propose a set of early termination algorithms to reduce the HEVC video encoding complexity by predicting both the split decisions of Coding Units (CUs) and the coding modes of Prediction Units (PUs). A video sequence-dependent approach is used in which frames belonging to the video being encoded are utilized for generating a classification model. At each CU depth level, features representing the given CU are extracted from both the current and previously encoded CUs. The feature vectors (FVs) are then utilized for generating dimensionality reduction and classification models. These models are in turn used at each coding depth to predict the split and mode decisions of subsequence CUs. In this work, we use stepwise regression, random forest feature importance, and Principal Component Analysis (PCA) for dimensionality reduction. Moreover, polynomial networks, random forests, and J48 decision trees are used for classification. Using seventeen video sequences with four different spatial resolution classes, the proposed solution is assessed in terms of the classification accuracy, Bjontegaard Delta bitrate (BD-rate), BD Peak Signal-to-Noise Ratio (BD-PSNR) and computational complexity reduction (CCR). On average, the CU early termination scheme achieved a CCR of 38.5% with an average classification accuracy of 78.1% at a negligible cost of 0.539% and -0.021 dB in terms of BD-rate and BD-PSNR, respectively. The PU early termination scheme attained an overall CCR of 20.9% with an average classification accuracy of 86.5% at the cost of a BD-rate of 0.248% and a BD-PSNR of -0.01 dB. When jointly implemented, an overall CCR of 50.1% was achieved with a BD-rate increase of 2% and a BD-PSNR decrease of 0.079 dB.

Keywords: *Video coding; HEVC (High Efficiency Video Coding); Machine learning.*

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI	All-Intra
AMP	Asymmetric Motion Partition
AMVP	Advanced Motion Vector Prediction
AVC	Advanced Video Coding
BD-PSNR	Bjontegaard Delta Peak Signal-To-Noise Ratio
BD-rate	Bjontegaard Delta Bitrate
B-frame	Bi-Directional-Frame
CCR	Computational Complexity Reduction
CTB	Coding Tree Block
CTC	Common Test Condition
CTU	Coding Tree Unit
CU	Coding Unit
DCT	Discrete Cosine Transform
FCD	Fast Discrete Cross Difference
f-random	Forward Random
FV	Feature Vector
GOP	Group of Pictures
HD	High Definition
HEVC	High Efficiency Video Coding
I-frame	Intra-Frame
IGAE	Information Gain Attribute
JCT-CV	Joint Collaborative Team on Video Coding

k-NN	k-Nearest Neighbours
LBP	Local Binary Pattern
LCU	Largest Coding Unit
LD	Low Delay
MB	Macroblock
MC	Motion Compensation
ME	Motion Estimation
MPEG	Moving Picture Experts Group
MSE	Mean Square Error
MSM	Merge/Skip Mode
MV	Motion Vector
MVP	Motion Vector Prediction
OOB	Out-of-Bag
PCA	Principal Component Analysis
P-frame	Predicted-Frame
POC	Picture Order Count
PoV	Proportion of Variance
PSNR	Peak Signal-To-Noise Ratio
PU	Prediction Unit
QP	Quantization Parameter
Qs	Quantization Scale
RA	Random Access
RD	Rate-Distortion
RDO	Rate-Distortion Optimization

RM	Reduced Multivariate Polynomial Model
RMD	Rough Mode Decision
RPS	Reference Picture Sets
RQT	Residual Quad-Trees
SAD	Sum of Absolute Differences
SAO	Sample-Adaptive-Offset
SCC	Subjective-Driven Complexity Control
SCU	Smallest Coding Unit
SMD	Skip Mode Decision
TC	Texture Complexity
TU	Transform Unit
VCEG	Video Coding Experts Group
VLC	Variable Length Coding
WEKA	Waikato Environment for Knowledge Analysis

Chapter 1. Introduction

In this chapter, an introduction to HEVC is presented along with the enhancements it introduces and the encountered drawbacks. Then, a brief summary of the proposed solutions and their performance evaluation methodologies is reported followed by the thesis contribution. Finally, a general organization of this thesis is outlined.

1.1. Overview

The HEVC standard, also known as H.265 or MPEG-H Part 2, is one of the successors of the well-known standard MPEG-4 AVC (H.264 or MPEG-4 Part 10). It is designed to target high quality digital video, Ultra High Definition (HD) content and low bitrate applications. The HEVC project was formally launched in January 2010, when a joint Call for Proposals was issued by the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) [1]. When it was completed in January 2013, the HEVC standard was found to offer a significant improvement to the compression performance relative to that presented by existing standards. In fact, HEVC currently provides twice the compression capabilities as that offered by its predecessor, the Advanced Video Coding (AVC). With minimal video quality level losses, enhanced compression or coding efficiency is achieved by HEVC, where around 50% bit-rate reduction is possible given that the appropriate encoder settings are used [2]. Nonetheless, this improvement comes at the cost of increasing the encoding computational complexity, which can reach up to 40% in comparison to that of H.264/AVC when only essential coding tools are enabled [3].

The aforementioned enhancement can be contributed to a number of factors, which mostly involves the introduction of flexible partitioning structures. HEVC uses quad-tree Coding Tree Units (CTUs), Prediction Units (PUs), and residual quad-trees (RQTs) rather than macroblocks (MBs), which were utilized in MPEG-2 and MPEG-4. A video frame or image is divided into CUs with a typical size of 64×64 pixels. In order to achieve the best configurations in terms of structure partitioning, an exhaustive Rate Distortion Optimization (RDO) process takes place, which greatly intensifies the computational complexity. Most of the encoding time involves recursively repeating the RDO process at each coding depth level for each structure (i.e. 64×64, 32×32,

16×16 and 8×8 pixels), testing every possible encoding structure combination and selecting the one that minimizes the rate-distortion (RD) cost [4]. Further details about these structures will be presented in Chapter 2.

1.2. Thesis Objectives

As mentioned earlier, HEVC presents a significant coding efficiency improvement when compared to that of its predecessors at the cost of increasing the computational complexity. Thus, the prime challenge of this work involves limiting this computational complexity without hurting the compression efficiency.

For this purpose, a fast partitioning decision algorithm is introduced for CUs and PUs. Here, the aim is to optimize the RDO process as to prevent full search from taking place at each CU depth level for all CTUs. The proposed system employs different video sequence-dependent approaches using machine learning techniques. In the first scheme, features are recorded for all CUs, which are used to implement an early termination algorithm for coding trees. In the second scheme, an early termination algorithm for PUs is applied using recorded attributes, which is sequentially utilized as the data sample. The final scheme combines both approaches to provide early termination for both CUs and PUs. The proposed system looks at different machine learning algorithms to allow for the early termination process to take place. The features may undergo feature selection or extraction before being fed into the selected classifier. The performance of the proposed solutions is evaluated using BD-rate, BD-PSNR, excessive bitrate, CCR, model generation time, and classification rates.

1.3. Research Contribution

The contributions of this research work can be summarized as follows:

- Propose a machine learning approach to predicting the split decisions of CUs.
- Use a video sequence-dependent approach to generate training models.
- Extract novel feature variables for CUs from both the underlying CU and previously encoded ones.
- Reduce the dimensionality of feature variables prior model training using a variety of dimensionality reduction techniques.
- Propose a novel early termination algorithm for PUs using dimensionality reduction and classification algorithms.

- Combine both CU split prediction and PU mode prediction to reduce the HEVC's computational complexity without significantly harming the video quality.

1.4. Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 provides background information on video encoding and different machine learning algorithms. Moreover, related work done in this field of research is discussed. The architecture of the proposed system is described in Chapter 3, while the experimental setup is given in Chapter 4. Chapter 5 focuses on the experimental results and analyses the performance of each of the proposed solutions. Lastly, Chapter 6 concludes this thesis and outlines possible future work.

Chapter 2. Background and Literature Review

The prime focus of this thesis involves attempting to minimize the computational complexity introduced during HEVC encoding without sacrificing the compression efficiency. In this chapter, video compression or coding is first explored, where the architecture and components present in MPEG-2 and HEVC, two video coding standards, are presented. Then, different machine learning algorithms are discussed alongside the motivation of using them to reduce the HEVC's computational complexity during the encoding phase. Additionally, the performance evaluation metrics are explored in this chapter. Finally, existing related work in this field of research is discussed.

2.1. Encoding

Data compression and decompression are frequently used concepts, especially when large amounts of data require storage and/or transmission. In a similar fashion, such a process takes place for audio-visual content. In multimedia systems, a number of stages exist during the coding process of a video, which is illustrated by the simplified block diagram shown in Figure 2.1.

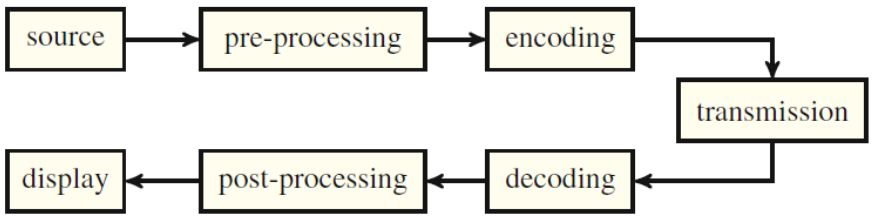

Figure 2.1: High-level overview of video coding as illustrated in [5].

In general terms, the raw uncompressed digital video sequence, to which pre-processing techniques such as trimming may apply, is fed into the encoder. The encoder converts the digital input into a coded bitstream, which usually involves a combination of lossless and lossy compression in order to meet the target transmission bitrate constraints. The bitstream is then either stored or transmitted over a channel to the receiver. At the receiver, decoding takes place, where the bitstream is transformed into a reconstructed video sequence. As lossy compression is involved in the process, a reconstruction error is introduced as some bits may be omitted in the encoding process, affecting the compression efficiency. Transmission error can also play a role in the

quality of the reconstructed video, which varies based on the channel over which the bitstream was transmitted. Finally, post-processing schemes may be applied as necessary to help in improving the video's quality.

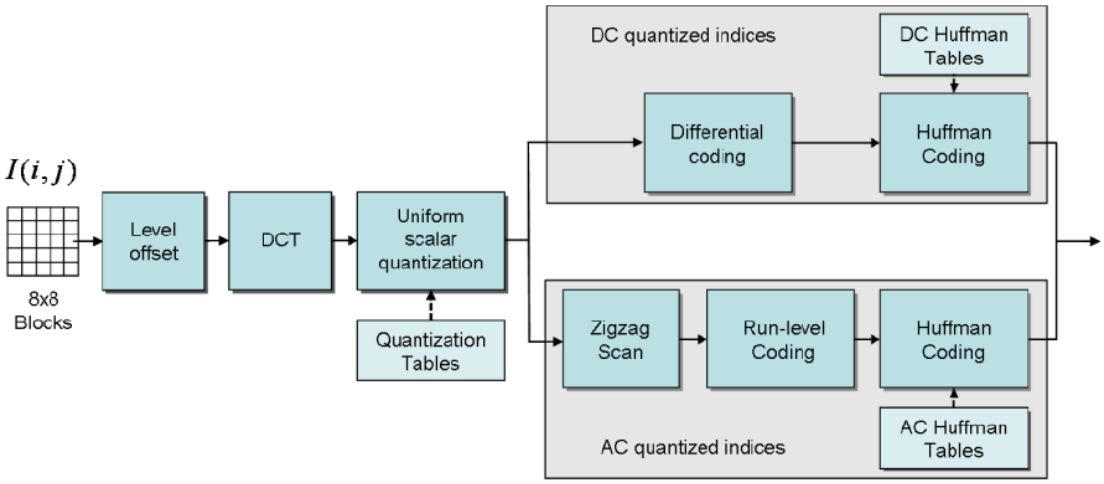

Figure 2.2: Block diagram showing how a block of pixels gets encoded [6].

The process of encoding, summarized in Figure 2.2, mainly involves generating frequency-based representations of pixel blocks of each frame in the video. Here, some high frequencies are usually discarded as they do not provide significant information in comparison to the low frequency ones. This is where lossy compression takes place and the bit omission mentioned earlier occurs. Regardless of which coding standard is used, some type of transformation algorithm such as the Discrete Cosine Transform (DCT) is performed on each pixel segment for all frames of the video, where pixels are transformed into their frequency representations. The exact partitioning manner will be discussed shortly once MPEG-2 and HEVC are explained. It should be noted that the frames are coded in what is called a coding order. After a block of pixels undergoes DCT, it is quantized, where lower frequency components of the transformations tend to be emphasized. A quantization scale is involved, where the higher its value is, the more information is lost. The top-left corner of the transformed block, which the zero-frequency coefficient (DC), is delta encoded by storing the difference between its value and the corresponding value from the previous block. The remainder of the block, which is scanned in a zigzag fashion, typically contains several long runs of zeros as a result of quantization. Therefore, these coefficients (ACs) are compressed using run-length encoding. The resultant of delta or differential coding and run-length coding

undergo Huffman coding, a variable length coding (VLC) algorithm, which is a lossless compression scheme to further increase the compression rate. This process is repeated, leading to a bitstream that is later stored or transmitted.

2.1.1. MPEG-2. Taking the encoding process described earlier in the context of MPEG-2 will help in understanding the procedure in more depth. In MPEG-2, each frame is organized in slices, where each slice is independently coded and consists of a set of adjacent blocks called macroblocks (MBs). Each MB constitutes a 16×16 block of luminance or grayscale samples, which are divided into four 8×8 blocks, and two 8×8 blocks of the matching chrominance (C_b and C_r), as seen in Figure 2.3.

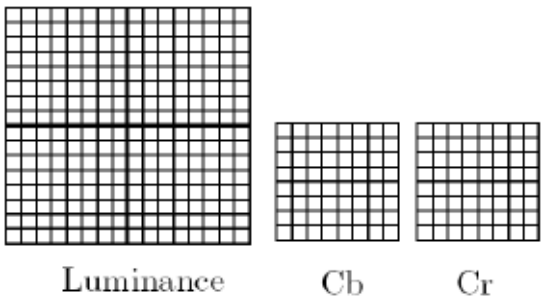

Figure 2.3: Structural composition of a MB structure [7].

During the coding process in MPEG-2, a raster scan is followed. Here, MB acts as the partitioning structure previously mentioned. A block diagram illustrating an overview of encoding in MPEG-2 can be seen in Figure 2.4.

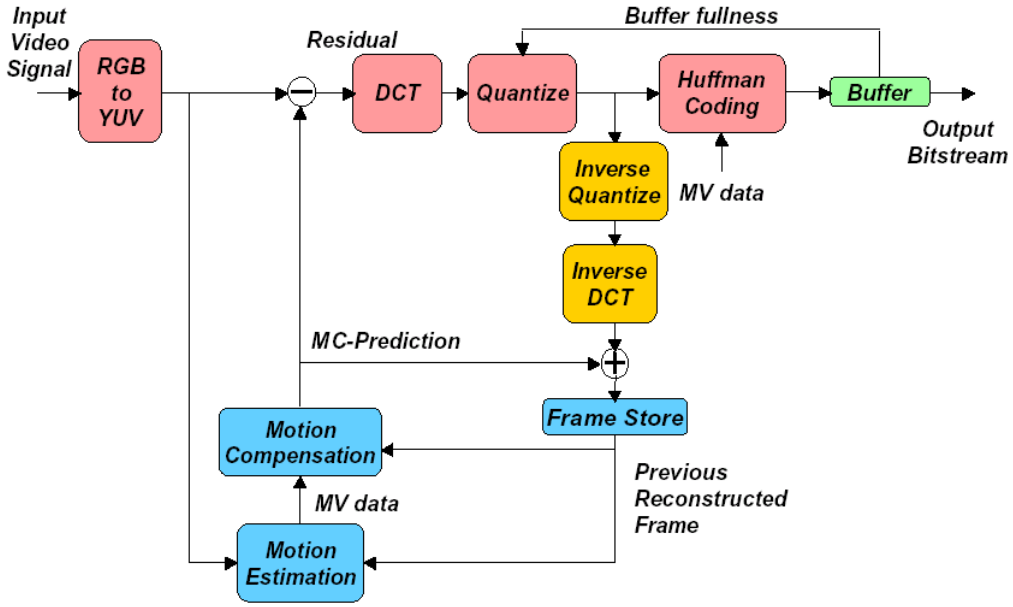

Figure 2.4: Detailed block diagram of a MPEG-2 encoder as illustrated [7].

At the encoder, the first frame, which is an intra-frame (I-frame), goes through DCT, quantization, and VLC. The local decoder is then used to generate the reconstructed I-frame. This is crucial as to avoid picture drift since the reconstructed frame will be used at the decoder. The reconstructed I-frame is stored into the frame store. When the next frame enters the encoder, motion estimation and motion compensation (MC) take place and the motion compensated reconstructed I-frame is subtracted from the current frame. In more details, motion estimation in the previous I-frame using a search window occurs, where motion vectors (MVs) are extracted. After that, motion compensation occurs, where a motion compensated picture of the previous frame is generated by taking best match location for each MB and inverting the MV corresponding to it. Thus, the MBs will coincide and the motion compensated picture can be subtracted from the current picture. Then, DCT, quantization, and VLC happen, and the process is repeated. It is important to note that the first frame in a video is an I-frame, which is intra-coded. This means that it is compressed by doing DCT, quantization, and DC and AC coding, as previously discussed.

The quantization scale (Qs) varies between 1 and 31 in MPEG-2, which is stored as the difference between the current and previous block for all blocks excluding the first one as the initial quantization scale factor is stored in the slice header. The following frames can be an I-frame, a Predicted-frame (P-frame), or a Bi-directional-frame (B-frame). A P-frame can be predicted from a previous reference frame within a group of pictures (GOP), while a B-frame can be predicted using previous and future reference frames. Here, each MB has a choice of either using motion estimation and compensation, which allows inter-coding, or immediately utilizing DCT and quantization, which allows intra-coding. This mainly depends on the MB type. On the other hand, prediction, which takes place when motion estimation and compensation are involved, can be of the types forward, backward or interpolated/bi-directional. A MB can also be skipped based on some requirements.

2.1.2. HEVC. HEVC operates in a fashion very similar to that of MPEG-2 with a number of additions that introduces significant improvements in the encoding efficiency. Figure 2.5 illustrates a block diagram representing a HEVC encoder.

One of the prime contributors to such an enhancement is the block partitioning structures, namely CUs, PUs, and Transform (TUs). In HEVC, each frame is also

divided into a set of slices, each of which is composed of equal-sized CTUs. Each CTU consists of one luminance coding tree block (CTB) and two chrominance CTBs. A CTU can be recursively further partitioned into smaller blocks called CUs, which can be of the sizes 64×64, which is the largest CU (LCU), 32×32, 16×16, and 8×8, which represents the smallest CU (SCU). This partitioning process generates a quad-tree structure with multiple coding depth levels, as observed in Figure 2.6.

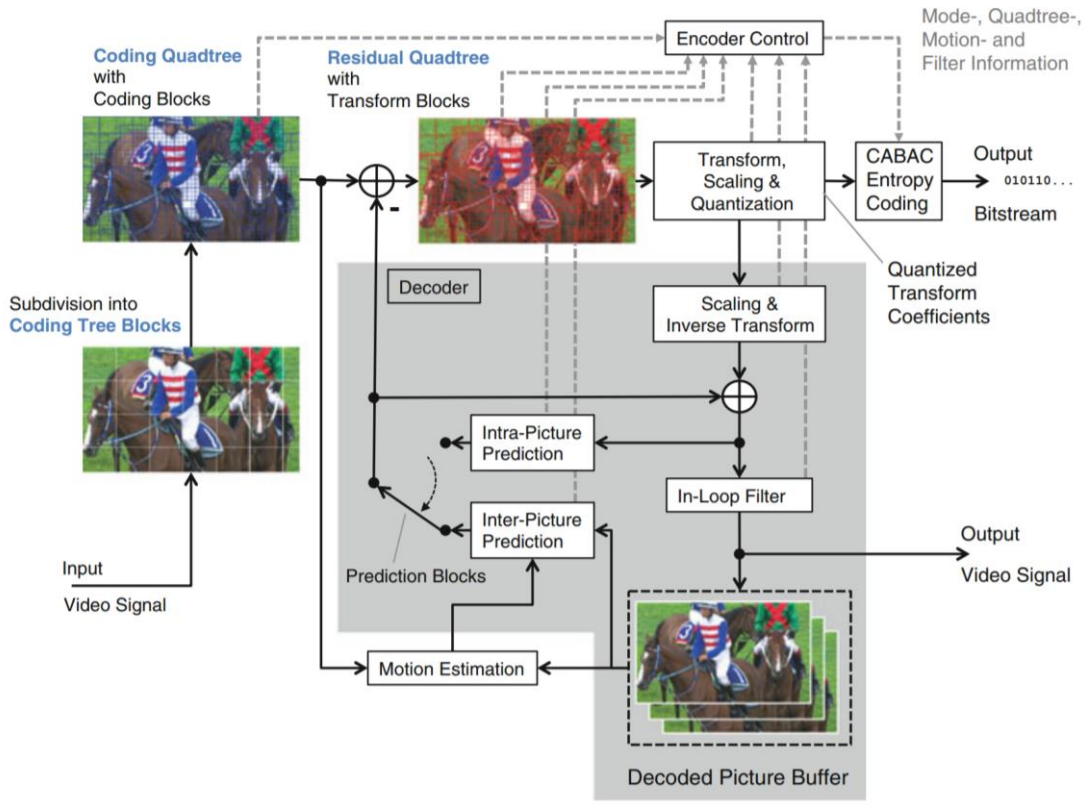

Figure 2.5: Detailed block diagram of a HEVC [8].

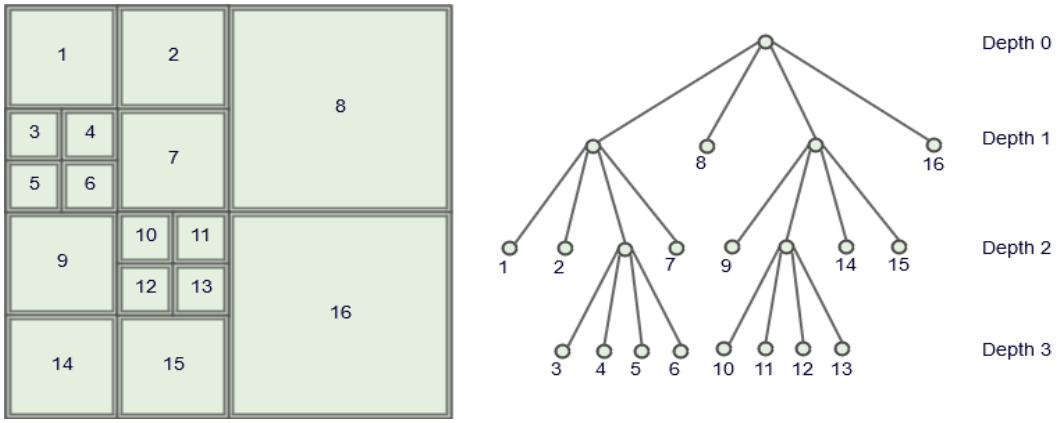

Figure 2.6: Block and tree representations of the CTU quad-tree structure.

During the splitting process, all partitioning possibilities $\mathcal{A}$ are evaluated in a RDO scheme based on the Lagrangian bit-allocation [8] represented in (1).

$$p^* = \arg\min_{\forall\, p \in \mathcal{A}} D(p) + \lambda . R(p), \tag{1}$$

where p^* represents the coding parameter that is determined by minimizing a weighted sum of the resultant distortion $D(p)$ and the associated number of bits $R(p)$. In other words, RDO denotes a measure of the amount of distortion affecting the quality of a video against the amount of data needed to encode that video. The Lagrange parameter λ, whose value is assigned based on the quantization scale, is a constant that determines the trade-off between $D(p)$ and $R(p)$.

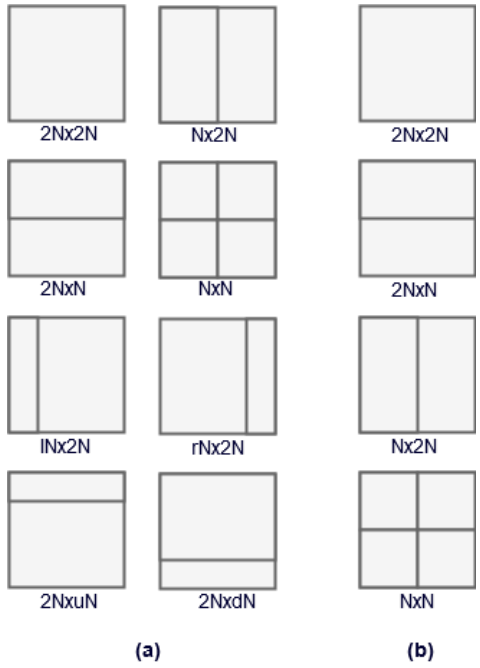

Figure 2.7: PU modes for (a) inter-coded CUs and (b) intra-coded CUs.

Each CU can also be divided into a number of PUs, which are predicted with either intra-frame or inter-frame prediction. Again, the optimal PU splitting mode is chosen through the RDO process, where all possible partitioning modes are evaluated. Figure 2.7 shows all possible PU partitioning modes excluding the Merge/Skip mode (MSM), which differ based on the current CU depth level. They can be of a symmetric or asymmetric type. In 8×8 CUs, asymmetric motion partitions (AMPs) are not tested to prevent the PUs smaller than 4×8 or 8×4. MSM is offered for all inter-predicted CU sizes and 2N×2N PUs, which is very similar to the skip mode in MPEG-2. MSM allows

the current PU to inherit the motion information from spatially and temporally neighboring PUs, resulting in a larger region.

The next structure to be evaluated is related to the transformation coding process, which was explained in Section 2.1, where transformation and quantization are involved. The quantization parameter (QP) acts in an identical manner to that of the quantization scale in MPEG-2. Here, each CU can be seen as the root of a quad-tree structure called residual quad-tree (RQT), which can be recursively partitioned into TUs. TU sizes can be of 32×32, 16×16, or 8×8 dimensionalities and does not depend on the underlying PU size. Once again, RDO determines that size for a given TU.

As previously explained in MPEG-2, prediction refers to the process of MV extraction. In HEVC, there is also a concept called Reference Picture Sets (RPS), which is divided into List0 and List1. List0 contains a list of picture order count (POC) used for forward prediction, while List1 contains a list of POC used for backward prediction. POC refers to the fame number in output/display order. It is worth noting that the same reference picture can be used for bi-directional prediction. When it comes to representing a MV, it is in the form of $\{d_x, d_y$, POC index in List0$\}$ and $\{d_x, d_y$, POC index in List1$\}$, where d_x and d_y are directions in the x-axis and y-axis, respectively. In place of ME, Advanced Motion Vector Prediction (AMVP) is used to find Motion Vector Predictions (MVPs). Up to two spatial candidate MVPs are derived from five spatial neighboring blocks and one temporal candidate MVPs is derived from two temporal co-located blocks in case the two spatial candidate MVPs were not available.

2.2. Machine Learning

Machine learning involves giving a computer or a machine the ability to learn and adapt without the need to explicitly program them to perform a particular task. It uses a set of previously collected data instances to detect patterns in this data, learn a predictive model, and adjust the program actions accordingly. This is done with the aim of optimizing a performance criterion using the data sample. Several applications where machine learning is found to be useful include pattern recognition, search customer relationship management, spam filtering, medical diagnoses, etc. Each of these applications poses a machine learning scenario that depends on the nature of the training data available, the method used in collecting the training sample, and the test data used

for evaluation. Two of the commonly seen learning scenarios [9] include supervised learning and unsupervised learning.

Before moving any further, it is important to understand some of the key terminologies used in machine learning [10]. An example is simply a data item or instance that can be used in a machine learning algorithm. This example can be part of the training sample, validation sample, or testing sample. A training sample is used to train a learning algorithm to generate a predictive model, while a testing sample is needed to evaluate the performance of a learning algorithm after a model has been constructed. Each example in any of the samples constitutes of features or variables, which are basically a set of attributes that represents the various features of that instance. These features are also called predictors in machine learning. In classification problems, which falls under supervised learning as will be seen later in this section, a label is used to identify a category to which an instance belongs to. During the evaluation of a learning algorithm, given a testing example whose label is known, a loss function is produced to measure the prediction error using the example's predicted label and its true label. In most cases, a model is to be built using the machine learning algorithm, which allows the generation of discriminant functions that are part of a hypothesis. The discriminant functions split the sample space into different regions representing different categories. In other words, decision boundaries imposed by discriminant functions are created based on which a given test example with certain features can successfully be mapped to the correct label with minimal error occurrence.

After the brief explanation given on various terminologies used in machine learning, it will be easy to understand the previously mentioned learning scenarios. An overview of the machine learning algorithms explained in this section and used in this work can be seen in Figure 2.8. In supervised learning, a set of labeled examples are used for both the training and testing phases of the machine learning algorithm. This is usually used in classification problems, as opposed to dimensionality reduction, which applies unsupervised learning. This type of learning considers unlabeled examples. More details about classification and dimensionality reduction will be seen in the following subsections.

It is crucial to understand that model generation can be a tricky process as the model is required to generalize well [9]. In other words, the decision boundaries

achieved during the training process should not be very complex to the point that over fits the training data, leading it to not perform well on the testing sample. Figure 2.9 gives a simple example that illustrates the generalization issue.

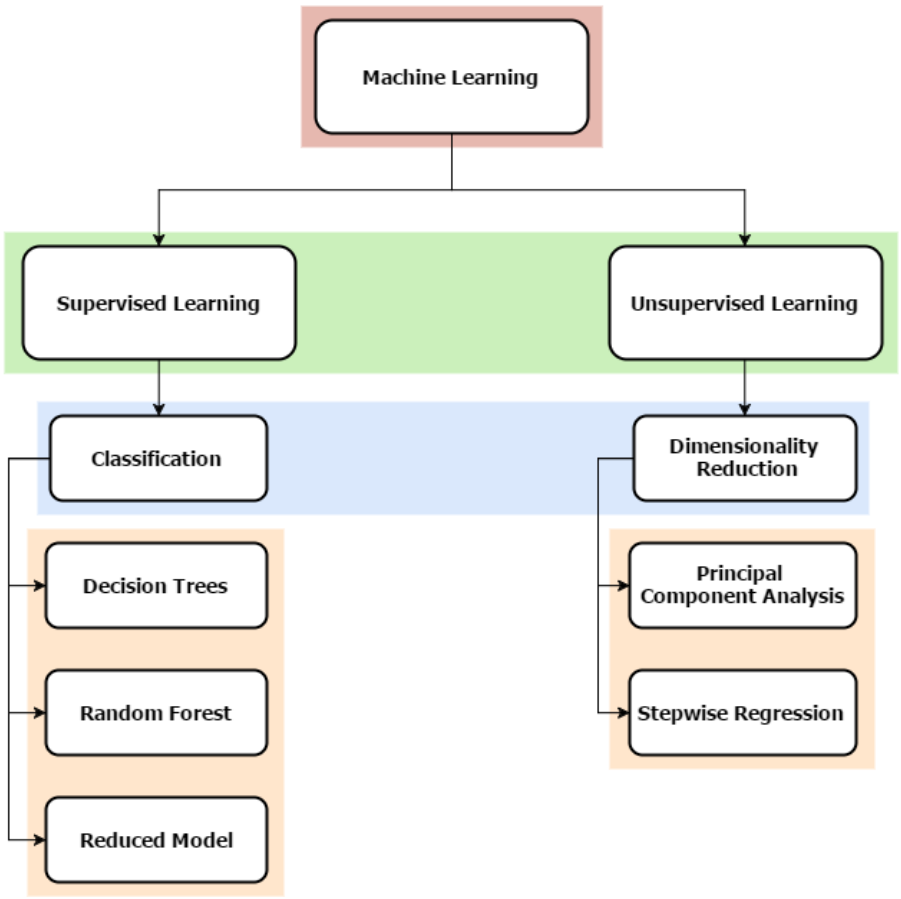

Figure 2.8: An overview of some machine learning algorithms and their learning scenarios.

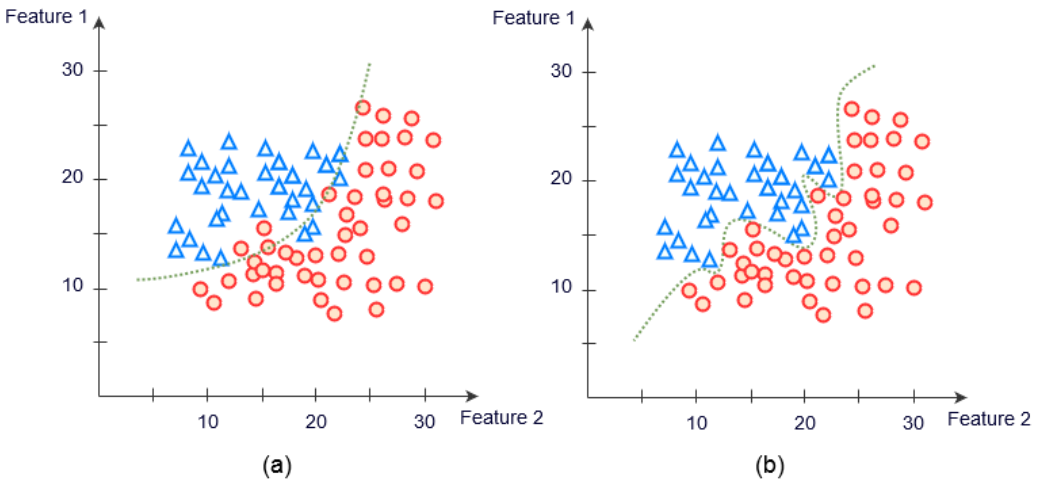

Figure 2.9: An illustration of different decision boundaries, where the less complex one (left) is likely to generalize better than the more complex one (right).

2.2.1. Classification models. During classification, labeled examples are considered, where a category is allocated to each instance. As mentioned earlier, each instance consists of values corresponding to the various selected features and its assigned label. A classification algorithm is then selected based on the nature of the problem to be tackled to build a predictive model. The goal is to optimize the discrimination between the data points from different classes represented by different labels, minimizing the error objective criterion. The following are the different classification algorithms utilized in this work.

2.2.1.1. ***Decision trees.*** A decision tree utilizes a "divide-and-conquer" approach to learn from a set of independent training instances and generate a tree-like model consisting of rules and possible outcomes or classes. It allows the classification process of a new instance to take place in a systematic manner. This tree can be constructed in a top-down recursive fashion [11] using algorithms such as ID3 [12] by Quinlan (1986), C4.5 [13] by Quinlan (1993), and CART [14] by Breiman et al. (1984).

C4.5, which is one of the classification algorithms used in this work, is an evolution of the ID3 algorithm. The way this decision tree inducer works is by following a series of simple steps. However, before exploring these steps, it is important to look at the attribute selection measure that it uses as the splitting criterion. Unlike ID3, which uses information gain as its attribute selection measure, C4.5 uses an extension to information gain known as the gain ratio. The issue with the information gain measure is that it is biased toward selecting attributes that have a bigger range of values. To overcome this issue, gain ratio applies some sort of normalization to the result of the information gain measure.

Let S be the set of training samples of length s with m distinct classes and T be the set of testing instances. To compute the gain ratio [15], based on information theory, the entropy or expected information needed to classify a given sample is first calculated and is given by

$$I(S) = -\sum_{i=1}^{m} p_i \log_2(p_i), \tag{2}$$

where p_i is the probability of a sample belonging to a particular class C_i. In the case of discrete attributes, let an attribute A have v distinct values, which can be represented as

$[a_1, a_2, \ldots, a_v]$. On the other hand, in the case of continuous attributes [16], whose values are numeric, $A \leq t$ is considered, where t represents a threshold value. This results in each value of A to have two outcomes: True and False. The vector A can be used to split S into v different subsets, denoted as S_j that is split on an a_j value. For continuous attributes, the threshold t, which is a possible split-point can be found by sorting the values corresponding to particular A in the training sample and taking the average of adjacent values. The entropy achieved by partitioning S into v subsets is defined as

$$I_A(S) = \sum_{j=1}^{v} I(S_j) \frac{|S_j|}{|S|}. \tag{3}$$

The gained encoding information by branching on A is

$$Gain(A) = I(S) - I_A(S). \tag{4}$$

As previously mentioned, gain ratio applies normalization to the information gain using a value that is given by

$$SplitInfo_A(S) = -\sum_{j=1}^{v} \left(|S_j|/|S|\right) \log_2\left(|S_j|/|S|\right). \tag{5}$$

This value represents the information generated by splitting S by A. The gain ratio can now be defined as

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(S)}. \tag{6}$$

The attribute with the highest gain ratio is selected as the splitting attribute to partition the current tree.

Moving back to discussing the way C4.5 works [17], at the beginning, the entire S set is placed at the root of the tree. Gain ratio is then used as the splitting criteria to determine the attribute that best differentiates the instances in S. The selected attribute is used as the value of the current tree node, which is the tree root in this case. Next, edges from this node are created, which represent a unique value for the chosen attribute. This process is repeated to further split the subsets of S until either the number of instances to be split is below a certain threshold or there are no remaining attributes to perform further partitioning. The leaf node is denoted based on the majority class,

where the aim is to reduce the impurity or uncertainty in data in order to decrease the misclassification error.

In order to classify a new unknown instance from the testing set T, it is routed down the grown tree based on attribute values. Once a leaf is reached, the instance is labeled according to the class assigned to that leaf. An example of a simple decision tree is illustrated in Figure 2.10, where only 2 attributes are considered, Feature 1 and Feature 2. Figure 2.10(a) demonstrates the splitting process in a 2D space with the order of partitioning shown. While Figure 2.10(b) shows the corresponding grown tree, which was done based on the splitting criterion previously discussed. In this work, J48 is used to build decision trees, which is an implementation of C4.5 in Waikato Environment for Knowledge Analysis (WEKA).

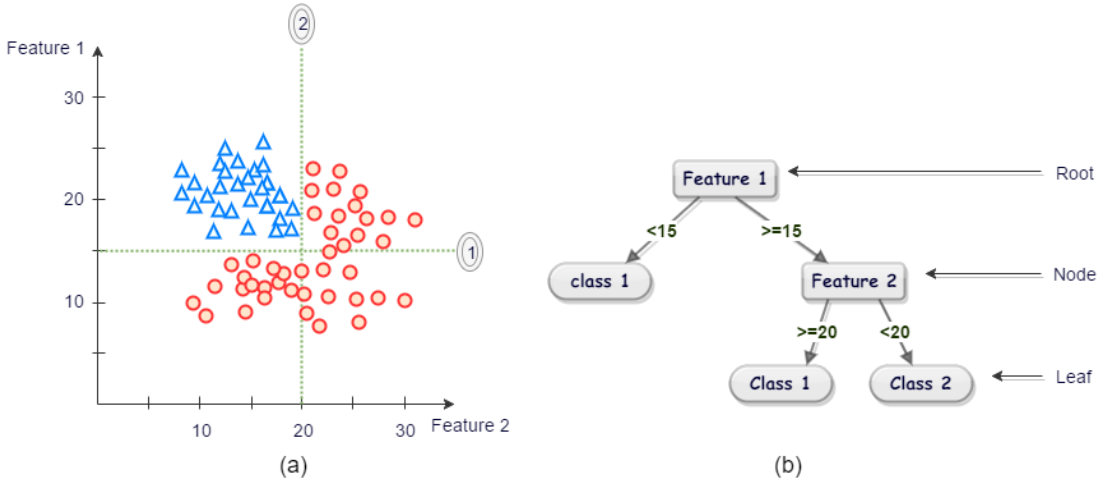

Figure 2.10: A simple example of a growing a decision tree.

2.2.1.2. *Random forest.* One of the issue of decision trees such as CART and C4.5 is that they tend to have empirically high variance [18]. In other words, these decision tree inducers are quite sensitive to the data used during the training phase. If the training set changes, the grown tree is likely to in turn change, resulting in producing different predictions. It is also important to note that these decision tree inducers employ a greedy approach that minimizes error by selecting the optimal attribute that splits the dataset at each node based on a certain data partitioning criterion.

In an effort to overcome the above, random forests [19] were introduced, where a random forest is an ensemble method that utilizes both bagging and decision trees. An ensemble method is a simple method that uses multiple machine learning algorithms

to enhance the classifiers' predictive performance. Bagging or bootstrap aggregation [20] is a procedure that is used to minimize the high variance seen in decision trees. It allows the resampling of a given training dataset, where large numbers of same-sized smaller samples called bootstrap samples are selected with replacement from the original dataset. This results in growing trees that are more independent. To tackle the drawback imposed by applying a greedy approach such as decision trees, a random forest [19] does not consider all the attributes and their values at the root of each tree to apply the splitting criterion. Instead, each tree is assigned a set of randomly selected features on which the splitting measure is applied.

Before looking at how a random forest operates, it is crucial to understand the splitting criterion it uses, which is the Gini index. As was the case for decision trees, let S be the set of training samples of length s with m distinct classes and T be the set of testing instances. The Gini index [15], which is another impurity measure similar to the gain ratio, is given by

$$Gini(S) = 1 - \sum_{i=1}^{m} {p_i}^2, \tag{7}$$

where p_i is the probability of a sample belonging to a particular class C_i. For each attribute A, the Gini index considers a binary split. The values of A, a_j, differs depending on its nature, which can be discrete or continuous as explained in the case of decision trees. Given that attribute A has v distinct values, the weighted sum of each of the partitions on A is computed as

$$Gini_A(S) = \sum_{j=1}^{v} \frac{|S_j|}{|S|} Gini(S_j). \tag{8}$$

The subset S_j that generates the minimum Gini index for A is selected as the splitting attribute to partition the current tree.

Figure 2.11 provides a visual explanation of the procedure followed to grow a random forest. For each tree, a random bootstrap sample of size N with replacement from the training set S is first taken [19]. Moreover, given M predictors or attributes, a random sample of constant $m \ll M$ predictors is selected for each tree. Based on the Gini index explained earlier, the attribute that best splits the sample space S is selected.

This process is repeated until the tree is as large as possible, without applying any pruning. It is worth noting that about one-third of the training dataset selected for a particular tree is left out of the sample and is called out-of-bag (OOB) data. This data is used to estimate the classification error as more trees are added to the forest and the variable importance.

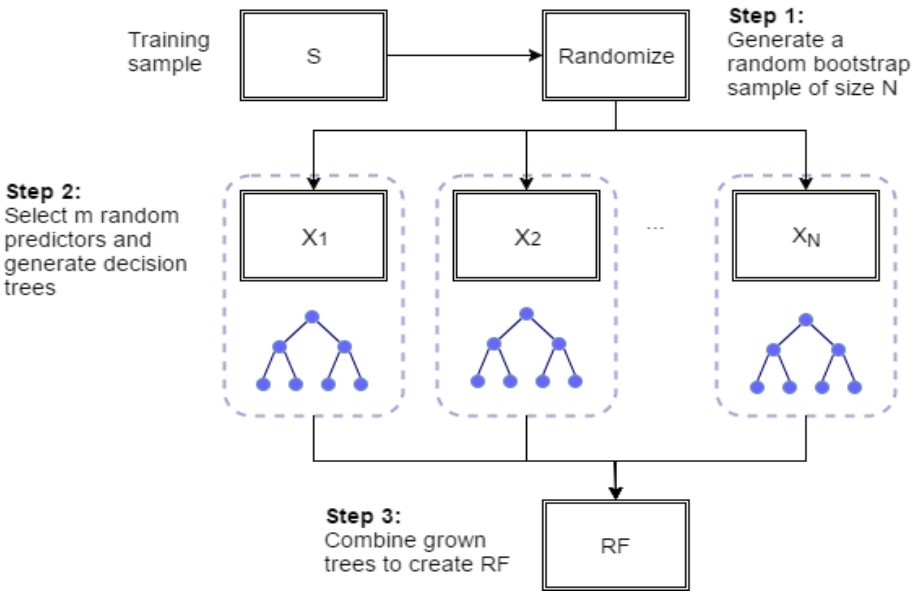

Figure 2.11: A simplified visual illustration of how to generate a random forest.

To classify an instance from the testing set T, the instance is routed down each of the grown trees in the forest based on attribute values. The instance is then labeled with the class that was assigned by most of the trees in the forest.

2.2.1.3. ***Reduced model.*** A multivariate polynomial model is capable of describing complex nonlinear relationships; however, for an rth order model with a l dimensional input or FV, the number of independent adjustable parameters can exponentially grow to up to l^r [21]. For that reason, Reduced Multivariate Polynomial Model (RM) is used instead, which provides approximately the same classification capabilities.

In order to generate the Reduced Model, a multinomial, a special case of multivariate polynomials, is first considered and can be expressed as

$$\hat{f}_{MN}(\alpha, x) = \alpha_0 + \sum_{j=1}^{r} \alpha_j (x_1 + x_2 + \cdots + x_l)^j, \tag{9}$$

where all inputs are lumped within each power term. Here, r is the degree of approximation, α_j is the weight parameter to be estimated, and $x = [x_1, x_2, \dots, x_l]^T$ is the regressor or FV containing l inputs. The above results in a non-linear estimation model with the weight parameters needing to be estimated in an unconventional manner. Therefore, a linearized model is considered.

On the multinomial function that is differentiable, two point α and α_1 are considered [22]. Taking only the FVs into account to simplify the expression, by the Mean Value Theorem, the multinomial function $f(\alpha) = \left(\alpha_{j1}x_1 + \alpha_{j2}x_1 + \cdots + \alpha_{jl}x_l\right)^j$ about reference point α_1, given that $j = 2, \dots, r$, can be re-written as

$$f(\alpha) = f(\alpha_1) + (\alpha - \alpha_1)^T \nabla f(\bar{\alpha}), \tag{10}$$

where $\bar{\alpha} = (1 - \beta)\alpha_1 + \beta\alpha$ for $0 \leq \beta \leq 1$. Including the summation of the weighted input terms back after removing the reference point α_1, the coefficients within $f(\alpha)$ and the gradient $\nabla f(\bar{\alpha})$ leads to the following multivariate model

$$\begin{aligned} \hat{f}_{RM'}(\alpha, x) = \alpha_0 + \sum_{j=1}^{l} \alpha_j x_j + \sum_{j=1}^{r} \alpha_{l+j} (x_1 + x_2 + \cdots + x_l)^j \\ + \sum_{j=2}^{r} \left(\alpha_j^T . x\right)(x_1 + x_2 + \cdots + x_l)^{j-1}, l, r \geq 2. \end{aligned} \tag{11}$$

The above formulation can be expanded to include more individual high-order terms, modifying the Reduced Model to become

$$\begin{aligned} \hat{f}_{RM}(\alpha, x) = \alpha_0 + \sum_{k=1}^{r} \sum_{j=1}^{l} \alpha_{kj} x_j^k + \sum_{j=1}^{r} \alpha_{rl+j} (x_1 + x_2 + \cdots + x_l)^j \\ + \sum_{j=2}^{r} \left(\alpha_j^T . x\right)(x_1 + x_2 + \cdots + x_l)^{j-1}, l, r \geq 2, \end{aligned} \tag{12}$$

where the total number of terms K is given by $K = 1 + r + l(2r - 1)$.

(12) is used to perform the classification presented in this work. It is worth mentioning that the weights denoted by α will be omitted during the expansion process.

Thus, the weights are recalculated using (13), which is a resultant of minimizing the objective function given by (14).

$$\alpha = (P^T P + bI)^{-1} P^T y \tag{13}$$

$$\begin{aligned} s(\alpha, x) &= \sum_{i=1}^{m} \left[y_i - \hat{f}_{RM}(\alpha, x_i) \right]^2 + b\|\alpha\|_2^2 \\ &= [y - P\alpha]^T [y - P\alpha] + b\alpha^T \alpha \end{aligned} \tag{14}$$

Here, for m data points, $P \in \mathcal{R}^{mxK}$ is the Jacobian matrix related to the expanded FVs, $y \in \mathcal{R}^{mx1}$ is the known inference vector from the training data, b is a regularization constant and I represents a $K \times K$ identity matrix. The $\|.\|_p$ operator is the second norm, where $p = 2$.

2.2.2. Dimensionality reduction. Dimensionality reduction is considered an important step, especially when the FV is relatively large since it can affect the classification process. In order to classify an observation correctly, it is preferable that the data points belonging to a particular class are clustered such that a certain density is reached, allowing the discrimination of those points from points belonging to a different class. However, maintaining the same density with more features is no easy task as more data points will be needed as a result, which is usually not possible. Consequently, feature selection and extraction algorithms are utilized during training, whose operation are explained in the following subsections.

2.2.2.1. *Random forest feature importance.* One of the features presented in a random forest classifiers is called the random forest variable importance. It analyzes the importance of a particular attribute in predicting the correct classification of a given test instance. First, for all grown trees, the number of correct classifications achieved using the OOB data is computed. The OOB data is the set of instances that were left out during the training process of a given tree in the random forest. Let A be an attribute having v distinct values. From the OOB data, each of the v distinct values are randomly permuted and tested for correct classification. The raw importance score is then given by

$$raw\_imp_A = \frac{correct\_class - correct\_class_A}{trees\_count}, \tag{15}$$

where $correct\_class$ is the number of correct classifications attained before applying the permutation and $correct\_class_A$ represents the number of correct classifications accomplished after applying the permutation. $trees\_count$ is total number of trees grown in the random forest. A given attribute is said to be important if it has a high raw importance score.

2.2.2.2. *Principal component analysis (PCA).* PCA [9] is an unsupervised projection method that allows projecting the training set S on a lower dimensional space. Given that the original feature space d, PCA employs a feature extraction approach that projects S on k dimensions, where $k < d$, with the objective of minimizing any information losses. This is achieved by maximizing the feature variance.

Let the projection of S in the direction of ω_1 be given by $z_1 = {\omega_1}^T S$. Additionally, let the variance of the projected training sample to be $Var(z_1) = {\omega_1}^T \sum \omega_1$, where $\sum$ denotes the covariance of S. To maximize (z_1) , it is important to subject it to $\|\omega_1\| = 1$, which is the L2-norm. This leads to a Lagrange problem that is expressed as

$$\max_{\omega_1} {\omega_1}^T \sum \omega_1 - \alpha({\omega_1}^T \omega_1 - 1) = 0. \tag{16}$$

The resultant of simplifying the above expression is $\sum \omega_1 = \alpha \omega_1$, where ω_1 is the eigenvector of $\sum$. The eigenvector ω_1 with the largest eigenvalue α maximizes $Var(z_1)$ and is selected as the first principal component. To find the second principal component, $Var(z_2)$ is maximized by subjected it to $\|\omega_2\| = 1$ and ensuring that it is orthogonal to ω_1. The expression then becomes

$$\max_{\omega_2} {\omega_2}^T \sum \omega_2 - \alpha({\omega_2}^T \omega_2 - 1) - \beta({\omega_2}^T \omega_1 - 0) = 0. \tag{17}$$

The resultant of simplifying the above expression is $\sum \omega_2 = \alpha \omega_2$, where ω_2 is another eigenvector of $\sum$. The eigenvector ω_1 with the largest eigenvalue α that maximizes $Var(z_2)$ is chosen as the second principal component. This process is usually repeated until the desired Proportion of Variance (PoV) explained is reached. Given that α_i are sorted in descending order, PoV is expressed as

$$\text{PoV} = \frac{\alpha_1 + \alpha_2 + \cdots + \alpha_k}{\alpha_1 + \alpha_2 + \cdots + \alpha_k + \cdots + \alpha_d}. \tag{18}$$

Typically, PoV is taken to be greater than 90% since it results in minimum information loss.

2.2.2.3. *Stepwise regression.* Stepwise regression is a subset/feature selection algorithm, where k of the d FVs are chosen with the aim of increasing discrimination or the classification rate given that $k < d$. In general, feature selection can either be forward, where the best feature is added at each step starting from the empty set until the best model is generated, or backward, where the starting point is a model with all features that are eliminated one-by-one, if possible, until the optimal model is reached. Stepwise regression is a combination of both schemes [23].

Given a set of variables $[x_1, x_2, \dots, x_l]^T$ that belong to a class r, f_{in} is the forward random (f-random) variable for adding a variable to the model and f_{out} is the f-random variable for removing a variable from the model. A f-random variable is a variable with the largest Pearson product moment correlation with r. At first, all variables are scanned and the variable with the highest statistics f is added to generate a one-variable model given by

$$h(x) = \alpha_0 + \alpha_1 x_1, \tag{19}$$

where $h(x)$ is the hypothesis and x_1 is one of the k features with the highest f value. For the remaining $k - 1$ variables, the variables are examined to choose the second best feature x_2, such that a two-variable model is generated in the form of

$$h(x) = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2. \tag{20}$$

x_2 is added such that $f_2 > f_{in}$, where f_2 is the statistics of x_2. f_2 is computed as

$$f_2 = \frac{SS_R(\alpha_1|\alpha_2\alpha_0)}{MSE(x_1, x_2)}, \tag{21}$$

where SS_R represents the regression sum squares error and MSE represents the mean square error. Next, f_1 is compared to f_{out} to check as to whether x_1 should be removed, where f_1 is calculated in a manner similar to that of f_2 as follows

$$f_1 = \frac{SS_R(\alpha_2|\alpha_1\alpha_0)}{MSE(x_2, x_1)}. \tag{22}$$

The same steps are repeated for the remaining $k-2$ variables until no more variables can be added to or removed from the model. The resultant of stepwise regression is the indices of the retained FVs.

2.2.3. Normalization. In order to normalize the FVs, z-score or zero-mean normalization is used, which acts as a measure of the distance between a data point (x_i) and the mean (μ) in terms of standard deviations (σ). It is given by

$$\mathrm{z} = \frac{(x_i - \mu)}{\sigma}. \tag{23}$$

Normalization is needed to standardize the range of independent features. Distinctive features can have different scales, which may affect the classification process.

2.2.4. PSNR, BD-rate, and BD-PSNR. When it comes to images, PSNR, given by (24), acts as a quality metric between the original and reconstructed image. The higher the value of this ration, the higher is the quality of the reconstructed image. It uses the Mean Square Error (MSE), which presents the cumulative squared error between the original and reconstructed image.

$$PSNR = 10.log_{10}\left(\frac{MAX_I^2}{MSE}\right) \tag{24}$$

Likewise, when it comes to video sequences, similar performance metrics exist to evaluate the quality of the reconstructed video in comparison to the original one. Bjontegaard's metrics BD-rate and BD-PSNR allow the computation of the average percentage saving in bitrate and the average gain in PSNR between two RD curves, correspondingly [24]. However, they do not take the encoder's complexity into account. Ideally, as BD-rate increases, BD-PSNR should decrease. In this work, we compare our coding solution to the regular HEVC coding approach. We aim to reduce the BD-rate and increase the BD-PSNR as must as possible. These performance metrics are among others that are used in the evaluation of the proposed solution in this thesis.

2.3. Related Work

Several state-of-art early termination algorithms for optimizing the encoding process in HEVC can be found in the literature, where the prime aim involves reducing the computational complexity while minimizing any performance degradation. Among

many, some approaches utilize the textural or structural characteristics of a given CU [25]-[35], while others use machine learning techniques [36]-[45], where they provide a method to view the issue at hand as a classification problem. The optimizations were not limited to HEVC inter-coding as some also considered enhancing intra-coding [31], [32], [34], [36]. It is important to understand that some approaches attempted to implement complexity control schemes, while others tried to apply complexity reduction algorithms to enhance the encoding process.

As previously mentioned, many papers focused on investigating the textural or structural characteristics of CUs at a given CU depth to optimize the HEVC encoding procedure. The work in [25] proposes an inter-prediction optimization scheme, where the CTU structure is analysed in a reverse order. Exploring CUs at higher depths first allowed the limiting of the PU modes to be tested and the speeding up of the motion estimation process. The encoding time was, as a result, enhanced by around 16.3% to 36.6% with BD-rate losses of around 0.3% to 2.2%. Alternatively, a subjective-driven complexity control (SCC) approach is proposed by [26] to control the HEVC encoding complexity. The authors investigated how the maximum depth of all largest CUs (LCUs) affects the encoding complexity and visual distortion. Based on that, an optimization formulation was computed in order to control the encoding complexity of HEVC with minimal visual distortion. It was observed that the encoding complexity greatly varied, where it could reach as low as 20% with the smallest complexity bias being 0.2%. However, the approach excelled in terms of control accuracy and visual quality. Another complexity control algorithm is proposed in [27], where an early termination condition is defined at each CU depth. The different parameters of the algorithm that determines the early termination condition dynamically changes on the fly based on the content of the video sequence being encoded, the configuration files and the target complexity, which can vary over time. A target complexity reduction of up to 60% was attainable while maintaining good results in terms of accuracy and coding efficiency.

In [28], the authors present a hierarchical structure-based fast mode decision scheme. The paper utilizes the depth information of co-located CUs to predict the current LCU's splitting. After that, the inter-prediction residual was analysed to optimize the PU mode decision process. Finally, fast discrete cross difference (FCD)

was used to predict the dominant direction of the current CU. The results accomplished from the algorithm reduced the encoding time by around 54.0%-68.4% with minimal degradation in the videos' quality. A fast CU decision algorithm is presented in [29], where the coded block flag and RD cost are checked to determine if intra- and inter-PUs are to be skipped. Experimental results demonstrate that the proposed algorithm saves around 35.39% of the encoding time with negligible losses. In [30], a two-layered motion estimation based fast CU decision process is proposed, which uses the latent Sum of Absolute Differences (SAD) estimation to extract the SAD costs for a CU and its sub-CUs. The relationship between the motion compensation RD cost and the SAD cost was then explored, from which the exponential model was generated and utilized to make the CU size decision. Consequently, an average encoding time saving of 52% and 58.4% with an average bit-rate increase of 1.61% and 2% were attained for Random Access (RA) and Low-Delay configuration, respectively. On the other hand, a fast encoding scheme is proposed in [31] to speed up the HEVC intra-coding to avoid running the full depth search procedure. Encoded CU depths and RD cost of co-located CTU were used to predict both the current CU's depth search range and the RD cost for CU splitting termination. Furthermore, PU modes to be used by the RDO process were limited through the fast mode decision step. This led to an encoding time reduction of 57% at the cost of a 0.6% increase in BD-rate.

Another fast CU size selection approach for I-frames is presented in [32], where local texture descriptors or image characteristics were used. CU split decisions were determined based on the histogram comparison of the Local Binary Patterns (LBP) of two consecutive CU depths. The speedup achieved here ranged on average between 5.4% and 80.2% with a performance loss of up to 0.87 dB in terms of BD-PSNR. [33] proposes an early texture-based inter-mode decision algorithm, where the current CU's texture, which was assigned based on the entropy, and the MV of the 2Nx2N PU mode were used for the early skip decision for the current CU. Furthermore, symmetric motion partition modes were optimized via the texture features calculated. This approach led to a reduction in the encoding time of around 40% with minimum performance degradation. A fast CU size decision algorithm for HEVC intra-coding is presented in [34]. Based on the texture and coding information of neighbouring CUs and adaptive thresholds built upon texture homogeneity, the splitting of the current CU

was limited. A CCR of up to 67%, was reached with around 0.06 dB loss in terms of PSNR and 1.08% increase in bitrate.

A spatiotemporal based CU encoding technique is explored in [35], where sample-adaptive-offset (SAO) parameters including the spatial encoding parameter were utilized to predict the texture complexity (TC) of the CU under encoding. The resultant was then used along with temporal encoding parameters such as MV and Transform Units sizes to enhance the early CU splitting decision. Moreover, RD cost comparisons of simple and complex TC classes, which are classified by the SAO parameters, were used to improve the early CU SKIP mode detection. The combined use of the proposed CU splitting decision and CU SKIP mode detection schemes gave rise to average encoding time savings of 49.6% and 42.7% with average BD-rate losses of 1.4% and 1.0% for RA and LD configuration, respectively.

Other approaches utilized the Bayesian decision rule and other machine learning techniques to improve the time complexity of an HEVC encoder. For instance, the proposed scheme in [36] involves three algorithms. The first is an early skip algorithm, where, based on the neighbouring PUs, RD cost computations for large PUs are skipped. The second is a PU skip algorithm using Bayes' rule that optimized the RD cost computation, while the third is a split termination algorithm that used the RD cost of rough mode decision (RMD) to prevent further PU splitting. An encoding time saving of 53.52% was achieved as a result while maintaining the same RD performance as that offered by the HM software. In contrast, three approaches involving a skip mode decision, a CU skip estimation, and an early CU termination are seen in [37]. The thresholds for each were allocated based on Bayes' rule with a complexity factor. The computational complexity was, as a result, reduced by 69% and 68% on average with a 2.99% and 2.46% BD-rate increase for RA and LD configuration profiles, respectively.

In [38], the authors present a joint online and offline learning-based fast CU partitioning method that uses the Bayesian decision rule to optimize the CU partitioning process. The proposed method was found to reduce the computational complexity to 53.6% on average with a 0.71% BD-rate increase for the RA configuration profile. The Bayesian decision theory is also utilized in [39] along with the correlation between the variances of the residual coefficients and the transform size to enhance the PU size

decision process. The algorithm was found to result in a 30-46% reduction in the computational complexity of transform processing with negligible coding efficiency losses. Alternatively, a fast CU splitting and pruning algorithm is proposed in [40], which is applied at each CU depth according to a Bayes decision rule method based on low-complexity RD costs and full RD costs. The various parameters governing this method are dynamically modified on the fly based on the varying signal characteristics. Experimental results performed using All-Intra (AI) configuration profile indicated that the approach could achieve around 50% complexity reduction with only 0.6% increase in the BD-rate. A fast CU size and PU mode prediction algorithm is provided by [41]. It utilizes the k-means clustering method to group 13 neighbouring CTUs into three classes that were used as reference CTUs to predict the current CTU's CU depth. In addition, rarely used PU modes were skipped, leading to PU selection complexity reduction. The encoding time was consequently reduced by 56.71% and 59.76%, and BD-rates by 1.0517% and 0.9918% for LD and RA configurations, respectively.

On the other hand, [42] presents an early mode decision algorithm based on Neyman-Pearson. In the paper, both skip mode and CU size decisions were modelled as binary classification problems with skip/non-skip and split/no-split class labels. The features used for model generation were the RD costs. Here, the Neyman-Pearson-based rule was used to balance the RD performance losses and the complexity reductions through minimizing the missed detection while limiting the incorrect decision rate. Online training and non-parametric likelihood estimation were utilized to update the RD cost probability density distribution for each QP at CU depth. The algorithm resulted in a CCR of 65% and 58% at the cost of a 1.29% and 1.08% increase in terms of BD-rate for RA and LD P configurations, respectively. In [43], a fast pyramid motion divergence-based CU selection algorithm is proposed, where a k-nearest neighbours (k-NN) like method is used to determine the optimal CU size. Experimental results show that an average time saving of 40% is achieved for LB-Main configuration profile with BD-rate losses of 2.21%. Whereas, an average time saving of 42.8% is attained for LP-Main configuration profile with BD-rate losses of 1.9%. The work in [44] utilizes a machine learning-based fast CU depth decision method to enhance the performance of a HEVC encoder, where the quad-tree CU depth levels are modelled as a three-level hierarchical binary decision problem. This was then used to

develop a flexible CU depth decision structure that allowed the generation of a three-output joint classifier that consists of multiple binary classifiers. Using Low Delay (LD) B-frame main configuration, the algorithm attained an average computational complexity of 51.45% with average BD-rate increase of 1.98% and BD-PSNR reduction of -0.061 dB.

The work proposed in [45] implemented early termination techniques on CUs, PUs, and TUs with the aim of optimizing the exhaustive recursive search process to avoid fully running the RDO algorithm. A set of decision trees were built from data collected during offline encodings with the aid of Waikato Environment for Knowledge Analysis (WEKA) [46], an open source DM tool. Based on the HEVC partitioning structure, the structure partitioning decisions were formulated into a data classification problem consisting of two classes, which was tackled through the usage of these decision trees. The attributes to be utilized for building the trees were chosen through the information gain attribute evaluation (IGAE) method in WEKA, which determined the information gain a variable offers. The C4.5 algorithm, specifically the J48 implementation, was used to train the decision trees, where best attributes or features were chosen and thresholds that are part of test nodes were computed. For each partitioning structure, observations from each class were taken to be of equal proportions to avoid data imbalance during training. An average CCR of up to 50% at the negligible cost of an increase of 0.56% in terms of BD-rate was obtained for RA profile when each of the schemes were separately implemented. Whereas, when jointly implemented, an average CCR of up to 65% was achieved with a compression efficiency loss of 1.36% in BD-rate.

Chapter 3. Methodology

In this chapter, the problem involving the HEVC video encoding complexity is formulated. Moreover, a proposed solution is discussed, where three early termination schemes are presented. Two of those schemes utilize two of the prime partitioning structures used during HEVC encoding, namely CUs and PUs, while the third combines both approaches. The proposed algorithms use different dimensionality reduction and classification algorithms to tackle the issue at hand.

3.1. Problem Formulation

As mentioned at the beginning of this thesis, one of the methods in which HEVC enhances the coding efficiency is through the usage of three flexible partitioning structures, i.e. CTUs, PUs, and the RQTs. Unfortunately, this improvement comes at the cost of massively increasing the computational complexity. It is true that the HEVC standard also uses several complexity reduction techniques, including Rough Mode Decision (RMD) for intra-frame prediction and the early Skip Mode Decision (SMD) algorithm; nevertheless, the computational complexity remains relatively high. As a result, it is crucial to apply different techniques to attempt and limit this rise in computational complexity without harming the compression efficiency in terms of video quality and bitrate consumption.

3.2. System Overview

To limit the increasing computational complexity, which is the resultant of the exhaustive rate-distortion optimization (RDO) process, a fast partitioning decision algorithm is introduced for both CUs and PUs using machine learning techniques. Both schemes implement a video sequence-dependent approach, where 10% of a given video sequence is considered to train a chosen classifier. The generated classification model is then used throughout the rest of the video sequence for testing. A general overview of the video sequence-dependent approach is given in Figure 3.1. Throughout this work, the HM reference software [47] is used. The assumption made for using a sequence-dependent approach is that since part of the same sequence is used for training a given model, the classification accuracy is likely to be higher. The behaviour of a data point from the training set belonging to a certain class is expected to provide a more accurate

methodology to predict the category to which a testing sample from the same video sequence belongs to.

Before training a given model, a video sequence is encoded to record a list of potentially important independent variables of size l that act as features in the form of a vector $[x_1, x_2, \ldots, x_l]^T$ and their corresponding response values r. The response represents the class label to which a data point with certain features belongs. Based on the early termination algorithm, the response can be related to a CU splitting option or a selected PU mode.

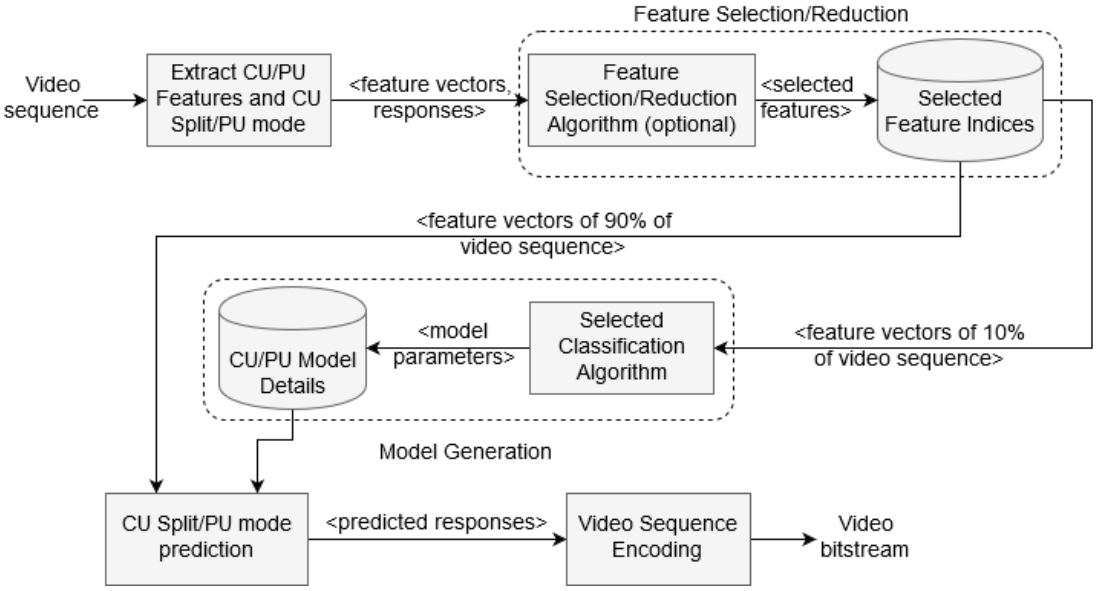

Figure 3.1: Block diagram representing the sequence-dependent approach.

Once all the features and class labels have been extracted from the training sample, dimensionality reduction is applied to select the key features needed for the model generation stage. The dimensionality reduction step is optional and will depend on the approach used. The features are then fed into the chosen classifier, where the model generation takes place. Using the built model, the testing sample is used to extract the values corresponding to the same attributes that were used during training and predicted responses are produced. The HEVC encoder is run again, but using the predicted responses and the effectiveness of the early termination algorithms is evaluated in terms several performance metrics, primarily BD-rate, BD-PSNR, and CCR. The fast partitioning decision algorithms for each partitioning structure are explained in more details in Sections 3.3 to 3.5.

3.3. Early CU Termination Scheme

The first set of algorithms implements a video sequence-dependent approach for early CU termination, where a split flag is computed at different coding depth levels for each CTU. This split flag allows the encoder to make an early decision in terms of whether splitting should occur at a given CU depth level without extensively running the RDO process. The split flag is calculated at CU depth level 0, CU depth level 1 and CU depth level 2. It is not computed at CU depth level 3, where the CU size is 8×8 pixels as this CU cannot be partitioned into four equally sized CUs. As this problem is viewed as a binary classification problem, the class labels considered are of the values 0 (do not split a CU into 4 sub CUs) or 1 (split a CU into 4 sub CUs), indicating whether the CU structure will be partitioned at a particular depth level.

3.3.1. Training phase. Two phases are involved in the training of the classification model: the data extraction stage and the model training stage, as seen in Figure 3.2.

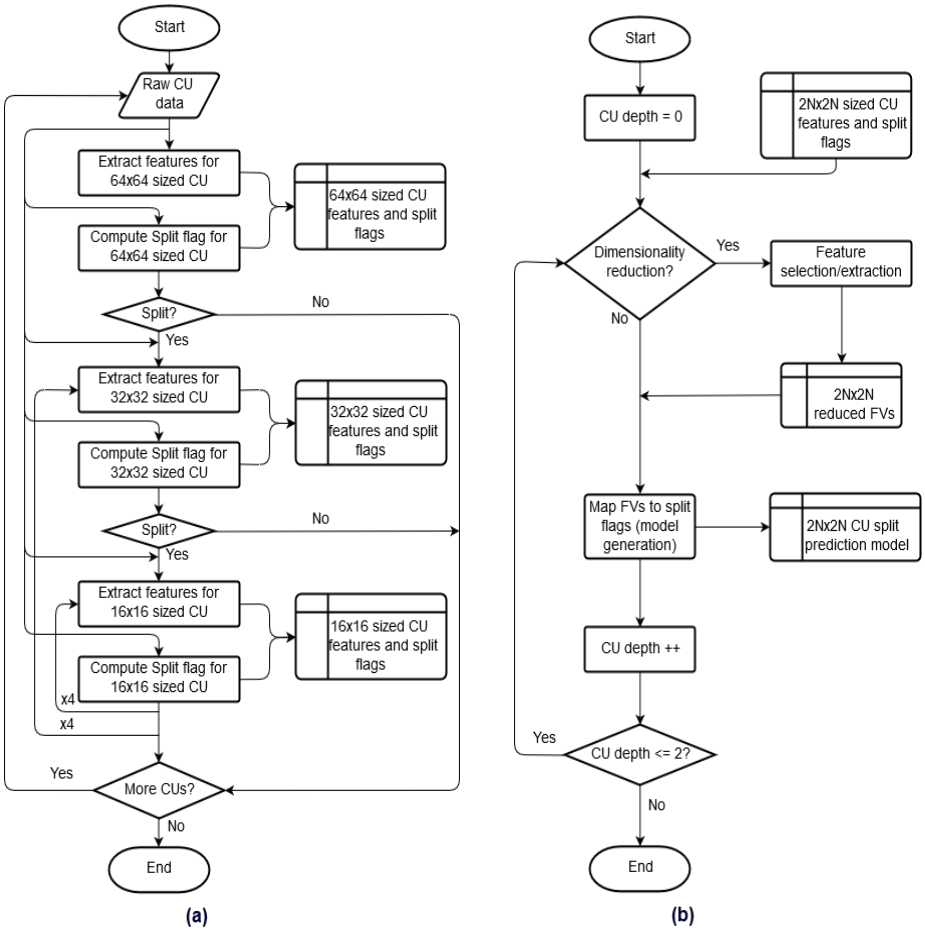

Figure 3.2: Flowchart representing the training phase for early CU termination scheme including (a) Data collection phase and (b) Model training phase.

During the first phase, data collected from running the unmodified encoder on the training sample is used, which is the first 10% of a given video sequence. As indicated in Figure 3.2(a), at each CU depth level, FVs are read, to which the reversed split flags are appended. In other words, for 64×64 sized CUs, features and their corresponding split flags are extracted. Based on the normal operation of the encoder, if the 64×64 CU structure was split for a given CTU, then features and split decision flags corresponding to the second depth level are computed. Again, if the 32×32 CU structure is split, features and split flags corresponding to the 16×16 CU sized structure are extracted. This process takes place recursively for all CTUs until all CUs in the training set have been processed.

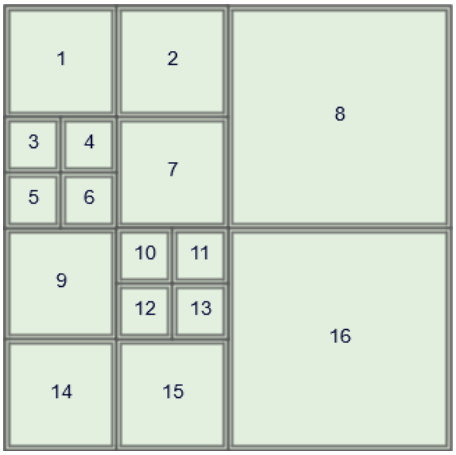

Figure 3.3: An example of a CTU structure.

For each CTU, 21 response values or split decision are recorded. As the process of splitting recursively takes place in a z-scan manner, the split values are stored in reverse. Thus, the split at each CU depth is reversed before being used for model generation. For instance, given the split flag values for the CTU seen in Figure 3.3 to be $[0\ 0\ 1\ 0\ 1 -\ 0\ 0\ 0\ 0\ 0 -\ 0\ 1\ 0\ 0\ 1 -\ 0\ 0\ 0\ 0\ 0 -\ 1]$, the first four numbers correspond to the split status of top left 16×16 blocks followed by split flag of the parent 32×32 block. Similarly, the next fifteen numbers act in the same way, but for the neighboring 32×32 blocks within the same CTU. The last number represents the split flag at depth 0. If the first block split is taken into account $[0\ 0\ 1\ 0\ 1]$, reversing the split flags involves taking last number representing splitting at depth 1 and placing it at the beginning, which results in $[1\ 0\ 0\ 1\ 0]$. This is done for all other blocks at depth 1, while the split flag at depth 0 is placed at the start of the 21 number sequence.

To further optimize the approach, the surrounding CTUs are analyzed and if the neighboring CUs are found to be mostly split, the current CTU's CU is also split. This is done as a CU is likely to behave in a way similar to that of most of its surrounding. Before training the models, for each depth level, FVs from each class are normalized and re-sampled to be of almost equal proportions in order to avoid data imbalance, which may lead to worsening the classification accuracy.

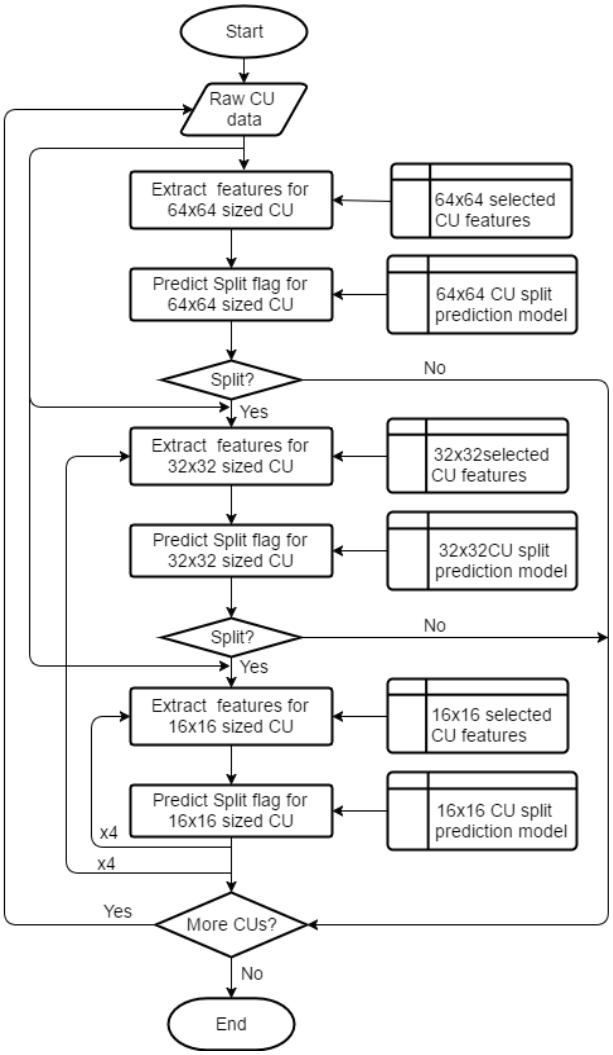

Figure 3.4: Flowchart representing the prediction phase for early CU termination scheme.

During the model training phase, at each CU depth level, the output of the first phase in terms of the FVs and their corresponding classes is fed into a feature selection or extraction algorithm of choice. This results in reducing the feature space. In case the feature selection algorithm does not choose any feature variable, all FVs excluding the ones related to the surrounding CUs are used as these CUs might not exist to begin

with, which depends on the location of the current CTU. Again, dimensionality reduction is an optional step and is dependent on the algorithm used. The resultant of this process is a CU prediction model for each of the three CU depth levels, which will be used during the prediction of the testing sample.

3.3.2. Prediction phase. After generating the CU prediction model, the testing sample, i.e. 90% of a given video sequence, is used to extract the values corresponding to the same attributes that are used in the trained model. As illustrated in Figure 3.4, at each CU depth level, the FVs are extracted and predicted responses are produced using the trained model. Based on the current depth level, the process is repeated for all sub CU structures until all CTUs have been processed.

3.3.3. Dimensionality reduction and classification algorithms. In this approach, five different combinations of dimensionality reduction and classification algorithms are considered. These approaches were selected after conducting a number of experiments and were found to produce better results in comparison. The different solutions are summarized in Table 3.1. The details on the operation of each of these dimensionality reduction and classification algorithms is explained in Chapter 2.

Table 3.1: Arrangement of classification solutions for CU early termination.

Solution	Classifier	Dimensionality reduction
Stepwise & Polynomial	Polynomial networks with second order expansion	Stepwise regression
PCA & Polynomial	Polynomial networks with second order expansion	PCA with PoV of 90
R.F. Select & R.F.	Random forest	Feature importance with random forests
R.F.	Random forest	Not used
J48	Decision trees	Not used

3.3.3.1. *Features and dimensionality reduction algorithms.* The features considered in the CU early termination approach are provided and explained in Table 3.2. The first 15 features are related to the current CU; whereas, the remaining 55 features belong to the surrounding CTUs. The total number of attributes initially considered for this approach is 70 features. Based on the solution, these attributes were either used with no modifications or reduced by either using a feature selection algorithm (feature importance based on random forests or stepwise regression) or a feature extraction algorithm (PCA).

Table 3.2: Attributes for CU early termination.

Feature/Attribute	Feature count	Description
CU depth	1	Coding depth level 0 (64×64), 1 (32×32), or 2 (16×16)
Prediction mode	1	Prediction mode 0 (inter-prediction) or 1 (intra-prediction)
Skip RD cost, 2N×2N RD cost, 2N×N RD cost, N×2N RD cost, N×N RD cost, 2N×uN RD cost, 2N×dN RD cost, lN×2N RD cost, rN×2N RD cost, 2N×2N-intra RD cost, N×N-intra RD cost	11	RD cost of choosing one of the PU splitting modes for the current CU structure, namely inter-PU modes (Skip, 2N×2N, 2N×N, N×2N, N×N, 2N×uN, 2N×dN, lN×2N, and rN×2N), and intra-PU modes (2N×2N and N×N)
CTU-L distortion, CTU-UL distortion, CTU-U distortion, CTU-UR distortion, CTU-T distortion	5	Total distortion cost of each of the surrounding CTUs, namely Left CTU (CTU-L), Upper Left CTU (CTU-UL), Upper CTU (CTU-U), Upper Right CTU (CTU-UR), and the Collocated CTU (CTU-T)
CTU-L avg. depth, CTU-UL avg. depth, CTU-U avg. depth, CTU-UR avg. depth, CTU-T avg. depth	5	Average depth of all CUs in each of the surrounding CTUs, namely Left CTU (CTU-L), Upper Left CTU (CTU-UL), Upper CTU (CTU-U), Upper Right CTU (CTU-UR), and the Collocated CTU (CTU-T)
CTU-L std. depth, CTU-UL std. depth, CTU-U std. depth, CTU-UR std. depth, CTU-T std. depth	5	Variance of all CUs in each of the surrounding CTUs, namely Left CTU (CTU-L), Upper Left CTU (CTU-UL), Upper CTU (CTU-U), Upper Right CTU (CTU-UR), and the Collocated CTU (CTU-T)
CTU-L avg. List0-x, CTU-L avg. List0-y, CTU-L std. List0-x, CTU-L std. List0-y, CTU-L avg. List1-x, CTU-L avg. List1-y, CTU-L std. List1-x, CTU-L std. List1-y	8	Average and variance of MV information in Left CTU (CTU-L) for both horizontal and vertical directions using the reference picture lists (List0 and List1)
CTU-UL avg. List0-x, CTU-UL avg. List0-y, CTU-UL std. List0-x, CTU-UL std. List0-y, CTU-UL avg. List1-x, CTU-UL avg. List1-y, CTU-UL std. List1-x, CTU-UL std. List1-y	8	Average and variance of MV information in Upper Left CTU (CTU-UL) for both horizontal and vertical directions using the reference picture lists (List0 and List1)
CTU-U avg. List0-x, CTU-U avg. List0-y, CTU-U std. List0-x, CTU-U std. List0-y, CTU-U avg. List1-x, CTU-U avg. List1-y, CTU-U std. List1-x, CTU-U std. List1-y	8	Average and variance of MV information in Upper CTU (CTU-U) for both horizontal and vertical directions using the reference picture lists (List0 and List1)
CTU-UR avg. List0-x, CTU-UR avg. List0-y, CTU-UR std. List0-x, CTU-UR std. List0-y, CTU-UR avg. List1-x, CTU-UR avg. List1-y, CTU-UR std. List1-x, CTU-UR std. List1-y	8	Average and variance of MV information in Upper Right CTU (CTU-UR) for both horizontal and vertical directions using the reference picture lists (List0 and List1)
CTU-T avg. List0-x, CTU-T avg. List0-y, CTU-T std. List0-x, CTU-T std. List0-y, CTU-T avg. List1-x, CTU-T avg. List1-y, CTU-T std. List1-x, CTU-T std. List1-y	8	Average and variance of MV information in the Collocated CTU (CTU-T) for both horizontal and vertical directions using the reference picture lists (List0 and List1)
Merge flag	1	Merge flag status, indicating if a CU has been predicted using MSM PU mode
Skip flag	1	Skip flag status, indicating if a CU has been predicted using Skip PU mode

The result of the dimensionality reduction algorithm is three sets of indices corresponding to the retained feature variables, one set per each coding depth level. It is important to take note that these indices were also used to reduce the dimensionality of FVs during the testing phase. Since a video-dependent approach is used in this work, the number of retained FVs achieved when using a feature selection algorithm or the projection dimensions selected when using a feature extraction algorithm can vary from one video sequence to the other.

One of the dimensionality reduction approaches utilized is based on the feature importance option provided by the usage of a random forest. At the beginning, a random forest of 100 trees is grown, where the maximum number of decision splits or branch nodes is set to be the initial set of 70 features. The training dataset is sampled for each decision tree with replacement and the feature variables selected at random for each decision split are chosen without replacement within the same decision tree. The importance of each of these features in predicting the correct classification of a test instance from the OOB data is computed and used to select the features whose raw importance score makes up 80% of the total importance score. The OOB data is the set of instances that were left out during the training process of a given tree in the random forest.

The second feature selection approach used is stepwise regression, whose operation is explained in Chapter 2. At first, one feature variable is selected and its correlation with the split decision is computed. Then, another feature variable is added, whose correlation with the split decision is also computed. The significance of adding the second feature variable is assessed at a level of significance of 0.05. If the added feature variable is found significant, then it is retained, otherwise it is removed from the list of variables. The algorithm revisits the features included in the retained features list, the first feature in this case, and reassess the significance of keeping it along with the newly added feature. The algorithm continues adding and removing feature variables in the same manner until all variables have been examined.

The final dimensionality reduction algorithm considered is the principle component analysis approach, which is a feature extraction algorithm. Here, an orthogonal transformation is used to convert the features into principle components based on maximizing the feature variance. The number of principle components

retained depends on the chosen PoV explained, which is 90% in this work. Again, the PCA is applied to the training dataset at 64×64, 32×32 and 16×16 coding levels. The resulting principle components are then stored and used for reducing the test data set.

3.3.3.2. ***Classification algorithms.*** Three different classifiers were used for early CU decision termination. The first involves applying the J48 algorithm, which is an implementation of C4.5 decision trees algorithm. The features proposed in [45] were used to generate the classification models. The J48 algorithm used is the one built in WEKA. The chosen confidence factor is 0.25, while the minimum number of instances per leaf was selected to be 2.

The second algorithm approach proposed involves using a random forest, where 100 trees are grown and the maximal number of decision splits or branch nodes is the square root of the number of retained feature variables. Based on the retained features, the training dataset is sampled for each decision tree with replacement. The variables selected at random for each decision split are then chosen within the same decision tree. As the purpose of growing the trees is classification, only one observation or class label can be seen per tree leaf. No pruning is applied to any of the grown trees as to avoid worsening the classification accuracy.

The last algorithm used involves polynomial networks with second order expansion. The Reduced Multivariate Polynomial Model with second degree of approximation presented in Chapter 2 is used for this purpose to perform second order polynomial classification.

3.4. Early PU Termination Scheme

The second set of algorithms implements a video sequence-dependent approach for early PU termination, where a PU mode flag is computed at different CU depth levels for each CTU. This PU mode flag allows the encoder to make an early decision in terms of whether, at a given coding depth level, the PU mode is of 2N×2N dimensionality or less. In other words, the flag indicates if the RDO process should run to evaluate all PU modes for a specific CU or to just consider PU modes of size 2N×2N. The PU mode flag is calculated at CU depth level 0, CU depth level 1, CU depth level 2, and CU depth level 3. As this problem is viewed as a binary classification problem,

the class labels considered are of the values 0 (consider PU modes of size less than 2N×2N) or 1 (consider PU modes of size 2N×2N) at a given CU depth level.

The reason for considering these two class labels is due to noticing that regardless of the video content of a given sequence, most of the time, either Skip PU mode, 2N×2N inter PU mode, or 2N×2N intra PU mode is chosen for any CU. Therefore, despite having around 11 different PU modes to consider at each CU depth level excluding CU depth level 3, only the abovementioned class labels were taken into account. Additionally, increasing the number of classes was seen to negatively affect the classification accuracy, leading to deteriorating the video's quality and increasing the bitrate consumption. Both inter-PU and intra-PU modes were considered in this scheme.

3.4.1. Training phase. Two phases are involved in the training of the classification model: the data extraction stage and the model training stage, as seen in Figure 3.5. During the first phase, data collected from running the unmodified encoder on the training sample is used, which is the first 10% of a given video sequence.

As indicated in Figure 3.5(a), at each depth level, the FVs are read, to which the PU mode flags are appended. In other words, for 64×64 sized CUs, features and their corresponding PU mode flags are extracted. Based on the normal operation of the encoder, if the 64×64 CU structure was split into sub CUs for a given CTU, then features and PU mode decision flags corresponding to the second depth level are computed. This process is recursively repeated for 16×16 and 8×8 sized CUs for all CTUs until all CUs in the training set have been processed.

Similar to the early CU termination algorithm, before training the models, FVs from each class are normalized and re-sampled to be of almost equal proportions in order to avoid data imbalance. No feature space reduction took place before generating the classification models. The resultant of this process is a PU mode prediction model for each of the four CU depth levels, which is used during the prediction of the testing sample.

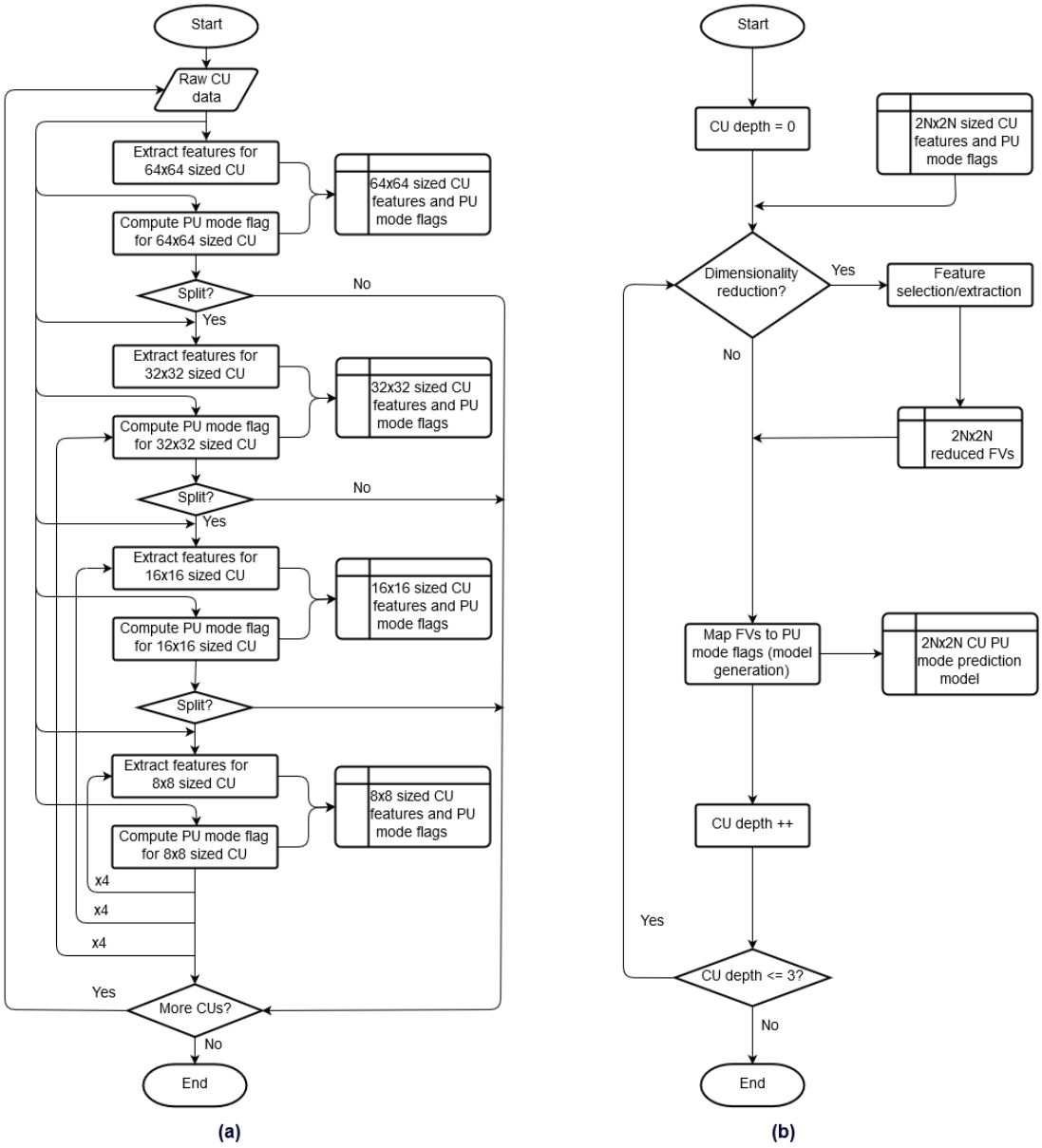

Figure 3.5: Flowchart representing the training phase for early PU termination scheme including (a) Data collection phase and (b) Model training phase.

3.4.2. Prediction phase. After generating the PU mode prediction model, the testing sample, i.e. 90% of a given video sequence, is used to extract the values corresponding to the same attributes that were used in the trained model. As illustrated in Figure 3.6, at each coding depth level, the FVs are extracted and predicted responses are produced using the trained model. Based on the current depth level, the process is repeated for each of the sub CU structures until all CTUs have been processed.

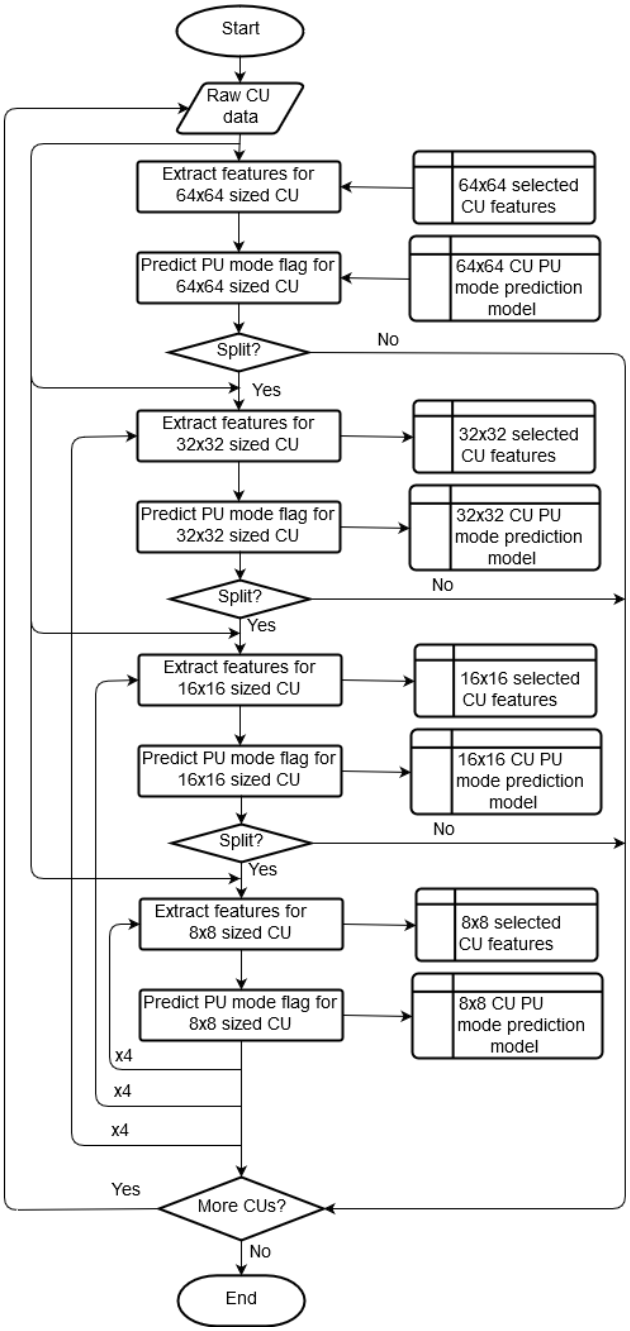

Figure 3.6: Flowchart representing the prediction phase for early PU termination scheme.

3.4.3. Dimensionality reduction and classification algorithms. In this approach, two different classification algorithms are taken into account. Unlike in early CU termination, dimensionality reduction was not used. These approaches were selected after conducting a number of experiments and were found to produce better results in comparison. The different solutions are summarized in Table 3.3. The details on the operation of the classification algorithms is given in Chapter 2.

Table 3.3: Arrangement of classification solutions for PU early termination.

Solution	Classifier	Dimensionality reduction
R.F.	Random forest	Not used
J48	Decision trees	Not used

3.4.3.1. *Features.* The features considered in the PU early termination approach are provided and explained in Table 3.4. The total number of attributes initially considered for this approach is 6 features. Regardless of the approach, these attributes were used without applying any dimensionality reduction techniques.

Table 3.4: Attributes for PU early termination.

Feature/Attribute	Feature count	Description
Skip RD cost	1	RD cost of choosing Skip PU splitting mode for the current CU structure
2N×2N RD cost	1	RD cost of choosing 2N×2N inter-PU splitting mode for the current CU structure
2N×2N Intra RD cost	1	RD cost of choosing 2N×2N intra-PU splitting mode for the current CU structure
N×N Intra RD cost	1	RD cost of choosing N×N intra-PU mode for the current CU structure
Best RD cost	1	Lowest RD cost among Skip PU mode, 2N×2N inter-PU mode, 2N×2N intra-PU mode, and N×N intra-PU mode
Upper CU div	1	Value indicating if the CU in the upper CU depth level was split

3.4.3.2. *Classification algorithms.* Two different classifiers were used for early PU decision termination. The first involves applying the J48 algorithm, which is an implementation of C4.5 decision trees algorithm. The features proposed in [45] were used to generate the classification models. The J48 algorithm used is the one built in WEKA. The chosen confidence factor is 0.25, while the minimum number of instances per leaf was selected to be 2.

The second approach proposed involves using a random forest, where 100 trees are grown and the maximal number of decision splits or branch nodes is the square root of the number of retained feature variables. Based on the retained features, the training dataset is sampled for each decision tree with replacement. The variables selected at random for each decision split are then chosen within the same decision tree. As the purpose of growing the trees is classification, only one observation or class label can

be seen per tree leaf. No pruning is applied to any of the grown trees as to avoid worsening the classification accuracy.

3.5. Early Joint Termination Scheme

The final scheme proposed implements a video sequence-dependent approach for both early CU and PU termination. It involves combining the aforementioned schemes to limit the RDO process. Three approaches were selected after conducting a number of experiments and were found to produce better results in comparison. The different solutions are summarized in Table 3.5. Based on the targeted structures, the operation of the selected algorithms is the same as that described in Sections 3.2 and 3.3.

Table 3.5: Arrangement of classification solutions for CU & PU early termination.

Solution	Classifier	Dimensionality reduction
R.F.	Random forest for CU & PU early termination	Not used
J48	Decision trees for CU & PU early termination	Not used
R.F. & J48	Random forest for CU early termination & Decision trees for PU early termination	Not used

Chapter 4. Experimental Setup

This chapter summarizes the experimental setup, including the set of configurations used to achieve the experimental results.

4.1. Testing Configurations

The proposed solutions were implemented using the HM reference software version 13.0 [47] in order to encode the video sequences used for both training and testing purposes. The baseline profile defined as the RA temporal configuration in the Joint Collaborative Team on Video Coding (JCT-VC) document containing the recommended common test conditions (CTCs) [48] was utilized to encode all the videos, where the QP values were set to 22, 27, 32, and 37. A total of 17 video sequences are used as reported in Table 4.1, where a mixture of 8 and 10 bit coding is considered.

Table 4.1: Video sequences used for the early termination approaches.

Class category	Video sequence	Frames encoded	Bit depth	Frame rate	Resolution
Class D	***RaceHorses***	100	8	30	384×192
	BlowingBubbles	100	8	50	384×192
	BQSquare	100	8	60	384×192
	BasketballPass	100	8	50	384×192
Class C	***RaceHorses***	100	8	30	832×448
	PartyScene	100	8	50	832×448
	BQMall	100	8	60	832×448
	BasketballDrill	100	8	50	832×448
Class B	***ParkScene***	100	8	24	1920×1024
	Kimono1	100	8	24	1920×1024
	Cactus	100	8	50	1920×1024
	BQTerrace	100	8	60	1920×1024
	BasketballDrive	100	8	50	1920×1024
Class A	***Traffic***	100	8	30	2560×1600
	PeopleOnStreet	100	8	30	2560×1600
	NebutaFestival	100	10	60	2560×1600
	SteamLocomotiveTrain	100	10	60	2560×1600

As per the JCT-VC document, the spatial resolutions that were used to obtain the experimental results are of Class A (2560×1600), Class B (1920×1080 pixels), Class C (832×480 pixels), and Class D (416×240 pixels). The number of frames to be encoded were set to be 100. It is important to note that all the video sequences underwent pre-processing, which involved cropping them such that the spatial resolution of each is a multiple of 64. This was crucial for the suggested methodologies to work. The experiments were conducted on a PC with an Intel Core i7-4770S, 3.10GHz CPU and a 16-GB DDR3 RAM installed. In addition to the HM software used for encoding, the MATLAB software version 2015a [49] was used to both train a given model and predict responses based on the data available in the testing sample. In order to use the J48 decision trees' implementation provided by WEKA [46], an efficient interface built by Dr. Sunghoon Lee, an Assistant Professor in the College of Information and Computer Science at the University of Massachusetts at Amherst, was used, which allows using WEKA in MATLAB. All schemes were first separately evaluated and then, a selection of them were jointly implemented.

Chapter 5. Results and Analysis

In this chapter, the experimental results achieved through the implementation of the proposed schemes are presented. Furthermore, the performance evaluation of using those solutions are discussed. All solutions are evaluated in terms of BD-rate, BD-PSNR, excessive bitrate, CCR, model generation time and decision accuracy. The results for each solution based on the partitioning structure it is applied to are presented from worse to best in terms of BD-rate and BD-PSNR. Clearly, a better encoding efficiency is acquired as the BD-rate decreases and the BD-PSNR increases. A negative BD-rate indicates that less bits are needed during the compression process, while a positive BD-PSNR specifies higher image quality. Furthermore, the results obtained are analysed in detail and compared to that presented in the literature.

5.1. Performance Metrics

As previously mentioned, the compression efficiency is quantified in terms of BD-rate and BD-PSNR, whose computation is explained at the end of Chapter 2. Moreover, the coding time saving acquired by using the proposed solutions are computed and compared with the corresponding times obtained by running the unmodified HEVC encoder. Lastly, the accuracy of the proposed classification systems is presented along with the model generation time and excessive coding bitrate.

In order to compute the encoding time savings and compare the results to that presented in the literature, two different equations are considered. The encoding time savings were computed after applying the predictive model generated by a proposed solution. The first formula presents the CCR achieved by a particular algorithm, which is given by

$$CCR\ (\%) = \frac{Time_{ref} - Time_{prop}}{Time_{ref}} \times 100, \tag{25}$$

where $Time_{ref}$ denotes the time taken to encode a specific video sequence using the HEVC model encoder and $Time_{prop}$ represents the time taken to encode the same video sequence using the HM software that utilizes the proposed solution. Using the variables given in (25), the second time saving equation is given by

$$\Delta\, Time\ (\%) = \frac{Time_{prop} - Time_{ref}}{Time_{prop}} \times 100. \qquad (26)$$

In order to compute the percentage of the time taken for a prediction model to be generated, (27) is used. Here, the total time taken to encode a specific video sequence using a proposed solution was added to the model generation and total prediction time. The model generation time was divided by this summation, resulting in

$$Time_{model\_t}(\%) = \frac{Time_{model}}{Time_{prop} + Time_{model} + Time_{pred}} \times 100, \qquad (27)$$

where $Time_{model}$ denotes the model generation time, $Time_{pred}$ the total prediction time and $Time_{prop}$ the time taken to encode a video sequence using the proposed solution.

5.2. Experimental Results

As mentioned in Chapter 4, a number of schemes were proposed with the aim of optimizing the HEVC encoding process. The first five algorithms, whose results are presented, utilize the coding tree to enhance the encoding process. These algorithms involve using J48 decision trees, random forests and a second order polynomial classifier along with different dimensionality reduction techniques. The second set of algorithms improves the coding efficiency by applying J48 decision trees and random forests algorithms on PUs. The last solution proposed combines both approaches, which led to using decision trees and random forests to generate three early termination algorithms for both CUs and PUs.

5.2.1. CU early termination algorithms. The results obtained by implementing a set of five machine learning algorithms to enhance the CU size selection are illustrated. These results are given in terms of BD-rate, BD-PSNR, excessive bitrate, computation complexity savings, model generation time, feature selection or extraction, and decision accuracy.

5.2.1.1. ***PCA with PoV of 90% and second order polynomial classifier.*** Tables 5.1 and 5.2 show the time savings and excessive bitrate per each QP for each of the test sequences acquired after applying PCA with PoV of 90% to select principal components to be used by the second order polynomial classifier, respectively. As the

QP value increases, it is observed that, on average, less coding bits and time are needed to encode a given video sequence.

Table 5.1: Time savings results per each QP using PCA with PoV of 90% and second order polynomial classifier for early CU termination.

Video Sequence	22		27		32		37	
	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)
RaceHorses (384×192)	14.7	-17.3	16.3	-19.4	17.9	-21.8	22.5	-29.0
BlowingBubbles (384×192)	23.5	-30.7	21.9	-28.0	36.4	-57.2	42.9	-75.1
BQSquare (384×192)	20.9	-26.4	33.3	-49.8	42.9	-75.2	50.8	-103.2
BasketballPass (384×192)	20.7	-26.1	30.5	-43.9	34.0	-51.6	41.1	-69.7
RaceHorses (832×448)	24.9	-33.2	19.8	-24.7	28.7	-40.2	38.0	-61.2
PartyScene (832×448)	19.8	-24.7	27.1	-37.3	36.5	-57.5	43.6	-77.3
BQMall (832×448)	23.7	-31.1	28.1	-39.1	35.5	-55.1	41.2	-70.1
BasketballDrill (832×448)	30.0	-42.9	33.5	-50.3	39.9	-66.3	48.5	-94.1
ParkScene (1920×1024)	28.6	-40.0	41.1	-69.9	51.0	-104.1	57.2	-133.4
Kimono1 (1920×1024)	34.0	-51.4	41.6	-71.3	45.6	-83.7	49.6	-98.2
Cactus (1920×1024)	26.9	-36.8	41.1	-69.8	45.4	-83.2	53.6	-115.6
BQTerrace (1920×1024)	24.1	-31.7	43.1	-75.8	58.9	-143.5	63.9	-177.1
BasketballDrive (1920×1024)	20.7	-26.1	37.9	-61.1	42.9	-75.2	50.7	-102.9
Traffic (2560×1600)	34.8	-53.4	44.3	-79.5	51.4	-105.8	58.7	-142.3
PeopleOnStreet (2560×1600)	25.5	-34.2	25.0	-33.2	23.1	-30.0	33.5	-50.4
NebutaFestival (2560×1600)	53.0	-112.6	45.8	-84.6	36.0	-56.3	54.3	-118.9
SteamLocomotiveTrain (2560×1600)	46.0	-85.1	48.2	-93.2	57.6	-135.9	63.3	-172.4
Average	***27.7***	***-41.4***	***34.0***	***-54.8***	***40.2***	***-73.1***	***47.8***	***-99.5***

Table 5.2: Excessive bitrate results per each QP using PCA with PoV of 90% and second order polynomial classifier for early CU termination.

Video Sequence	Excessive Bitrate (%)			
	QP = 22	QP = 27	QP = 32	QP = 37
RaceHorses (384×192)	1.06	0.43	-0.05	-0.44
BlowingBubbles (384×192)	0.25	-0.18	-0.39	-0.40
BQSquare (384×192)	-0.19	-0.41	-0.52	-0.15
BasketballPass (384×192)	0.56	1.47	0.22	0.98
RaceHorses (832×448)	1.07	0.27	0.99	0.33
PartyScene (832×448)	0.64	0.22	-0.07	-0.50
BQMall (832×448)	0.52	0.70	0.36	0.38
BasketballDrill (832×448)	1.01	0.74	0.53	0.61
ParkScene (1920×1024)	0.27	-0.09	-0.38	-0.47
Kimono1 (1920×1024)	0.15	0.35	0.06	-0.01
Cactus (1920×1024)	-0.17	-0.02	0.20	-0.07
BQTerrace (1920×1024)	-0.25	-0.72	-0.84	-0.93
BasketballDrive (1920×1024)	-0.38	0.14	0.29	0.37
Traffic (2560×1600)	-0.16	0.06	0.11	-0.26
PeopleOnStreet (2560×1600)	2.71	1.89	2.16	0.29
NebutaFestival (2560×1600)	0.04	0.28	-0.05	-0.09
SteamLocomo-tiveTrain (2560×1600)	0.09	-0.22	0.02	-0.25
Average	***0.42***	***0.29***	***0.16***	***-0.04***

Overall, a CCR of 37.5% is attained at the cost of introducing performance losses of 1.355% and -0.053 dB in terms BD-rate and BD-PSNR, respectively. The results imply that, as the spatial resolution increases, more complexity reduction is accomplished. These results can be observed in Table 5.3.

Table 5.3: BD-rate, BD-PSNR, avg. excessive bitrate, and time savings results using PCA with PoV of 90% and second order polynomial classifier for early CU termination.

Video Sequence	BD-rate (%)	BD-PSNR (dB)	Exc. Bitrate (%)	CCR (%)	ΔTime (%)	BD-rate/CCR (%)
RaceHorses (384×192)	1.420	-0.068	0.25	17.8	-21.9	7.959
BlowingBubbles (384×192)	0.874	-0.035	-0.18	31.2	-47.8	2.805
BQSquare (384×192)	0.751	-0.035	-0.32	37.0	-63.7	2.032
BasketballPass (384×192)	2.454	-0.112	0.81	31.6	-47.8	7.770
Average	***1.375***	***-0.063***	***0.14***	***29.4***	***-45.3***	***5.141***
RaceHorses (832×448)	2.094	-0.083	0.67	27.8	-39.8	7.522
PartyScene (832×448)	1.452	-0.067	0.07	31.8	-49.2	4.570
BQMall (832×448)	1.835	-0.076	0.49	32.1	-48.8	5.712
BasketballDrill (832×448)	1.739	-0.071	0.72	38.0	-63.4	4.580
Average	***1.780***	***-0.074***	***0.49***	***32.4***	***-50.3***	***5.596***
ParkScene (1920×1024)	1.293	-0.041	-0.17	44.5	-86.8	2.908
Kimono1 (1920×1024)	0.761	-0.025	0.14	42.7	-76.2	1.783
Cactus (1920×1024)	1.103	-0.024	-0.02	41.8	-76.4	2.641
BQTerrace (1920×1024)	0.906	-0.017	-0.69	47.5	-107.0	1.906
BasketballDrive (1920×1024)	1.169	-0.026	0.11	38.1	-66.3	3.072
Average	***1.046***	***-0.027***	***-0.13***	***42.9***	***-82.5***	***2.462***
Traffic (2560×1600)	1.516	-0.052	-0.06	47.3	-95.2	3.205
PeopleOnStreet (2560×1600)	3.724	-0.161	1.76	26.8	-37.0	13.919
NebutaFestival (2560×1600)	0.041	0.000	0.05	47.3	-93.1	0.087
SteamLocomotiveTrain (2560×1600)	-0.094	0.000	-0.09	53.8	-121.6	-0.174
Average	***1.297***	***-0.053***	***0.41***	***43.8***	***-86.7***	***4.259***
Overall Average	***1.355***	***-0.053***	***0.208***	***37.5***	***-67.2***	***4.253***

Table 5.4 summarizes the percentage of the time taken for the prediction model to be generated. This information is important as the proposed solution is a sequence-dependent one. The number of principal components retained when using PCA with a PoV value of 90% is given in Table 5.5. This information is not easy to interpret in comparison to simply selecting features. The reason behind this is that a principal component can be seen as the combination of different features selected with the aim of maximizing the feature variance. On average, 26 principal components are selected on which a data sample is projected.

Table 5.4: Model generation time to encoding time using modified encoder ratios using PCA with PoV of 90% and second order polynomial classifier for early CU termination.

Video Sequence	$Time_{model\_t}$ (%)			
	QP = 22	**QP = 27**	**QP = 32**	**QP = 37**
RaceHorses (384×192)	0.07	0.10	0.09	0.09
BlowingBubbles (384×192)	0.05	0.06	0.07	0.11
BQSquare (384×192)	0.04	0.07	0.10	0.11
BasketballPass (384×192)	0.06	0.05	0.06	0.07
Average	***0.05***	***0.07***	***0.08***	***0.09***
RaceHorses (832×448)	0.03	0.04	0.05	0.06
PartyScene (832×448)	0.04	0.05	0.07	0.08
BQMall (832×448)	0.04	0.06	0.07	0.07
BasketballDrill (832×448)	0.04	0.05	0.07	0.08
Average	***0.04***	***0.05***	***0.07***	***0.07***
ParkScene (1920×1024)	0.06	0.07	0.09	0.13
Kimono1 (1920×1024)	0.04	0.07	0.07	0.08
Cactus (1920×1024)	0.06	0.07	0.08	0.11
BQTerrace (1920×1024)	0.06	0.10	0.13	0.15
BasketballDrive (1920×1024)	0.05	0.06	0.07	0.09
Average	***0.05***	***0.07***	***0.09***	***0.11***
Traffic (2560×1600)	0.21	5.10	11.90	0.20
PeopleOnStreet (2560×1600)	4.15	5.74	4.94	7.34
NebutaFestival (2560×1600)	2.83	0.06	6.51	12.17
SteamLocomotiveTrain (2560×1600)	4.38	7.15	8.89	0.60
Average	***2.90***	***4.51***	***8.06***	***5.08***
Overall Average	***0.718***	***1.112***	***1.956***	***1.267***

Table 5.5: Retained principal components per CU size using PCA with PoV of 90% and second order polynomial classifier for early CU termination.

Video Sequence	64×64	32×32	16×16
RaceHorses (384×192)	8	24	25
BlowingBubbles (384×192)	20	26	26
BQSquare (384×192)	20	22	22
BasketballPass (384×192)	19	20	21
Average	***16***	***23***	***23***
RaceHorses (832×448)	19	30	30
PartyScene (832×448)	29	30	30
BQMall (832×448)	27	29	29
BasketballDrill (832×448)	24	24	25
Average	***24***	***28***	***28***
ParkScene (1920×1024)	29	31	31
Kimono1 (1920×1024)	29	29	27
Cactus (1920×1024)	27	28	29
BQTerrace (1920×1024)	31	31	32
BasketballDrive (1920×1024)	29	29	29
Average	***29***	***29***	***29***
Traffic (2560×1600)	22	23	24
PeopleOnStreet (2560×1600)	30	34	34
NebutaFestival (2560×1600)	29	30	31
SteamLocomotiveTrain (2560×1600)	28	29	30
Average	***27***	***29***	***29***
Overall Average	***24***	***27***	***27***

The decision accuracies achieved by using the proposed algorithm can be seen in Table 5.6. It is important to understand that the resultant CU size predictions are not used blindly by the modified encoder. Some checking mechanism takes place by the encoder when the predicted CU split flag indicates that split should take place at a given coding depth. A classification rate of around 83.0% is achieved by the proposed scheme.

Table 5.6: Classification rates per each CU size using PCA with PoV of 90% and second order polynomial classifier for early CU termination.

Video Sequence	64×64	32×32	16×16	True Overall
RaceHorses (384×192)	51.5	72.8	74.0	75.0
BlowingBubbles (384×192)	65.0	80.6	83.0	82.4
BQSquare (384×192)	76.4	87.2	87.4	86.5
BasketballPass (384×192)	81.4	83.5	83.0	82.9
Average	***68.5***	***81.0***	***81.8***	***81.7***
RaceHorses (832×448)	75.6	75.6	80.2	80.2
PartyScene (832×448)	86.2	85.3	85.7	85.1
BQMall (832×448)	86.8	80.9	81.0	80.5
BasketballDrill (832×448)	83.1	84.6	86.0	85.0
Average	***82.9***	***81.6***	***83.2***	***82.7***
ParkScene (1920×1024)	84.6	85.5	88.1	86.2
Kimono1 (1920×1024)	72.6	74.7	71.5	77.3
Cactus (1920×1024)	86.4	84.4	84.5	83.9
BQTerrace (1920×1024)	87.0	87.4	89.0	87.6
BasketballDrive (1920×1024)	84.7	81.5	80.9	80.3
Average	***83.0***	***82.7***	***82.8***	***83.0***
Traffic (2560×1600)	84.8	87.9	90.5	88.7
PeopleOnStreet (2560×1600)	88.1	79.6	78.7	79.2
NebutaFestival (2560×1600)	60.2	74.8	82.9	82.1
SteamLocomotiveTrain (2560×1600)	79.8	86.4	88.0	88.0
Average	***78.2***	***82.2***	***85.0***	***84.5***
Overall Average	***78.5***	***81.9***	***83.2***	***83.0***

5.2.1.2. *Stepwise regression and second order polynomial classifier.* Tables 5.7 and 5.8 show the time savings and excessive bitrate per each QP for each of the test sequences acquired after applying stepwise regression to select the features to be used by the second order polynomial classifier, respectively. As the QP value increases, it is evident that, on average, less coding bits and time are required to encode a given video sequence. Overall, a CCR of 39.1% is attained at the cost of introducing performance losses of 1.339% and -0.054 dB in terms BD-rate and BD-PSNR, respectively. The results imply that, as the spatial resolution increases, more complexity reduction is accomplished, while increasing the BD-PSNR. These results can be observed in Table 5.9.

Table 5.7: Time savings results per each QP using stepwise regression and second order polynomial classifier for early CU termination.

Video Sequence	22		27		32		37	
	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)
RaceHorses (384×192)	19.4	-24.1	22.4	-28.9	25.5	-34.1	33.6	-50.7
BlowingBubbles (384×192)	23.5	-30.8	30.5	-44.0	36.6	-57.8	44.3	-79.5
BQSquare (384×192)	21.1	-26.7	34.4	-52.4	43.4	-76.4	48.6	-94.4
BasketballPass (384×192)	28.9	-40.7	30.8	-44.4	37.3	-59.3	42.3	-73.3
RaceHorses (832×448)	20.3	-25.5	27.4	-37.7	33.7	-50.9	40.0	-66.8
PartyScene (832×448)	23.0	-29.9	29.3	-41.5	36.2	-56.7	41.2	-70.1
BQMall (832×448)	24.3	-32.2	34.2	-52.1	35.8	-55.7	40.1	-67.1
BasketballDrill (832×448)	36.3	-57.0	34.8	-53.4	39.8	-66.1	47.1	-88.9
ParkScene (1920×1024)	29.0	-40.8	38.8	-63.3	47.5	-90.6	57.4	-134.5
Kimono1 (1920×1024)	32.2	-47.5	40.6	-68.3	43.6	-77.2	49.8	-99.1
Cactus (1920×1024)	43.4	-76.8	37.5	-60.1	46.4	-86.6	59.2	-145.1
BQTerrace (1920×1024)	25.5	-34.2	43.3	-76.3	57.6	-135.8	62.9	-169.5
BasketballDrive (1920×1024)	28.8	-40.5	37.5	-59.9	43.7	-77.7	48.9	-95.8
Traffic (2560×1600)	36.7	-58.1	47.0	-88.7	52.5	-110.3	53.1	-113.2
PeopleOnStreet (2560×1600)	20.2	-25.2	23.9	-31.4	27.0	-36.9	43.7	-77.5
NebutaFestival (2560×1600)	54.1	-118.1	46.7	-87.5	46.2	-85.8	54.0	-117.2
SteamLocomotiveTrain (2560×1600)	46.1	-85.6	48.1	-92.6	57.5	-135.8	62.1	-163.7
Average	***30.2***	***-46.7***	***35.7***	***-57.8***	***41.8***	***-76.1***	***48.7***	***-100.4***

Table 5.8: Excessive bitrate results per each QP using stepwise regression and second order polynomial classifier for early CU termination.

Video Sequence	Excessive Bitrate (%)			
	QP = 22	QP = 27	QP = 32	QP = 37
RaceHorses (384×192)	0.63	0.35	-0.60	-0.93
BlowingBubbles (384×192)	-0.27	0.42	0.44	-0.47
BQSquare (384×192)	-0.13	-0.25	0.48	0.19
BasketballPass (384×192)	1.98	1.86	3.02	0.08
RaceHorses (832×448)	0.69	0.05	-0.06	-0.83
PartyScene (832×448)	0.43	0.09	-0.28	-0.61
BQMall (832×448)	0.49	0.54	0.05	-0.19
BasketballDrill (832×448)	0.70	0.71	0.29	0.02
ParkScene (1920×1024)	-0.30	-0.31	-0.66	-0.71
Kimono1 (1920×1024)	0.10	0.23	-0.19	-0.16
Cactus (1920×1024)	-0.35	-0.06	-0.16	-0.49
BQTerrace (1920×1024)	-0.50	-0.83	-0.86	-0.84
BasketballDrive (1920×1024)	-0.15	0.50	0.50	0.38
Traffic (2560×1600)	-0.65	-0.45	-0.50	-0.78
PeopleOnStreet (2560×1600)	0.87	0.14	0.09	-0.11
NebutaFestival (2560×1600)	0.03	0.27	0.16	-0.36
SteamLocomo-tiveTrain (2560×1600)	0.01	-0.22	0.14	-0.09
Average	***0.21***	***0.18***	***0.11***	***-0.35***

Table 5.9: BD-rate, BD-PSNR, avg. excessive bitrate, and time savings results using stepwise regression and second order polynomial classifier for early CU termination.

Video Sequence	BD-rate (%)	BD-PSNR (dB)	Exc. Bitrate (%)	CCR (%)	ΔTime (%)	BD-rate/CCR (%)
RaceHorses (384×192)	2.575	-0.124	-0.14	25.2	-34.5	10.202
BlowingBubbles (384×192)	1.840	-0.073	0.03	33.7	-53.0	5.452
BQSquare (384×192)	1.229	-0.055	0.07	36.8	-62.5	3.337
BasketballPass (384×192)	4.338	-0.211	1.74	34.8	-54.4	12.461
Average	***2.495***	***-0.116***	***0.43***	***32.7***	***-51.1***	***7.863***
RaceHorses (832×448)	1.740	-0.069	-0.04	30.4	-45.2	5.731
PartyScene (832×448)	1.036	-0.048	-0.09	32.4	-49.5	3.194
BQMall (832×448)	1.944	-0.082	0.22	33.6	-51.8	5.781
BasketballDrill (832×448)	1.585	-0.065	0.43	39.5	-66.4	4.012
Average	***1.576***	***-0.066***	***0.13***	***34.0***	***-53.2***	***4.680***
ParkScene (1920×1024)	0.828	-0.027	-0.50	43.2	-82.3	1.919
Kimono1 (1920×1024)	0.427	-0.015	-0.01	41.5	-73.0	1.028
Cactus (1920×1024)	0.806	-0.018	-0.27	46.6	-92.1	1.728
BQTerrace (1920×1024)	0.749	-0.015	-0.76	47.3	-103.9	1.583
BasketballDrive (1920×1024)	1.597	-0.035	0.31	39.7	-68.5	4.018
Average	***0.881***	***-0.022***	***-0.24***	***43.7***	***-84.0***	***2.055***
Traffic (2560×1600)	0.657	-0.023	-0.60	47.3	-92.6	1.388
PeopleOnStreet (2560×1600)	1.348	-0.060	0.25	28.7	-42.7	4.704
NebutaFestival (2560×1600)	0.105	0.000	0.03	50.2	-102.1	0.209
SteamLocomotiveTrain (2560×1600)	-0.042	0.000	-0.04	53.4	-119.4	-0.079
Average	***0.517***	***-0.021***	***-0.09***	***44.9***	***-89.2***	***1.556***
Overall Average	***1.339***	***-0.054***	***0.038***	***39.1***	***-70.2***	***3.922***

Table 5.10: Model generation time to encoding time using modified encoder ratios using stepwise regression and second order polynomial classifier for early CU termination.

Video Sequence	$Time_{model\_t}$ (%)			
	QP = 22	**QP = 27**	**QP = 32**	**QP = 37**
RaceHorses (384×192)	0.08	0.12	0.11	0.11
BlowingBubbles (384×192)	0.20	0.08	0.11	0.11
BQSquare (384×192)	0.12	0.16	0.17	0.20
BasketballPass (384×192)	0.10	0.14	0.09	0.15
Average	***0.13***	***0.12***	***0.12***	***0.14***
RaceHorses (832×448)	0.09	0.10	0.10	0.10
PartyScene (832×448)	0.23	0.16	0.21	0.16
BQMall (832×448)	0.11	0.17	0.13	0.13
BasketballDrill (832×448)	0.10	0.08	0.13	0.09
Average	***0.13***	***0.13***	***0.14***	***0.12***
ParkScene (1920×1024)	0.15	2.23	0.24	0.26
Kimono1 (1920×1024)	0.05	0.20	0.11	0.10
Cactus (1920×1024)	0.26	0.19	0.15	0.39
BQTerrace (1920×1024)	0.34	0.16	0.27	0.43
BasketballDrive (1920×1024)	0.13	0.20	0.05	0.07
Average	***0.19***	***0.60***	***0.17***	***0.25***
Traffic (2560×1600)	1.39	0.26	1.76	0.21
PeopleOnStreet (2560×1600)	0.19	1.02	0.30	0.20
NebutaFestival (2560×1600)	0.08	0.07	0.14	0.10
SteamLocomotiveTrain (2560×1600)	0.11	0.11	0.14	0.13
Average	***0.44***	***0.37***	***0.58***	***0.16***
Overall Average	***0.219***	***0.321***	***0.249***	***0.172***

Table 5.10 summarizes the time percentage taken for the prediction model to be generated. This information is important as the proposed solution is a sequence-dependent one.

Table 5.11: Selected features per CU size using stepwise regression and second order polynomial classifier for early CU termination.

Video Sequence	64×64	32×32	16×16
RaceHorses (384×192)	10	11	15
BlowingBubbles (384×192)	4	13	12
BQSquare (384×192)	7	17	17
BasketballPass (384×192)	9	15	13
Average	***7***	***14***	***14***
RaceHorses (832×448)	7	14	19
PartyScene (832×448)	14	27	28
BQMall (832×448)	14	21	22
BasketballDrill (832×448)	13	13	16
Average	***12***	***18***	***21***
ParkScene (1920×1024)	20	30	26
Kimono1 (1920×1024)	17	16	14
Cactus (1920×1024)	20	22	28
BQTerrace (1920×1024)	19	24	27
BasketballDrive (1920×1024)	16	18	23
Average	***18***	***22***	***23***
Traffic (2560×1600)	22	25	26
PeopleOnStreet (2560×1600)	17	28	31
NebutaFestival (2560×1600)	16	20	24
SteamLocomotiveTrain (2560×1600)	18	20	24
Average	***18***	***23***	***26***
Overall Average	***14***	***19***	***21***

Table 5.12: Classification rates per each CU size using stepwise regression and second order polynomial classifier for early CU termination.

Video Sequence	64×64	32×32	16×16	True Overall
RaceHorses (384×192)	67.6	82.1	82.4	83.2
BlowingBubbles (384×192)	75.5	83.7	84.1	83.9
BQSquare (384×192)	80.6	86.4	88.7	87.1
BasketballPass (384×192)	84.6	86.2	86.3	85.7
Average	***77.1***	***84.6***	***85.4***	***85.0***
RaceHorses (832×448)	80.4	83.9	85.5	85.5
PartyScene (832×448)	89.1	87.5	87.4	86.8
BQMall (832×448)	90.1	86.8	86.1	85.7
BasketballDrill (832×448)	87.0	88.0	88.8	87.8
Average	***86.6***	***86.6***	***87.0***	***86.5***
ParkScene (1920×1024)	88.9	90.3	89.4	88.4
Kimono1 (1920×1024)	80.2	76.6	78.6	81.2
Cactus (1920×1024)	89.0	86.6	87.0	86.0
BQTerrace (1920×1024)	88.0	89.9	89.7	88.7
BasketballDrive (1920×1024)	86.3	84.1	85.5	84.5
Average	***86.5***	***85.5***	***86.0***	***85.7***
Traffic (2560×1600)	89.9	91.2	91.2	89.9
PeopleOnStreet (2560×1600)	90.3	86.7	82.8	83.7
NebutaFestival (2560×1600)	64.4	80.3	87.5	87.6
SteamLocomotiveTrain (2560×1600)	83.5	86.0	88.6	88.4
Average	***82.0***	***86.0***	***87.5***	***87.4***
Overall Average	***83.2***	***85.7***	***86.4***	***86.1***

The number of features retained when using stepwise regression is given in Table 5.11. On average, 18 features are selected, which are used to determine the dimensions over which a data sample is projected. The decision accuracies achieved by using the proposed algorithm can be seen in Table 5.12. A classification rate of around 86.1% is attained by the proposed scheme.

5.2.1.3. ***J48 decision trees classifier.*** Table 5.13 and 5.14 show the time savings and excessive bitrate per each QP for each of the test sequences acquired by applying the J48 classifier, respectively. As the QP value increases, it is observed that, on average, less coding bits and time are needed to encode a given video sequence. Overall, a CCR of 41.2% is attained at the cost of introducing performance losses of 0.745% and -0.029 dB in terms BD-rate and BD-PSNR, respectively. The results imply that, as the spatial resolution increases, more complexity reduction is accomplished, while an increase is seen in BD-rate. These results can be observed in Table 5.15.

Table 5.13: Time savings results per each QP using J48 decision trees classifier for early CU termination.

Video Sequence	**22**		**27**		**32**		**37**	
	CCR (%)	**ΔTime (%)**	**CCR (%)**	**ΔTime (%)**	**CCR (%)**	**ΔTime (%)**	**CCR (%)**	**ΔTime (%)**
RaceHorses (384×192)	19.4	-24.1	21.8	-27.9	22.9	-29.6	35.5	-55.0
BlowingBubbles (384×192)	21.6	-27.5	34.0	-51.4	36.4	-57.3	49.6	-98.5
BQSquare (384×192)	20.6	-25.9	37.6	-60.1	48.5	-94.0	55.5	-124.6
BasketballPass (384×192)	21.1	-26.7	31.4	-45.7	33.6	-50.6	48.2	-93.3
RaceHorses (832×448)	28.7	-40.3	25.9	-35.0	34.8	-53.2	41.9	-72.2
PartyScene (832×448)	24.0	-31.6	30.1	-43.0	39.9	-66.3	47.8	-91.6
BQMall (832×448)	22.6	-29.2	33.8	-51.2	39.3	-64.7	44.7	-80.8
BasketballDrill (832×448)	32.2	-47.5	40.6	-68.3	40.6	-68.2	49.1	-96.5
ParkScene (1920×1024)	34.4	-52.4	44.2	-79.3	53.2	-113.6	63.0	-169.9
Kimono1 (1920×1024)	24.3	-32.1	33.5	-50.5	44.6	-80.4	38.9	-63.7
Cactus (1920×1024)	40.5	-67.9	41.6	-71.2	51.5	-106.0	60.9	-155.6
BQTerrace (1920×1024)	23.7	-31.1	50.6	-102.6	63.1	-170.7	67.5	-207.3
BasketballDrive (1920×1024)	35.3	-54.6	39.6	-65.5	46.5	-86.7	52.5	-110.5
Traffic (2560×1600)	36.4	-57.2	51.2	-105.1	60.2	-151.0	66.2	-196.0
PeopleOnStreet (2560×1600)	23.9	-31.4	24.6	-32.7	27.5	-38.0	38.9	-63.7
NebutaFestival (2560×1600)	61.1	-156.8	46.6	-87.1	47.6	-91.0	61.6	-160.1
SteamLocomotiveTrain (2560×1600)	44.9	-81.5	54.4	-119.2	64.3	-179.8	69.8	-231.5
Average	***30.3***	***-48.1***	***37.7***	***-64.4***	***44.4***	***-88.3***	***52.4***	***-121.8***

Table 5.14: Excessive bitrate results per each QP using J48 decision trees classifier for early CU termination.

Video Sequence	Excessive Bitrate (%)			
	QP = 22	QP = 27	QP = 32	QP = 37
RaceHorses (384×192)	0.25	-0.04	-0.06	-0.13
BlowingBubbles (384×192)	-0.42	-0.11	-0.28	-0.28
BQSquare (384×192)	-0.25	-0.32	-0.54	0.22
BasketballPass (384×192)	0.02	0.14	-0.18	-0.25
RaceHorses (832×448)	0.54	0.00	-0.03	-0.31
PartyScene (832×448)	0.15	-0.07	-0.36	-0.18
BQMall (832×448)	-0.15	-0.19	-0.49	-0.14
BasketballDrill (832×448)	-0.01	-0.13	-0.08	-0.30
ParkScene (1920×1024)	-0.28	-0.19	-0.39	-0.51
Kimono1 (1920×1024)	0.01	0.13	0.01	-0.14
Cactus (1920×1024)	-0.21	-0.22	-0.12	-0.09
BQTerrace (1920×1024)	-0.29	-0.43	-0.39	-0.21
BasketballDrive (1920×1024)	-0.38	0.05	-0.05	0.17
Traffic (2560×1600)	-0.36	-0.30	-0.11	-0.24
PeopleOnStreet (2560×1600)	0.64	0.16	0.18	0.10
NebutaFestival (2560×1600)	0.47	0.46	0.33	-0.19
SteamLocomo-tiveTrain (2560×1600)	-0.28	-0.28	-0.33	-0.55
Average	***-0.03***	***-0.08***	***-0.17***	***-0.18***

Table 5.15: BD-rate, BD-PSNR, avg. excessive bitrate, and time savings results using J48 decision trees classifier for early CU termination.

Video Sequence	BD-rate (%)	BD-PSNR (dB)	Exc. Bitrate (%)	CCR (%)	ΔTime (%)	BD-rate/CCR (%)
RaceHorses (384×192)	2.279	-0.106	0.01	24.9	-34.2	9.154
BlowingBubbles (384×192)	1.207	-0.049	-0.27	35.4	-58.7	3.410
BQSquare (384×192)	0.568	-0.026	-0.22	40.5	-76.2	1.400
BasketballPass (384×192)	0.720	-0.036	-0.07	33.6	-54.1	2.145
Average	***1.193***	***-0.054***	***-0.14***	***33.6***	***-55.8***	***4.027***
RaceHorses (832×448)	1.772	-0.070	0.05	32.8	-50.2	5.398
PartyScene (832×448)	0.666	-0.030	-0.12	35.4	-58.1	1.880
BQMall (832×448)	0.551	-0.024	-0.24	35.1	-56.5	1.570
BasketballDrill (832×448)	0.749	-0.031	-0.13	40.6	-70.1	1.844
Average	***0.935***	***-0.039***	***-0.11***	***36.0***	***-58.7***	***2.673***
ParkScene (1920×1024)	0.348	-0.011	-0.34	48.7	-103.8	0.715
Kimono1 (1920×1024)	0.446	-0.015	0.00	35.3	-56.7	1.262
Cactus (1920×1024)	0.512	-0.010	-0.16	48.6	-100.2	1.054
BQTerrace (1920×1024)	0.558	-0.010	-0.33	51.2	-127.9	1.090
BasketballDrive (1920×1024)	0.612	-0.014	-0.05	43.5	-79.3	1.408
Average	***0.495***	***-0.012***	***-0.18***	***45.5***	***-93.6***	***1.106***
Traffic (2560×1600)	0.598	-0.020	-0.25	53.5	-127.3	1.117
PeopleOnStreet (2560×1600)	1.124	-0.050	0.27	28.7	-41.4	3.910
NebutaFestival (2560×1600)	0.306	0.000	0.27	54.2	-123.7	0.564
SteamLocomotiveTrain (2560×1600)	-0.343	0.000	-0.36	58.4	-153.0	-0.588
Average	***0.421***	***-0.017***	***-0.02***	***48.7***	***-111.4***	***1.251***
Overall Average	***0.745***	***-0.029***	***-0.115***	***41.2***	***-80.7***	***2.196***

Table 5.16: Model generation time to encoding time using modified encoder ratios using J48 decision trees classifier for early CU termination.

Video Sequence	$Time_{model\ t}$ (%)			
	QP = 22	**QP = 27**	**QP = 32**	**QP = 37**
RaceHorses (384×192)	0.09	0.11	0.13	0.15
BlowingBubbles (384×192)	0.10	0.15	0.15	0.18
BQSquare (384×192)	0.12	0.14	0.16	0.21
BasketballPass (384×192)	0.19	0.12	0.15	0.16
Average	***0.12***	***0.13***	***0.15***	***0.18***
RaceHorses (832×448)	0.08	0.09	0.09	0.09
PartyScene (832×448)	0.08	0.03	0.11	0.10
BQMall (832×448)	0.07	0.09	0.09	0.16
BasketballDrill (832×448)	0.08	0.10	0.10	0.10
Average	***0.08***	***0.08***	***0.10***	***0.11***
ParkScene (1920×1024)	0.12	0.64	0.23	0.83
Kimono1 (1920×1024)	0.10	1.51	0.51	0.12
Cactus (1920×1024)	0.44	0.10	0.18	1.26
BQTerrace (1920×1024)	0.27	0.63	0.33	0.29
BasketballDrive (1920×1024)	0.38	0.14	0.10	0.25
Average	***0.26***	***0.60***	***0.27***	***0.55***
Traffic (2560×1600)	0.19	0.17	0.31	0.44
PeopleOnStreet (2560×1600)	0.17	0.34	0.14	0.39
NebutaFestival (2560×1600)	0.21	0.39	0.41	0.42
SteamLocomotiveTrain (2560×1600)	0.22	0.12	0.62	0.14
Average	***0.20***	***0.26***	***0.37***	***0.35***
Overall Average	***0.170***	***0.286***	***0.224***	***0.310***

Table 5.16 summarizes the percentage of the time taken for the prediction model to be generated. This information is important as the proposed solution is a sequence-dependent one. The decision accuracies achieved by using the proposed algorithm can be seen in Table 5.17. A classification rate of around 87% is attained by the proposed scheme.

Table 5.17: Classification rates per each CU size using J48 decision trees classifier for early CU termination.

Video Sequence	64×64	32×32	16×16	True Overall
RaceHorses (384×192)	64.3	82.7	81.7	77.1
BlowingBubbles (384×192)	79.9	82.8	83.2	84.3
BQSquare (384×192)	82.1	88.8	87.0	88.9
BasketballPass (384×192)	87.8	87.4	84.4	86.9
Average	***78.5***	***85.4***	***84.1***	***84.3***
RaceHorses (832×448)	83.5	83.6	86.0	85.8
PartyScene (832×448)	88.8	87.8	87.2	87.8
BQMall (832×448)	90.7	87.2	85.6	87.0
BasketballDrill (832×448)	87.2	87.4	87.4	88.6
Average	***87.5***	***86.5***	***86.5***	***87.3***
ParkScene (1920×1024)	89.1	90.3	89.1	90.1
Kimono1 (1920×1024)	80.3	74.2	77.4	82.3
Cactus (1920×1024)	88.2	85.7	86.7	87.8
BQTerrace (1920×1024)	87.9	89.2	88.2	89.4
BasketballDrive (1920×1024)	86.2	82.0	84.9	85.9
Average	***86.3***	***84.3***	***85.3***	***87.1***
Traffic (2560×1600)	90.0	91.2	91.8	92.2
PeopleOnStreet (2560×1600)	91.3	86.9	82.5	84.2
NebutaFestival (2560×1600)	61.7	78.7	87.1	89.1
SteamLocomotiveTrain (2560×1600)	82.0	85.8	89.8	91.1
Average	***81.2***	***85.7***	***87.8***	***89.1***
Overall Average	***83.6***	***85.4***	***85.9***	***87.0***

5.2.1.4. ***Random forest feature importance and Random forest classifier.*** Tables 5.18 and 5.19 show the time savings and excessive bitrate attained per each QP for each of the test sequences, respectively. As the QP value increases, less coding bits and time are needed on average to encode a given video sequence. Overall, a CCR of 39.2% is attained at the cost of introducing performance losses of 0.558% and -0.022 dB in terms BD-rate and BD-PSNR, respectively. The results imply that, as the spatial resolution increases, more complexity reduction is accomplished, while a reduction is seen in BD-rate. These results can be observed in Table 5.20.

Table 5.18: Time savings results per each QP using random forest feature importance and random forest classifier for early CU termination.

Video Sequence	22		27		32		37	
	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)
RaceHorses (384×192)	16.3	-19.4	23.0	-29.9	25.3	-33.8	34.5	-52.5
BlowingBubbles (384×192)	20.2	-25.3	28.9	-40.6	36.2	-56.9	42.6	-74.3
BQSquare (384×192)	19.8	-24.7	30.2	-43.2	41.0	-69.5	49.9	-99.6
BasketballPass (384×192)	22.5	-29.1	26.6	-36.1	35.0	-53.7	38.4	-62.2
RaceHorses (832×448)	26.0	-35.2	26.5	-36.0	34.3	-52.2	40.0	-66.6
PartyScene (832×448)	24.8	-33.0	28.3	-39.4	35.3	-54.5	44.0	-78.5
BQMall (832×448)	24.2	-31.8	30.4	-43.6	35.2	-54.4	42.4	-73.5
BasketballDrill (832×448)	31.6	-46.2	36.3	-57.1	41.5	-71.0	47.3	-89.7
ParkScene (1920×1024)	30.2	-43.3	39.9	-66.3	49.2	-96.8	56.2	-128.5
Kimono1 (1920×1024)	40.6	-68.3	36.8	-58.2	45.1	-82.2	54.0	-117.2
Cactus (1920×1024)	28.0	-38.9	36.6	-57.7	48.7	-95.0	51.8	-107.3
BQTerrace (1920×1024)	25.2	-33.7	44.5	-80.2	59.0	-143.6	65.1	-186.3
BasketballDrive (1920×1024)	34.1	-51.8	41.4	-70.6	45.3	-82.8	49.4	-97.7
Traffic (2560×1600)	34.9	-53.6	46.7	-87.8	54.3	-118.7	59.0	-143.8
PeopleOnStreet (2560×1600)	26.9	-36.8	30.1	-43.0	26.9	-36.7	38.7	-63.1
NebutaFestival (2560×1600)	54.2	-118.2	46.4	-86.5	47.6	-91.0	58.8	-142.7
SteamLocomotiveTrain (2560×1600)	45.8	-84.4	52.4	-110.0	58.0	-138.2	64.3	-179.8
Average	***29.7***	***-45.5***	***35.6***	***-58.0***	***42.2***	***-78.3***	***49.2***	***-103.7***

Table 5.19: Excessive bitrate results per each QP using random forest feature importance and random forest classifier for early CU termination.

Video Sequence	Excessive Bitrate (%)			
	QP = 22	QP = 27	QP = 32	QP = 37
RaceHorses (384×192)	0.11	-0.23	-0.32	-0.67
BlowingBubbles (384×192)	-0.57	-0.48	-0.31	-0.37
BQSquare (384×192)	-0.16	-0.29	-0.26	-0.12
BasketballPass (384×192)	0.14	-0.06	0.03	-0.44
RaceHorses (832×448)	0.17	-0.15	-0.31	-0.61
PartyScene (832×448)	0.15	-0.24	-0.32	-0.57
BQMall (832×448)	-0.24	-0.27	-0.48	-0.40
BasketballDrill (832×448)	0.12	-0.10	-0.06	-0.46
ParkScene (1920×1024)	-0.44	-0.45	-0.67	-0.81
Kimono1 (1920×1024)	0.17	0.21	-0.14	-0.06
Cactus (1920×1024)	-0.39	-0.28	-0.18	-0.32
BQTerrace (1920×1024)	-0.52	-0.95	-0.89	-0.53
BasketballDrive (1920×1024)	-0.50	-0.11	-0.16	-0.15
Traffic (2560×1600)	-0.73	-0.55	-0.45	-0.71
PeopleOnStreet (2560×1600)	0.35	0.01	0.02	-0.07
NebutaFestival (2560×1600)	0.02	0.18	0.07	-0.29
SteamLocomo-tiveTrain (2560×1600)	-0.05	-0.31	-0.13	-0.46
Average	***-0.14***	***-0.24***	***-0.27***	***-0.41***

Table 5.20: BD-rate, BD-PSNR, avg. excessive bitrate, and time savings results using random forest feature importance and random forest classifier for early CU termination.

Video Sequence	BD-rate (%)	BD-PSNR (dB)	Exc. Bitrate (%)	CCR (%)	ΔTime (%)	BD-rate/CCR (%)
RaceHorses (384×192)	1.583	-0.075	-0.28	24.8	-33.9	6.397
BlowingBubbles (384×192)	0.572	-0.023	-0.43	32.0	-49.3	1.790
BQSquare (384×192)	0.460	-0.021	-0.21	35.2	-59.3	1.306
BasketballPass (384×192)	0.334	-0.017	-0.08	30.6	-45.3	1.092
Average	***0.737***	***-0.034***	***-0.25***	***30.6***	***-46.9***	***2.646***
RaceHorses (832×448)	0.835	-0.033	-0.23	31.7	-47.5	2.635
PartyScene (832×448)	0.687	-0.032	-0.25	33.1	-51.4	2.076
BQMall (832×448)	0.484	-0.021	-0.35	33.0	-50.8	1.466
BasketballDrill (832×448)	0.520	-0.021	-0.13	39.2	-66.0	1.328
Average	***0.632***	***-0.027***	***-0.24***	***34.2***	***-53.9***	***1.876***
ParkScene (1920×1024)	0.630	-0.020	-0.59	43.9	-83.7	1.436
Kimono1 (1920×1024)	0.468	-0.016	0.05	44.1	-81.5	1.061
Cactus (1920×1024)	0.561	-0.014	-0.29	41.3	-74.7	1.359
BQTerrace (1920×1024)	0.478	-0.010	-0.72	48.4	-111.0	0.987
BasketballDrive (1920×1024)	0.458	-0.010	-0.23	42.6	-75.7	1.075
Average	***0.519***	***-0.014***	***-0.36***	***44.1***	***-85.3***	***1.184***
Traffic (2560×1600)	0.540	-0.019	-0.61	48.7	-101.0	1.108
PeopleOnStreet (2560×1600)	1.066	-0.047	0.08	30.6	-44.9	3.481
NebutaFestival (2560×1600)	0.047	0.000	0.00	51.7	-109.6	0.091
SteamLocomotiveTrain (2560×1600)	-0.234	0.000	-0.24	55.1	-128.1	-0.424
Average	***0.355***	***-0.017***	***-0.19***	***46.5***	***-95.9***	***1.064***
Overall Average	***0.558***	***-0.022***	***-0.265***	***39.2***	***-71.4***	***1.663***

Table 5.21: Model generation time to encoding time using modified encoder ratios using random forest feature importance and random forest classifier for early CU termination.

Video Sequence	$Time_{model\_t}$ (%)			
	QP = 22	**QP = 27**	**QP = 32**	**QP = 37**
RaceHorses (384×192)	8.52	8.39	12.27	15.06
BlowingBubbles (384×192)	10.42	14.07	16.93	20.07
BQSquare (384×192)	12.93	16.62	19.99	25.17
BasketballPass (384×192)	12.25	13.98	16.68	19.20
Average	***11.03***	***13.26***	***16.47***	***19.88***
RaceHorses (832×448)	4.18	4.30	5.00	5.47
PartyScene (832×448)	4.82	5.33	6.27	7.22
BQMall (832×448)	4.50	5.14	5.51	6.10
BasketballDrill (832×448)	4.86	4.83	5.94	6.36
Average	***4.59***	***4.90***	***5.68***	***6.29***
ParkScene (1920×1024)	3.33	3.25	3.08	2.80
Kimono1 (1920×1024)	2.08	1.76	1.87	2.27
Cactus (1920×1024)	3.40	2.76	2.77	2.65
BQTerrace (1920×1024)	3.44	3.65	3.71	3.31
BasketballDrive (1920×1024)	2.23	1.90	1.84	1.93
Average	***2.90***	***2.66***	***2.65***	***2.59***
Traffic (2560×1600)	4.40	4.24	4.62	3.90
PeopleOnStreet (2560×1600)	7.40	5.24	3.87	3.96
NebutaFestival (2560×1600)	1.18	1.18	1.88	2.77
SteamLocomotiveTrain (2560×1600)	2.83	2.93	2.87	2.86
Average	***3.95***	***3.40***	***3.31***	***3.38***
Overall Average	***5.459***	***5.857***	***6.771***	***7.712***

Table 5.22: Selected features per CU size using random forest feature importance and random forest classifier for early CU termination.

Video Sequence	64×64	32×32	16×16
RaceHorses (384×192)	14	11	13
BlowingBubbles (384×192)	8	13	10
BQSquare (384×192)	12	14	14
BasketballPass (384×192)	12	13	14
Average	***11***	***12***	***12***
RaceHorses (832×448)	9	14	14
PartyScene (832×448)	15	15	15
BQMall (832×448)	13	14	14
BasketballDrill (832×448)	14	15	13
Average	***12***	***14***	***14***
ParkScene (1920×1024)	15	14	14
Kimono1 (1920×1024)	15	14	14
Cactus (1920×1024)	15	15	12
BQTerrace (1920×1024)	15	15	14
BasketballDrive (1920×1024)	15	15	14
Average	***15***	***14***	***13***
Traffic (2560×1600)	15	15	15
PeopleOnStreet (2560×1600)	14	14	14
NebutaFestival (2560×1600)	13	13	14
SteamLocomotiveTrain (2560×1600)	15	15	14
Average	***14***	***14***	***14***
Overall Average	***13***	***14***	***13***

Table 5.21 summarizes percentage of the time taken for the prediction model to be generated. This information is important as the proposed solution is a sequence-dependent one. The number of features retained when using the feature importance option accessible through the usage of the random forest algorithm is given in Table 5.22.

Table 5.23: Classification rates per each CU size using random forest feature importance and random forest classifier for early CU termination.

Video Sequence	64×64	32×32	16×16	True Overall
RaceHorses (384×192)	79.0	83.8	83.0	83.9
BlowingBubbles (384×192)	85.1	84.7	83.7	83.9
BQSquare (384×192)	82.6	87.7	87.2	86.6
BasketballPass (384×192)	87.1	88.1	83.6	84.7
Average	***83.4***	***86.1***	***84.4***	***84.8***
RaceHorses (832×448)	85.3	84.7	86.0	86.2
PartyScene (832×448)	89.7	88.3	87.8	87.4
BQMall (832×448)	91.6	87.7	85.2	85.5
BasketballDrill (832×448)	89.1	88.9	87.7	87.4
Average	***88.9***	***87.4***	***86.7***	***86.6***
ParkScene (1920×1024)	88.4	90.6	89.7	88.9
Kimono1 (1920×1024)	80.3	77.0	82.3	83.3
Cactus (1920×1024)	89.1	86.4	88.2	88.6
BQTerrace (1920×1024)	88.1	90.1	89.6	88.9
BasketballDrive (1920×1024)	86.6	83.8	86.6	85.5
Average	***86.5***	***85.5***	***87.3***	***87.0***
Traffic (2560×1600)	89.6	91.1	91.2	90.2
PeopleOnStreet (2560×1600)	91.3	87.4	83.4	84.4
NebutaFestival (2560×1600)	64.0	80.5	88.3	89.0
SteamLocomotiveTrain (2560×1600)	82.7	87.0	90.7	90.0
Average	***81.9***	***86.5***	***88.4***	***88.4***
Overall Average	***85.3***	***86.3***	***86.7***	***86.7***

On average, 14 features are selected, which are used to determine the dimensions over which a data sample is projected. The decision accuracies achieved by using the proposed algorithm can be seen in Table 5.23. A classification rate of around 86.7% is attained by the proposed scheme.

5.2.1.5. ***Random forest classifier.*** Tables 5.24 and 5.25 show the time savings and excessive bitrate per each QP for each of the test sequences that are acquired by applying the random forest classifier, respectively. As the QP value increases, it is observed that, on average, less coding bits and time are required to encode a given video sequence. Overall, a CCR of 38.9% is attained at the cost of introducing performance losses of 0.539% and -0.021 dB in terms BD-rate and BD-PSNR, respectively. The results imply that, as the spatial resolution increases, more complexity reduction is accomplished, while a reduction is seen in BD-rate. These results can be observed in Table 5.26.

Table 5.24: Time savings results per each QP using random forest classifier for early CU termination.

Video Sequence	**22**		**27**		**32**		**37**	
	CCR (%)	**ΔTime (%)**	**CCR (%)**	**ΔTime (%)**	**CCR (%)**	**ΔTime (%)**	**CCR (%)**	**ΔTime (%)**
RaceHorses (384×192)	20.0	-25.1	21.6	-27.5	25.4	-34.0	34.1	-51.8
BlowingBubbles (384×192)	21.9	-28.0	28.7	-40.2	34.3	-52.2	42.2	-73.0
BQSquare (384×192)	18.0	-22.0	32.6	-48.4	42.0	-72.4	50.1	-100.1
BasketballPass (384×192)	24.3	-32.0	28.7	-40.2	35.5	-55.0	39.5	-65.4
RaceHorses (832×448)	25.4	-34.0	27.6	-38.1	32.6	-48.3	38.6	-62.8
PartyScene (832×448)	24.5	-32.4	26.1	-35.3	38.3	-62.1	44.7	-80.9
BQMall (832×448)	22.8	-29.5	28.9	-40.7	36.3	-56.9	40.5	-68.1
BasketballDrill (832×448)	32.5	-48.2	35.2	-54.3	40.2	-67.3	45.8	-84.4
ParkScene (1920×1024)	29.4	-41.6	40.6	-68.5	48.0	-92.3	56.3	-129.0
Kimono1 (1920×1024)	40.0	-66.8	39.4	-64.9	40.9	-69.2	50.5	-102.2
Cactus (1920×1024)	22.2	-28.6	38.7	-63.1	42.3	-73.2	52.1	-108.7
BQTerrace (1920×1024)	31.1	-45.1	44.6	-80.4	58.1	-138.4	65.2	-187.4
BasketballDrive (1920×1024)	32.1	-47.4	38.4	-62.3	45.3	-82.8	49.4	-97.5
Traffic (2560×1600)	34.6	-52.8	47.8	-91.7	51.8	-107.5	60.7	-154.6
PeopleOnStreet (2560×1600)	21.8	-27.9	23.2	-30.2	30.0	-42.9	36.5	-57.4
NebutaFestival (2560×1600)	56.5	-129.7	50.1	-100.3	47.5	-90.3	57.4	-134.8
SteamLocomotiveTrain (2560×1600)	48.5	-94.3	53.1	-113.0	57.8	-136.8	64.3	-180.4
Average	***29.7***	***-46.2***	***35.6***	***-58.8***	***41.5***	***-75.4***	***48.7***	***-102.3***

Table 5.25: Excessive bitrate results per each QP using random forest classifier for early CU termination.

Video Sequence	Excessive Bitrate (%)			
	QP = 22	QP = 27	QP = 32	QP = 37
RaceHorses (384×192)	0.39	-0.07	-0.36	-0.70
BlowingBubbles (384×192)	-0.35	-0.28	-0.28	-0.50
BQSquare (384×192)	-0.14	-0.72	-0.45	-0.05
BasketballPass (384×192)	0.11	-0.07	-0.13	0.08
RaceHorses (832×448)	0.35	-0.12	-0.32	-0.66
PartyScene (832×448)	0.09	-0.38	-0.31	-0.68
BQMall (832×448)	-0.14	-0.25	-0.29	-0.57
BasketballDrill (832×448)	0.00	-0.09	-0.08	-0.47
ParkScene (1920×1024)	-0.44	-0.45	-0.68	-0.85
Kimono1 (1920×1024)	0.14	0.09	-0.20	-0.20
Cactus (1920×1024)	-0.37	-0.31	-0.22	-0.45
BQTerrace (1920×1024)	-0.46	-0.84	-0.87	-0.56
BasketballDrive (1920×1024)	-0.47	-0.12	-0.07	-0.11
Traffic (2560×1600)	-0.71	-0.50	-0.50	-0.81
PeopleOnStreet (2560×1600)	0.38	0.03	0.01	-0.20
NebutaFestival (2560×1600)	0.02	0.26	0.17	-0.33
SteamLocomo-tiveTrain (2560×1600)	-0.07	-0.27	-0.19	-0.59
Average	***-0.10***	***-0.24***	***-0.28***	***-0.45***

Table 5.26: BD-rate, BD-PSNR, avg. excessive bitrate, and time savings results using random forest classifier for early CU termination.

Video Sequence	BD-rate (%)	BD-PSNR (dB)	Exc. Bitrate (%)	CCR (%)	ΔTime (%)	BD-rate/CCR (%)
RaceHorses (384×192)	1.478	-0.071	-0.19	25.3	-34.6	5.848
BlowingBubbles (384×192)	0.525	-0.021	-0.35	31.8	-48.4	1.653
BQSquare (384×192)	0.419	-0.019	-0.34	35.7	-60.7	1.174
BasketballPass (384×192)	0.504	-0.025	0.00	32.0	-48.2	1.575
Average	***0.732***	***-0.034***	***-0.22***	***31.2***	***-48.0***	***2.563***
RaceHorses (832×448)	0.782	-0.031	-0.19	31.0	-45.8	2.520
PartyScene (832×448)	0.508	-0.024	-0.32	33.4	-52.7	1.521
BQMall (832×448)	0.384	-0.016	-0.31	32.1	-48.8	1.197
BasketballDrill (832×448)	0.581	-0.024	-0.16	38.4	-63.5	1.511
Average	***0.564***	***-0.024***	***-0.25***	***33.7***	***-52.7***	***1.687***
ParkScene (1920×1024)	0.621	-0.020	-0.61	43.6	-82.8	1.426
Kimono1 (1920×1024)	0.379	-0.013	-0.04	42.7	-75.8	0.887
Cactus (1920×1024)	0.522	-0.012	-0.34	38.8	-68.4	1.345
BQTerrace (1920×1024)	0.686	-0.013	-0.68	49.7	-112.8	1.379
BasketballDrive (1920×1024)	0.500	-0.010	-0.19	41.3	-72.5	1.211
Average	***0.542***	***-0.014***	***-0.37***	***43.2***	***-82.5***	***1.250***
Traffic (2560×1600)	0.449	-0.016	-0.63	48.7	-101.6	0.921
PeopleOnStreet (2560×1600)	0.981	-0.044	0.06	27.9	-39.6	3.519
NebutaFestival (2560×1600)	0.112	0.000	0.03	52.9	-113.8	0.212
SteamLocomotiveTrain (2560×1600)	-0.265	0.000	-0.28	55.9	-131.1	-0.474
Average	***0.319***	***-0.015***	***-0.21***	***46.3***	***-96.5***	***1.045***
Overall Average	***0.539***	***-0.021***	***-0.267***	***38.9***	***-70.7***	***1.613***

Table 5.27 summarizes percentage of the time taken for the prediction model to be generated. This information is important as the proposed solution is a sequence-dependent one. The decision accuracies achieved by using the proposed algorithm can

be seen in Table 5.28. A classification rate of around 86.5% is attained by the proposed scheme.

Table 5.27: Model generation time to encoding time using modified encoder ratios using random forest classifier for early CU termination.

Video Sequence	$Time_{model\ t}$ (%)			
	QP = 22	**QP = 27**	**QP = 32**	**QP = 37**
RaceHorses (384×192)	13.06	12.53	18.32	22.00
BlowingBubbles (384×192)	16.88	20.84	24.44	29.05
BQSquare (384×192)	17.99	23.99	28.03	33.24
BasketballPass (384×192)	18.51	21.71	24.46	28.09
Average	***16.61***	***19.77***	***23.81***	***28.09***
RaceHorses (832×448)	6.07	6.22	7.00	7.46
PartyScene (832×448)	6.92	7.19	8.56	10.13
BQMall (832×448)	6.33	7.29	7.71	8.56
BasketballDrill (832×448)	7.03	6.95	8.06	9.19
Average	***6.59***	***6.91***	***7.83***	***8.83***
ParkScene (1920×1024)	4.82	4.60	4.25	4.03
Kimono1 (1920×1024)	2.68	2.42	2.28	2.51
Cactus (1920×1024)	4.91	4.04	4.04	3.82
BQTerrace (1920×1024)	4.95	4.74	5.21	8.84
BasketballDrive (1920×1024)	3.16	2.46	2.52	2.40
Average	***4.10***	***3.65***	***3.66***	***4.32***
Traffic (2560×1600)	5.97	4.79	4.06	3.55
PeopleOnStreet (2560×1600)	5.56	4.86	4.42	4.35
NebutaFestival (2560×1600)	0.96	1.11	1.68	2.55
SteamLocomotiveTrain (2560×1600)	2.96	2.77	2.51	2.23
Average	***3.86***	***3.38***	***3.17***	***3.17***
Overall Average	***7.574***	***8.147***	***9.268***	***10.705***

Table 5.28: Classification rates per each CU size using random forest classifier for early CU termination.

Video Sequence	64×64	32×32	16×16	True Overall
RaceHorses (384×192)	91.1	83.8	83.2	83.9
BlowingBubbles (384×192)	84.1	84.5	84.2	83.9
BQSquare (384×192)	82.9	87.8	87.6	86.7
BasketballPass (384×192)	89.4	88.5	84.2	85.0
Average	***86.9***	***86.1***	***84.8***	***84.9***
RaceHorses (832×448)	85.8	84.6	86.0	86.0
PartyScene (832×448)	90.0	88.5	87.7	87.3
BQMall (832×448)	91.5	87.4	84.9	85.1
BasketballDrill (832×448)	89.4	89.1	87.7	87.2
Average	***89.2***	***87.4***	***86.6***	***86.4***
ParkScene (1920×1024)	89.1	90.9	89.6	88.8
Kimono1 (1920×1024)	80.5	77.7	81.5	82.6
Cactus (1920×1024)	89.4	86.7	87.3	86.4
BQTerrace (1920×1024)	88.5	90.3	89.9	89.0
BasketballDrive (1920×1024)	86.4	83.6	86.5	85.2
Average	***86.8***	***85.8***	***87.0***	***86.4***
Traffic (2560×1600)	90.3	91.2	91.3	90.2
PeopleOnStreet (2560×1600)	91.7	87.5	82.9	84.1
NebutaFestival (2560×1600)	62.9	81.1	90.0	89.5
SteamLocomotiveTrain (2560×1600)	82.5	87.0	90.5	89.7
Average	***81.8***	***86.7***	***88.7***	***88.4***
Overall Average	***86.2***	***86.5***	***86.8***	***86.5***

5.2.1.6. *Summary of CU split prediction results.* Tables 5.29-5.32 illustrate a summary of the overall averages of each of the proposed early CU termination schemes.

Table 5.29: BD-rate, BD-PSNR, avg. excessive bitrate, and time savings results for the proposed early CU termination algorithms.

Proposed early CU termination algorithms	BD-rate (%)	BD-PSNR (dB)	Exc. Bitrate (%)	CCR (%)	ΔTime (%)	BD-rate/CCR (%)
PCA90+RM2	1.355	-0.053	0.208	37.5	-67.2	4.253
Stepwise+RM2	1.339	-0.054	0.038	39.1	-70.2	3.922
DecisionTrees	0.745	-0.029	-0.115	41.2	-80.7	2.196
RFselect+RF	0.558	-0.022	-0.265	39.2	-71.4	1.663
RF	0.539	-0.021	-0.267	38.9	-70.7	1.613

Table 5.30: Model generation time to encoding time using modified encoder ratios for all proposed early CU termination algorithms.

Proposed early CU termination algorithms	$Time_{model\_t}$ (%)			
	QP = 22	QP = 27	QP = 32	QP = 37
PCA90+RM2	0.718	1.112	1.956	1.267
Stepwise+RM2	0.219	0.321	0.249	0.172
DecisionTrees	0.170	0.286	0.224	0.310
RFselect+RF	5.459	5.857	6.771	7.712
RF	7.574	8.147	9.268	10.705

Table 5.31: Selected features per CU size for proposed early CU termination algorithms that utilize feature selection.

Proposed early CU termination algorithms	64×64	32×32	16×16
Stepwise +RM2	14	19	21
RFselect+RF	13	14	13

Table 5.32: Classification rates per each CU size for all proposed early CU termination algorithms.

Proposed early CU termination algorithms	64×64	32×32	16×16	True Overall
PCA90+RM2	78.5	81.9	83.2	83.0
Stepwise+RM2	83.2	85.7	86.4	86.1
DecisionTrees	83.6	85.4	85.9	87.0
RFselect+RF	85.3	86.3	86.7	86.7
RF	86.2	86.5	86.8	86.5

Based on the overall averages of these solutions, it is evident that the scheme that utilizes the random forest approach outperforms the rest in terms of BD-rate, BD-PSNR and excessive bitrate. Its usage resulted in a BD-rate of 0.539%, BD-PSNR of -0.021 dB, and bitrate reduction of 0.267%. Nonetheless, this led to introducing a CCR of 38.9%, which does not represent the highest attained time saving in comparison to the other proposed schemes. Utilizing its predictive model led to a decision accuracy of 86.5%, which is very close to the highest overall true accuracy offered by one of the solutions proposed, i.e. the J48 decision trees. Nonetheless, the time needed to generate these model is significantly higher than that of other solutions. Random forest model generation time is significantly higher due to using the entire set of initial features and

not applying any pruning to the built trees. Relative to the other approaches, J48 decision trees resulted in the highest CCR, the least time needed to generate the predictive models and the highest true overall decision accuracy. However, its performance in terms of BD-rate, BD-PSNR and excessive bitrate was average in comparison to that of other proposed schemes. On the other hand, the least performance enhancement was seen when combining PCA with the second order polynomial classifier, resulting in a DB-rate of 1.355%, DB-PSNR of -0.053 dB, excessive bitrate of 0.208%, an overall decision accuracy of 83% and a CCR of 37.5%.

5.2.2. PU early termination algorithms. The results obtained by implementing two machine learning algorithms to enhance the PU mode selection are presented. These results are given in terms of BD-rate, BD-PSNR, excessive bitrate, computation complexity savings, model generation time, feature selection or extraction, and decision accuracy.

5.2.2.1. *Random forest classifier.* Tables 5.33 and 5.34 show the time savings and excessive bitrate per each QP for each of the test sequences that are acquired by applying the random forest classifier, respectively. As the QP value increases, it is evident that, on average, less coding bits and time are needed to encode a given video sequence.

Table 5.33: Time savings results per each QP using random forest classifier for early PU termination.

Video Sequence	22		27		32		37	
	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)
RaceHorses (384×192)	15.2	-17.9	22.8	-29.6	31.5	-46.0	31.2	-45.3
BlowingBubbles (384×192)	16.2	-19.3	22.6	-29.3	30.3	-43.4	35.1	-54.1
BQSquare (384×192)	14.1	-16.4	28.6	-40.1	28.9	-40.6	41.2	-70.1
BasketballPass (384×192)	19.6	-24.3	25.4	-34.1	33.6	-50.7	36.1	-56.4
RaceHorses (832×448)	14.2	-16.5	32.4	-47.9	23.4	-30.6	37.9	-61.0
PartyScene (832×448)	16.7	-20.0	29.9	-42.6	30.2	-43.2	32.4	-47.9
BQMall (832×448)	17.9	-21.8	24.9	-33.2	31.2	-45.4	35.7	-55.4
BasketballDrill (832×448)	16.5	-19.7	31.3	-45.6	34.1	-51.8	38.5	-62.7
ParkScene (1920×1024)	20.9	-26.5	30.1	-43.0	29.4	-41.7	47.6	-90.7
Kimono1 (1920×1024)	21.0	-26.6	27.0	-37.0	29.6	-41.9	39.4	-64.9
Cactus (1920×1024)	14.0	-16.3	15.3	-18.1	29.1	-41.1	41.0	-69.4
BQTerrace (1920×1024)	21.9	-28.0	40.7	-68.7	41.8	-71.9	44.3	-79.4
BasketballDrive (1920×1024)	15.1	-17.7	18.8	-23.1	31.9	-46.7	36.9	-58.4
Traffic (2560×1600)	28.2	-39.3	33.7	-50.9	45.9	-84.7	46.3	-86.3
PeopleOnStreet (2560×1600)	25.6	-34.5	14.9	-17.5	28.4	-39.6	32.0	-47.1
NebutaFestival (2560×1600)	10.2	-11.3	12.7	-14.5	13.3	-15.3	57.0	-132.6
SteamLocomotiveTrain (2560×1600)	19.2	-23.8	26.7	-36.4	37.3	-59.5	52.1	-108.8
Average	***18.0***	***-22.4***	***25.8***	***-36.0***	***31.2***	***-46.7***	***40.3***	***-70.0***

Table 5.34: Excessive bitrate results per each QP using random forest classifier for early PU termination.

Video Sequence	Excessive Bitrate (%)			
	QP = 22	QP = 27	QP = 32	QP = 37
RaceHorses (384×192)	1.02	0.86	0.54	-0.11
BlowingBubbles (384×192)	0.07	0.03	0.06	-0.14
BQSquare (384×192)	-0.03	-0.14	-0.18	0.07
BasketballPass (384×192)	0.02	0.10	0.08	0.46
RaceHorses (832×448)	0.10	1.21	0.34	0.24
PartyScene (832×448)	0.31	0.34	0.11	0.16
BQMall (832×448)	0.04	-0.02	-0.23	0.11
BasketballDrill (832×448)	0.02	0.11	-0.08	-0.23
ParkScene (1920×1024)	0.00	0.07	-0.09	0.00
Kimono1 (1920×1024)	0.06	0.06	-0.02	0.18
Cactus (1920×1024)	-0.12	0.08	0.01	0.12
BQTerrace (1920×1024)	0.02	-0.09	-0.11	0.09
BasketballDrive (1920×1024)	0.08	-0.05	-0.09	0.14
Traffic (2560×1600)	-0.06	0.01	0.08	0.01
PeopleOnStreet (2560×1600)	0.11	0.02	0.09	-0.06
NebutaFestival (2560×1600)	0.05	0.19	0.01	0.07
SteamLocomo-tiveTrain (2560×1600)	-0.10	-0.02	0.23	0.33
Average	***0.09***	***0.16***	***0.04***	***0.08***

Overall, a CCR of 28.8% is attained at the cost of introducing performance losses of 0.437% and -0.264 dB in terms BD-rate and BD-PSNR, respectively. The results imply that, as the spatial resolution increases, more complexity reduction is accomplished, while a reduction in BD-rate and an increase in BD-PSNR is seen. These results can be observed in Table 5.35.

Table 5.35: BD-rate, BD-PSNR, avg. excessive bitrate, and time savings results using random forest classifier for early PU termination.

Video Sequence	BD-rate (%)	BD-PSNR (dB)	Exc. Bitrate (%)	CCR (%)	ΔTime (%)	BD-rate/CCR (%)
RaceHorses (384×192)	1.500	-0.074	0.58	25.2	-34.7	5.956
BlowingBubbles (384×192)	0.538	-0.021	0.01	26.1	-36.5	2.067
BQSquare (384×192)	0.192	-4.213	-0.07	28.2	-41.8	0.681
BasketballPass (384×192)	0.494	-0.025	0.17	28.7	-41.4	1.723
Average	***0.681***	***-1.083***	***0.17***	***27.0***	***-38.6***	***2.607***
RaceHorses (832×448)	1.381	-0.054	0.47	27.0	-39.0	5.121
PartyScene (832×448)	0.626	-0.029	0.23	27.3	-38.4	2.295
BQMall (832×448)	0.242	-0.010	-0.03	27.4	-39.0	0.882
BasketballDrill (832×448)	0.240	-0.010	-0.05	30.1	-45.0	0.797
Average	***0.622***	***-0.026***	***0.16***	***27.9***	***-40.3***	***2.274***
ParkScene (1920×1024)	0.537	-0.017	-0.01	32.0	-50.5	1.679
Kimono1 (1920×1024)	0.078	-0.003	0.07	29.2	-42.6	0.267
Cactus (1920×1024)	0.211	-0.004	0.02	24.9	-36.2	0.849
BQTerrace (1920×1024)	0.462	-0.009	-0.02	37.2	-62.0	1.243
BasketballDrive (1920×1024)	0.165	-0.003	0.02	25.6	-36.5	0.643
Average	***0.291***	***-0.007***	***0.02***	***29.8***	***-45.6***	***0.936***
Traffic (2560×1600)	0.287	-0.009	0.01	38.5	-65.3	0.745
PeopleOnStreet (2560×1600)	0.298	-0.013	0.04	25.2	-34.7	1.181
NebutaFestival (2560×1600)	0.070	0.000	0.08	23.3	-43.4	0.299
SteamLocomotiveTrain (2560×1600)	0.108	0.000	0.11	33.8	-57.1	0.318
Average	***0.191***	***-0.005***	***0.06***	***30.2***	***-50.1***	***0.636***
Overall Average	***0.437***	***-0.264***	***0.096***	***28.8***	***-43.8***	***1.573***

Table 5.36 summarizes the time percentage taken for the prediction model to be generated. This information is important as the proposed solution is a sequence-dependent one.

Table 5.36: Model generation time to encoding time using modified encoder ratios using random forest classifier for early PU termination.

Video Sequence	$Time_{model\_t}$ (%)			
	QP = 22	QP = 27	QP = 32	QP = 37
RaceHorses (384×192)	5.01	5.88	6.81	7.12
BlowingBubbles (384×192)	5.50	6.22	7.56	8.77
BQSquare (384×192)	5.16	6.43	7.79	9.16
BasketballPass (384×192)	5.42	6.31	7.62	8.39
Average	***5.27***	***6.21***	***7.44***	***8.36***
RaceHorses (832×448)	1.65	3.01	3.23	3.71
PartyScene (832×448)	2.15	3.18	3.23	3.19
BQMall (832×448)	2.37	2.74	3.08	3.65
BasketballDrill (832×448)	2.53	3.42	3.35	3.24
Average	***2.17***	***3.09***	***3.22***	***3.45***
ParkScene (1920×1024)	1.75	2.07	2.23	2.19
Kimono1 (1920×1024)	1.46	2.53	2.20	2.23
Cactus (1920×1024)	0.62	1.38	2.01	1.65
BQTerrace (1920×1024)	1.27	1.80	2.18	1.87
BasketballDrive (1920×1024)	1.89	1.69	1.95	1.45
Average	***1.40***	***1.89***	***2.12***	***1.88***
Traffic (2560×1600)	1.08	1.07	1.16	0.83
PeopleOnStreet (2560×1600)	0.86	0.71	0.94	0.83
NebutaFestival (2560×1600)	0.43	0.68	0.99	1.50
SteamLocomotiveTrain (2560×1600)	0.96	1.63	1.96	1.94
Average	***0.83***	***1.02***	***1.26***	***1.27***
Overall Average	***7.574***	***8.147***	***9.268***	***10.705***

The decision accuracies achieved by using the proposed algorithm can be seen in Table 5.37. A classification rate of around 69.0% is attained by the proposed scheme.

Table 5.37: Classification rates per each CU size using random forest classifier for early PU termination.

Video Sequence	64×64	32×32	16×16	8×8	True Overall
RaceHorses (384×192)	74.0	58.1	66.0	69.1	66.8
BlowingBubbles (384×192)	69.3	64.3	69.3	67.8	67.7
BQSquare (384×192)	72.7	65.2	69.8	73.3	70.3
BasketballPass (384×192)	72.3	71.2	75.1	73.0	72.9
Average	***72.1***	***64.7***	***70.0***	***70.8***	***69.4***
RaceHorses (832×448)	67.5	63.6	63.9	64.1	64.8
PartyScene (832×448)	65.6	66.2	67.4	70.1	67.3
BQMall (832×448)	74.5	70.6	71.7	68.6	71.3
BasketballDrill (832×448)	69.8	67.7	71.9	77.1	71.6
Average	***69.3***	***67.0***	***68.7***	***70.0***	***68.7***
ParkScene (1920×1024)	68.7	67.3	67.8	72.8	69.1
Kimono1 (1920×1024)	69.0	67.9	70.3	71.6	69.7
Cactus (1920×1024)	72.5	73.0	74.9	75.9	74.1
BQTerrace (1920×1024)	65.4	61.0	61.6	57.9	61.5
BasketballDrive (1920×1024)	66.5	64.1	66.5	69.9	66.7
Average	***68.4***	***66.6***	***68.2***	***69.6***	***68.2***
Traffic (2560×1600)	65.5	68.9	73.9	74.3	70.6
PeopleOnStreet (2560×1600)	72.1	73.3	74.9	75.8	74.0
NebutaFestival (2560×1600)	73.4	66.4	68.4	67.0	68.8
SteamLocomotiveTrain (2560×1600)	66.1	62.8	64.0	69.2	65.5
Average	***69.3***	***67.8***	***70.3***	***71.5***	***69.7***
Overall Average	***69.7***	***66.5***	***69.2***	***70.4***	***69.0***

5.2.2.2. *J48 decision trees classifier.* Tables 5.38 and 5.39 show the time savings and excessive bitrate per each QP for each of the test sequences that are attained by applying the J48 classifier, respectively. As the QP value increases, it can be seen that less coding bits and time are needed on average to encode a given video sequence. Overall, a CCR of 20.9% is attained at the cost of introducing performance losses of 0.248% and -0.010 dB in terms BD-rate and BD-PSNR, respectively. The results imply that, as the spatial resolution increases, more complexity reduction is accomplished, while a reduction is seen in BD-rate. These results can be observed in Table 5.40.

Table 5.38: Time savings results per each QP using J48 decision trees classifier for early PU termination.

Video Sequence	22		27		32		37	
	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)
RaceHorses (384×192)	15.6	-18.5	14.0	-16.3	14.2	-16.5	16.3	-19.4
BlowingBubbles (384×192)	9.0	-9.8	15.1	-17.8	21.6	-27.5	29.3	-41.4
BQSquare (384×192)	11.9	-13.5	20.8	-26.3	23.9	-31.4	33.3	-49.9
BasketballPass (384×192)	13.7	-15.9	19.7	-24.5	21.5	-27.4	30.6	-44.1
RaceHorses (832×448)	5.9	-6.2	11.8	-13.4	15.9	-18.9	18.9	-23.3
PartyScene (832×448)	9.1	-10.0	13.4	-15.5	20.5	-25.8	30.2	-43.3
BQMall (832×448)	15.1	-17.7	19.5	-24.2	24.0	-31.5	28.4	-39.6
BasketballDrill (832×448)	14.1	-16.5	20.4	-25.6	26.9	-36.9	32.8	-48.8
ParkScene (1920×1024)	13.6	-15.7	19.5	-24.3	27.0	-37.0	35.9	-55.9
Kimono1 (1920×1024)	12.4	-14.2	20.7	-26.1	20.2	-25.3	28.7	-40.2
Cactus (1920×1024)	15.7	-18.7	26.0	-35.1	33.1	-49.4	38.4	-62.4
BQTerrace (1920×1024)	10.6	-11.9	21.2	-26.9	25.3	-33.8	29.9	-42.7
BasketballDrive (1920×1024)	15.6	-18.5	13.4	-15.5	20.6	-26.0	28.2	-39.3
Traffic (2560×1600)	20.3	-25.5	27.1	-37.3	34.8	-53.3	42.0	-72.4
PeopleOnStreet (2560×1600)	8.2	-9.0	12.8	-14.6	19.3	-23.9	23.7	-31.1
NebutaFestival (2560×1600)	10.8	-12.2	9.9	-11.0	17.2	-20.8	29.6	-42.0
SteamLocomotiveTrain (2560×1600)	18.3	-22.5	18.2	-22.3	23.7	-31.1	28.9	-40.6
Average	***12.9***	***-15.1***	***17.9***	***-22.1***	***22.9***	***-30.4***	***29.7***	***-43.3***

Table 5.39: Excessive bitrate results per each QP using J48 decision trees classifier for early PU termination.

Video Sequence	Excessive Bitrate (%)			
	QP = 22	QP = 27	QP = 32	QP = 37
RaceHorses (384×192)	2.60	0.94	0.24	0.31
BlowingBubbles (384×192)	0.08	-0.04	-0.27	0.03
BQSquare (384×192)	0.13	-0.18	-0.19	0.25
BasketballPass (384×192)	0.00	-0.21	0.00	-0.02
RaceHorses (832×448)	0.02	0.60	0.30	-0.27
PartyScene (832×448)	0.19	0.06	0.01	-0.12
BQMall (832×448)	-0.04	-0.04	0.02	-0.04
BasketballDrill (832×448)	0.02	0.03	0.11	-0.14
ParkScene (1920×1024)	0.04	0.02	-0.15	0.12
Kimono1 (1920×1024)	0.06	0.07	0.01	0.24
Cactus (1920×1024)	-0.04	-0.02	-0.10	0.05
BQTerrace (1920×1024)	0.01	0.02	0.07	0.10
BasketballDrive (1920×1024)	0.08	-0.03	0.01	0.21
Traffic (2560×1600)	-0.03	0.00	-0.03	-0.04
PeopleOnStreet (2560×1600)	0.03	0.00	-0.03	-0.14
NebutaFestival (2560×1600)	0.02	0.17	-0.05	-0.15
SteamLocomo-tiveTrain (2560×1600)	-0.04	0.09	0.14	-0.06
Average	***0.18***	***0.09***	***0.01***	***0.02***

Table 5.40: BD-rate, BD-PSNR, avg. excessive bitrate, and time savings results using J48 decision trees classifier for early PU termination.

Video Sequence	BD-rate (%)	BD-PSNR (dB)	Exc. Bitrate (%)	CCR (%)	ΔTime (%)	BD-rate/CCR (%)
RaceHorses (384×192)	1.247	-0.062	1.02	15.0	-17.7	8.312
BlowingBubbles (384×192)	0.162	-0.006	-0.05	18.8	-24.1	0.864
BQSquare (384×192)	0.187	-0.008	0.00	22.5	-30.3	0.832
BasketballPass (384×192)	0.145	-0.008	-0.06	21.4	-28.0	0.678
Average	***0.435***	***-0.021***	***0.23***	***19.4***	***-25.0***	***2.672***
RaceHorses (832×448)	0.652	-0.026	0.16	13.1	-15.5	4.976
PartyScene (832×448)	0.113	-0.006	0.04	18.3	-23.6	0.618
BQMall (832×448)	0.214	-0.008	-0.03	21.7	-28.3	0.986
BasketballDrill (832×448)	0.214	-0.009	0.01	23.6	-31.9	0.908
Average	***0.298***	***-0.012***	***0.04***	***19.2***	***-24.8***	***1.872***
ParkScene (1920×1024)	0.248	-0.008	0.01	24.0	-33.2	1.033
Kimono1 (1920×1024)	0.286	-0.010	0.10	20.5	-26.5	1.395
Cactus (1920×1024)	0.133	-0.002	-0.03	28.3	-41.4	0.470
BQTerrace (1920×1024)	0.043	-0.001	0.05	21.8	-28.8	0.198
BasketballDrive (1920×1024)	0.221	-0.004	0.07	19.5	-24.8	1.135
Average	***0.186***	***-0.005***	***0.04***	***22.8***	***-30.9***	***0.846***
Traffic (2560×1600)	0.091	-0.004	-0.03	31.1	-47.1	0.293
PeopleOnStreet (2560×1600)	0.192	-0.009	-0.04	16.0	-19.6	1.200
NebutaFestival (2560×1600)	0.000	0.000	0.00	16.9	-21.5	-0.001
SteamLocomotiveTrain (2560×1600)	0.060	0.000	0.03	22.3	-29.1	0.269
Average	***0.086***	***-0.003***	***-0.01***	***21.6***	***-29.3***	***0.440***
Overall Average	***0.248***	***-0.010***	***0.074***	***20.9***	***-27.7***	***1.422***

Table 5.41 summarizes percentage of the time taken for the prediction model to be generated. This information is important as the proposed solution is a sequence-dependent one.

Table 5.41: Model generation time to encoding time using modified encoder ratios using J48 decision trees classifier for early PU termination.

Video Sequence	$Time_{model\_t}$ (%)			
	QP = 22	QP = 27	QP = 32	QP = 37
RaceHorses (384×192)	0.11	0.11	0.20	0.19
BlowingBubbles (384×192)	0.22	0.12	0.21	0.24
BQSquare (384×192)	0.11	0.15	0.21	0.25
BasketballPass (384×192)	0.12	0.13	0.13	0.23
Average	***0.14***	***0.13***	***0.19***	***0.23***
RaceHorses (832×448)	0.09	0.13	0.14	0.16
PartyScene (832×448)	0.11	0.13	0.16	0.25
BQMall (832×448)	0.13	0.13	0.19	0.21
BasketballDrill (832×448)	0.14	0.15	0.16	0.15
Average	***0.12***	***0.14***	***0.16***	***0.19***
ParkScene (1920×1024)	0.53	0.19	0.20	0.31
Kimono1 (1920×1024)	0.56	0.36	0.41	0.36
Cactus (1920×1024)	0.14	0.46	0.18	1.12
BQTerrace (1920×1024)	0.60	0.28	0.71	0.79
BasketballDrive (1920×1024)	0.34	0.93	0.40	0.87
Average	***0.43***	***0.44***	***0.38***	***0.69***
Traffic (2560×1600)	0.81	0.54	0.62	1.29
PeopleOnStreet (2560×1600)	0.46	0.49	0.97	0.87
NebutaFestival (2560×1600)	0.47	0.81	1.24	4.36
SteamLocomotiveTrain (2560×1600)	0.69	1.35	1.81	1.35
Average	***0.61***	***0.80***	***1.16***	***1.97***
Overall Average	***0.331***	***0.380***	***0.467***	***0.765***

The decision accuracies achieved by using the proposed algorithm can be seen in Table 5.42. A classification rate of around 78.1% is attained by the proposed scheme.

Table 5.42: Classification rates per each CU size using J48 decision trees classifier for early PU termination.

Video Sequence	64×64	32×32	16×16	8×8	True Overall
RaceHorses (384×192)	85.5	68.8	69.3	68.4	73.0
BlowingBubbles (384×192)	83.0	75.1	77.1	74.3	77.3
BQSquare (384×192)	79.1	77.9	80.1	76.6	78.4
BasketballPass (384×192)	81.8	79.6	82.0	79.8	80.8
Average	***82.3***	***75.3***	***77.1***	***74.8***	***77.4***
RaceHorses (832×448)	85.4	72.5	69.9	66.9	73.7
PartyScene (832×448)	83.8	80.9	79.7	74.7	79.8
BQMall (832×448)	87.2	80.7	82.2	78.7	82.2
BasketballDrill (832×448)	79.6	78.8	79.9	80.8	79.8
Average	***84.0***	***78.2***	***77.9***	***75.3***	***78.8***
ParkScene (1920×1024)	79.2	79.5	81.0	77.4	79.3
Kimono1 (1920×1024)	78.6	76.2	77.8	75.3	77.0
Cactus (1920×1024)	85.2	83.0	83.9	81.2	83.3
BQTerrace (1920×1024)	78.4	79.9	80.8	74.9	78.5
BasketballDrive (1920×1024)	76.3	75.2	74.6	73.0	74.8
Average	***79.5***	***78.7***	***79.6***	***76.3***	***78.6***
Traffic (2560×1600)	78.2	80.3	83.8	82.8	81.3
PeopleOnStreet (2560×1600)	86.0	79.9	76.3	73.3	78.9
NebutaFestival (2560×1600)	77.4	67.6	68.5	66.6	70.0
SteamLocomotiveTrain (2560×1600)	80.1	81.2	82.1	77.7	80.3
Average	***80.4***	***77.3***	***77.7***	***75.1***	***77.6***
Overall Average	***81.4***	***77.5***	***78.2***	***75.4***	***78.1***

5.2.2.3. *Summary of PU split prediction results.* Tables 5.43-5.45 illustrate a summary of the overall averages of each of the proposed early PU termination schemes.

Table 5.43: BD-rate, BD-PSNR, avg. excessive bitrate, and time savings results for the proposed early PU termination algorithms.

Proposed early PU termination algorithms	BD-rate (%)	BD-PSNR (dB)	Exc. Bitrate (%)	CCR (%)	ΔTime (%)	BD-rate/CCR (%)
RF	0.437	-0.264	0.096	28.8	-43.8	1.573
DecisionTrees	0.248	-0.010	0.074	20.9	-27.7	1.422

Table 5.44: Model generation time to encoding time using modified encoder ratios for all proposed early PU termination algorithms.

Proposed early PU termination algorithms	$Time_{model\ t}$ (%)			
	QP = 22	QP = 27	QP = 32	QP = 37
RF	7.574	8.147	9.268	10.705
DecisionTrees	0.331	0.380	0.467	0.765

Table 5.45: Classification rates per each CU size for all proposed early PU termination algorithms.

Proposed early PU termination algorithms	64×64	32×32	16×16	8×8	True Overall
RF	69.7	66.5	69.2	70.4	69.0
DecisionTrees	81.4	77.5	78.2	75.4	87.1

Based on the overall averages of the proposed solutions, it is evident that the scheme that utilizes J48 decision trees outperforms the random forest approach in terms of almost all aspects. Its usage resulted in a BD-rate of 0.248%, BD-PSNR of -0.010 dB and excessive bitrate of 0.074%. Nonetheless, this led to introducing a CCR of 20.9%, which is lower than that demonstrated by utilizing random forests. However, the time needed to generate the predictive model was significantly less and it displayed a higher accuracy of 78.1%. Random forest model generation time is significantly higher due to using the entire set of initial features and not applying any pruning to the built trees.

5.2.3. CU and PU early termination algorithms. The results obtained by combining the previous two approaches to enhance the CU size and PU mode selection are displayed. These results are given in terms of BD-rate, BD-PSNR, excessive bitrate, and computation complexity savings.

5.2.3.1. ***Random forest classifier for CU and PU predictions.*** Tables 5.46 and 5.47 show the time savings and excessive bitrate per each QP for each of the test sequences that are acquired by applying the random forest classifier for both CU and PU predictions, respectively. As the QP value increases, it is observed that, on average, less coding bits and time are required to encode a given video sequence.

Table 5.46: Time savings results per each QP using random forest classifier for CU and PU predictions.

Video Sequence	22		27		32		37	
	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)
RaceHorses (384×192)	39.3	-64.8	46.3	-86.1	47.5	-90.5	54.0	-117.5
BlowingBubbles (384×192)	34.0	-51.5	45.1	-82.3	51.2	-104.9	55.0	-122.0
BQSquare (384×192)	33.1	-49.5	45.9	-84.9	53.9	-116.7	64.1	-178.3
BasketballPass (384×192)	36.0	-56.2	43.3	-76.5	51.8	-107.1	59.0	-143.7
RaceHorses (832×448)	41.1	-69.7	55.2	-123.4	53.1	-113.0	61.8	-161.9
PartyScene (832×448)	40.0	-66.5	49.2	-96.9	51.4	-105.9	55.6	-125.1
BQMall (832×448)	30.8	-44.4	45.6	-83.9	55.4	-124.0	64.0	-177.9
BasketballDrill (832×448)	41.9	-72.2	56.3	-128.8	60.9	-155.5	66.9	-201.6
ParkScene (1920×1024)	43.7	-77.7	52.9	-112.5	61.5	-160.0	77.8	-350.1
Kimono1 (1920×1024)	53.5	-115.1	57.4	-134.8	59.2	-145.3	67.1	-203.6
Cactus (1920×1024)	37.9	-60.9	53.9	-117.0	63.0	-170.2	73.5	-276.7
BQTerrace (1920×1024)	44.6	-80.6	56.6	-130.2	71.2	-247.7	81.1	-430.0
BasketballDrive (1920×1024)	50.8	-103.1	50.2	-100.6	59.5	-147.0	66.0	-194.1
Traffic (2560×1600)	51.1	-104.3	60.4	-152.4	73.7	-279.5	77.9	-351.9
PeopleOnStreet (2560×1600)	47.1	-88.9	41.2	-70.0	46.0	-85.2	58.1	-138.6
NebutaFestival (2560×1600)	62.9	-169.5	58.9	-143.4	57.4	-134.7	81.3	-434.6
SteamLocomotiveTrain (2560×1600)	61.2	-157.8	62.4	-165.9	70.4	-238.0	80.6	-416.4
Average	***44.1***	***-84.3***	***51.8***	***-111.2***	***58.1***	***-148.5***	***67.3***	***-236.7***

Table 5.47: Excessive bitrate results per each QP using random forest classifier for CU and PU predictions.

Video Sequence	Excessive Bitrate (%)			
	QP = 22	QP = 27	QP = 32	QP = 37
RaceHorses (384×192)	4.51	4.33	2.10	0.67
BlowingBubbles (384×192)	0.82	0.72	0.62	0.08
BQSquare (384×192)	0.19	-0.14	-0.16	0.48
BasketballPass (384×192)	1.39	1.61	0.53	0.40
RaceHorses (832×448)	1.93	3.09	2.21	0.88
PartyScene (832×448)	1.25	0.82	0.58	-0.20
BQMall (832×448)	0.45	1.20	1.31	0.82
BasketballDrill (832×448)	0.56	0.72	0.59	0.15
ParkScene (1920×1024)	0.28	0.38	-0.15	-0.40
Kimono1 (1920×1024)	0.56	0.83	0.41	0.01
Cactus (1920×1024)	-0.22	0.24	0.39	-0.10
BQTerrace (1920×1024)	-0.19	-0.87	-0.88	-0.74
BasketballDrive (1920×1024)	-0.11	0.38	0.21	0.39
Traffic (2560×1600)	-0.13	0.26	0.17	-0.34
PeopleOnStreet (2560×1600)	2.24	1.59	1.73	1.45
NebutaFestival (2560×1600)	0.19	0.72	0.66	-0.33
SteamLocomo-tiveTrain (2560×1600)	0.18	0.17	0.18	-0.34
Average	***0.82***	***0.94***	***0.62***	***0.17***

Table 5.48: BD-rate, BD-PSNR, avg. excessive bitrate, and time savings results using random forest classifier for CU and PU predictions.

Video Sequence	BD-rate (%)	BD-PSNR (dB)	Exc. Bitrate (%)	CCR (%)	ΔTime (%)	BD-rate/CCR (%)
RaceHorses (384×192)	9.351	-0.434	2.90	46.8	-89.7	19.989
BlowingBubbles (384×192)	3.964	-0.157	0.56	46.3	-90.2	8.557
BQSquare (384×192)	1.684	-0.078	0.09	49.3	-107.4	3.419
BasketballPass (384×192)	3.426	-0.164	0.98	47.5	-95.9	7.209
Average	***4.606***	***-0.208***	***1.13***	***47.5***	***-95.8***	***9.794***
RaceHorses (832×448)	6.512	-0.251	2.03	52.8	-117.0	12.337
PartyScene (832×448)	3.028	-0.139	0.61	49.0	-98.6	6.174
BQMall (832×448)	3.363	-0.139	0.95	48.9	-107.6	6.870
BasketballDrill (832×448)	2.337	-0.095	0.51	56.5	-139.5	4.137
Average	***3.810***	***-0.156***	***1.02***	***51.8***	***-115.7***	***7.380***
ParkScene (1920×1024)	2.395	-0.076	0.03	59.0	-175.1	4.060
Kimono1 (1920×1024)	1.692	-0.055	0.45	59.3	-149.7	2.853
Cactus (1920×1024)	2.077	-0.048	0.08	57.1	-156.2	3.641
BQTerrace (1920×1024)	1.719	-0.033	-0.67	63.4	-222.1	2.712
BasketballDrive (1920×1024)	1.579	-0.035	0.22	56.6	-136.2	2.790
Average	***1.892***	***-0.049***	***0.02***	***59.1***	***-167.9***	***3.211***
Traffic (2560×1600)	2.538	-0.086	-0.01	65.7	-222.0	3.861
PeopleOnStreet (2560×1600)	4.429	-0.194	1.75	48.1	-95.7	9.211
NebutaFestival (2560×1600)	0.505	0.000	0.31	65.1	-220.5	0.775
SteamLocomotiveTrain (2560×1600)	0.092	0.000	0.05	68.7	-244.5	0.134
Average	***1.891***	***-0.070***	***0.53***	***61.9***	***-195.7***	***3.495***
Overall Average	***2.982***	***-0.117***	***0.637***	***55.3***	***-145.2***	***5.808***

Overall, a CCR of 55.3% is attained at the cost of introducing performance losses of 2.982% and -0.117 dB in terms BD-rate and BD-PSNR, respectively. The results imply that, as the spatial resolution increases, more complexity reduction is accomplished, while a reduction is seen in BD-rate. These results can be observed in Table 5.48.

5.2.3.2. ***J48 decision trees classifier for CU and PU predictions.*** Tables 5.49 and 5.50 show the time savings and excessive bitrate per each QP for each of the test sequences that are acquired by applying the J48 classifier for both CU and PU predictions, respectively. As the QP value increases, it is observed that, on average, less coding bits and time are needed to encode a given video sequence.

Table 5.49: Time savings results per each QP using J48 decision trees classifier for CU and PU predictions.

Video Sequence	22		27		32		37	
	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)
RaceHorses (384×192)	41.2	-69.9	33.5	-50.3	32.8	-48.8	45.2	-82.6
BlowingBubbles (384×192)	28.9	-40.6	37.2	-59.3	52.5	-110.3	58.9	-143.5
BQSquare (384×192)	30.4	-43.6	44.3	-79.5	59.9	-149.5	66.1	-195.3
BasketballPass (384×192)	33.5	-50.4	41.6	-71.3	47.6	-90.9	59.5	-147.0
RaceHorses (832×448)	33.3	-49.9	44.7	-80.8	47.4	-90.1	50.1	-100.3
PartyScene (832×448)	32.2	-47.6	38.7	-63.2	46.8	-87.9	59.0	-143.8
BQMall (832×448)	35.7	-55.5	43.7	-77.5	50.9	-103.4	60.4	-152.5
BasketballDrill (832×448)	41.8	-71.9	48.3	-93.5	54.0	-117.3	60.7	-154.5
ParkScene (1920×1024)	40.7	-68.8	52.0	-108.4	60.3	-152.1	68.3	-215.8
Kimono1 (1920×1024)	40.3	-67.6	48.3	-93.3	55.2	-123.0	59.3	-145.8
Cactus (1920×1024)	38.2	-61.8	52.8	-111.9	63.7	-175.5	67.3	-205.5
BQTerrace (1920×1024)	36.8	-58.2	56.0	-127.4	69.9	-232.0	73.6	-278.5
BasketballDrive (1920×1024)	48.1	-92.6	47.1	-89.1	54.7	-120.7	61.1	-156.7
Traffic (2560×1600)	47.6	-90.9	59.9	-149.5	69.1	-223.9	76.2	-319.9
PeopleOnStreet (2560×1600)	33.2	-49.6	33.8	-51.1	43.0	-75.5	50.5	-102.1
NebutaFestival (2560×1600)	67.3	-205.7	51.4	-105.9	52.8	-111.8	70.7	-241.0
SteamLocomotiveTrain (2560×1600)	53.5	-115.0	62.1	-163.5	71.0	-244.9	76.3	-322.4
Average	***40.2***	***-72.9***	***46.8***	***-92.7***	***54.8***	***-132.8***	***62.5***	***-182.8***

Table 5.50 Excessive bitrate results per each QP using J48 decision trees classifier for CU and PU predictions.

Video Sequence	Excessive Bitrate (%)			
	QP = 22	QP = 27	QP = 32	QP = 37
RaceHorses (384×192)	5.47	2.40	0.55	0.55
BlowingBubbles (384×192)	0.11	0.17	0.20	0.20
BQSquare (384×192)	0.10	0.12	-0.01	-0.01
BasketballPass (384×192)	0.94	0.94	0.66	0.66
RaceHorses (832×448)	1.14	2.02	1.54	1.54
PartyScene (832×448)	0.68	0.40	0.01	0.01
BQMall (832×448)	0.68	0.92	0.70	0.70
BasketballDrill (832×448)	0.41	0.40	0.25	0.25
ParkScene (1920×1024)	0.01	0.07	-0.23	-0.23
Kimono1 (1920×1024)	0.34	0.54	0.38	0.38
Cactus (1920×1024)	-0.15	0.16	0.22	0.22
BQTerrace (1920×1024)	-0.24	-0.41	-0.29	-0.29
BasketballDrive (1920×1024)	0.03	0.30	0.27	0.27
Traffic (2560×1600)	-0.08	0.12	0.19	0.19
PeopleOnStreet (2560×1600)	1.70	1.15	0.98	0.98
NebutaFestival (2560×1600)	0.61	0.75	0.30	0.30
SteamLocomo-tiveTrain (2560×1600)	-0.22	-0.15	-0.04	-0.04
Average	***0.68***	***0.58***	***0.33***	***0.33***

Overall, a CCR of 51.1% is attained at the cost of introducing performance losses of 2.189% and -0.086 dB in terms BD-rate and BD-PSNR, respectively. The results imply that, as the spatial resolution increases, more complexity reduction is accomplished, while a reduction is seen in BD-rate. These results can be observed in Table 5.51.

Table 5.51: BD-rate, BD-PSNR, avg. excessive bitrate, and time savings results using J48 decision trees classifier for CU and PU predictions.

Video Sequence	BD-rate (%)	BD-PSNR (dB)	Exc. Bitrate (%)	CCR (%)	ΔTime (%)	BD-rate/CCR (%)
RaceHorses (384×192)	6.878	-0.316	2.08	38.2	-62.9	18.026
BlowingBubbles (384×192)	2.896	-0.115	0.15	44.4	-88.4	6.526
BQSquare (384×192)	1.463	-0.067	0.06	50.2	-117.0	2.917
BasketballPass (384×192)	2.881	-0.139	0.56	45.6	-89.9	6.322
Average	***3.530***	***-0.159***	***0.71***	***44.6***	***-89.6***	***8.448***
RaceHorses (832×448)	4.908	-0.190	1.12	43.9	-80.3	11.189
PartyScene (832×448)	1.778	-0.082	0.26	44.2	-85.6	4.023
BQMall (832×448)	2.734	-0.114	0.75	47.7	-97.2	5.737
BasketballDrill (832×448)	1.607	-0.065	0.31	51.2	-109.3	3.138
Average	***2.757***	***-0.113***	***0.61***	***46.7***	***-93.1***	***6.022***
ParkScene (1920×1024)	1.312	-0.042	-0.12	55.4	-136.3	2.370
Kimono1 (1920×1024)	1.269	-0.041	0.34	50.8	-107.4	2.500
Cactus (1920×1024)	1.771	-0.039	0.09	55.5	-138.7	3.191
BQTerrace (1920×1024)	1.185	-0.022	-0.30	59.1	-174.0	2.006
BasketballDrive (1920×1024)	1.377	-0.030	0.19	52.7	-114.8	2.611
Average	***1.383***	***-0.035***	***0.04***	***54.7***	***-134.2***	***2.536***
Traffic (2560×1600)	1.798	-0.062	0.06	63.2	-196.1	2.844
PeopleOnStreet (2560×1600)	3.246	-0.143	1.15	40.1	-69.6	8.089
NebutaFestival (2560×1600)	0.328	0.000	0.29	60.5	-166.1	0.542
SteamLocomotiveTrain (2560×1600)	-0.215	0.000	-0.27	65.7	-211.5	-0.327
Average	***1.289***	***-0.051***	***0.30***	***57.4***	***-160.8***	***2.787***
Overall Average	***2.189***	***-0.086***	***0.394***	***51.1***	***-120.3***	***4.806***

5.2.3.3. ***Random forest classifier for CU predictions and J48 decision trees classifier PU predictions.*** Tables 5.52 and 5.53 show the time savings and excessive bitrate per each QP for each of the test sequences that are acquired by predicting CUs using the random forest classifier and predicting PUs using the J48 classifier, respectively. As the QP value increases, it is observed that, less overall coding bits and time are required to encode a given video sequence. Overall, a CCR of 50.1% is attained at the cost of introducing performance losses of 2.007% and -0.079 dB in terms BD-rate and BD-PSNR, respectively. The results imply that, as the spatial resolution increases, more complexity reduction is accomplished, while reduction is seen in BD-rate. These results can be observed in Table 5.54.

Table 5.52: Time savings results per each QP using random forest classifier for CU predictions and J48 decision trees classifier PU predictions.

Video Sequence	22		27		32		37	
	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)	CCR (%)	ΔTime (%)
RaceHorses (384×192)	38.1	-61.4	36.6	-57.7	34.6	-52.9	43.9	-78.1
BlowingBubbles (384×192)	28.6	-40.2	36.8	-58.2	48.1	-92.8	49.1	-96.2
BQSquare (384×192)	40.9	-69.1	44.9	-81.7	53.6	-115.3	63.4	-172.9
BasketballPass (384×192)	33.3	-49.8	39.6	-65.6	47.1	-89.1	57.4	-134.6
RaceHorses (832×448)	28.8	-40.4	42.1	-72.7	45.9	-84.9	48.8	-95.2
PartyScene (832×448)	30.6	-44.0	39.5	-65.3	46.3	-86.3	56.7	-131.1
BQMall (832×448)	35.7	-55.5	41.7	-71.4	50.0	-99.8	56.8	-131.4
BasketballDrill (832×448)	39.1	-64.2	46.6	-87.2	53.8	-116.4	61.2	-157.5
ParkScene (1920×1024)	38.6	-62.8	49.5	-98.1	58.8	-142.9	67.9	-211.7
Kimono1 (1920×1024)	50.4	-101.5	50.8	-103.2	53.7	-115.8	61.6	-160.7
Cactus (1920×1024)	37.4	-59.8	51.8	-107.5	59.7	-147.9	66.9	-202.1
BQTerrace (1920×1024)	31.2	-45.3	58.1	-138.7	70.0	-233.1	72.9	-269.4
BasketballDrive (1920×1024)	47.7	-91.3	45.4	-83.1	52.3	-109.6	58.8	-142.7
Traffic (2560×1600)	47.2	-89.5	57.6	-135.8	65.9	-193.1	73.5	-276.6
PeopleOnStreet (2560×1600)	30.5	-43.8	35.3	-54.6	42.9	-75.0	50.4	-101.6
NebutaFestival (2560×1600)	64.0	-178.1	55.0	-122.1	55.0	-122.1	70.2	-236.0
SteamLocomotiveTrain (2560×1600)	54.6	-120.5	60.1	-150.6	66.3	-196.6	73.0	-270.6
Average	***39.8***	***-71.6***	***46.6***	***-91.4***	***53.2***	***-122.0***	***60.7***	***-168.7***

Table 5.53: Excessive bitrate results per each QP using random forest classifier for CU predictions and J48 decision trees classifier PU predictions.

Video Sequence	Excessive Bitrate (%)			
	QP = 22	QP = 27	QP = 32	QP = 37
RaceHorses (384×192)	5.34	2.52	0.44	-0.13
BlowingBubbles (384×192)	0.24	0.01	-0.35	-0.23
BQSquare (384×192)	0.04	-0.18	-0.36	-0.02
BasketballPass (384×192)	1.16	0.88	0.28	0.47
RaceHorses (832×448)	0.93	1.72	1.08	-0.24
PartyScene (832×448)	0.58	0.22	-0.11	-0.42
BQMall (832×448)	0.65	0.79	0.66	0.30
BasketballDrill (832×448)	0.28	0.33	0.36	-0.04
ParkScene (1920×1024)	-0.11	-0.13	-0.59	-0.62
Kimono1 (1920×1024)	0.47	0.63	-0.01	-0.13
Cactus (1920×1024)	-0.29	0.02	0.16	-0.22
BQTerrace (1920×1024)	-0.45	-0.89	-0.81	-0.79
BasketballDrive (1920×1024)	-0.08	0.19	0.11	0.10
Traffic (2560×1600)	-0.37	-0.01	-0.17	-0.50
PeopleOnStreet (2560×1600)	1.38	0.99	1.06	0.64
NebutaFestival (2560×1600)	0.21	0.60	0.19	-0.54
SteamLocomo-tiveTrain (2560×1600)	-0.05	-0.11	-0.12	-0.51
Average	***0.58***	***0.45***	***0.11***	***-0.17***

Table 5.54: BD-rate, BD-PSNR, avg. excessive bitrate, and time savings results using random forest classifier for CU predictions and J48 decision trees classifier PU predictions.

Video Sequence	BD-rate (%)	BD-PSNR (dB)	Exc. Bitrate (%)	CCR (%)	ΔTime (%)	BD-rate/CCR (%)
RaceHorses (384×192)	6.217	-0.289	2.04	38.3	-62.5	16.242
BlowingBubbles (384×192)	2.263	-0.090	-0.08	40.7	-71.9	5.566
BQSquare (384×192)	1.256	-0.058	-0.13	50.7	-109.7	2.477
BasketballPass (384×192)	2.620	-0.125	0.70	44.3	-84.8	5.910
Average	***3.089***	***-0.141***	***0.63***	***43.5***	***-82.2***	***7.549***
RaceHorses (832×448)	4.000	-0.156	0.87	41.4	-73.3	9.665
PartyScene (832×448)	1.729	-0.080	0.07	43.3	-81.7	3.995
BQMall (832×448)	2.679	-0.111	0.60	46.0	-89.5	5.821
BasketballDrill (832×448)	1.597	-0.065	0.23	50.2	-106.3	3.184
Average	***2.501***	***-0.103***	***0.44***	***45.2***	***-87.7***	***5.666***
ParkScene (1920×1024)	1.550	-0.050	-0.36	53.7	-128.9	2.886
Kimono1 (1920×1024)	1.258	-0.042	0.24	54.1	-120.3	2.324
Cactus (1920×1024)	1.690	-0.037	-0.08	54.0	-129.3	3.133
BQTerrace (1920×1024)	1.168	-0.022	-0.74	58.0	-171.6	2.012
BasketballDrive (1920×1024)	0.988	-0.021	0.08	51.0	-106.7	1.935
Average	***1.331***	***-0.034***	***-0.17***	***54.2***	***-131.4***	***2.458***
Traffic (2560×1600)	1.741	-0.059	-0.26	61.0	-173.7	2.853
PeopleOnStreet (2560×1600)	3.338	-0.147	1.02	39.8	-68.8	8.395
NebutaFestival (2560×1600)	0.200	0.000	0.12	61.1	-164.6	0.327
SteamLocomotiveTrain (2560×1600)	-0.170	0.000	-0.20	63.5	-184.6	-0.267
Average	***1.277***	***-0.052***	***0.17***	***56.3***	***-147.9***	***2.827***
Overall Average	***2.007***	***-0.079***	***0.242***	***50.1***	***-113.4***	***4.498***

5.3. Performance Evaluation

The coding efficiency of both the individual and joint algorithms are analysed. The proposed solutions involve algorithms applied to perform early CU size and PU mode decisions. For RA configuration profile, the compression efficiency of each algorithm is evaluated in terms of the complexity reduction attained at the cost of some BD-rate and BD-PSNR losses. These are the most important metric to be considered as they directly affect the quality of the encoded video sequence. BD-rate to CCR ratio is also considered as it provides a good indication as to the improvement introduced by utilizing the proposed solution. The smaller this ratio is, the more enhancement is likely seen in the HEVC encoder.

5.3.1. Analysis of the proposed algorithms. Both Figure 5.1 and Table 5.55 present a comparison between all the proposed solutions in terms of the complexity reduction accomplished and the corresponding compression efficiency degradation. The table illustrates that the joint schemes combining both CU and PU early termination approaches yield the largest CCR, ranging between 50.1% to 55.3%. The justification

to such large reductions can be due to trying to limit the RDO process on both CUs and PUs, two main contributors to the massive computational complexity seen during HEVC encoding. However, this reduction negatively impacted the RD results, leading to BD-rate values ranging between 2% to 2.9% and BD-PSNR losses ranging between 0.079 dB and 0.117 dB. The algorithm that utilizes random forests for CU predictions and J48 for PU predictions is observed to outperform the other joint approaches with a BD-rate to CCR ratio of 4.5%.

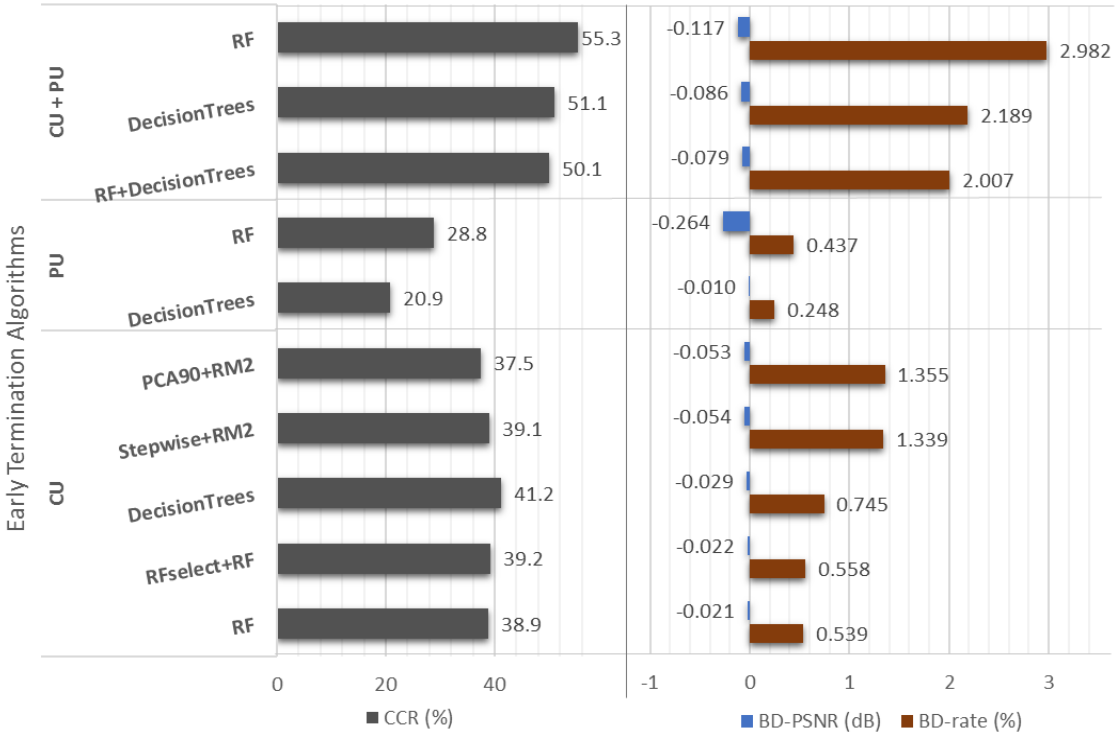

Figure 5.1: A comparison between all the proposed solutions.

Table 5.55: A comparison between all the proposed solutions in terms of the CCR and compression efficiency degradation.

Proposed early termination algorithms		BD-rate (%)	BD-PSNR (dB)	CCR (%)	BD-rate/CCR (%)
CU	***RF***	0.539	-0.021	38.9	1.613
	RFselect+RF	0.558	-0.022	39.2	1.663
	DecisionTrees	0.745	-0.029	41.2	2.196
	Stepwise+RM2	1.339	-0.054	39.1	3.922
	PCA90+RM2	1.355	-0.053	37.5	4.253
PU	***DecisionTrees***	0.248	-0.010	20.9	1.422
	RF	0.437	-0.264	28.8	1.573
CU + PU	***RF+DecisionTrees***	2.007	-0.079	50.1	4.498
	DecisionTrees	2.189	-0.086	51.1	4.806
	RF	2.982	-0.117	55.3	5.808

As seen in the results presented in Section 5.2, high resolution videos usually displayed huge complexity reductions. This can be attributed to the structural characteristics of the video sequence, which usually contains large homogeneous areas, resulting in having large CUs. Consequently, the splitting of the CU is discouraged, which enhances the time in comparison to running the unmodified HM software.

On the other hand, the smallest CCR is seen when separately considering the PU early termination algorithms, which led to CCR ranging between 20.9% to 28.8%. The improvement these algorithms display depends on the motion characteristics of the video, which might explain the reason behind the limited gain. Video sequences containing slow-motion scenes allows the selection of MSM/Skip or 2Nx2N inter-prediction modes. This limits the RDO process from checking other PU modes that are likely to rarely take place. In return, this approach resulted in very small performance degradations of 0.248% to 0.437% in terms of BD-rate and -0.010 dB to -0.264 dB in terms of BD-PSNR. *RaceHorses (384×192)* mostly reflecting the worst RD efficiency can be attributed not only to its small spatial resolution, which was further reduced during the pre-processing of the video sequence, but also to the nature of motion seen in the video. The algorithm that utilizes J48 for PU predictions is observed to outperform the other approach with a BD-rate to CCR ratio of 1.4%. A CCR of 38.9% is seen when using random forests for CU predictions with a BD-rate increase of 0.539% and BD-PSNR reduction of -0.021 dB.

Figures 5.2 and 5.3 show the RD efficiency of the three proposed schemes for two video sequences encoded with different bitrates. The two video sequences were chosen based on their RD efficiency results, where *RaceHorses* displayed the one of the highest RD losses in comparison to other sequences and insignificant RD losses are seen in *Traffic*. The curves represent the results for the unmodified reference encoder (HM 13.0), and the best performing algorithm from each of the individual and joint schemes. It is notified that for both video sequences, the proposed schemes present very good results as their curves overlap with that of the original encoder. Nonetheless, when applying the CU+PU scheme to the *RaceHorses* video sequence, it is observed that the RD efficiency is not as good as that achieved by using the original encoder.

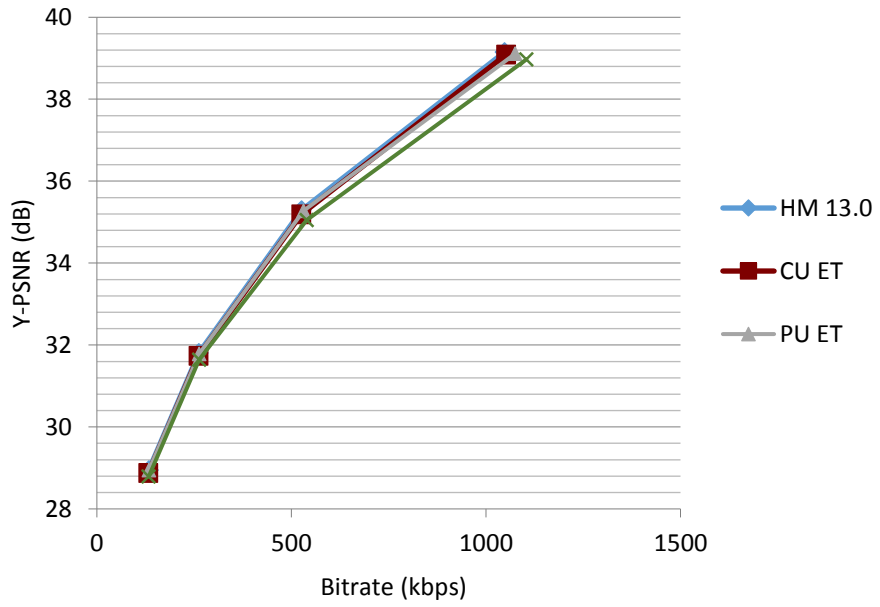

Figure 5.2: RD efficiency for *RaceHorses (384×192)* video sequence encoded with the unmodified HM 13.0 software and three early termination schemes.

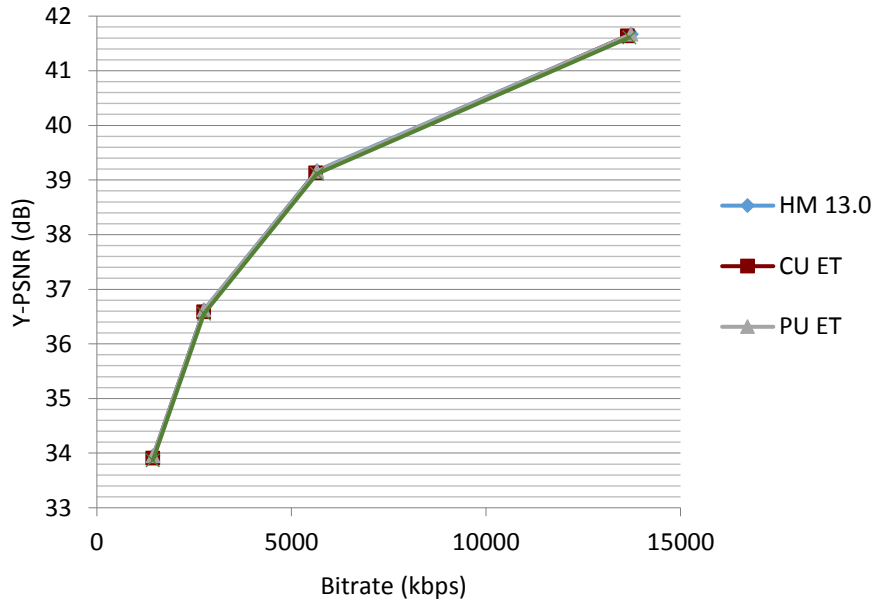

Figure 5.3: RD efficiency for *Traffic (2560×1600)* video sequence encoded with the unmodified HM 13.0 software and three early termination schemes.

5.3.2. Comparison with existing work. Various related work in this field are analysed and compared to the schemes proposed in this thesis in terms of BD-rate and complexity reduction values. To allow a fair comparison, only those solutions that used QPs 22, 27, 32, and 37, RA temporal configuration, and were tested on at least eight video sequences belonging to at least four different spatial resolutions were considered.

The BD-rate to complexity reduction ratio combines the evaluation metrics, introducing an appropriate measure that is used to compare the proposed solutions with that of other work. Thus, the analysis is done in terms of the RD efficiency losses encountered when a certain amount of complexity reduction is attained. The smaller this value is, the more computational complexity saved, the less RD efficiency loss seen, or both.

Table 5.56: Comparison with related work that use (25) to compute complexity reduction.

Category	Related Work	BD-rate (%)	CCR (%)	BD-rate/CCR (%)
CU early termination	*[31]*	2.000	58.4	3.425
	[36]	1.400	49.6	2.823
	[38]	1.430	62.0	2.306
	[46]	0.280	37.0	0.757
	Proposed	0.539	38.9	1.386
PU early termination	*[46]*	0.560	50.0	1.120
	Proposed	0.248	20.9	1.187
CU+PU early termination	*[42]*	0.992	59.8	1.660
	[43]	1.290	65.1	1.981
	[46]	1.330	63.0	2.111
	Proposed	2.007	50.1	4.006

A number of related work considered used (25) to compute the CCR. The results are illustrated in Table 5.56. These results are categorized based on the partitioning structure it targets. The best performing proposed solution in each category is utilized for the comparison done in this section. When it comes to the CU early termination category, the best performing related work solution [46] produced a BD-rate/CCR ratio of 0.757, which is smaller than the one produced by the proposed scheme. Nevertheless, it is important to take note that the CCR achieved by the proposed scheme is slightly higher, which resulted in the increase of the performance losses. Additionally, only 8 sequences were used in common. On the other hand, the proposed work evidently outperforms the other solutions presented in this category. The proposed PU early termination scheme produces a BD-rate/CCR ratio that is very similar to that seen in [46]. The seemingly small complexity reduction seen when using the proposed algorithm is at the cost of reducing performance losses. Unfortunately, despite the high computational complexity introduced when using the proposed solution that combines both the CU and PU approaches in comparison to the other proposed algorithms, the BD-rate to CCR ratio is significantly higher than that illustrated by related work.

Table 5.57: Comparison with related work that use (26) to compute complexity reduction.

Related Work	BD-rate (%)	ΔTime (%)	BD-rate/ΔTime (%)
[30]	0.820	-37.4	2.193
[39]	0.710	-53.6	1.325
[41]	0.700	-46.7	1.499
[44]	1.540	-45.1	3.415
Proposed	0.539	-70.7	0.762

Other related work in this field used (26) to compute the reduction in time complexity and a CU early termination approach. The comparison is presented in Table 5.57. It is evident that the proposed CU early termination solution outperforms all the solutions presented in the table by a great margin. It is important to note that, unlike the first set of related works presented in this section, these solutions use all the video sequences used in this work.

Chapter 6. Conclusion and Future Work

In comparison to its predecessors, HEVC introduced a significant improvement in terms of the coding efficiency, but at the cost of increasing the computational complexity. The proposed schemes in this thesis aim to counter this computational complexity by proposing a fast partitioning decision algorithm for CUs and a fast mode selection for PUs using dimensionality reduction and classification techniques. A video sequence-dependent approach was considered, where the CU split decision was treated as a 2-class problem. Different evaluation metrics were considered to evaluate the schemes, including the classification accuracy, BD-rate, BD-PSNR and CCR. Experimental results illustrated that a CCR of 38.5% at the negligible cost of 0.539% and -0.021 dB in terms of BD-rate and BD-PSNR, respectively, was achieved for the best performing CU early termination scheme proposed. The best performing PU early termination scheme proposed attained an overall CCR of 20.9% at the cost of a BD-rate of 0.248% and a BD-PSNR of -0.01 dB. When jointly implemented, an overall BD-rate increase of 2.007% and BD-PSNR decrease of 0.079 dB was observed, leading to a CCR of 50.1%. In comparison to existing work, it was shown that the proposed solutions are superior in terms of coding efficiency as evident in the DB-rate and DB-PSNR results. The proposed solutions are also superior in terms of time savings when it comes to applying the generated models to predict the CU split and PU coding modes.

In future work, the proposed algorithms can be tested in a video sequence-independent setting. In this case, a comprehensive set of video sequences can be used for offline training and model generation. The generated models can then be used to optimize the encoding of any other video sequence. Likewise, for video-dependent training models, a future work direction can focus on repeating the training whenever a scene cut is deducted. The complexity of such solutions should be analysed carefully as the retraining time might be significant. Additionally, this thesis only focused on testing the proposed schemes using RA profile configuration. As future work, more test conditions can be considered, including Low Delay with P-frames and Low Delay with B-frames configurations. Lastly, a fast decision algorithm can be implemented for TU structures and combined with the individual algorithms proposed in this work.

References

[1] G. Sullivan, J. Ohm, W. Han and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649-1668, Sep. 2012.

[2] J. R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan and T. Wiegand, "Comparison of the Coding Efficiency of Video Coding Standards—Including High Efficiency Video Coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1669-1684, Dec. 2012.

[3] J. Vanne, M. Viitanen, T. D. Hamalainen, and A. Hallapuro, "Comparative Rate-distortion-complexity Analysis of HEVC and AVC Video Codecs," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1885-1898, Dec. 2012.

[4] G. J. Sullivan and T. Wiegand, "Rate-distortion Optimization for Video Compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74-90, Nov. 1998.

[5] M. Wien, *High Efficiency Video Coding*, 1st ed. Heidelberg: Springer, 2015.

[6] T. Dutoit and F. Marques, *Applied Signal Processing*, 1st ed. New York: Springer, 2009.

[7] J. McVeigh, G. K. Chen, J. Goldstein, A. Gupta, M. Keith and S. Wood, "A Software-based Real-time MPEG-2 Video Encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 7, pp. 1178-1184, Oct. 2000.

[8] V. Sze, M. Budagavi and G. Sullivan, *High Efficiency Video Coding (HEVC):Algorithms and Architectures*, 1st ed. Heidelberg: Springer, 2014.

[9] E. Alpaydın, *Introduction to Machine Learning*, 3rd ed. Cambridge, MA: MIT Press, 2014.

[10] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning. Cambridge*, MA: MIT Press, 2012.

[11] O. Z. Maimon and L. Rokach, *Data Mining and Knowledge Discovery Handbook*, 1st ed. New York: Springer, 2010.

[12] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, no. 1, pp. 81-106, 1986.

[13] J. R. Quinlan, *C4.5*, 1st ed. San Mateo, Calif.: Morgan Kaufmann Publishers, 1993.

[14] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. CRC press, 1984.

[15] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Waltham, MA: Morgan Kaufmann, 2012.

[16] J. R. Quinlan, "Improved Use of Continuous Attributes in C4.5," *Journal of Artificial Intelligence Research*, vol. 4, no. 1, pp. 77-90, Jan. 1996.

[17] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. Amsterdam: Elsevier, 2011.

[18] R. E. Schapire and Y. Freund, *Boosting: Foundations and Algorithms*. Cambridge, MA: MIT Press, 2014.

[19] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, Jan. 2001.

[20] L. Breiman, "Bagging Predictors," *Machine Learning*, vol. 24, no. 2, pp. 123-140, Aug. 1996.

[21] C.M. Bishop, *Neural Networks for Pattern Recognition*. New York: Oxford Univ. Press Inc., 1995.

[22] K. Toh, Q. Tran and D. Srinivasan, "Benchmarking a Reduced Multivariate Polynomial Pattern Classifier," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 740-755, Apr. 2004.

[23] W. Mendenhall and T. Sincich, *Statistics for Engineering and the Sciences*, 1st ed. Upper Saddle River, N.J.: Pearson Prentice-Hall, 2007.

[24] G. Bjøntegaard, Calculation of Average PSNR Differences Between RD-Curves, Technical Report VCEG-M33, ITU-T SG16/Q6, Austin, Texas, USA, 2001.

[25] Zupancic, S. G. Blasi, E. Peixoto and E. Izquierdo, "Inter-Prediction Optimizations for Video Coding Using Adaptive Coding Unit Visiting Order," *IEEE Transactions on Multimedia*, vol. 18, no. 9, pp. 1677-1690, Sep. 2016.

[26] X. Deng, M. Xu, L. Jiang, X. Sun and Z. Wang, "Subjective-Driven Complexity Control Approach for HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 91-106, Jan. 2016.

[27] A. Jiménez-Moreno, E. Martínez-Enríquez and F. Díaz-de-María, "Complexity Control Based on a Fast Coding Unit Decision Method in the HEVC Video Coding Standard," *IEEE Transactions on Multimedia*, vol. 18, no. 4, pp. 563-575, Apr. 2016.

[28] W. Zhao, T. Onoye and T. Song, "Hierarchical Structure-Based Fast Mode Decision for H.265/HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 10, pp. 1651-1664, Oct. 2015.

[29] H. Yoo and J. Suh, "Fast Coding Unit Decision Based on Skipping of Inter and Intra Prediction Units," *Electronics Letters*, vol. 50, no. 10, pp. 750-752, Aug. 2014.

[30] J. Xiong, H. Li, F. Meng, Q. Wu and K. N. Ngan, "Fast HEVC Inter CU Decision Based on Latent SAD Estimation," *IEEE Transactions on Multimedia*, vol. 17, no. 12, pp. 2147-2159, Dec. 2015.

[31] S. Park, "CU Encoding Depth Prediction, Early CU Splitting Termination and Fast Mode Decision for Fast HEVC Intra-coding," *Signal Processing: Image Communication*, vol. 42, pp. 79-89, Mar. 2016.

[32] M. Radosavljević, G. Georgakarakos, S. Lafond and D. Vukobratović, "Fast Coding Unit Selection Based on Local Texture Characteristics for HEVC Intra

Frame," in *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 1377-1381, Orlando, FL, Dec. 2015.

[33] K. Goswami, J. Lee and B. Kim, "Fast Algorithm for the High Efficiency Video Coding (HEVC) Encoder Using Texture Analysis," Information Sciences, vol. 364-365, pp. 72-90, Oct. 2016.

[34] L. Shen, Z. Zhang and Z. Liu, "Effective CU Size Decision for HEVC Intracoding," *IEEE Transactions on Image Processing*, vol. 23, no. 10, pp. 4232-4241, Oct. 2014.

[35] S. Ahn, B. Lee and M. Kim, "A Novel Fast CU Encoding Scheme Based on Spatiotemporal Encoding Parameters for HEVC Inter Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 3, pp. 422-435, Mar. 2015.

[36] K. Lim, J. Lee, S. Kim and S. Lee, "Fast PU Skip and Split Termination Algorithm for HEVC Intra Prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 8, pp. 1335-1346, Aug. 2015.

[37] J. Lee, S. Kim, K. Lim and S. Lee, "A Fast CU Size Decision Algorithm for HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 3, pp. 411-421, Mar. 2015.

[38] H. Kim and R. Park, "Fast CU Partitioning Algorithm for HEVC Using an Online-Learning-Based Bayesian Decision Rule," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 130-138, Jan. 2016.

[39] L. Shen, Z. Zhang, X. Zhang, P. An and Z. Liu, "Fast TU Size Decision Algorithm for HEVC Encoders Using Bayesian Theorem Detection," *Signal Processing: Image Communication*, vol. 32, pp. 121-128, Mar. 2015.

[40] S. Cho and M. Kim, "Fast CU Splitting and Pruning for Suboptimal CU Partitioning in HEVC Intra Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 9, pp. 1555-1564, Sep. 2013.

[41] Z. Liu, T. Lin and C. Chou, "Efficient Prediction of CU Depth and PU Mode for Fast HEVC Encoding Using Statistical Analysis," *Journal of Visual Communication and Image Representation*, vol. 38, pp. 474-486, Jul. 2016.

[42] Q. Hu, X. Zhang, Z. Shi and Z. Gao, "Neyman-Pearson-Based Early Mode Decision for HEVC Encoding," *IEEE Transactions on Multimedia*, vol. 18, no. 3, pp. 379-391, Mar. 2016.

[43] J. Xiong, H. Li, Q. Wu and F. Meng, "A Fast HEVC Inter CU Selection Method Based on Pyramid Motion Divergence," *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 559-564, Feb. 2014.

[44] Y. Zhang, S. Kwong, X. Wang, H. Yuan, Z. Pan and L. Xu, "Machine Learning-Based Coding Unit Depth Decisions for Flexible Complexity Allocation in High Efficiency Video Coding," *IEEE Transactions on Image Processing*, vol. 24, no. 7, pp. 2225-2238, Jul. 2015.

[45] G. Correa, P. A. Assuncao, L. V. Agostini and L. A. da Silva Cruz, “Fast HEVC Encoding Decisions Using Data Mining,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 4, pp. 660-673, Apr. 2015.

[46] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA Data Mining Software: An Update,” *ACM SIGKDD Explorations Newslett.*, vol. 11, no. 1, pp. 10-18, Jun. 2009.

[47] K. McCann, B. Bross, W. Han, I. Kim, K. Sugimoto, G. Sullivan, High Efficiency Video Coding (HEVC) Test Model 13 (HM 13) Encoder Description, document JCTVC-O1002, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC-JCT1/SC29/WG11, Geneva, Switzerland, Oct. 2013.

[48] Common Test Conditions and Software Reference Configurations, document JCTVC-J110, ISO/IEC-JCT1/SC29/WG11, Stockholm, Sweden, 2012.

[49] MATLAB version 8.5.0.197613 (R2015a), The MathWorks Inc., Natick, Massachusetts, USA, 2015.

Vita

Mahitab Hassan was born in 1994, in Alexandria, Egypt. She received her primary and secondary education in Sharjah, UAE. She received her B.Sc. degree in Computer Engineering and completed the requirements for a Computer Science minor from the American University of Sharjah in 2015.

In September 2015, she joined the Computer Engineering master's program in the American University of Sharjah as a graduate teaching assistant. During her master's studies, she co-authored 2 papers, which were presented in local and international conferences. Her research interests include machine learning, digital video processing, and ubiquitous learning systems.