Fast Intra-frame Coding Algorithm for HEVC Based on TCM and Machine Learning

by

Yi Shan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2016

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

High Efficiency Video Coding (HEVC) is the latest video coding standard. Compared with the previous standard H.264/AVC, it can reduce the bit-rate by around 50% while maintaining the same perceptual quality. This performance gain on compression is achieved mainly by supporting larger Coding Unit (CU) size and more prediction modes. However, since the encoder needs to traverse all possible choices to mine out the best way of encoding data, this large flexibility on block size and prediction modes has caused a tremendous increase in encoding time. In HEVC, intra-frame coding is an important basis, and it is widely used in all configurations. Therefore, fast algorithms are always required to alleviate the computational complexity of HEVC intra-frame coding.

In this thesis, a fast intra coding algorithm based on machine learning is proposed to predict CU decisions. Hence the computational complexity can be significantly reduced with negligible loss in the coding efficiency. Machine learning models like Bayes decision, Support Vector Machine (SVM) are used as decision makers while the Laplacian Transparent Composite Model (LPTCM) is selected as a feature extraction tool. In the main version of the proposed algorithm, a set of features named with Summation of Binarized Outlier Coefficients (SBOC) is extracted to train SVM models. An online training structure and a performance control method are introduced to enhance the robustness of decision makers.

When applied on All Intra Main (AIM) full test and compared with HM 16.3, the main version of the proposed algorithm can achieve, on average, 48% time reduction with 0.78% BD-rate increase. Through adjusting parameter settings, the algorithm can change the trade-off between encoding time and coding efficiency, which can generate a performance curve to meet different requirements. The maximum average time reduction in the test can be 51.37% with 1.07% BD-rate increase while the minimum BD-rate increase could be 0.14% with 29.88% time reduction. By testing different methods on the same machine, the performance of proposed method has outperformed all CU decision based HEVC fast intra algorithm in the benchmarks.

In addition, experiments and explorations are carried out to provide useful insights for fast intra algorithm design, like the composition of encoding time and the experimental lower bound of time reduction. To obtain the lower bound for time reduction, we have output the optimal CU decisions in a file by encoding a sequence by one pass. When encoding the sequence again, the encoder can utilize that file to decide the best CU partition without any trial. Through this two-pass test, the encoding time can be reduced by around 70%, which shows the time reduction potential of CU decision based fast algorithms.

Acknowledgements

First of all, I need to express my prior gratitude to my supervisor, Prof. En-hui Yang, for his insightful guidance and enduring support every step of the way. It is him who inspires me with new ideas and encourages me when I am thinking of giving up. I have learned a lot from his spirit of research and knowledge of the field.

I would like to thank other professors who teach my courses: Mohamed Kamel, Weihua Zhuang, Amir Keyvan Khandani. They have given excellent lectures, and provided me with extensive knowledge, which forms the foundation of my current research.

I would like to thank Prof. Zhou Wang and Prof. Alfred C.H. Yu for being readers of my thesis and providing valuable comments and suggestions.

I would also like to thank my labmates in the multimedia communications lab for their helps and companionship: Jingyun Bian, Nan Hu, Jeffrey Erbrecht, Jiasen Xu, Hossam Amer, Jin Meng, and Xiang Yu. I would like to thank visiting scholars Xiangwen Wang and Jing He. I have obtained a lot of new knowledge and useful advises through discussions with them.

Last but not least, I would like to thank my family and friends for their endless support and encouragement.

Table of Contents

List of Tables

List of Figures

List of Frequent Abbreviations

AC	Alternating Current.
AIM	All Intra Main.
BD-rate	Bjntegaard-Delta Bit-Rate.
CTU	Coding Tree Unit.
CU	Coding Unit.
CDM	CU Decision Map.
DCT	Discrete Cosine Transform.
DST	Discrete Sine Transform.
DC	Direct Current.
DM	Decision Maker.
DCC	Data Compression Conference.
FP	False Positive.
FN	False Negative.
HEVC	High Efficiency Video Coding.

HM	HEVC Test Model.
HD	High Definition.
ICIP	International Conference on Image Processing.
IPM	Intra Prediction Mode.
JCT-VC	Joint Collaborative Team on Video Coding.
LPTCM	Laplacian Transparent Composite Model.
OBF	Outlier Block Flag.
PU	Prediction Unit.
PDF	Probability Density Function.
PSNR	Peak Signal-to-noise Ratio.
QP	Quantization Parameter.
RMD	Rough Mode Decision.
RD	Rate Distortion.
RDO	Rate Distortion Optimization.
SBOC	Summation of Binarized Outlier Coefficients.
SVM	Support Vector Machine.

SATD	Summation of Hadamard Transformed Coefficients.
TU	Transform Unit.
TCSVT	IEEE Transaction on Circuits and Systems for Video Technology.
TP	True Positive.
TN	True Negative.
MPM	Most Probable Mode.
MPEG	ISO/IEC Moving Picture Expert Group.
VCEG	ITU-T Video Coding Experts Group.
WVGA	Wide Video Graphics Array.
WQVGA	Wide Quarter Video Graphics Array.

Chapter 1

Introduction

1.1 Motivation

High Efficiency Video Coding (HEVC) is the latest video coding standard. It is finalized by Joint Collaborative Team on Video Coding (JCT-VC) in January 2013 [33]. Compared with the previous video standard H.264/AVC, HEVC can reduce the bit-rate by around 50% while maintaining the same perceptual quality. This performance gain on coding efficiency is mainly achieved by supporting larger block sizes and more prediction modes. However, since the encoder needs to traverse all possible choices to mine out the best way to encode the data, this large flexibility on block size and prediction modes causes a tremendous increase in the encoding time and energy consumption of HEVC encoders. Based on our experiment, the encoding time of HEVC intra-frame coding has increased by around 5 times, which is very significant.

The trade-off in data compression is not only made between rate and distortion, but also with the complexity. Sacrificing too much in computational cost may make HEVC less practical for some applications like energy-limited system and real-time cases. Therefore, how to reduce the encoding complexity of HEVC becomes a crucial problem. Currently, a fast HEVC encoder x.265 based on parallel computing and highly simplification can achieve real time encoding [1]. While compared with the HEVC Test Model (HM), the x.265 has sacrificed the coding efficiency by around 25% in average, which is too much. Hence, there is an urgent need for fast HEVC encoding algorithms that can achieve more time reduction while maintain the coding efficiency. For real-time applications, better coding efficiency is desired when real-time encoding can be achieved. For non-real-time applications, faster

coding algorithm with the same level of coding efficiency is also needed to save energy and computational resources.

In our research, the fast coding algorithm is only focused on intra-frame coding. Intra-frame coding is an important basis of video coding, which is widely used in all configurations of HEVC. In HEVC intra coding, a frame is processed as squared blocks and predicted by numbers of prediction modes. This is the main reason for the high complexity of HEVC. The largest block size in HEVC is 64×64 while the smallest block size is 4×4 [34]. Unlike the 16×16 largest block in H.264/AVC, this promoted flexibility allows HEVC to have a better adaptation to different video content. In addition, each block for prediction can support 35 Intra Prediction Mode (IPM). While, how to select the best Coding Unit (CU) size and IPM is a problem for all HEVC encoders. In the HM, the best choices of CU size and IPM are selected through full Rate Distortion Optimization (RDO), where the encoder has to apply a full encoding process for all possible combinations of partition and IPM. Therefore, to reduce the complexity, the strategy is to predict those decisions by using less computationally expensive methods. These decisions based fast algorithm can be further combined with parallel computing and application-specific hardware to fit in real applications.

There are already many fast algorithms for HEVC intra-frame coding in the current literature. However, most of them are traditional algorithms that are fixed designs through experimental observation. Therefore, many improvements could be made for those algorithms. On the one hand, video data can have huge variety. Those traditional algorithms may be effective for test sequences, but there is no guarantee that they can work well in real applications. To achieve better performance and adaptivity, one solution is to use machine learning in fast intra-frame coding. However, different machine learning models may have different characteristics. A problem in the design of machine learning based fast coding algorithms could be how to select models. Moreover, variety of features can be extracted from the video data to train machine learning models. The feature extraction methods are crucial since they may directly decide the bound of performance. Besides, some meta-algorithms are also needed to ensure robustness and boost performance. Therefore, for machine learning based fast algorithms, lots of innovations can be made in feature extraction methods, model training, and framework design.

1.2 Contributions

In this thesis, we have proposed a fast algorithm to reduce the computational complexity of HEVC intra-frame coding and maintain the coding efficiency. The proposed algorithm

is based on a machine learning framework. The Laplacian Transparent Composite Model (LPTCM) is utilized as a feature extraction tool. Machine learning models like Bayes Decision, Support Vector Machine (SVM) are used as decision makers. When applied on All Intra Main (AIM) full test, the algorithm can achieve, on average, 48% time reduction with 0.78% BD-rate increase. Also, through adjusting the setting, the algorithm can change the trade-off between encoding time and BD-rate, which can generate a performance curve for different requirements. The maximum time reduction can be 51.37% with 1.07% BD-rate increase while the minimum BD-rate increase could be 0.14% with 29.88% time reduction.

Since the proposed algorithm is implemented on the base of HM, the encoder currently can't achieve real-time coding. However, intra coding is widely used in all configurations and how to select the CU size is a problem for all HEVC implementations. Therefore, a fast and precise CU size decision method can benefit any HEVC encoder. For those real-time implementation that based on parallel computing and application-specific hardware, if assisted by our algorithm, they are expected to achieve better performance.

To compare the performance with the state of the art, we have chosen wide range of recent works as our benchmarks. Because the coding time highly depends on the machine, to make a fair comparison, we have regenerated the result of some top papers on the same machine with our test. The results show that our algorithm not only can adjust the trade-off but also outperforms the state of the art in both time reduction and BD-rate. What's more, in the hope of achieving good performance not only for the test sequences but also in real application, the proposed algorithm chooses to use on-line learning with the performance control that have made the decision model more robust.

Besides the performance, we have summarized a general model of designing fast video coding algorithm which can be followed by video coding of the similar framework or even the next generation of video standard.

In addition, some experiments and explorations have been carried out to provide useful insights for fast intra algorithm design, like the compositions of encoding time and the experimental lower bound of time reduction. To obtain the lower bound for time reduction, we have output the optimal CU decisions in a file by encoding a sequence by one pass. When encoding the sequence again, the encoder can utilize that file to decide the best CU partition without any trials. Through this two-pass test, the encoding time has been reduced by around 70%, which has clearly shown the potential of time reduction.

1.3 Thesis Organization

Our thesis is organized as follow:

Chapter 2 are some background knowledges. Firstly, we will give a brief overview of HEVC standard. Then some important techniques in intra-frame coding that are highly relevant to our fast algorithm will be introduced. Those techniques are novel quad-tree structure, spatial sample prediction, and the transformation in HEVC. After that, it comes to the state of the art that will include the adopted rough mode decision methods and many latest innovations.

In chapter 3, the proposed machine learning based fast HEVC intra-frame algorithm will be introduced in detail, which includes the logic structure of the proposed algorithm, how the performance is measured, how the features are extracted, and how the models are trained. Furthermore, some meta-algorithms like flexible train-test period and model switch are proposed.

In chapter 4, experimental result will be demonstrated. A professional test will be done to test the effectiveness of our algorithm. To make fair comparison, we have regenerated the results of some top papers on the same computer. By adjusting the parameters, a performance curve with different trade-off points will be generated to compare our algorithm with wide range of benchmarks. In addition, some valuable experimental results was given as insights for our design.

At last, in chapter 5, we will make a conclusion and further discuss some potential application as well as future works.

Chapter 2

Background

2.1 Overview of HEVC

HEVC is the latest video coding standard. Its first edition is finalized in January 2013 by JCT-VC, which is a partnership of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Expert Group (MPEG) [33]. Two main requirements has been emphasised in this standard, one is to attain large coding efficiency enhancement in contrast to its predecessor, H.264/AVC. Another requirement is to achieve relatively low complexity to enable high-resolution and high-quality video applications in different scenarios, including wireless transmission and mobile devices [6].

Similar with previous video coding standards, the HEVC intra coding follows the famous hybrid coding architecture, which bases on spatial sample prediction, transform coding, and some post-processing. As a fixed framework, frames are normally partitioned into image blocks before being processed and encoded. But unlike the 16×16 macroblock in H.264/AVC, HEVC has utilized a novel quad-tree structure to enable far more variable block sizes to handle more complex video content and higher resolution data. Since the largest coding unit increase to 64×64, flexible transform size (from 4×4 to 32×32) has also been adopted in HEVC to allow strong energy compaction and smaller quantization errors. In terms of intra prediction, where image blocks are firstly extrapolated by restructured pixels from their spatial neighbors, H.264/AVC only support 9 ways for extrapolation. Whereas, to effectively capture more directional patterns in image texture, HEVC have introduced 35 prediction modes. Since the intra prediction are performed on blocks of size ranging from 4×4 to 32×32, there are 140 combinations of block sizes and prediction modes for the codec to address [34]. The best combination is selected by RDO [33].

The coding efficiency achieved by HEVC intra coding, compared with H.264/AVC, is reported to be up to 36% and 20% on average [30]. This significant performance increase mainly come from its novel quad-tree structure and rich prediction modes. However, the success of HEVC consists of many small contributions from detailed designs.

2.2 HEVC Intra Frame Coding

In this section, a brief introduction of the HEVC intra-frame coding techniques is given. The content will mainly focus on relevant architectures and algorithms in intra frame coding, including the quad-tree-based blocks partition scheme, spatial sample prediction, transform coding, and some basic concepts in HEVC standard. These background knowledge forms the basic materials for us to analyze and design fast intra algorithms.

2.2.1 Quad-tree Structure in HEVC

Since video data are highly unstationary, encoder should process data adaptively based on different contents. HEVC handles various video contents by partitioning each frame into blocks of various sizes. These image blocks are then processed as basic units in core encoding processes: prediction, transformation, and entropy coding. Since the hybrid coding structure consist of those three main steps, there are also three concepts of basic units: CU, Prediction Unit (PU), and Transform Unit (TU) [19].

In HEVC intra coding, even the prediction and transformation can be done independently, TU is defined under PU while PU is defined under CU, which means the size of TU is less than or equal to its PU and the size of PU is less than or equal to its CU. Due to this hierarchical relationship, decision of CU partition has the largest flexibility and also causes the main load of computational complexity. The size of a CU can vary from 64×64 to 8×8. When recursively partitioning a large CU into smaller sub-CUs, the process naturally forms a novel quad-tree structure. The largest coding block at the root of a quad-tree is defined as Coding Tree Unit (CTU) and it is at depth 0. Each CTU is of fixed size as large as 64×64 or smaller based on the configuration; it can be encoded either as a single large CU when the video content is simple or as small sub-CUs when the video content is complex.

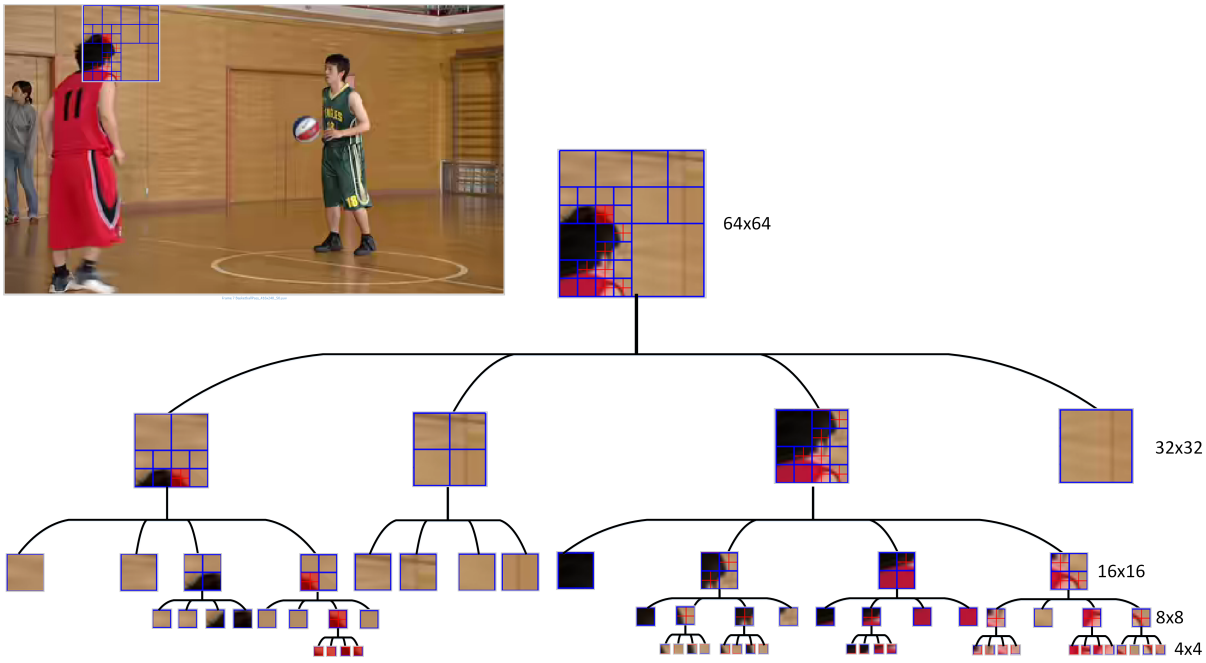

Figure 2.1: Quad-tree in HEVC

The quad-tree describes the recursive partition process via which every CU, represented as a node of the tree, can be split into four sub-CUs within the maximum depth. Figure 2.1 shows an example of quad-tree when the size of CTU is 64×64. The data in this example comes from the 8th frame of standard test sequence BasketballPass_416×240. The quad-tree decision is made under the quantization parameter 32, which is a neutral trade-off between quantization error and bitrate. As shown in this example, the 64×64 image block contains inhomogenous content and can not be properly modelled as a large block. Therefore it is partitioned into sub-blocks. For those sub-blocks which can be predicted well, they will terminate as leaf nodes of the quad-tree. However, for those blocks that still contain enough complex textures, to achieve better coding efficiency, they may keep partitioning into smaller blocks until they can be effectively predicted or reach the maximum depth.

2.2.2 Spatial Sample Prediction

The essence of intra coding is to remove the redundancy of data in spatial domain. Therefore, intra prediction is also called spatial sample prediction. The core algorithms of spatial

sample prediction can be summarized into three stages. The first stage is to construct reference sample array as the base materials for prediction. Then, in the second stage, prediction blocks can be generated by applying different prediction modes. To achieve a better prediction result, the last stage is post-processing, which will conditionally filter the prediction blocks to reduce discontinuities on the block boundaries [34].

Table 2.1: Indexes of intra prediction modes

Mode Type	Mode Index	Target Content
Angular	2-34	Directional
Planar	1	Gradually Changing
DC	0	Honogenous

As mentioned before, there are 35 intra prediction modes in HEVC. In H.264/AVC, the codec provides only nine intra prediction modes. While this number become insufficient for HEVC since larger block sizes are now utilized. Based on their characters, 35 modes can be categorized into 3 sets, planar, Direct Current (DC), and angular (33 directional modes). Those modes are designed with intentions to handle various types of content that typically appear in video data and also to achieve fine trade off between computational complexity and coding efficiency [23]. To be specific, angular modes, as the largest category, are defined to capture directional pattern or edges in image blocks. DC mode is useful when predicting homogeneous blocks and planar can effectively model image contents that are gradually changing with the space. It need to be noticed that DC and Planar also can be applied to complex texture that cannot be effectively predicted by any simple directional mode.

Although large number of modes have been introduced in HEVC intra coding, essentially, all intra prediction modes can be treated as mapping functions. Since the redundancy lays in the correlation between spatially neighboring pixels, these functions take restructured pixels from neighboring boundary as input, and generate prediction data for current PU as output. Just as equation 2.1 shows, M is prediction mode, R is an array of reference samples. Prediction function $F(M, R)$ processes R based on type of M and output P_M as the prediction data. For more details about how the function $F(M, R)$ is designed, please see reference[34].

$$P_M = F(M, R) \tag{2.1}$$

The left side of figure 2.2 shows 33 directions defined in HEVC. The maximum sample accuracy of each direction is 1/32 pixel [34], and each direction has an angular parameter.

The angular parameters and their corresponding modes are shown on the top and left boundary of the figure. Particularly, mode 10 is horizontal mode and 26 is vertical mode. It is found, from the statistical result, that vertical and horizontal patterns present more frequently in natural images than other directional patterns. Based on this observation, 33 modes are intentionally deigned to achieve more prediction accuracy on vertical and horizontal modes. Reflected from the figure 2.2, the arrows near mode 10 and mode 26 are more dense than that of diagonal arrows.

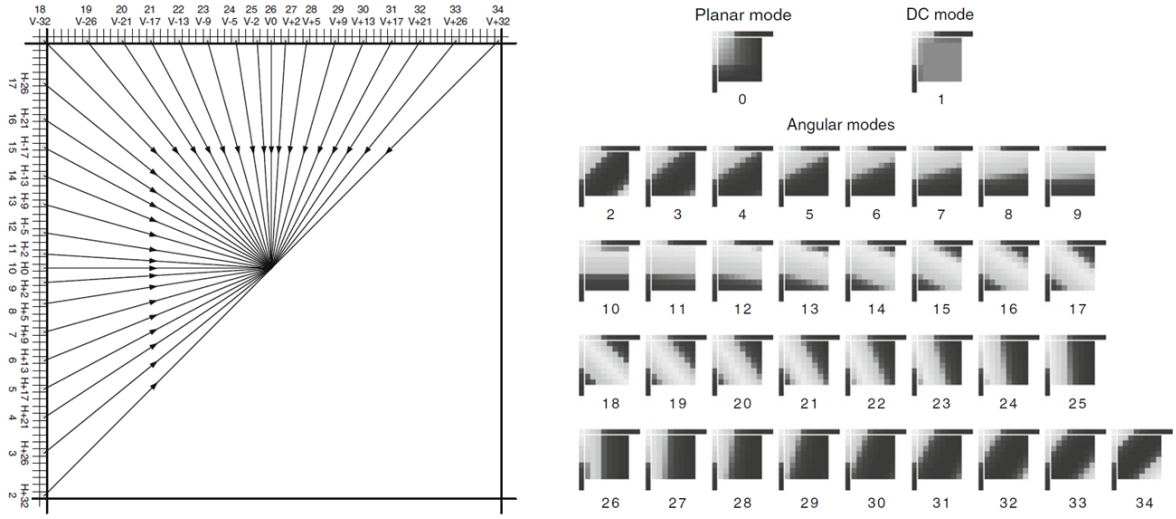

Figure 2.2: Intra Prediction Modes in HEVC

The right part of figure 2.2 shows 35 predicted image blocks generated by different prediction modes when given the same reference samples. It can be noticed that, for mode 10, mode 26 and DC mode, some boundaries of prediction blocks are conditionally processed by filters to provide better continuity of image block. These filtering operations are named with post-processing for predicted samples.

Apart from applying prediction modes, there are two unusual modes to encode a CU, I_PCM and transform skipping mode. However, they are not treated as prediction modes since they don't involve prediction. The main application scenario for I_PCM mode is when the CU data is too complex that no other mode can handle it well. Unlike I_PCM, transform skipping mode will skip the transformation and encode the residual data directly. It is especially effective when encoding the videos that generated by computers.

All the prediction methods introduced above are performed within PU. As the name

implies, PU is the smallest unit for prediction, which means that all pixels in one PU should be predicted by using the same prediction mode. Usually, the size of PU is smaller than the size of its corresponding CU. In inter coding, PU is even not necessary to be a square block. While in intra coding, they are strictly square and have little flexibility for partition under its CU. Experiments have proved that the efficiency gain from various PU sizes within a large CU is minor in intra coding [34]. To reduce the mode selection complexity, HEVC intra coding has restricted PU size to be the same with CU size from depth 0 to depth 2. Only in depth 3, where CU size is 8×8, a PU can be partitioned into four 4×4 PUs [20].

Even though PU is defined as the unit of prediction, it only restricts that data within a PU should be predicted by the same IPM. However, the prediction method can be applied in TU level if available TU sizes are smaller than the size of current PU. This flexible scheme allow the intra prediction algorithm to always take the nearest reference samples as input, which makes full use of spatial correlation within a PU. According to the experiments, the performance gain from TU-based intra prediction is 1.2% in comparison with the method that applies prediction for a whole PU [33].

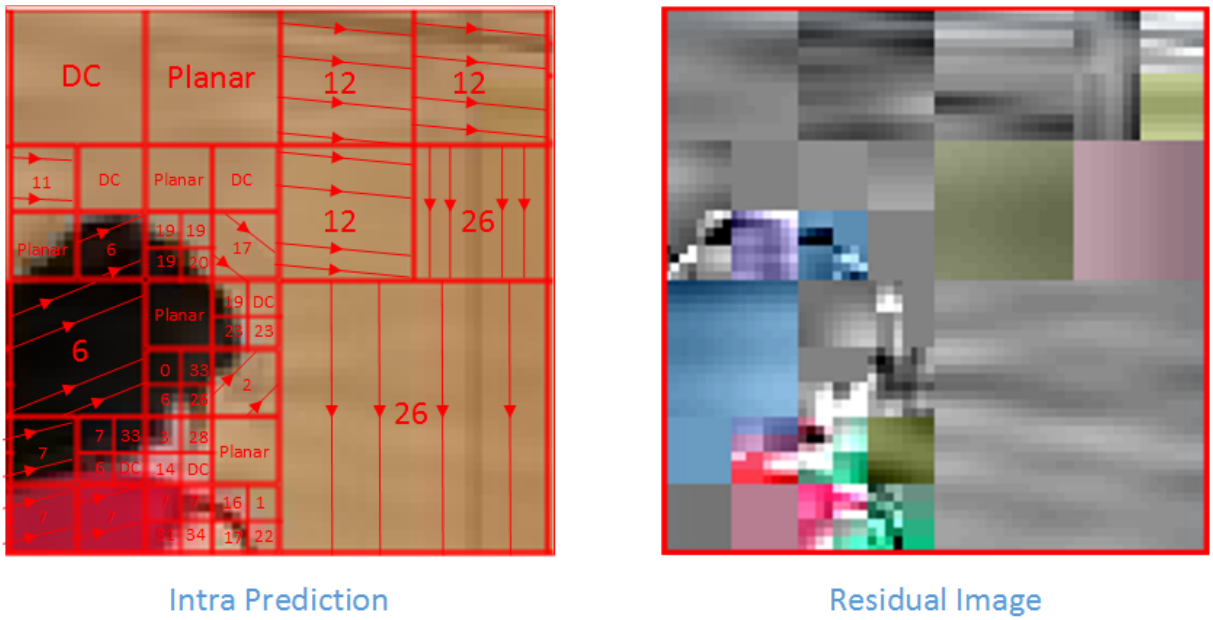

Figure 2.3: Example of Prediction and Residual

Figure 2.3 illustrates an image of predicted data and an image of residual data. It is a

CTU of size 64×64. The block partition decision and mode decision can be clearly seen from this example. Because the block can not be well predicted as a whole 64×64 CU, it is recursively partitioned into multiple small blocks. Each block can use its own prediction mode to generate the prediction. For blocks contain vertical pattern, like the background in the right-bottom corner of the CTU, the best mode is 26. While for blocks that are very homoneneous, like the CU in the left-top corner of the CTU, DC mode could be more effective in this case. After the prediction been generated, it will be subtracted from original image to generate residual image, which will be further transformed, quantized and encoded. It can be noticed that the energy of residual become much smaller. However, most of the content are noise-like pattern that can not be removed by intra prediction. Those content are related to high frequency component in DCT coefficients. Later on, this will become one of our inspirations to find good features for the fast intra algorithm.

2.2.3 Transform in HEVC

After intra prediction, there are still data left in the residual image. To further remove the spatial correlation of residual image, DCT or Discrete Sine Transform (DST) will be applied to transform the data from spatial domain into frequency domain. In HEVC, DST is only applied for 4×4 luma blocks, while DCT is applied in all other situations. Because we are going to extract feature from DCT coefficients, we will introduce more about DCT in our thesis. Let $\mathbf{X}$ represent $N \times N$ residual matrix, transform matrix $\mathbf{D}$ is used to project the residual matrix on several orthogonal basis matrices. And $\mathbf{Y}$ is the transformed coefficient matrix.

$$\mathbf{Y} = \mathbf{D}\mathbf{X}\mathbf{D}^T \tag{2.2}$$

Transform matrix $\mathbf{D}$, based on its size N, has its mathematical expression as

$$\mathbf{D} = \begin{bmatrix} \sqrt{\frac{1}{N}} & \sqrt{\frac{1}{N}} & \cdots & \sqrt{\frac{1}{N}} \\ \sqrt{\frac{2}{N}}\cos\frac{\pi}{2N} & \sqrt{\frac{2}{N}}\cos\frac{3\pi}{2N} & \cdots & \sqrt{\frac{2}{N}}\cos\frac{(2N-1)\pi}{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \sqrt{\frac{2}{N}}\cos\frac{(N-1)\pi}{2N} & \sqrt{\frac{2}{N}}\cos\frac{3(N-1)\pi}{2N} & \cdots & \sqrt{\frac{2}{N}}\cos\frac{(2N-1)(N-1)\pi}{2N} \end{bmatrix}. \tag{2.3}$$

However, in HEVC, the calculation of $\mathbf{Y}$ is based on integer. Elements in matrix $\mathbf{D}$ are also approximated by integers. Treating 4×4 as en example, the transform matrix defined

in the standard document is:

$$\mathbf{D} = \frac{1}{128}\begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{bmatrix}. \tag{2.4}$$

Then the coefficient matrix $\mathbf{Y}$ can be calculated as

$$\mathbf{Y} = \frac{1}{128 \times 128}\begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{bmatrix} \mathbf{X} \begin{bmatrix} 64 & 83 & 64 & 36 \\ 64 & 36 & -64 & -83 \\ 64 & -36 & -64 & 83 \\ 64 & -83 & 64 & -36 \end{bmatrix}. \tag{2.5}$$

In terms of DST, it is following the similar principle but using different transform matrix. In HEVC, the transform matrix $\mathbf{S}$ of 4×4 DST is

$$\mathbf{S} = \frac{1}{128}\begin{bmatrix} 29 & 55 & 74 & 84 \\ 74 & 74 & 0 & -74 \\ 84 & -29 & -74 & 55 \\ 55 & -84 & 74 & -29 \end{bmatrix}. \tag{2.6}$$

In the real implementation, transforms can be calculated by butterfly algorithm to achieve a faster speed. After the transform process, coefficients will be quantized and scanned into symbol sequences. Together with the symbols about the mode information, entropy coder will encode all symbols into binary bits. To have a better understanding of DCT, let's explore the meaning of each coefficient. Linear transform actually represents the object to be transformed into a linear combination of basis functions. The example in figure 2.4 shows 64 basis functions of 8×8 DCT. In a $N \times N$ DCT, basis functions are orthogonal to each other and form a complete description of $N \times N$ matrix. In other words, any $N \times N$ images can be represented by the summation of those basis matrix $\mathbf{B}_{i,j}$ times with the corresponding coefficient $Y_{i,j}$.

$$\mathbf{X} = \sum_{i=1}^{N}\sum_{j=1}^{N} \mathbf{B}_{i,j} Y_{i,j} \tag{2.7}$$

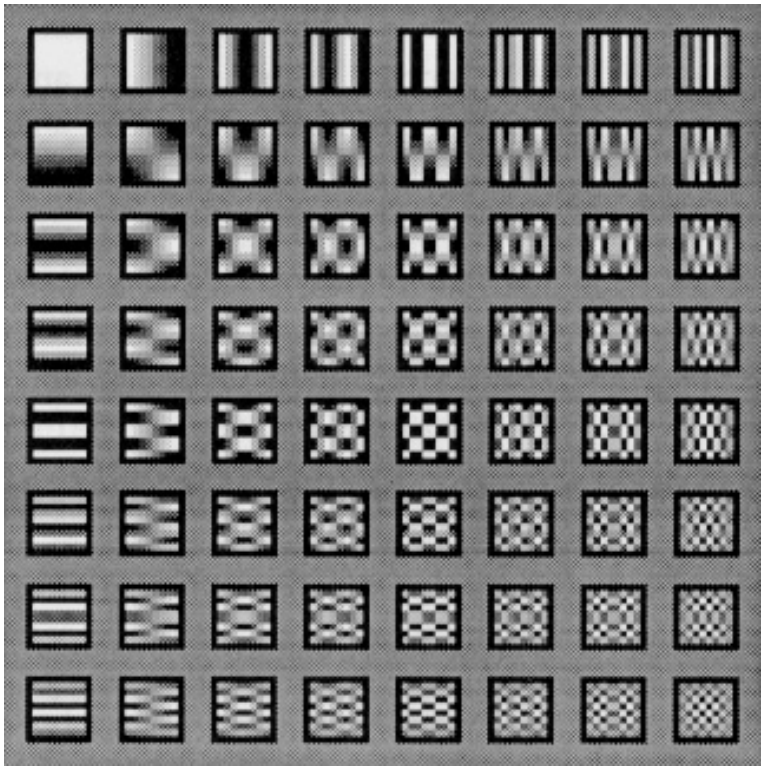

Figure 2.4: Basis Functions of 8×8 DCT[26]

For the coefficient matrix $\mathbf{Y}$, value of $Y_{1,1}$ is called DC coefficient, which is associated with the average value. Besides DC, other coefficients are called AC coefficients. From the figure 2.4, AC coefficients on the first row $Y_{1,j}$ are associated with vertical pattern while value of $Y_{i,1}$ are corresponded to horizontal patterns of the image. The frequency in image sense reflects the changing behaviour on space. For example, if the content of image is rapidly changing on a short distance, like edges, its coefficient matrix tends to have large value on high frequency component. Therefore, from result of DCT, we can gain an intuitive understanding of the property of images. For images contain homogeneous or gradually changing content, most of the energy will be concentrated on low frequency area. For those complex pattern in images, they may result in large efficients which are closer to the right-bottom corner of the $\mathbf{Y}$.

2.3 Recursive Compression Structure

Our thesis focuses on fast intra algorithms, in other word, reducing the complexity of intra encoder. Therefore it is necessary to introduce some core functions of the current implementation of intra encoder. Only by a deep understanding of standard implementation can we find clues to come up with better solution. Since large CU can be partitioned into sub-CUs in the quad-tree, to make methods consistent across different block size, the CU compression process is implemented through recursion.

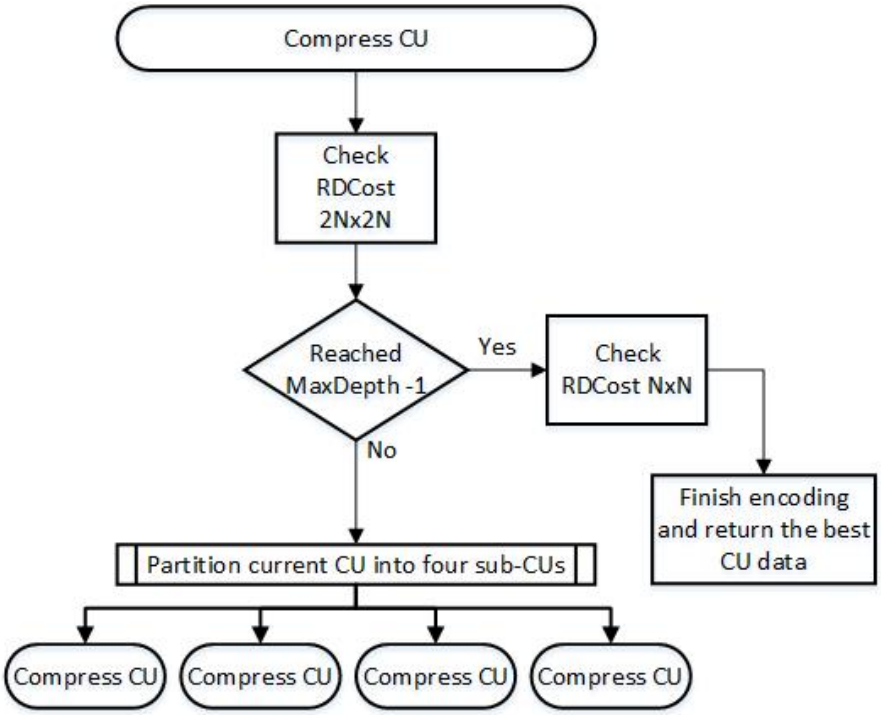

Figure 2.5: Recursive Process of CTU Encoding (intra) in HM 16.0

Figure 2.5 shows the core function CompressCU in $HM16.3$. At the first level of recursion, the program starts with the largest CU with size 64×64. To explore the best way to encode the CU at current depth, it will call a function named with CheckRDcost2Nx2N which will try to predict and encode the data as a whole block by using different IPMs. After that, CheckRDcost2Nx2N will return the best CU data including best IPM, best RD-cost, and other information. While those best CU data are derived just from current level, if partition CU into smaller CUs, better CU data could be obtained. Hence the program

will call the same function CompressCU on four sub-CUs and further explore the best CU data. The definition of RD-cost is shown in equation 2.8. SSE denotes the summation of square error between original image $I(x, y)$ and reconstructed image $I'(x, y)$. R_{Total} is the total number of bits used to encode current CU, and λ is the Lagrange multiplier that is decided by Quantization Parameter (QP) and frame type.

$$J_{RDO} = SSE + \lambda * R_{Total} \tag{2.8}$$

$$SSE = \sum_{x,y} |I(x, y) - I'(x, y)|^2 \tag{2.9}$$

When the quad-tree reach the maximum depth, where CU size is 8×8, the program reach the last level of recursion. Then the function CheckRDCostNxN will be called to split 8×8 PU into four 4×4 PUs and check the possibly better RD-cost. After that, at the end of recursion, the best CU data among all choices will be returned to the top, level by level, and the compression of current CTU is finished. The program then turns to next CTU and call the same function CompressCU.

2.4 State of the Art

2.4.1 Rough Mode Decision

Because the high computational complexity of HEVC intra frame coding mainly comes from its quad-tree structure and large number of prediction modes, nearly all fast intra coding algorithms can be categorized into two types: quad-tree decision algorithms and IPM decision algorithms. To achieve better encoding time reduction, some papers choose to combine these two. But those are still following the similar categorization. In intra coding, PU is the same with its CU until the last depth (depth 3). Therefore, quad-tree decision is mainly focused on CU size decision. For IPM based method, the main strategy for accelerating IPM selection is to reduce the number of candidates for its internal stages.

In terms of prediction modes, HEVC has already adopted some fast algorithms. In the early stage of HEVC, a Rough Mode Decision (RMD) was first proposed and adopted by HM1.0 in 2010 [28]. To reduce the high computational cost of RDO, the unified intra mode decision method was proposed to select a smaller RDO-candidate set from 35-modes full set based on an approximated value of RD-cost. This approximation, denoted as J_{RMD}, is

chosen as absolute Summation of Hadamard Transformed Coefficients (SATD) of residual data plus the estimated rate cost R_{Est}. In this algorithm, 35 modes are first applied to the image block to obtain 35 SATD values. After that, only N modes with minimum SATD are selected for further RDO. Since the RDO is much more time consuming thanSATD calculation, encoding time can be saved by this two stage algorithm.

$$J_{RMD} = SATD + \lambda * R_{Est} \tag{2.10}$$

After that, L. Zhao and L. Zhang introduced a three-stage method in 2011 that further improved the RMD [39]. This innovation is later adopted and implemented in the current HM. The three stages are RMD, Most Probable Mode (MPM) generation and RDO. This process is illustrated in figure 2.6.

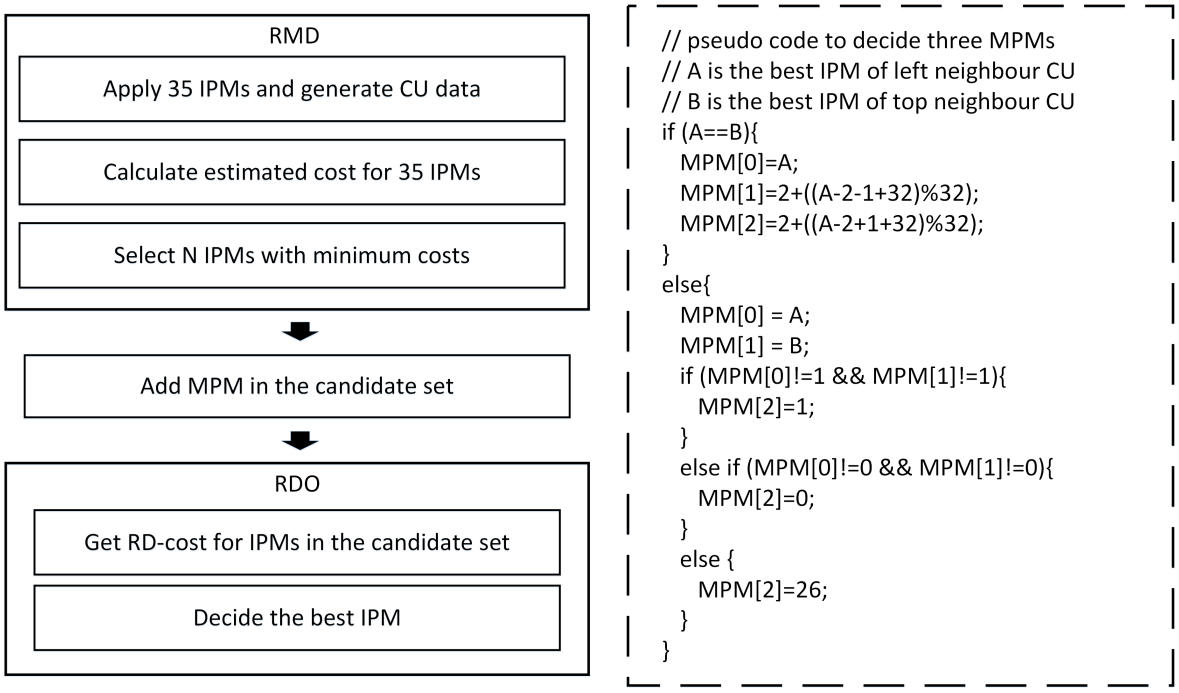

Figure 2.6: Three-stage Mode Selection Algorithm

Besides N IPMs with minimum J_{RMD}, three MPMs decided from the best IPM of neighbour CUs are also added in RDO-candidate set. The detail logic to find MPMs is shown in right side of figure 2.6. The value of N in different versions are compared in

2.2. It can be seen that the N in [39] is much smaller than that in [28]. Therefore, the computational cost in the RDO can be further reduced by [39].

Table 2.2: Value of N in different RMD

PU size	N in [28]	N in [39]
64×64	5	1
32×32	4	2
16×16	4	2
8×8	9	4
4×4	9	4

2.4.2 Latest Innovations

In 2012, a gradient based fast mode decision algorithm for HEVC intra prediction is proposed by W. Jiang[16]. Gradient vectors are extracted by applying Sobel masks on original image. Intra prediction modes are decided by amplitude and angle of gradient vectors. The encoding time reduction is reported to be 20% while the BD-rate increase is relatively large. However, the idea of using gradient features has widely inspired other papers. In 2014, H.Zhang and Z.Ma proposed a hybrid intra mode decision algorithm[37]. The proposed method consisted of two levels: micro-level and macro-level. The micro-level method focused on IPM selection while the macro-level focused on CU size decision. The micro-level method only reduced the number of modes to be tested in RMD and follow the same RDO process in default HM. To reduce number of modes in the RMD, a progressive rough mode decision(pRMD) was utilized. In pRMD, a smaller set of modes, as a sub-sampled set of 35 modes, was first tested, where the SATDs of each mode in the subset would be calculated as the rough cost. Then another RMD would be applied to the neighbour modes of the mode with the smallest SATD and add the mode with lowest SATD to the result set. Later, MPM would also be added into the result set. When the result set contained the specific number of modes, the candidate set for RDO was finally locked down. In terms of the macro-level method, the CU splitting process was early terminated if the predicted RDcost of split CUs was larger than that of current CU. The algorithm proposed in this paper had some drawbacks. For example, the progressive RMD proposed in the paper was not suitable for parallel computation. For a computer of strong parallel-computing ability, because different modes are independent, test 35 modes and obtain the SATDs can be very fast while follow a progressive method to test them sequentially could be even slower.

In April 2015, G. Correa proposed a fast encoding decision algorithm for HEVC by using data mining methods [9]. Several Decision trees were introduced as decision maker to accelerate both intra and inter encoding process. The Waikato Environment for Knowledge Analysis(WEKA), a famous open-source data mining package, was used in this paper to train offline models and fulfill classification tasks. Features include RD-cost, depth of neighbour CUs, Because some video sequences are used as training data and only ten sequences are tested, the paper didn't provide a full-test result as the official HEVC documents of test conditions required [5].

Similarly, in May 2015, Biao Min and Ray C.C.Cheung proposed a fast CU size algorithm, which utilized the gradient and variance of luminance pixels[27]. Multiple features, including global edge complexities and local edge complexities, are retrieved from the CU data. Global edge complexities are variances calculated from different partition styles while local edge complexities are variances of outputs from four local edge filters. After feature extraction, a manually designed decision tree was used to generate three kinds of flags, which controlled the decision of CU size. In the decision tree, there are only two parameters: local threshold and global threshold. Both of these two parameters are trained offline, using part of test sequences. The paper provides a full test result and source code. As a general method, the algorithm proposed in that paper is concise and effective. However,from the stand of statistics or machine learning, a little drawback of their algorithm is that they have applied the model in a testing set which contains data overlapping with its training set.

In September 2015, Nan Hu proposed a fast mode selection algorithm for HEVC intra-frame coding based on Bayes Decision method[15]. This paper have also used LPTCM as feature extraction method. The feature used in this paper is the number of Outlier Block Flag (OBF). The first 2 frames of 20-frame period are used to collect statistics and a Bayes Decision model can be trained. Since only 2 frames are used for the training, the Bayes model may not converge well. Also, because the feature is only a single number, the information utilized is not sufficient for accurate prediction. Through our test, the performance is not very satisfactory.

In April 2016, T. Zhang proposed a fast algorithm to decide both intra prediction modes and CU size [38]. Apart from SATD, RD-cost and depth of neighbour CUs, the paper also extracted average gradients in the horizontal direction (AGH) and vertical direction (AGV) as features. Equations 2.11 show the mathematical expression of Zhang's features.

$$
\begin{aligned}
G_x(i,j) &= p(i,j-1) - p(i,j+1) \\
G_y(i,j) &= p(i-1,j) - p(i+1,j) \\
AGH &= \tfrac{1}{N*N} \textstyle\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |G_x(i,j)| \\
AGV &= \tfrac{1}{N*N} \textstyle\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |G_y(i,j)|
\end{aligned}
\tag{2.11}
$$

Then the features are processed and utilized to make encoding decisions. In terms of CU size decision, CUs are first classified as homogeneous CUs and non-homogeneous CUs by comparing the maximum value of AGH and AGV with a threshold. The CU splitting of homogeneous CUs were terminated at this stage, while non-homogeneous CU were further processed to obtain more features and passed into two offline-trained SVMs for CU decision. In terms of intra prediction modes, the mode set for RMD was narrowed down into smaller subsets of original 35 modes by comparing AGH/AGV (ratio of AGH and AHV) with a series of thresholds. Hence, the complexity for checking all modes in the RMD can be reduced.

So far, numbers of fast intra algorithms have been introduced. But there are still several problems that haven't been effectively solved yet. Firstly, most of the current methods are using single-dimensional feature with offline trained thresholds. As we know, video data are highly unstationary, offline trained parameters may works well for standard test sequences while it may not be effective for general video data. What's more, off-line models have lost the flexibility to achieve more accurate prediction. Therefore, self-adaptive and on-line trained algorithms are more desirable than fixed design.

Secondly, most algorithms are trying to predict the encoding decisions by utilizing correlation between features and decisions. To train a robust threshold-based model from experimental insights, the feature should be as simple as possible. Ideally, it is expected to be a single number. However, through our observations, the encoding process is complex and there are not too many strong features that can highly correlate to encoding decision. Instead, numbers of weak features can be generated in the encoding process. Each of them may partially contains the information about the encoding decisions. The two crucial conditions for us to make good prediction are that we must take enough information as input and we should process it with right methods. Therefore, the problem is how to utilize high dimensional data to predict the encoding decisions as accurate as possible under the trade-off between complexity and coding efficiency.

Another problem is that, if we want to further achieve better results based on current encoding framework, the design of codec could be more and more complex. Despite of whether we could afford the tough work to manually mine out those decision rules, as engineers, we should always think about how to simplify the problem and automate the work. Hence, machine learning is the trend and also could be the best choice for future video coding.

2.5 Summary

In this chapter, we have given an brief overview of HEVC. As the basics for designing fast intra algorithms, some crucial contents of HEVC intra frame coding that relate to our algorithm, like quad-tree structure, spatial sample prediction, and recursive implementation are discussed in detail. It is pointed out that the coding efficiency enhancement of HEVC is achieved at the cost of tremendous increase in computational complexity. This computational cost, in intra coding, mainly comes from exhaustive search on large amount of quad-tree structures and IPMs. To look for more inspirations for fast algorithms, typical methodologies and some recent innovations on this topic are introduced and analyzed. We have known that the current HEVC has already adopted a three-stage RMD in the standard test model. To reduce the encoding complexity, a fast algorithm should focus on accurately predicting CU size. What's more, to achieve good prediction, feature extraction is the key. In the literature, various information like gradient, variance, depth differences, RMD-cost, RD-cost, OBF have been used as features.

Chapter 3

ML Based Fast HEVC Intra Algorithm

3.1 Algorithm Structure

3.1.1 General Model

Since the HEVC been released, there are many HEVC fast intra algorithms coming up. However, most of them are focusing on specific methods and processes, and nearly no paper has summarized a high-level model or methodology that can be followed to design fast algorithms more efficiently. Through comprehensive reading and analyzing, we are trying to describe the problem more concisely and summarize some methodologies that can be generally applied for HEVC or even next generation of video coding.

Therefore, in this section, we will introduce a machine learning framework for fast intra algorithm that is applicable for any models and features. As shown in the 3.1, the encoding process of one CTU can be shown as a progress bar. To accelerate the encoding without changing the high level structure of encoder, the strategy is to skip some unnecessary steps and jump from one point of the progress bar to another point. Figure 3.1 has compared the coding progress bar of normal HM with that of fast algorithm. For the functional blocks to help making those decisions, we name them with Decision Maker (DM). DMs are different in three aspects: what decision the a DM makes, what type of decision model a DM uses, and what kind of features or information inputs a DM takes.

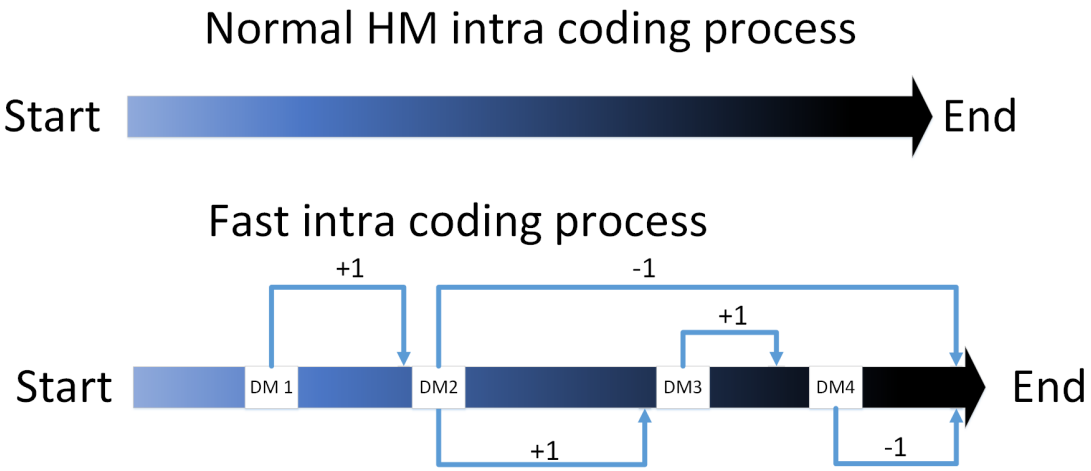

Figure 3.1: General Model of Fast Intra Algorithm

Theoretically, decisions of a DM could be jumps from one point to any point in the progress bar. However, our purpose is to reduce encoding time and maintain the coding efficiency. Therefore, not all jumps are meaningful or effective. On one hand, we encourage a DM to jump to points after it as far as possible so that we can skip more operations and save encoding time. On the other hand, we may limit this stride to ensure that a DM can make precise decision, otherwise the coding performance will suffer. Hence, the first step is to design reasonable decisions that can easily achieve high time reduction as well as good coding performance.

As mentioned before, the encoding of HEVC is following a quad-tree from depth zero to the maximum depth and the effectiveness of intra sample prediction depends on the content of current CU image. Homogeneous content may tend to be encoded into large blocks while complex content are more often encoded into small blocks. If a CU is found to be homogeneous, a possible decision is to finish the encoding on current depth and stop any further exploration. This decision is called CU termination. While, if a DM finds the CU on current depth is very complex, as a rule of thumb, a decision could be to skip the encoding trials on current depth and directly jump to the next depth of the quad-tree. This decision is called CU skip.

CU skip is to skip the whole depth at the beginning of the function CompressCU. However, within one depth, with the encoding process going on, many intermediate data will be generated. Those data provide more information that could allow DMs to make more precise decisions. For example. after the RMD, new data like the SATDs and

residuals are available. If a DM think the CU can't be encoded efficiently in current depth after added those new information to the input. An available choice is to skip the following RDO and jump to the next depth. This decision is called RDO skip. Because RDO is the most time-consuming part in CompressCU, encoding time can still be effectively saved by this half-way decision.

3.1.2 Specific Logic

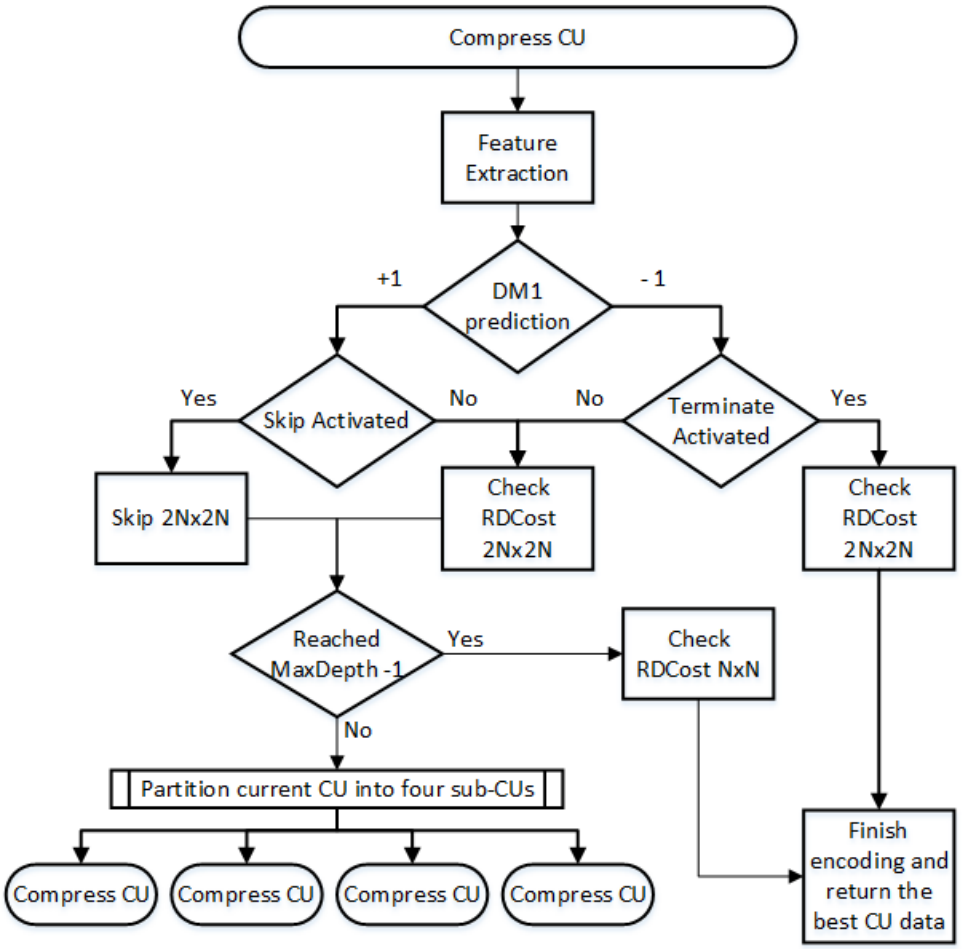

Figure 3.2: Flowchart Diagram of Proposed Algorithm

Here we give a specific logic of our algorithm. The flowchart diagram is illustrated in figure 3.2. The first DM(DM1) is inserted at the begin of CompressCU since at this point it has the maximum potential for time reduction. Since the encoding of CU just starts, the features utilized by the DM1 are all from the original image. Let $F(CU)$ represent the general feature extraction method that take CU as input and output a N dimensional feature vector $\boldsymbol{x}$. One example of $F(CU)$ is using LPTCM to detect outlier DCT coefficients.

$$\boldsymbol{x} = F(CU) \tag{3.1}$$

When the feature vector is extracted from the CU, it will be fed into DM1.

$$y = f_{DM}(\boldsymbol{x}) \tag{3.2}$$

At this point, we first suppose that the decision models are already available and the training method of those decision models will be introduced later. When DM1 takes $\boldsymbol{x}$ as input, it could have two possible value of output y. If the y is +1, it means CU skip; if the y is -1, it means CU termination. In our design, rather than trust those decisions directly, the output need to pass the performance control block to make sure the effectiveness of the decision. Inside the DM1, there may be several models. The performance of each model will be evaluated in the validation stage. The model with the best performance will be chosen to make the decision. Also, there is a performance threshold as the minimum requirement. If none of the models can meet the requirement. The corresponding decision will be switched off. In this case, the encoding process will not be affected by the inactive DM so that the coding performance can be ensured.

3.2 Performance Measure

3.2.1 Measure for Classification Performance

Here we give some definitions of performance measure for our machine learning framework. Models for classification, in our application scenario, are considered as decision makers. A decision maker can have a targeted decision, like terminating or skip some process. If the predicted output or the ground-truth label is the same with the targeted decision, we denote it with +1, otherwise with -1. When passing a sample into a models in the testing process, there are four possible results, as shown in table 3.1. If prediction is +1 and label

is also +1, the result is called True Positive (TP). If prediction is +1 while label is -1, the result is False Positive (FP). Similarly, when prediction is -1, based on different label values, another two results are called False Negative (FN) and True Negative (TN).

Table 3.1: Four Classification Results

Prediction		+1	-1
Label	+1	True Positive(TP)	False Negative(FN)
	-1	False Positive(FP)	True Negative(TN)

Based on four results we defined above, accuracy, expressed as 3.3, can be calculated to measure the general performance for a classification model. However, the precision to make targeted decision is more important in our framework. It defines as number of right positive decision divided by the number of all samples predicted as positive 3.4. On the other hand, sensitivity of the model also matters. It is the percentage of real positive samples that a classification model can successfully catch 3.5.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.3}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.4}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{3.5}$$

3.2.2 Measure for Encoding Performance

Since accuracy and precision are only used to measure the effectiveness of classification models, we still need a measure that can directly evaluate the computational complexity and coding efficiency. In terms of computational complexity, it is measured by encoding time reduction. The calculation of time saving strictly follows the equation 3.6. To always make fair comparisons, all results of fast algorithms are compared with its corresponding HM with same version. For example, our proposed algorithm is implemented in $HM16.3$, hence the result is compared with default $HM16.3$. We also run others' encoders on the same machine. Since we only have their executable files. Defined as a general rule, for

encoders which are implemented based on HMX, their results are compared with default HMX.

$$TS = \frac{Time_{HMX} - Time_{proposed}}{Time_{HMX}} \times 100\% \tag{3.6}$$

Bjntegaard-Delta Bit-Rate (BD-rate), proposed by Erlend Bjntegaard, is widely accepted as a standard way to assess the video coding efficiency. It is generated through fitting of Rate Distortion (RD) curve [3]. Treating the result of original HM as benchmark, if the BD-rate is negative, the proposed encoder achieves smaller bit-rate when given the same level of distortion. Otherwise, if the BD-rate is positive, it means that the proposed method has introduced degradation in coding efficiency. The RD curve typically contains four points with different QP. As a standard practice, the four QP values are set to be 22, 27, 32, and 37. In our thesis, without additional remarks, BD-rate is calculated follow this standard way. Table 3.2 ,as an example, shows the encoding result of $HM16.3$ and proposed algorithm. The result is from the first 60 frames of sequence PeopleOnStreet_2560x1600_30_crop.yuv.

Table 3.2: An example of encoding result, 60 Frame Test

Sequence: PeopleOnStreet_2560x1600_30_crop.yuv						
Original HM16.3 x64						
QP	Bitrate	Y-PSNR	U-PSNR	V-PSNR	YUV-PSNR	Enc-Time/s
22	173807.20	43.243618	45.600892	45.354502	43.868260	1590
27	100790.82	39.801365	43.216357	43.536167	40.699809	1354
32	57310.10	36.675014	41.224019	41.878033	37.779509	1198
37	33371.34	33.825296	39.743744	40.602537	35.107585	1087
Proposed HM16.3 x64						
QP	Bitrate	Y-PSNR	U-PSNR	V-PSNR	YUV-PSNR	Enc-Time/s
22	174003.53	43.215166	45.601975	45.358991	43.846989	899
27	101192.92	39.778306	43.220455	43.548244	40.682313	681
32	57531.78	36.650874	41.222478	41.876591	37.758535	600
37	33568.82	33.817566	39.695861	40.540127	35.094960	531

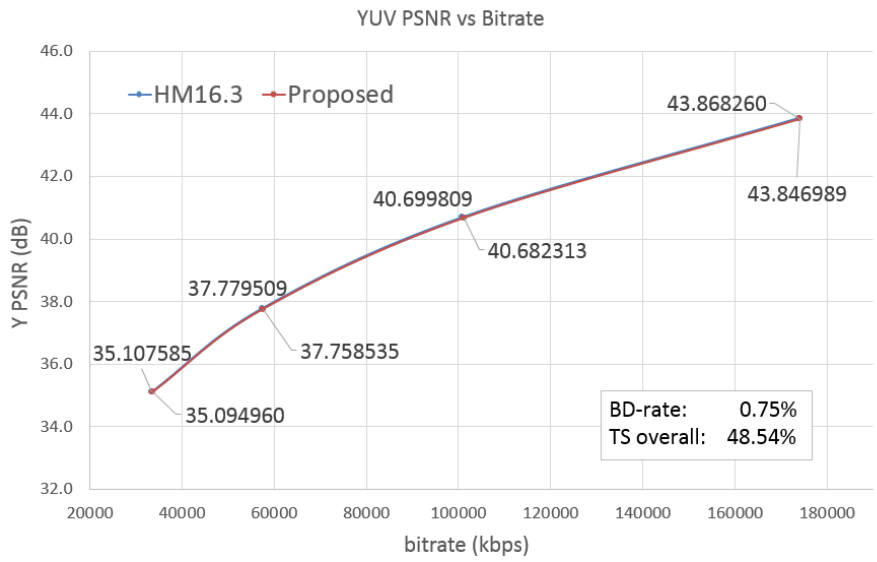

Figure 3.3: RD-curve of original HM16.3 and proposed method. Input video sequence: PeopleOnStreet_2560×1600_30_crop.yuv

3.3 Feature extraction

3.3.1 Transparent Composite Model

The feature extraction methods in the literature mostly focus on pixel domain, like methods based on variance and gradient. However, gradient and variance in pixel domain may not be effective since the RD-cost more depends on residual image rather than original image. To be exact, if an image can be well predicted by intra prediction, even with large variance and gradient, it still tends to be non-split case. Therefore, we are trying to find some new aspects of feature extraction.

As we discussed in the spatial sample prediction, for complex CU, it may not be modeled well by a single directional predictor. That will result in large energy in residual image. Since DC and low frequency components are relatively easy to be removed through intra prediction. The residual will mainly contain large coefficients of high frequency components. On the other hand, if a CU is homogeneous, most energy are focused in low frequency component, which will show low energy in high frequency components.

This shows us a rationale to capture the information about image texture and complexity in the frequency domain. Therefore, to achieve image understanding from this aspect, one basic question is how DCT coefficients are distributed. Many statistical models have been used in the literature to model the distribution of DCT coefficients[29][31][10]. They includes Laplacian distribution, Cauchy distribution, Gaussian distribution, mixtures thereof, and generalized Gaussian(GG) distribution. Based on the observation of [36], the histogram of DCT coefficients tends to have a heavy tail. Those large value on the tail usually contain important information about the image. However, the probability density functions of Laplacian, Gaussian and GG distributions all decrease too fast and can't have a good model accuracy on the tail. Therefore, (TCM), which uses a parametric distribution to model the main-body while uses uniform distribution to model the tail, has been proposed in [36]. If the parametric distribution is chosen as Laplacian, the TCM is called LPTCM. LPTCM turns out to be a good balance between the estimation complexity and model accuracy. Because we are designing fast algorithm and complexity is always taken into consideration, we have chosen LPTCM to model DCT coefficients of Alternating Current (AC) frequencies. The probability density function of LPTCM [36] is given in (3.7)

$$p(y|y_c,b,\lambda) \triangleq \begin{cases} \frac{b}{1-e^{-y_c/\lambda}}\frac{1}{2\lambda}e^{-|y|/\lambda} & \text{if } |y| < y_c \\ \frac{1-b}{2(a-y_c)} & \text{if } y_c < |y| \le a \\ \max\{\frac{b}{1-e^{-y_c/\lambda}}\frac{1}{2\lambda}e^{-|y|/\lambda}, \frac{1-b}{2(a-y_c)}\} & \text{if } |y| = y_c \\ 0 & \text{otherwise,} \end{cases} \tag{3.7}$$

where $0 \le b \le 1$, $0 < y_c < a$, and a represents the largest magnitude a sample DCT coefficient y can take. The parameters y_c, b, λ of the LPTCM are derived online via maximum likelihood estimation. DCT coefficients in the tail ($y_c < |y| \le a$), which are modeled by the uniform distribution, are called outliers. In terms of energy, outliers account for a very small percentage of AC coefficient (about 1.2%). However, they contain very important information about the content of images.

3.3.2 Summation of Binarized Outlier Coefficients

As we mentioned before, the feature used in [15] is only a flag that indicates whether a 4×4 DCT block contain outlier or not. To achieve better performance, we need to retain more information. Therefore, a set of features called Summation of Binarized Outlier Coefficients (SBOC) is defined. As a better reflection of image complexity and property, it has contained not only the number of the outliers but also the information about the

position of outliers. Then we will introduce how features are extracted from the image data. Firstly, the original frame is segmented into 4×4 blocks and processed by 4×4 DCT. After that, a map of DCT coefficients can be obtained. Those coefficients can be used to estimate the parameters of LPTCM. Given the PDF of LPTCM, we are able to detect AC coefficients with significantly large magnitudes. If a AC coefficient is larger than y_c, it will be marked as outlier coefficient. After that, the DCT coefficient map will be processed into a binarized outlier map, where outliers will be quantized into 1 and normal coefficients will be suppressed to 0. Meanwhile, all DC coefficient will be set to zero. Within each 4×4 block, the binarized outlier will be summed up to form a single number called SBOC.

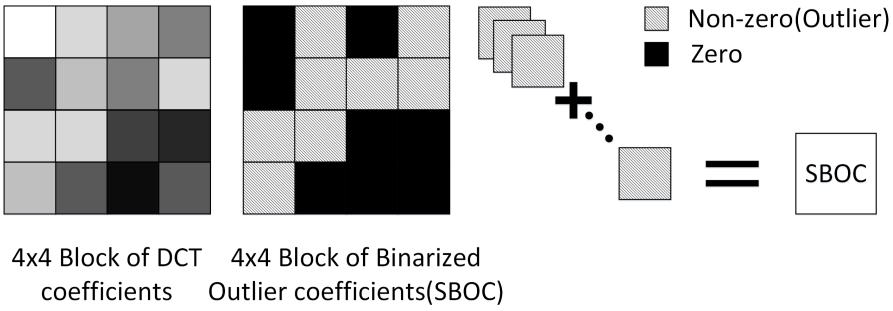

Figure 3.4: How SBOC is generated from a 4×4 matrix of DCT coefficients

Figure 3.4 has shown a example of how the SBOC is generated from a 4×4 DCT block. Trough this process, the map of SBOC can be generated. Since a CU will contain multiple 4×4 blocks, the SBOC map of a CU will be raster scanned into a vector, which is called SBOC vector. An example of extracting a SBOC vector from a 16×16 CU is shown in figure 3.5

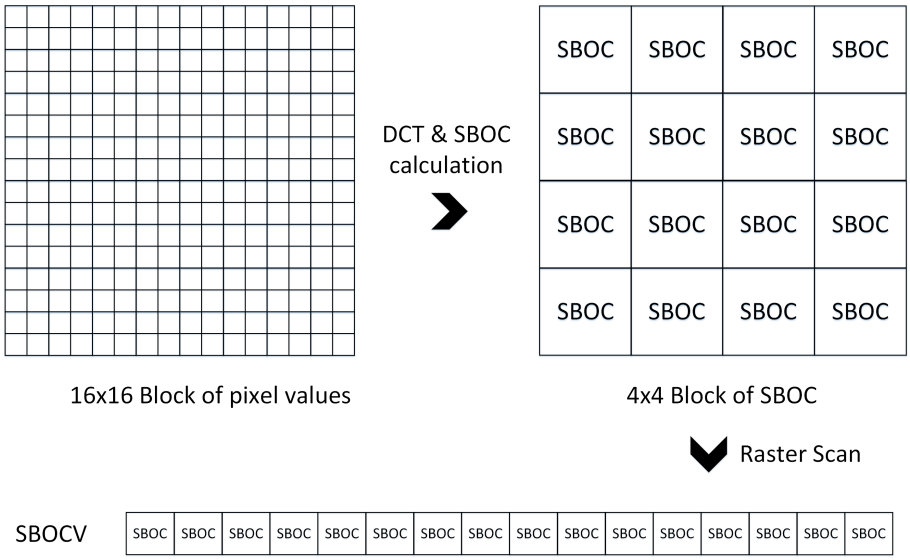

Figure 3.5: A example of extracting SBOC vector from a 16×16 CU

3.4 Training of Classification Models

3.4.1 Bayes Decision

In this section, we will introduce classification model that is available for our algorithm and the how those model are trained. The first one is Bayes decision. It is very effective for classification when the relation between feature and label fit well with the probabilistic assumption. Also it is intuitively understandable which can serve as an useful tool for us to find some insights about how to design the fast algorithm.

Let $\boldsymbol{X}$ denotes the feature variable and Y is used as label variable. In our application, the feature $\boldsymbol{X}$ is usually an one-dimensional value. In gerneral, it can be a N-dimensional vector, but for Bayes methods, that may need more training data to make the model converged. In terms of Y, it is always one-dimensional discrete variable. Generally, we assume that we have M labels:

$$Y \in \{y_1, y_2, y_3, ...y_M\}. \tag{3.8}$$

If $\boldsymbol{X}$ is discrete, from the training set, the probability mass function(PMF) can be estimated by empirical probability. Given the value of $\boldsymbol{X}$, we have the conditional probability of Y

calculated by Bayes formula. Then the prediction $\hat{\mathbf{Y}}$ should equal to the y_i that maximize this conditional probability

$$P(Y = y_i|\boldsymbol{X} = \boldsymbol{x}) = \frac{P(\boldsymbol{X} = \boldsymbol{x}|Y = y_i)P(Y = y_i)}{P(\boldsymbol{X} = \boldsymbol{x})} \tag{3.9}$$

$$\hat{Y} = \arg \max_{y_i \in \{y_1, y_2, y_3, \ldots, y_M\}} P(Y = y_i|\boldsymbol{X} = \boldsymbol{x}). \tag{3.10}$$

If $\boldsymbol{X}$ is continuous, there are two practical options. One is to quantize the $\boldsymbol{X}$ into discrete variable $\tilde{\boldsymbol{X}}$ and follow the same way with discrete case. Another is to model the of $\boldsymbol{X}$ with parametric distributions and get its Probability Density Function (PDF). Then the conditional probability become:

$$P(Y = y_i|\boldsymbol{X} = \boldsymbol{x}) = \frac{f_{\boldsymbol{X}}(\boldsymbol{x}|Y = y_i)P(Y = y_i)}{f_{\boldsymbol{X}}(\boldsymbol{x})}. \tag{3.11}$$

Since the classification may have error and different error could have different penalty, to reflect the cost of each decision on the optimization target, a more general formulation is weighted Bayes decision. We can define $W_{j,i}$ as the weight for the case that real label is y_j while the prediction is y_i. Usually, when $i = j$, $W_{j,i}$ is positive, which means reward. When $i \neq j$, $W_{j,i}$ is negative, which means penalty. Therefore, the best prediction is the y_i that maximize the expected reward or minimize the expected penalty.

$$\hat{Y} = \arg \max_{y_i \in \{y_1, y_2, y_3, \ldots, y_M\}} \sum_{j=1}^{M} P(Y = y_i|\boldsymbol{X} = \boldsymbol{x})W_{j,i} \tag{3.12}$$

In our application, four very useful value of $\hat{Y}$ are CU skip, CU termination, RDO skip and unsure decision. The weight could be the average RD-cost loss due to errors. However, selection of those design should depend on the actual effectiveness in the test. What's more, if we use multiple DMs, for a single DM, there may be only one decision with very high precision will be targeted.

3.4.2 Support Vector Machine

In most cases, it is hard to find a strong feature that can effectively discriminate the data. What is more common is that numbers of weak features are available [35]. Each feature only contains small part of information about the decision. If their information are not

totally overlapped, they all have value to contribute to the classification. In this case, the difficulty is how to combine those weak features to form a strong classifier, knowing that an improper processing of extra information can make the result even worse. As the most direct way, one solution is to keep the dimensionality and represent each data sample as a point in a N-dimensional space. Then a decision boundary can be trained as the classifier [22]. If this is the case, Bayes model may show its limitation. Even though it is a nice theory, it may need large amount of data to support the convergence of high-dimensional probabilistic model [2].

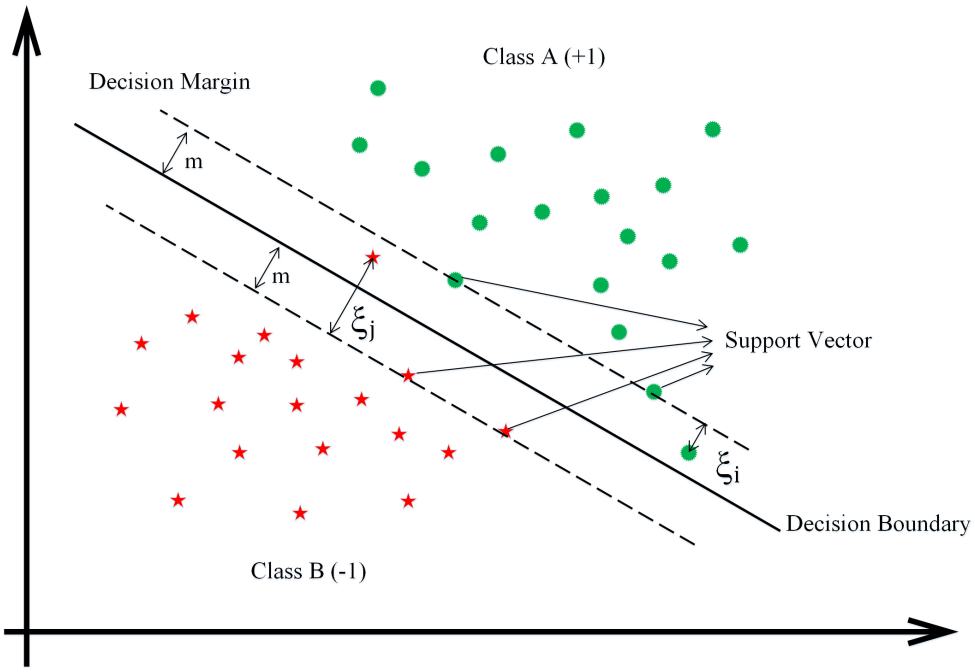

Figure 3.6: Demonstration of Support Vector Machine

To handle multidimensional data, Support Vector Machine(SVM) is an reasonable option. It is boundary-based, which uses hyperplanes to separate the data points in the space. The best hyperplane is chosen as the one that minimized the systemic error. To be exact, if one hyperplane is used to separate two classes of data points, the optimal one should stand in the middle of two classes and keep the same distance with the closest points. Those points that are closest to the decision boundary are called support vectors. Also the hyperplane can be uniquely decided by support vectors chosen from two classes.

To introduce the formulation of SVM, we first consider the two-class classification

where label $y \in \{-1, +1\}$. Let $\boldsymbol{x} \in R^N$ denote feature point in N-dimensional space. The hyperplane can be represented as equation 3.13, where $\boldsymbol{w}$ is the hyperplane's coefficient vector and b is the offset.

$$\boldsymbol{w}^T \boldsymbol{x} + b = 0 \tag{3.13}$$

Because $\boldsymbol{w}$ can has arbitrary magnitude, we normalize this expression by

$$\begin{cases} \boldsymbol{w}^T \boldsymbol{x}^+ + b = +1 \\ \boldsymbol{w}^T \boldsymbol{x}^- + b = -1 \end{cases} \tag{3.14}$$

$\boldsymbol{x}^+$ denotes the positive support vector and $\boldsymbol{x}^-$ is the negative support vector. If we use m to represent the distance from one support vector to the hyperplane. The optimization target is to maximize the decision margin $2m$.

$$2m = \boldsymbol{w}^T (\boldsymbol{x}^+ - \boldsymbol{x}^-) / \|\boldsymbol{w}\| \tag{3.15}$$

from the normalization, we have

$$2m = \frac{2}{\|\boldsymbol{w}\|} \tag{3.16}$$

To maximize $2m$ is equivalent to minimize $\|\boldsymbol{w}\|/2$. What's more, minimizing $\|\boldsymbol{w}\|/2$ has same result with minimizing $\|\boldsymbol{w}\|^2/2$. Hence, the optimization problem can be expressed as

$$\min_{\boldsymbol{w},b} \frac{1}{2} \boldsymbol{w}^T \boldsymbol{w} \tag{3.17}$$

subject to

$$y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1 \geq 0 \quad \forall (\boldsymbol{x}_i, y_i) \in S_t \tag{3.18}$$

, where S_t is the training set that contains feature-label pairs. When the $y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1$ equal to 0, it means the current $\boldsymbol{x}_i$ is support vector. Then Lagrange multiplier can be used to combine the constraint into the minimization[4].

$$\min_{\boldsymbol{w},b,\lambda} \{\frac{1}{2} \boldsymbol{w}^T \boldsymbol{w} - \sum_{i=1}^{|S_t|} \lambda_i [y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1]\} \tag{3.19}$$

Since $y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1 \geq 0$, one special case is to set λ_i to infinity and the minimization is achieved. To avoid this meaningless result, the problem should be reformed as

$$\min_{\boldsymbol{w},b} \max_{\lambda \geq 0} \{\frac{1}{2} \boldsymbol{w}^T \boldsymbol{w} - \sum_{i=1}^{|S_t|} \lambda_i [y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1]\} \tag{3.20}$$

Then one solution is to set $\lambda_i = 0$ for those $(\boldsymbol{x}_i, y_i)$ that meet $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 > 0$. Therefore $\lambda_i > 0$ only when the $\boldsymbol{x}_i$ is support vector. The problem can be solved by Karush-Kuhn Tucker, and $\boldsymbol{w}$ can be represent as

$$\boldsymbol{w} = \sum_{i=1}^{|S_t|} \lambda_i y_i \boldsymbol{x}_i \tag{3.21}$$

This solution of $\boldsymbol{w}$ can be substituted into objective function 3.20 and we have

$$\max_{\lambda}\{\sum_{i=1}^{|S_t|} \lambda_i - \frac{1}{2}\sum_{i=1}^{|S_t|}\sum_{j=1}^{|S_t|} \lambda_i\lambda_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j\} \tag{3.22}$$

subjected to

$$\sum_{i=1}^{|S_t|} \lambda_i y_i = 0, \lambda_i \geq 0 \tag{3.23}$$

After the best λ been decided, the $\boldsymbol{w}$ is calculated by equation 3.21, and the b can be calculated as

$$b = y_s - \boldsymbol{w}^T \boldsymbol{x}_s \tag{3.24}$$

where x_s is the support vector with label y_s. So far, the hyperplane can be decided. While for new feature vector $\boldsymbol{x}_{new}$ in the testing set, the rule for classification is

$$\begin{cases} f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b \\ y_{new} = sign(f(\boldsymbol{x}_{new})). \end{cases} \tag{3.25}$$

The formulation above is the simplest case of SVM. In most application, the data can not be perfectly separated by a linear boundary. One solution for non-separable cases is soft-margin SVM that allows some errors in the training. Also introducing of soft-margin can enlarge the decision margin and make the model more robust to noise. Another solution is to map the data point $\boldsymbol{x}$ into higher-dimensional space by using nonlinear mapping function $\phi(\boldsymbol{x})$, which can make SVM a non-linear classifier. What's more, similar with what we mentioned in the Bayes decision, different error may have different penalty. Therefore, in a more versatile case, the classification function becomes

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\phi(\boldsymbol{x}) + b \tag{3.26}$$

and the optimization becomes

$$\min_{\boldsymbol{w},b}\{\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C(W^+\sum_{i=1}^{|S_t^+|}\xi_i + W^-\sum_{j=1}^{|S_t^-|}\xi_j)\} \tag{3.27}$$

subject to

$$y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i)+b) \geq 1-\xi_i, and\ \xi_i \geq 0 \quad \forall(\boldsymbol{x}_i, y_i) \in S_t \tag{3.28}$$

where C is the penalty parameter and ξ_i is the slack variable. There are several definition for ξ_i in current SVM algorithms [12][7]. A typical definition is

$$\xi_i = \max(1-\boldsymbol{w}^T\phi(\boldsymbol{x}), 0) \tag{3.29}$$

After solved the optimization problem, the $\boldsymbol{w}$ is

$$\boldsymbol{w} = \sum_{i=1}^{|S_t|} \lambda_i y_i \phi(\boldsymbol{x}_i) \tag{3.30}$$

Because the calculation of $\phi(\boldsymbol{x})$ is usually computational expensive, a more practical way is to use kernel trick to meet the same purpose. From the optimization process, we can find that only $\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$ need to be calculated, while it is not necessary to calculate $\phi(\boldsymbol{x}_i)^T$ and $\phi(\boldsymbol{x}_j)$ individually. For some nonlinear mapping function $\phi(\boldsymbol{x}_i)$, the calculation of $\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$ can be much more efficient than that of $\phi(\boldsymbol{x}_j)$. Therefore, kernel functions,

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) =< \phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2) > \tag{3.31}$$

,are defined as an equivalent solution for nonlinear mapping. There are many choices for the kernel. A popular one is radial basis function(RBF), which is suitable for data that is Gaussian distributed[17].

$$k_{RBF}(\boldsymbol{x}_1, \boldsymbol{x}_2) = e^{-\gamma(\boldsymbol{x}_1-\boldsymbol{x}_2)^2} \tag{3.32}$$

And the decision function for nonlinear SVM becomes:

$$\begin{cases} f(\boldsymbol{x}) = \sum_{i=1}^{|S_t|} \lambda_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b \\ y_{new} = sign(f(\boldsymbol{x}_{new})). \end{cases} \tag{3.33}$$

3.5 Flexible Train-Test Period

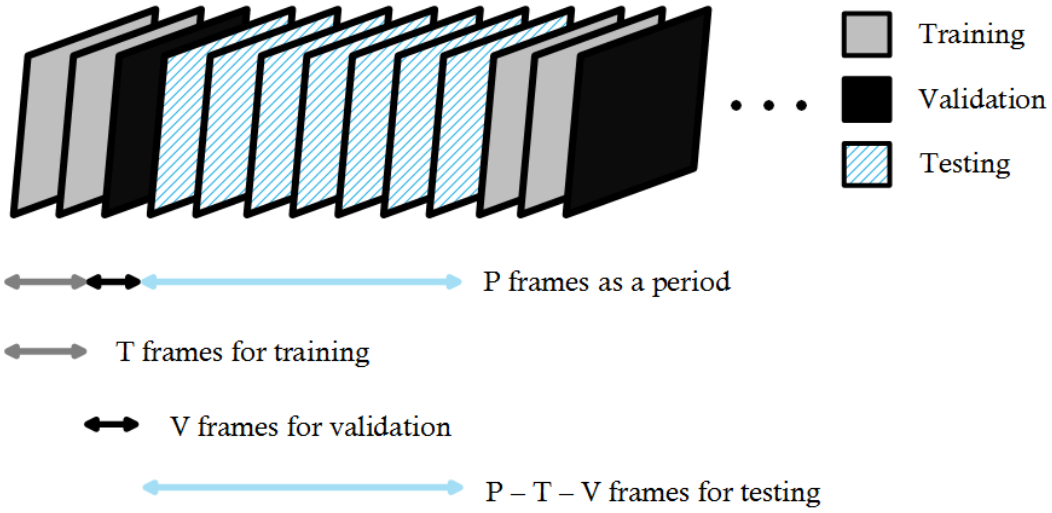

Figure 3.7: Demonstratiion of Train-Test Period

After known how to assess the performance, we then discuss the periodical structure that classification models are trained and applied. Data extracted from the video sequences are divided into 3 subsets: training set, validation set and testing set. Machine learning models are generated from training set to predict testing set, and validation set is used to tuning parameters and switch models so that effectiveness and robustness of the algorithm can be ensured.

Because the property of video data can change with time, it may not be suitable to train a model and use it for a long time. To avoid error propagation and performance degradation caused by motion and scene changes, models are trained, validated, and tested periodical. This periodical structure is illustrated in figure 3.7.

As shown in the figure 3.7, the period is defined as P frames. The first T frames, filled with grey, are used for extracting training data. Features needed by all models and the labels will be saved, while the encoding process is not affected. After that, another V frames will be used to validate effectiveness of each model. In the validation stage, all models trained from training stage will be applied to predict the label. But their decisions are only used to calculate their performance measures rather than adopted by the encoder. Since some models may have bad performance due to ineffective features,

sample-insufficiency or over-fitting, those decisions with unsatisfactory performance will be disable in the testing stage until a new period begin. At the rest $P - T - V$ frames, the encoding will enter the testing stage, where effective decision predicted models will be used to guide the encoder to achieve a faster coding speed. After that, at the beginning of another P frames, the encoder will clear all the models and repeat the same process. In the main version of our algorithm, the P is set to be 60 while the T and V are set to be 2 and 1 respectly.

Another design is to flexibly balance the number of sample for the training. Table 3.3 has shown the number of sample per frame for different video resolution. For low resolution data, the number of sample is very small. For example, when the resolution is 416×240, there is only 18 samples available for depth 0. While for high resolution data, the number of sample can be too large for large depth. For 2560×1600 videos, there are 64000 samples on depth 3.

Table 3.3: Number of sample per frame for different video resolution

Resolution	Depth 0	Depth 1	Depth 2	Depth 3
2560×1600	1000	4000	16000	64000
1920×1080	480	1980	8040	32400
1024×768	192	768	3072	12288
832×480	91	390	1560	6240
416×240	18	91	390	1560

If there is not enough data for the training, the model can not be effective. This problem is especially important for Bayes Decision. Later on , we will show the experimental result that the Bayes decision totally lose its effectiveness when applied to low resolution data. On the other hand, if there is too much data for the training, the training time will increase, which is desired in our application.

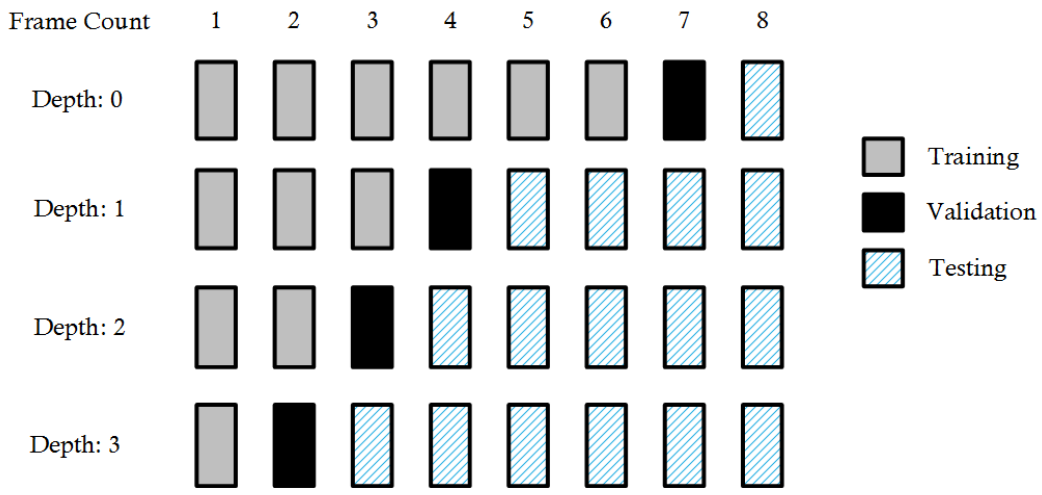

Figure 3.8: Flexible Train-Test Period

Therefore, the train-test period are setting separately for different depth. For depth 0 and depth 1, because the number of sample is relatively small, the the number of training frames could be more than that of depth 2 and depth 3. This flexible train-test period design has been shown in figure 3.8. In this example, the is number of training frames for depth 0 and depth 1 are set to be 6 and 3 respectively. While the number of training frames for depth 2 and depth 3 are only 2 and 1.

3.6 Validation Frame and Model Switch

The periodical learning structure have solved the problem of performance degradation along with the changing of input data. The purpose of validation frame is to keep the performance under control and also choose the best model for current data input. Since we have introduced different ML models, like Bayes Model and SVM , each of them with different parameters may have a possibility to be the best for particular input data. Therefore, to achieve better classification performance, one reasonable strategy is model switch. In the first stage of validation, multiple models will be applied in prediction and their performance will be calculated. After that, the model with the best performance will be chosen to be the dominated decision maker. On the other hand, for those models that can't predict the decision well in the validation frame, we have defined a threshold as the minimum requirement for performance. If the performance of a model is less than the threshold, their prediction will not be adopted in the following testing frames.

3.7 Summary

In this chapter, the proposed machine learning based HEVC fast intra algorithm is introduced. We have summarised a general model of fast algorithm and then given a specific logic structure of our algorithm. The performance measure of classification like accuracy, precision, and sensitivity are defined to judge the effectiveness of classification models. While encoding time reduction and BD-rate are introduced to measure the overall performance. LPTCM is used as an image understanding method to detect the significantly large value of DCT coefficient. Those outliers are then processed to form the feature vector that used to train the classification models and make CU decision. One typical feature SBOC that used in our method is intoduced in detail. In the section of model training, we have discussed the Bayes Decision and SVM. Furthermore, to ensure the robustness of the algorithm, we have meta-algorithm like flexible train-test period and model switch.

Chapter 4

Experiments and Analysis

4.1 Observations on Encoding Time

4.1.1 Potential of Time Reduction

Before starting designing the algorithm to decide the CU partition, we have a strong wish of knowing the potential of this CU decision based fast algorithms and we have been curious about one question. The question is how much computational cost the encoder has spent on exploring the best CU decision. In other word, if a DM can achieve 100% precision as well as sensitivity, how much time reduction can it achieves.

Therefore, an experiment has been done to answer this question. Through encoding the video sequences by the normal encoder, all information about the CU partition has been stored in a file called CU Decision Map (CDM). Then, assisted by the CDM, the sequences will be encoded in the second pass. In the second pass, rather than exploring the best CU partition by exhaustive search, the encoder will read the best CU decision directly from the CDM. Therefore, the time of the second pass encoding has shown the minimum computational cost when the best quad-tree is given in advance. The result of the this experiment has been shown in table 4.1. The result is generated from encoding result of standard video sequences provided by HEVC document. The encoder version is $HM16.3$. Depends on the application and property, those sequences has been categorized into six classes. Class A are down-sampled 4K×2K video sequences with resolution 2560×1600. Class B are the traditional High Definition (HD) 1080P videos with resolution 1920×1080. Class C and class D are videos with size 832×480 and 416×240 respectively, which are resolutions standardized by Wide Video Graphics Array (WVGA) and Wide Quarter Video

Table 4.1: Encoding time reduction in each class when CU decisions are given

Class	Resolution	TS LowerBound
Class A	2560 × 1600	73.76%
Class B	1920 × 1080	69.36%
Class C	832 × 480	56.68%
Class D	416 × 240	52.28%
Class E	1280 × 720	72.08%
Class F	mixed	61.86%

Graphics Array (WQVGA). Class E are 720P video conference records while class F are screen content videos. In this experiment, the time reduction compared with the normal HM has been calculated for each class. It can be noticed that, the time saving is around 70% for high resolution video, like class A, B, and E. However, it is only 50% time can be saved for low resolution data like class D.

4.1.2 Encoding Time Distribution

Because the number of CU candidates for searching in the recursion is increasing exponentially with depth while the size of CU are decreasing with depth, CU skip and CU termination in different depths may have different potential for time reduction. To focus our efforts on the most promising parts, we has done an experiment to explore how much time the encoder spends on different depths and functions. This result is illustrated in table 4.2, where the numbers are generated by averaging the percentages of encoding time over 24 standard test sequences. The encoder version used to run the test is $HM16.3$.

Table 4.2: Encoding time spent in different Depths and Functions

Functions	Encoding Time
CompressCU on Depth 0 + Other Overhead	13%
CompressCU on Depth 1	11%
CompressCU on Depth 2	14%
CheckRDcost2Nx2N on Depth 3	21%
CheckRDcostNxN on Depth 3	41%

From the table, it can be seen that time spent on depth 0 , 1, and 2 are very similar, around 10% on each depth. However, encoding time on depth 3 is over 60% and about

40% is used to run function CheckRDCostNxN. Function CheckRDCostNxN is called only when the CU reaches the minimum size (8×8), where the 8×8 PU is still allowed to split into four 4×4 PUs. However, except for videos with very complex content, only the minority of video data will be encoded into smallest block size. If those minority CUs can effectively detected be DMs, significant time reduction can be achieved. Therefore, decisions on depth 3 could be more important than that on other depths.

Another observation is that CU termination, in all depths, will achieve much more time reduction than the CU skip. For example, if a homogeneous CU on depth 0 is terminated, up to 87% time can be saved. While, if a complex CU is skipped on depth 0, this decision can only save less than 13% of the time. Later on, we will show that these observations work as important inspiration for us to improve fast intra algorithms.

4.2 Observations on Classification Model

4.2.1 Result of Bayes Decision

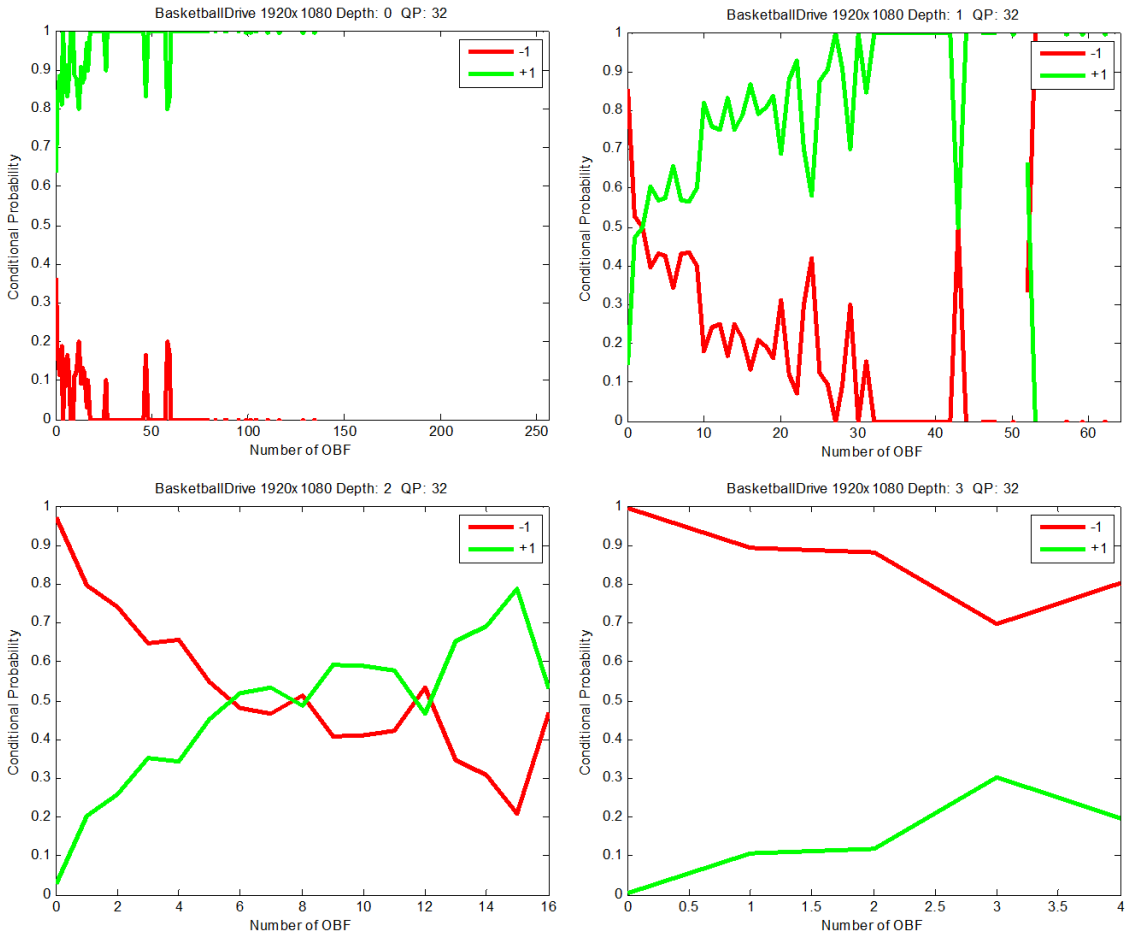

Figure 4.1: Conditional Probability in Bayes Model

In this subsection, we will first provide the classification result of Bayes method to get some insight about the decision maker. Figure 4.1 shows the conditional probability of Bayes Model in each depth, and the feature is chosen as N_{OBF}, which is the number of SBOC [15]. It is a single number that equals to the summation of all OBF within a CU.

The green line shows the estimated probability of CU skip(+1) when the $N_{OBF} = x$

$$P(Y = +1|N_{OBF} = x) = \frac{N(Y = +1, N_{OBF} = x)}{N(N_{OBF} = x)} \tag{4.1}$$

The red line shows, given the N_{OBF}, the probability of CU termination.

$$P(Y = -1|N_{OBF} = x) = \frac{N(Y = -1, N_{OBF} = x)}{N(N_{OBF} = x)} \tag{4.2}$$

Because Bayes method need relatively more training sample, to provide enough training sample for the Bayes model, the data we choose is extracted from the first two frame of high resolution sequence BasketballDrive1920 × 1080. It can be seen from the figure 4.1 that, for depth 0, the the green line is always above the red line. In this case, all CU in depth 0 will be classified as CU skip. The same thing happened in depth 3, where all CU are classified as CU termination and the decisions have lost the diversity. In terms of the classification performance, the precision and sensitivity of Bayes method are given in table 4.3. The sensitivity of CU skip in depth 0 and the sensitivity of CU termination in depth 3 are all zero, which is consistent with that showing in figure 4.1. As we mentioned in the introduction of validation frame, to control the BD-rate increase, all decisions are managed by a precision threshold. In our main version, this threchold is set to be 80%. In this case, only CU skip in depth 0 and CU termination in depth 1,2,3 will be switched on because their precision is larger than 80%.

Table 4.3: Performance Example of Bayes Decision

Model:Bayes	Sequence:BasketballDrive		QP:32	
	CU Skip		CU Termination	
Depth	Precision	Sensitivity	Precision	Sensitivity
0	84.4	100	0	0
1	71.3	70.1	82.1	82.9
2	57.8	23.3	90.6	97.7
3	0	0	97.7	100

When we set period parameter to be $T = 2$, $V = 1$, $P = 60$ and encode the first 60 frames of BasketballDrive1920 × 1080 by this Bayes method, the time reduction is 67% while the BD-rate increases by 3.75%. From the result in this example, it shows that, when quantizing the outliers into flags and summing them up, many information has been lost and the N_{OBF} itself don't have enough discriminative power to separate two classes.

If without the validation frame, the encoder will suffer more BD-rate increase and this problem can be more significant for low resolution videos, like class D (416×240), where the training sample are insufficient.

In [15], the author has used the RD-cost loss due to miss-classification as the weight for the Bayes model. Therefore the decision function becomes

$$\hat{Y} = \arg\min_{i} P(Y = \bar{i}|N_{OBF} = x)W_{x,i,\bar{i}} \tag{4.3}$$

, where $W_{x,i,\bar{i}}$ is the average RD-cost loss when the prediction is i while the true label is $\bar{i}$ given $N_{OBF} = x$. In [15], this weight $W_{x,i,\bar{i}}$ is learned from the first two frames of the input sequence. Table 4.4 has shown the classification result of this weighted Bayes method. In depth 1 and 2, the precision of CU termination has increased slightly. But for the depth 0 and 3, the decision maker still loses the diversity of decisions.

Table 4.4: Performance Example of Weighted Bayes Decision

Model:Weighted Bayes	Sequence:BasketballDrive		QP:32	
	CU Skip		CU Termination	
Depth	Precision	Sensitivity	Precision	Sensitivity
0	84.4	100	0	0
1	68.6	74.7	83.6	79
2	50.5	42.3	92.5	94.5
3	0	0	97.7	100

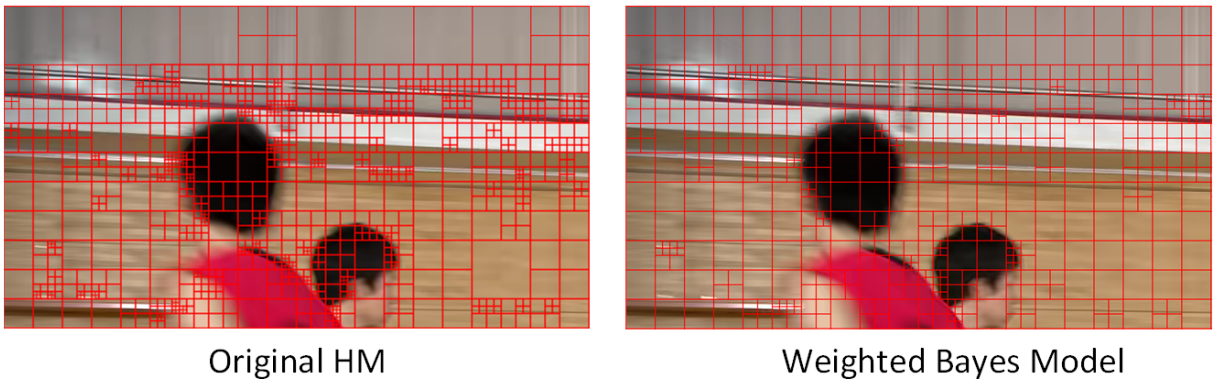

Figure 4.2: CU decision comparison between original HM and Weighted Bayes Model

In figure 4.2, we have made a comparison about the CU partition between original HM and the weighted Bayes method. For result of original HM, it contains blocks of all size from 64×64 to 4×4. However, on the right-hand side, since the Bayes method has classified all CU in depth 0 into +1 and all CU in depth 3 into -1, there is no 64×64 and 4×4 block in the encoding result.

4.2.2 Result of SVM

From the result of Bayes method, we can find out that classifying the CU decision is not a nice and clean problem that can be solved perfectly. Under the constrain of information limitation and computational complexity, the target of the decision makers is to catch those precise decisions as many as they can. Hence, two linear SVM with different purposes have been trained to fulfill this task.

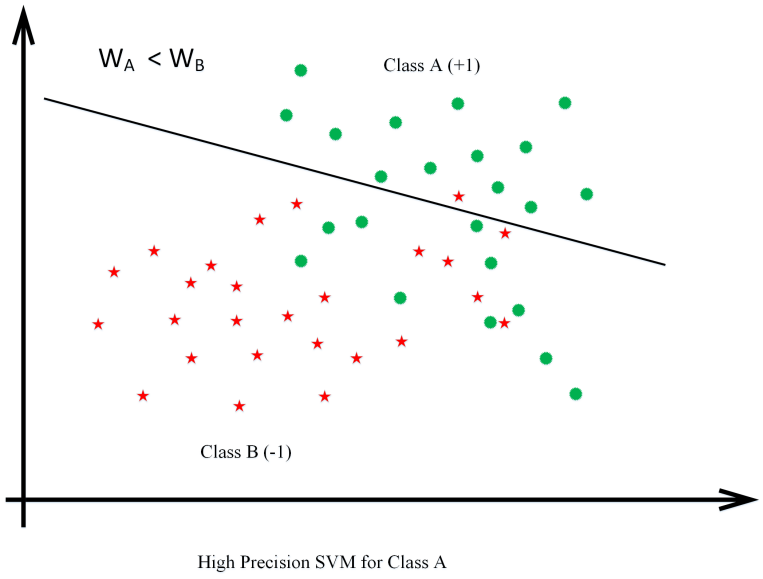

Figure 4.3: Demonstratiion of Weighted SVM1

By setting the W^- larger than W^+, SVM1 can be trained to have high precision of decision +1. This rationale has been illustrated in figure 4.3. When the W^- is large, the false positive error will cause more penalty in the objective function. Hence the decision boundary will more close to the positive class to alleviate the penalty. Usually, the sensitivity of positive decision will decrease in this case. But, in out application, the precision

should be put in the first place. If the precision can't meet the requirement, even the sensitivity is high, the corresponding decision will be switched off by the performance control and will not make contribution for the time reduction. In the main version of our proposed

Table 4.5: Performance Example of SVM1

Model:SVM1	Sequence:BasketballDrive		QP:32	
	CU Skip		CU Termination	
Depth	Precision	Sensitivity	Precision	Sensitivity
0	95	65.4	29.5	80.8
1	100	0.4	61.9	100
2	0	0	88.3	100
3	0	0	97.7	100

algorithm, the parameter setting for SVM1 is $C = 0.000002$, $W^+ = 1$, $W^- = 20$. The feature used in SVM is the SBOC we introduced in chapter 3. The classification performance of SVM1 is given in 4.5. Another model, SVM2, is trained with same feature but to achieve high precision for CU termination. This has been demonstrated in figure 4.4

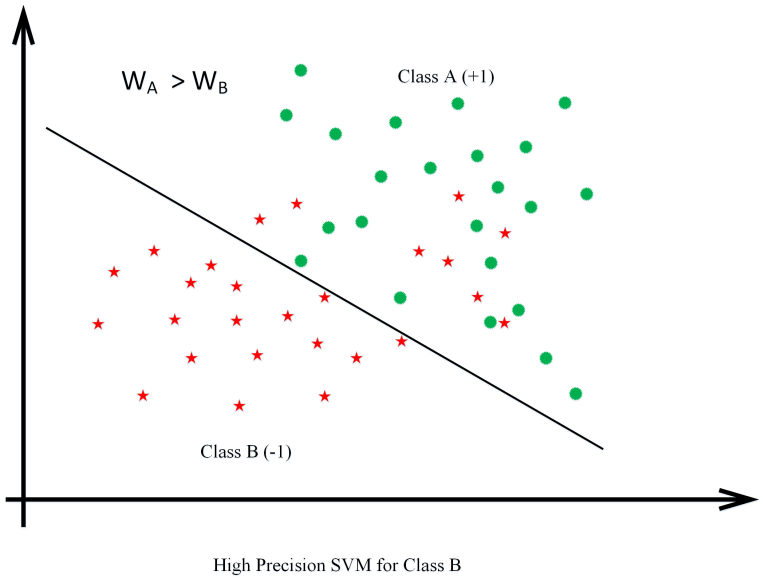

Figure 4.4: Demonstratiion of Weighted SVM2

For SVM2, the W^+ is larger than W^- to avoid too many false negative errors. The

parameter setting for SVM2 is $C = 0.000002$, $W^+ = 50$, $W^- = 1$. And the classification performance has been shown in table 4.6.

Table 4.6: Performance Example of SVM2

Model:SVM2	Sequence:BasketballDrive		QP:32	
	CU Skip		CU Termination	
Depth	Precision	Sensitivity	Precision	Sensitivity
0	93.7	69	30	74
1	68.8	74.9	83.6	79
2	38.8	84.9	97.6	82.3
3	13.6	83.9	99.6	87.5

It can be seen from the result that, SVM1 have better CU skip precision than that of SVM2 while SVM2 have higher CU termination precision than SVM1. Typically, the decison maker will trust the SVM1 about CU skip decision and trust the SVM2 about the CU termination decision. For general video sequences, the situation can be more complex. Therefore, in our algorithm, the strategy is to switch between SVM1 and SVM2 based on their performance in validation frame.

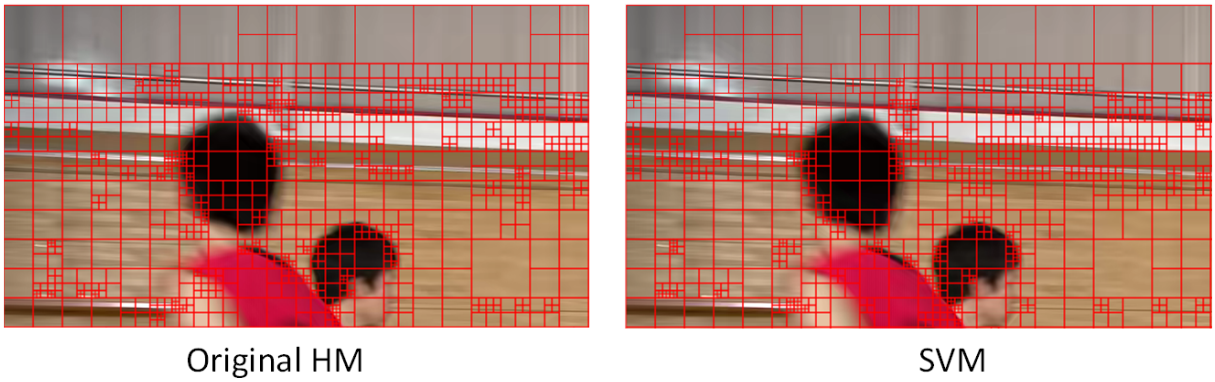

Figure 4.5: CU decision comparison between original HM and SVM

Similar as what we do for Bayes method, we have compared the CU decision result of the proposed SVM based method with that of original HM. It can be seen that the predicted partition result is very close to that of normal encoder. Unlike the Bayes method, the

proposed method can well retain the diversity of different block size and achieve better precision.

4.3 Full Test and Comparison

4.3.1 Test Conditions

In this section, we will test the proposed algorithm from the perspective of encoding. A professional experiment has been done to test the performance of several fast algorithms which includes our proposed method and some high-performance benchmarks. The result of normal HM is first generated as reference. Then encoding results of fast algorithms are compared with the reference in terms of BD-rate and encoding time reduction. To make a more fair comparison, each fast algorithm is compared with its base version of HM. For example, our fast algorithm is implemented on $HM16.3$, so it will be compared with normal $HM16.3$. The configuration of encoders is chosen as AIM that follows the common test condition provided by official HEVC document [5]. Four QP values in the test condition are chosen as 22, 27, 32, and 37.By default, our standard results are generated from the whole test sequences.

In terms of test environment, the computer that is used to generate our experimental results is of Intel(R) Core(TM) i7-4790 3.60GHz CPU, and 8GB RAM. The operation system is 64 bits Windows 7 Enterprise.

4.3.2 Full Test Result

In the HEVC's official document, there are totally 24 test sequences, which are categorized into 6 classes. They are selected to cover different video resolutions and contents in wide range of applications. Since the time reduction and BD-rate are showing the similar property within each class, the full test result will be analyzed in the order of classes.

The result of Class A and class B are illustrated in table 4.7. Class A are down-sampled 4K×2K video sequences with resolution 2560×1600. Class B are the traditional high-definition(HD) 1080P videos with resolution 1920×1080. From the table 4.7, we can see that the proposed algorithm can achieve very significant time reduction for high resolution video. The average time reduction for class A is 54.76% with only 0.53% BE-rate increase. For class B, the TS is 48.90% while the BD-rate increase is 0.78%. The maximum time reduction is from NebutaFestival, which is 64.05% with only 0.24% BD-rate increase.

This may because the edges pattern in this video is relatively simple and the motion is very slow. Besides the BD-rate, it also can be seen that the Peak Signal-to-noise Ratio (PSNR) loss caused by the fast algorithm is very small, which is only 0.01 dB. This property make our algorithm more suitable for the application that requires fast speed and high video resolution or quality.

Table 4.7: Encoding Performance of Class A and Class B

Class	Sequence	Δ Bitrate(%)	Δ PSNR(dB)	BD-Rate	TS
Class A	PeopleOnStreet	0.40%	-0.0181	0.73%	48.23%
	Traffic	0.47%	-0.0144	0.90%	49.06%
	SteamLocomotiveTrain	0.00%	-0.0099	0.26%	57.70%
	NebutaFestival	0.28%	0.0021	0.24%	64.05%
Class Average		0.29%	-0.0101	0.53%	54.76%
Class B	Kimono1	0.19%	-0.0169	0.73%	57.84%
	ParkScene	0.16%	-0.0157	0.63%	37.43%
	Cactus	0.41%	-0.0115	0.88%	45.22%
	BascketballDrive	0.32%	-0.0125	0.81%	52.08%
	BQTerrace	0.58%	-0.0059	0.87%	51.94%
Class Average		0.33%	-0.0125	0.78%	48.90%

Class C and class D are videos with size 832×480 and 416×240 respectively, which are resolutions standardized by WVGA and WQVGA. The encoding performance of class C and class D are given in table 4.8.

Table 4.8: Encoding Performance of Class C and Class D

Class	Sequence	Δ Bitrate(%)	Δ PSNR(dB)	BD-Rate	TS
Class C	BasketballDrill	0.31%	-0.0256	0.90%	43.49%
	BQMall	0.68%	-0.0199	1.14%	43.64%
	PartyScene	0.36%	-0.0094	0.40%	38.93%
	RaceHorsesC	0.58%	-0.0102	0.86%	43.03%
Class Average		0.48%	-0.0163	0.83%	42.27%
Class D	BasketballPass	0.46%	-0.0216	0.97%	43.96%
	BQSquare	0.91%	-0.0121	1.23%	47.90%
	BlowingBubbles	0.20%	-0.0113	0.32%	35.73%
	RaceHorsesD	0.60%	-0.0093	0.82%	41.00%
Class Average		0.54%	-0.0136	0.83%	42.15%

In our observation on the time reduction lower bound, the time reduction potential is relatively smaller than that of high resolution video. This is also shown in our test result. The average time reduction in class C is only 42.27% and the BD-rate is 0.83%. Class D has shown a very similar result: 42.15% TS and 0.83% BD-rate increase.

Table 4.9: Encoding Performance of Class E and Class F

Class	Sequence	Δ Bitrate(%)	Δ PSNR(dB)	BD-Rate	TS
Class E	Johnny	0.30%	-0.0081	0.53%	52.37%
	KristenAndSara	0.54%	-0.0112	0.88%	55.39%
	FourPeople	0.27%	-0.0237	0.81%	47.59%
Class Average		0.37%	-0.0143	0.74%	51.78%
Class F	BasketballDrillText	0.22%	-0.0316	0.85%	43.88%
	ChinaSpeed	0.65%	-0.0196	0.84%	54.17%
	SlideEditing	0.90%	-0.1364	1.91%	45.08%
	SlideShow	0.17%	-0.0146	0.31%	52.96%
Class Average		0.49%	-0.0505	0.98%	49.02%

Class E are 720P video conference records while class F are screen content videos. The result of class E and class F have been shown in 4.9. Screen content can be very challenging

since the edges in the this kind of video is very sharp and the pattern can be very complex, like text. For example, SlideEditing is the screen record of editing PowerPoint slides. In this video there are some all-blank frames, which may cause some risks for online model. Also it can be seen that the PSNR loss in class F is very significant which up to 0.05.

4.3.3 Comparison with Benchmarks

To compare our algorithm with top methods, we have chosen many papers of high quality as our benchmarks. It is no exaggeration to say that IEEE Transaction on Circuits and Systems for Video Technology (TCSVT) has represented the highest level of video coding technology. Therefore we have chosen some relevant works that recently published in TCSVT. Also our benchmarks have included some top conferences like International Conference on Image Processing (ICIP) and Data Compression Conference (DCC). Because, in different papers, results are generated on different machines. To make a fair comparison, we have regenerated some of their results in our computer. However, for those benchmarks that we can't obtain their encoders, we have no choice but to directly refer the results that are reported in their papers. Here, I would like to thank Tao Zhang, Hao Zhang, Biao Min for their sharing the source code or executable files.

Table 4.10: Performance comparison with benchmarks, 600-frames full test

Encoder Version	Time Saving	BD-rate	Source of data
Proposed(MainVersion)	48.03%	0.78%	Through test
[37]H. Zhang, TCSVT 2014(Overall)	58.83%	1.33%	Through test
[37]H. Zhang, TCSVT 2014(CU Size)	45%	0.8%	From paper
[27]B. Min, TCSVT 2015	43.97%	1.16%	Through test
[15]N. Hu, TCSVT 2015	39.42%	0.46%	Through test
[38]T. Zhang, TCSVT 2016	48.15%	0.92%	Through test
[18]M. Khan, ICIP 2013	44.00%	1.27%	From paper
[32]X. Shang, ICIP 2015	37.91%	0.66%	From paper
[11]F. Duanmu, ICIP 2015	36.80%	3.00%	From paper
[25]Y. Liu, DCC 2015	46.50%	2.20%	From paper

Together with our main version algorithm, the full test results of benchmarks has been shown in table 4.10. Our overall result is 48.03% time reduction with 0.78% BD-rate increase. Compared with others' work, the proposed algorithm has outperformed the

algorithms in [28], [18], [32], [11] by both BD-rate and time reduction. Since the overall version in [37] has combined both fast CU size decision and fast IPM selection, it achieves the best time reduction performance. However, in their paper, they also reported the result of micro-version that only fast CU size decision is switched on. The time reduction of their macro-version is 45% TS and 0.8% BD-rate increase.

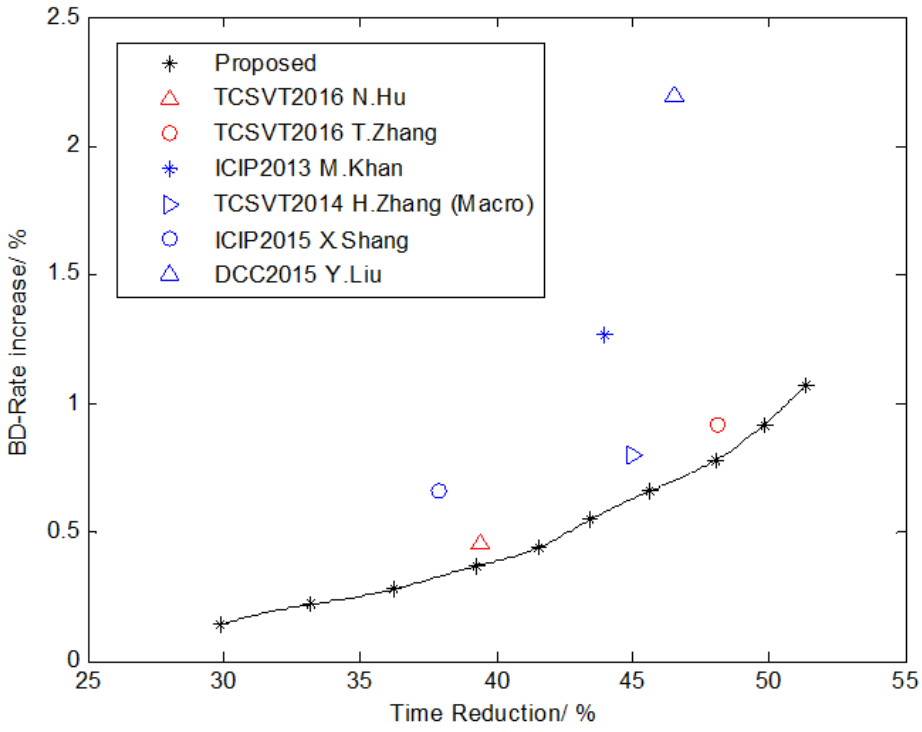

Figure 4.6: Performance Comparison With Benchmarks, Curve of different Trade-off

Because most of the results provided in the literature are only single TS BD-rate points, to make a better comparison, we have generated a performance curve by adjusting the precision threshold. The figure 4.6 has shown the comparison between our performance curve and benchmarks. 10 points are tested to generate this curve, where the precision threshold is changing from 0.76 to 0.94. Each point is obtained from the overall average result of 24 test sequences and the curve is generated by cubic spline data interpolation to form a smooth shape. With the changing of precision threshold, the TS is changing from 51.37% to 29.88% while the BD-rate is changing from 1.07% to 0.14%. For benchmarks,

the red points are full test result that regenerated on our computer, and the blue points are data reported in the literature.

From this curve, we can clearly see that all the benchmark points are above the curve, which means, given the same TS, the point on curve has a smaller BD-rate. For points that are close to the curve, we can always find a black point that have both larger TS and smaller BD-rate. Specific data in figure 4.6 is attached in the appendix. Because the overall result of [37] has combined two fast algorithms and achieved a TS that we can not achieve, it is not plotted in the figure. But, in terms of their CU decision based part, our test result is still better than the performance reported in [37].

4.4 Summary

In this chapter, we have first done an observation on potential time reduction and encoding time distribution. The time reduction potential in CU size decision is very promising, which can be around 70%. In the HM, most of the encoding time has been spent on a function called CompressCU. While, within this CompressCU, encoding time on each depth are different. About 60% encoding time has been spent on the depth 3. Therefore, we have concluded that the focus of the fast CU size decision algorithm should be focused on depth 3.

After that, we have done an experiment to observe the performance of classification model. For Bayes model, the result is unsatisfactory. Due to the lack of training samples, the Bayes Model can't utilize high dimensional data with large variety. However, even when applying single dimensional feature OBF in high resolution video, the Bayes have lost the effectiveness in depth 0 and depth 3. And the classification result in depth 1 and depth 2 are also not good. For SVM, two linear SVMs has been trained to handle different decisions. One is trained to achieve highl precision of CU skip, another is designed to better predict CU termination. By switching between two models, the classification result has increased significantly.

A professional full test has been done to verify the encoding performance of the proposed algorithm and some benchmarks. By generating a curve with different TS BD-rate trade-off, we have clearly shown that our algorithm has out performed all benchmarks in terms of both time reduction and BD-rate. Another advantage of current design is that, the proposed algorithm is only based the original image and CU decision. Therefore is can be easily combined with other fast algorithm like fast IPM selection methods to achieve more time redcuction.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we have proposed a fast algorithm to reduce the computational complexity of HEVC intra-frame coding and maintain the coding efficiency. The proposed algorithm is based on a machine learning framework. The LPTCM is utilized as a feature extraction tool. When applied on AIM full test, the main version algorithm have achieved, on average, 48% time reduction with 0.8% BD-rate increase. Also, through adjusting the setting, the algorithm can change the trade-off between encoding time and BD-rate, which can generate a performance curve for different requirements. The maximum time reduction can be 51.37% with 1.07% BD-rate increase while the minimum BD-rate increase could be 0.14% with 29.88% time reduction. While for Bayes Decision, through our observation, it can be concluded that Bayes Decision is not very suitable for this kind of algorithm. Generally speaking, the Bayes Decision requires more training data than other models. If we extend the training period to obtain more samples, the potential for encoding time reduction will been squeezed. Also the encoding decision is relatively complex that can't be effectively predicted through one-dimensional feature. If the dimensionality increase, the requirement for training sample will also increase, which makes Bayes Decision less effective. Even we can come up with an effective way to combine many information into one dimension, the processing itself will also introduce extra computational cost. Therefore, the Support Vector Machine is chosen as the main decision maker and has achieved a good performance in the test result.

5.2 Application and Future Work

Since the HEVC becomes the newest video standard, our algorithm will have a wide range of application. As one basis of video compression, intra-frame coding has been widely used in all configuration of HEVC. Since the proposed method is all on the software level, assisted by parallel computing and application-specific hardware, the HEVC encoder using our algorithm can achieve more significant time reduction. In some application, the video can even be encoded into all-intra configuration, where the whole video is compressed by intra coding. An typical application of all-intra coding is the rear camera in parking aid systems, where the requirement for real-time display is strict and there is almost not limitation on transmission bandwidth. Another example is the lunar rover. For a lunar rover, because the information collected are very precious while the computational resources are quite limited, video must be recorded with high quality and low energy consumption. In this case, all-intra video coding may be used and a fast intra algorithm is very important.

For fast video coding algorithm, one of our future work could be applying the machine learning methods in inter frame coding, where may have more potential for encoding time reduction. On the other hand, machine learning method can also be used for better prediction, quantization and entropy coding to improve the coding efficiency. Another furture work could be provide more insights and data visualization so that we can understand the property of machine learning better.

In current machine learning methods we used, the training and testing are separated while the models are updated periodically. However, even we don't have a specific solution, the most desirable algorithm should be incremental and semi-supervised learning methods. In that kind of method, as the data keep coming in, the model can be continually updated when partially given some labels. In this case, the model can be more self-adaptive and the training and predicting are progressing at the same time.

References

[1] x.265. In *https://bitbucket.org/multicoreware/x265/wiki/Home*. Web Link, retrieved 2016 Sep.

[2] R Bellman. Dynamic programming: Princeton univ. press, 1957.

[3] Gisle Bjontegaard. Calcuation of average psnr differences between rd-curves. *Doc. VCEG-M33 ITU-T Q6/16, Austin, TX, USA, 2-4 April 2001*, 2001.

[4] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[5] F Bossen and HM Common. test conditions and software reference configurations, jct-vc doc. *L1100, Jan*, 2013.

[6] Frank Bossen, Benjamin Bross, Karsten Suhring, and David Flynn. Hevc complexity and implementation analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1685–1696, 2012.

[7] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[8] Seunghyun Cho and Munchurl Kim. Fast cu splitting and pruning for suboptimal cu partitioning in hevc intra coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(9):1555–1564, 2013.

[9] Guilherme Correa, Pedro A Assuncao, Luciano Volcan Agostini, and Luis A da Silva Cruz. Fast hevc encoding decisions using data mining. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(4):660–673, 2015.

[10] Minh N Do and Martin Vetterli. Wavelet-based texture retrieval using generalized gaussian density and kullback-leibler distance. *IEEE transactions on image processing*, 11(2):146–158, 2002.

[11] Fanyi Duanmu, Zhan Ma, and Yao Wang. Fast cu partition decision using machine learning for screen content compression. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 4972–4976. IEEE, 2015.

[12] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[13] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.

[14] Nan Hu and En-Hui Yang. Erratum to fast mode selection for hevc intra-frame coding with entropy coding refinement based on a transparent composite model.

[15] Nan Hu and En-Hui Yang. Fast mode selection for hevc intra-frame coding with entropy coding refinement based on a transparent composite model. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(9):1521–1532, 2015.

[16] Wei Jiang, Hanjie Ma, and Yaowu Chen. Gradient based fast mode decision algorithm for intra prediction in hevc. In *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, pages 1836–1840. IEEE, 2012.

[17] S Sathiya Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural computation*, 15(7):1667–1689, 2003.

[18] Muhammad Usman Karim Khan, Muhammad Shafique, and Jorg Henkel. An adaptive complexity reduction scheme with fast prediction unit decision for hevc intra encoding. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pages 1578–1582. IEEE, 2013.

[19] Il-Koo Kim, Junghye Min, Tammy Lee, Woo-Jin Han, and JeongHoon Park. Block partitioning structure in the hevc standard. *IEEE transactions on circuits and systems for video technology*, 22(12):1697–1706, 2012.

[20] J Kim and B Jeon. Encoding complexity reduction by removal of n× n partition type. In *Document JCTVC-D087, 4th JCT-VC Meeting, Daegu, Korea (January 2011)*, 2011.

[21] Donald Knuth. *The TEXbook*. Addison-Wesley, Reading, Massachusetts, 1986.

[22] Ludmila I Kuncheva. *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2004.

[23] Jani Lainema, Frank Bossen, Woo-Jin Han, Junghye Min, and Kemal Ugur. Intra coding of the hevc standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1792–1801, 2012.

[24] Leslie Lamport. *LATEX — A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

[25] Yen-Chun Liu, Zong-Yi Chen, Jiunn-Tsair Fang, and Pao-Chi Chang. Svm-based fast intra cu depth decision for hevc. In *Data Compression Conference (DCC), 2015*, pages 458–458. IEEE, 2015.

[26] Dave Marshall. https://www.cs.cf.ac.uk/dave/multimedia/node231.html, relationship between dct and fft. 2001.

[27] Biao Min and Ray CC Cheung. A fast cu size decision algorithm for the hevc intra encoder. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(5):892–896, 2015.

[28] JH Min, S Lee, IK Kim, WJ Han, J Lainema, and K Ugur. Te4: Results for simplification of unified intra prediction. *JCTVC-C042, Guangzhou, China*, 2010.

[29] F Muller. Distribution shape of two-dimensional dct coefficients of natural images. *Electronics Letters*, 29(22):1935–1936, 1993.

[30] Jens-Rainer Ohm, Gary J Sullivan, Heiko Schwarz, Thiow Keng Tan, and Thomas Wiegand. Comparison of the coding efficiency of video coding standardsincluding high efficiency video coding (hevc). *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1669–1684, 2012.

[31] I-Ming Pao and Ming-Ting Sun. Modeling dct coefficients for fast video encoding. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(4):608–616, 1999.

[32] Xiwu Shang, Guozhong Wang, Tao Fan, and Yan Li. Fast cu size decision and pu mode decision algorithm in hevc intra coding. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 1593–1597. IEEE, 2015.

[33] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

[34] Vivienne Sze, Madhukar Budagavi, and Gary J Sullivan. High efficiency video coding (hevc). In *Integrated Circuit and Systems, Algorithms and Architectures*, pages 1–375. Springer, 2014.

[35] Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[36] En-Hui Yang, Xiang Yu, Jin Meng, and Chang Sun. Transparent composite model for dct coefficients: Design and analysis. *IEEE Trans. Image Process.*, 23(3):1303–1316, 2014.

[37] Hao Zhang and Zhan Ma. Fast intra mode decision for high efficiency video coding (hevc). *IEEE Transactions on circuits and systems for video technology*, 24(4):660–668, 2014.

[38] Tao Zhang, Ming-Ting Sun, Debin Zhao, and Wen Gao. Fast intra mode and cu size decision for hevc. 2016.

[39] Liang Zhao, Li Zhang, Siwei Ma, and Debin Zhao. Fast mode decision algorithm for intra prediction in hevc. In *Visual Communications and Image Processing (VCIP), 2011 IEEE*, pages 1–4. IEEE, 2011.

APPENDICES

Appendix A

Detailed Experimental Result

A.1 Time Reduction Lower Bound Given CU Size Decision

A.2 600 Frame Full Test Result of Proposed Main Version

A.3 600 Frame Full Test Result of Benchmarks

A.4 Data in Figure 4.6

Table A.1: Experimental Lower Bound for Encoding Time Reduction

Class	Sequence	TS LowerBound	Class Average
Class A	PeopleOnStreet	62.69%	73.76%
	Traffic	67.07%	
	SteamLocomotiveTrain	83.67%	
	NebutaFestival	81.61%	
Class B	Kimono1	81.12%	69.36%
	ParkScene	66.77%	
	Cactus	66.54%	
	BascketballDrive	71.45%	
	BQTerrace	60.94%	
Class C	BasketballDrill	56.75%	56.68%
	BQMall	59.21%	
	PartyScene	46.11%	
	RaceHorsesC	64.66%	
Class D	BasketballPass	60.08%	52.28%
	BQSquare	48.61%	
	BlowingBubbles	43.87%	
	RaceHorsesD	56.54%	
Class E	Johnny	75.88%	72.08%
	KristenAndSara	73.78%	
	FourPeople	66.57%	
Class F	BasketballDrillText	55.03%	61.86%
	ChinaSpeed	60.95%	
	SlideEditing	54.59%	
	SlideShow	76.85%	

Table A.2: 600 Frame Full Test Result of Proposed Main Version

Class	Sequence	ΔRate(%)	ΔPSNR(dB)	BD-Rate	TS
Class A	PeopleOnStreet	0.40%	-0.0181	0.73%	48.23%
	Traffic	0.47%	-0.0144	0.90%	49.06%
	SteamLocomotiveTrain	0.00%	-0.0099	0.26%	57.70%
	NebutaFestival	0.28%	0.0021	0.24%	64.05%
Class B	Kimono1	0.19%	-0.0169	0.73%	57.84%
	ParkScene	0.16%	-0.0157	0.63%	37.43%
	Cactus	0.41%	-0.0115	0.88%	45.22%
	BascketballDrive	0.32%	-0.0125	0.81%	52.08%
	BQTerrace	0.58%	-0.0059	0.87%	51.94%
Class C	BasketballDrill	0.31%	-0.0256	0.90%	43.49%
	BQMall	0.68%	-0.0199	1.14%	43.64%
	PartyScene	0.36%	-0.0094	0.40%	38.93%
	RaceHorsesC	0.58%	-0.0102	0.86%	43.03%
Class D	BasketballPass	0.46%	-0.0216	0.97%	43.96%
	BQSquare	0.91%	-0.0121	1.23%	47.90%
	BlowingBubbles	0.20%	-0.0113	0.32%	35.73%
	RaceHorsesD	0.60%	-0.0093	0.82%	41.00%
Class E	Johnny	0.30%	-0.0081	0.53%	52.37%
	KristenAndSara	0.54%	-0.0112	0.88%	55.39%
	FourPeople	0.27%	-0.0237	0.81%	47.59%
Class F	BasketballDrillText	0.22%	-0.0316	0.85%	43.88%
	ChinaSpeed	0.65%	-0.0196	0.84%	54.17%
	SlideEditing	0.90%	-0.1364	1.91%	45.08%
	SlideShow	0.17%	-0.0146	0.31%	52.96%
Overoll Average		0.41%	-0.0195	0.78%	48.03%

Table A.3: 600 Frame Full Test Result of Hao Zhang TCSVT2014[37]

Class	Sequence	ΔRate(%)	ΔPSNR(dB)	BD-Rate	TS
Class A	PeopleOnStreet	0.35%	-0.0365	1.02%	56.98%
	Traffic	0.37%	-0.0339	1.05%	59.07%
	SteamLocomotiveTrain	0.03%	-0.0174	0.67%	67.19%
	NebutaFestival	0.08%	-0.0116	0.25%	59.03%
Class B	Kimono1	-0.13%	-0.0252	0.63%	65.57%
	ParkScene	-0.40%	-0.0412	0.61%	58.55%
	Cactus	0.09%	-0.0370	1.17%	58.29%
	BascketballDrive	0.34%	-0.0308	1.36%	61.93%
	BQTerrace	0.60%	-0.0359	1.18%	58.41%
Class C	BasketballDrill	0.81%	-0.0405	1.70%	54.49%
	BQMall	0.17%	-0.0500	1.16%	56.78%
	PartyScene	-0.30%	-0.0827	0.96%	50.81%
	RaceHorsesC	0.03%	-0.0501	0.99%	57.22%
Class D	BasketballPass	0.19%	-0.0614	1.28%	57.55%
	BQSquare	-0.02%	-0.0712	0.96%	50.31%
	BlowingBubbles	-0.31%	-0.0837	1.05%	50.67%
	RaceHorsesD	-0.04%	-0.0638	1.11%	54.79%
Class E	Johnny	1.00%	-0.0387	2.02%	67.04%
	KristenAndSara	0.78%	-0.0403	1.67%	64.99%
	FourPeople	0.49%	-0.0416	1.31%	60.86%
Class F	BasketballDrillText	0.71%	-0.0512	1.66%	53.96%
	ChinaSpeed	1.22%	-0.0881	2.29%	57.94%
	SlideEditing	2.49%	-0.1676	3.79%	55.51%
	SlideShow	1.02%	-0.0818	2.01%	73.91%
Overoll Average		0.40%	-0.0534	1.33%	58.83%

Table A.4: 600 Frame Full Test Result of Min Biao TCSVT2015[27]

Class	Sequence	ΔRate(%)	ΔPSNR(dB)	BD-Rate	TS
Class A	PeopleOnStreet	0.28%	-0.0193	0.62%	45.40%
	Traffic	0.69%	-0.0091	0.84%	46.74%
	SteamLocomotiveTrain	1.63%	-0.0325	3.17%	53.94%
	NebutaFestival	1.86%	-0.1596	4.83%	22.94%
Class B	Kimono1	2.13%	-0.0198	2.71%	50.21%
	ParkScene	0.78%	-0.0066	0.74%	38.75%
	Cactus	0.64%	-0.0085	0.82%	39.42%
	BascketballDrive	0.62%	-0.0173	1.16%	43.19%
	BQTerrace	0.66%	-0.0137	0.82%	34.99%
Class C	BasketballDrill	0.42%	-0.0341	0.98%	44.01%
	BQMall	0.60%	-0.0186	0.89%	44.57%
	PartyScene	0.05%	-0.0102	0.14%	31.66%
	RaceHorsesC	0.62%	-0.0022	0.64%	41.02%
Class D	BasketballPass	0.20%	-0.0230	0.53%	42.14%
	BQSquare	0.19%	-0.0177	0.42%	29.08%
	BlowingBubbles	0.04%	-0.0092	0.11%	27.75%
	RaceHorsesD	0.43%	-0.0162	0.65%	36.61%
Class E	Johnny	1.02%	-0.0357	1.94%	62.04%
	KristenAndSara	0.64%	-0.0320	1.32%	59.42%
	FourPeople	0.39%	-0.0246	0.81%	53.66%
Class F	BasketballDrillText	0.30%	-0.0329	0.77%	43.25%
	ChinaSpeed	0.30%	-0.0517	0.88%	47.19%
	SlideEditing	0.57%	-0.0554	0.94%	47.87%
	SlideShow	0.46%	-0.0595	1.12%	69.43%
Overoll Average		0.65%	-0.0296	1.16%	43.97%

Table A.5: 600 Frame Full Test Result of Nan Hu TCSVT2015[15]

Class	Sequence	Δ RD(%)	Δ PSNR(dB)	BD-Rate	TS
Class A	PeopleOnStreet	0.34%	-0.0079	0.50%	33.37%
	Traffic	0.25%	-0.0055	0.38%	33.20%
	SteamLocomotiveTrain	0.01%	-0.0093	0.27%	46.63%
	NebutaFestival	0.21%	0.0011	0.18%	50.84%
Class B	Kimono1	0.19%	-0.0103	0.52%	45.22%
	ParkScene	-0.01%	-0.0136	0.32%	32.26%
	Cactus	0.15%	-0.0095	0.46%	36.40%
	BascketballDrive	0.23%	-0.0086	0.54%	42.09%
	BQTerrace	0.26%	-0.0031	0.32%	36.05%
Class C	BasketballDrill	0.07%	-0.0181	0.38%	36.78%
	BQMall	0.40%	-0.0044	0.50%	31.50%
	PartyScene	0.09%	-0.0153	0.25%	33.49%
	RaceHorsesC	0.22%	-0.0103	0.45%	35.13%
Class D	BasketballPass	0.27%	-0.0079	0.39%	37.02%
	BQSquare	0.13%	-0.0113	0.24%	34.24%
	BlowingBubbles	0.10%	-0.0221	0.37%	32.17%
	RaceHorsesD	0.18%	-0.0130	0.37%	31.26%
Class E	Johnny	0.51%	-0.0076	0.72%	43.20%
	KristenAndSara	0.43%	-0.0149	0.81%	44.72%
	FourPeople	0.25%	-0.0120	0.45%	35.93%
Class F	BasketballDrillText	0.15%	-0.0173	0.40%	36.10%
	ChinaSpeed	0.40%	-0.0181	0.57%	47.20%
	SlideEditing	0.36%	-0.0137	0.46%	50.60%
	SlideShow	0.76%	-0.0267	1.06%	57.62%
Overoll Average		0.25%	-0.0116	0.46%	39.29%

Table A.6: 600 Frame Full Test Result of Tao Zhang TCSVT2016[38]

Class	Sequence	Δ Rate(%)	Δ PSNR(dB)	BD-Rate	TS
Class A	PeopleOnStreet	0.29%	-0.0306	0.86%	45.72%
	Traffic	0.26%	-0.0244	0.75%	48.12%
	SteamLocomotiveTrain	0.22%	-0.0171	0.83%	59.14%
	NebutaFestival	0.32%	-0.0102	0.47%	53.82%
Class B	Kimono1	0.34%	-0.0124	0.71%	55.75%
	ParkScene	-0.04%	-0.0267	0.58%	44.75%
	Cactus	-0.07%	-0.0237	0.61%	49.03%
	BascketballDrive	0.13%	-0.0186	0.75%	53.95%
	BQTerrace	0.07%	-0.0212	0.49%	46.24%
Class C	BasketballDrill	0.07%	-0.0265	0.63%	37.65%
	BQMall	0.24%	-0.0275	0.76%	46.47%
	PartyScene	-0.02%	-0.0389	0.57%	34.63%
	RaceHorsesC	-0.05%	-0.0299	0.51%	46.04%
Class D	BasketballPass	0.19%	-0.0300	0.69%	43.97%
	BQSquare	-0.06%	-0.0409	0.52%	37.07%
	BlowingBubbles	-0.17%	-0.0366	0.41%	30.31%
	RaceHorsesD	0.06%	-0.0299	0.55%	40.46%
Class E	Johnny	0.60%	-0.0253	1.26%	60.08%
	KristenAndSara	0.48%	-0.0318	1.12%	58.41%
	FourPeople	0.28%	-0.0340	0.91%	51.80%
Class F	BasketballDrillText	0.25%	-0.0344	0.88%	38.72%
	ChinaSpeed	0.58%	-0.0723	1.46%	51.04%
	SlideEditing	1.35%	-0.1414	2.34%	51.68%
	SlideShow	1.55%	-0.1601	3.44%	70.77%
Overoll Average		0.29%	-0.0393	0.92%	48.15%

Table A.7: Data in Figure 4.6, Full Test Result of Proposed Method

Precision Threshold	TS	BD-rate
0.76	51.37%	1.07%
0.78	49.83%	0.92%
0.80	48.03%	0.78%
0.82	45.60%	0.66%
0.84	43.45%	0.55%
0.86	41.53%	0.44%
0.88	39.26%	0.37%
0.90	36.24%	0.28%
0.92	33.16%	0.22%
0.94	29.88%	0.14%