Complexity reduction of VVC encoder using machine learning techniques : intra-prediction

Naïma Zouidi

COLLEGE DOCTORAL BRETAGNE
MATHS, TELECOMS
INFORMATIQUE, SIGNAL
SYSTEMES, ELECTRONIQUE

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

THÈSE DE DOCTORAT DE

L'INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : *Signal, Image and Vision*

Par
Naima ZOUIDI

Complexity reduction of VVC encoder using machine learning techniques

«intra prediction»

Thèse présentée et soutenue à INSA Rennes, le « 19 decémbre 2023 »
Unité de recherche : Institut d'Electronique et des Technologies du numéRique (IETR) - UMR CNRS 6164
Thèse N° : 23ISAR 44 / D23 - 44

Rapporteurs avant soutenance :

Anissa MOKRAOUI	Professor at the University of Sorbonne, Paris
Mohammed ATRI	Professor at the University King Khalid, Abha, Saudi Arabia

Président :	Francois-Xavier COUDOUX	Professor at the University of Polytechnic, Hauts-de-France

Examinateurs :	Anissa MOKRAOUI	Professor at the University of Sorbonne, Paris
	Mohammed ATRI	Professor at the University King Khalid, Abha, Saudi Arabia
	Maher JRIDI	Associate Professor at ISEN YNCREA Ouest, Nantes
	Amina KESSENTINI	Assistant Professor at ISM Gabès, Tunisia
	Wassim HAMIDOUCHE	Principal Researcher at TII, United Arab Emirates
Dir. de thèse :	Daniel MENARD	Professor at INSA Rennes
Co-dir. de thèse :	Nouri MASMOUDI	Professor at ENI Sfax, Tunisia

Acknowledgement

First and foremost, I would like to express my gratitude to my thesis advisors, Mr. **Nouri MASMOUDI**, professor at the University of Sfax, Tunisia, and Mr. **Daniel MENARD**, professor at the University of Rennes 1 for their availability, support, and advice. Your modesty and your professional skills will always inspire me as I move forward in my career!

My thanks also extend to Mrs. **Amina KESSENTINI**, assistant professor at the University of Gabes, Tunisia, and Mr. **Wassim HAMIDOUCHE** assistant professor at the University of Rennes 1. I want to thank them for the competent help they provided me, as well as their patience and encouragement. Their feedback was very helpful in structuring the work and improving its quality.

I would also like to express my sincere thanks to the jury members Mr. **François-Xavier COUDOUX**, professor at the Univesity of Polytechnic, Hauts-de-France, Mrs. **Anissa MOKRAOUI**, professor at the University of Sorbonne, Paris, Mr. **Mohamed ATRI**, professor at the University of King Khalid, Saudi Arabia and Mr. **Maher JREDI**, assistant professor at ISEN YNCREA west, Nantes for agreeing to judge this work despite their multiple occupations.

My heartfelt gratitude also goes to all my family and friends who, from near or far, allowed through their counsel and encouragement the completion of this thesis.

A special mention to **Campus France** for their assistance, and funding of this thesis work with the **PHC Maghreb**.

Table of Contents

List of Figures

List of Tables

Acronyms

ADAM Adaptive Moment Estimation. 66

AI All Intra. 40, 92, 96, 120, 126

ALF Adaptive Loop Filtering. 26

AMVP Adaptive Motion Vector Prediction. 21, 22

AMVR Adaptive Motion Vector Resolution. 22, 128

AR Augmented Reality. 125

AVC Advanced Video Coding. 15, 78

BD-PSNR Bjontegaard Delta Peak Signal to Noise Ratio. 14, 15

BD-BR Bjontegaard Delta Bitrate. 12, 14, 42, 43, 97, 105, 108, 120, 122, 123, 124, 127

BT Binary Tree. 30, 51, 80

CABAC Context Adaptive Binary Arithmetic Coding. 25, 37, 39, 126

CB Coding Block. 18, 30

CC-ALF Cross-Component Adaptive Loop Filtering. 26

CNN Convolutional Neural Network. 13, 5, 63, 66, 67, 70, 71, 82, 83, 84, 85, 86, 88, 89, 91, 94, 95, 96, 97, 100, 102, 104, 126, 127, 128

CPU Central Processing Unit. 128

CTB Coding Tree Block. 17, 18, 30

CTC Common Test Conditions. 40, 42, 46, 50, 52, 96, 120

CTU Coding Tree Unit. 17, 18, 30, 79, 80, 110, 113, 114, 127

CU Coding Unit. 11, 5, 6, 11, 17, 18, 32, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 93, 95, 96, 100, 101, 104, 106, 108, 109, 110, 111, 113, 115, 120, 122, 123, 124, 127, 128, 131, 132

DBF Deblocking Filter. 26

DNN Deep Neural Network. 62, 63, 64

DT Decision Tree. 58, 61, 76, 82, 109, 116, 117, 118, 119, 120, 122, 124, 127, 128

FIFO First In First Out. 22

FIPMD Final Intra Prediction Mode Decision. 36, 39, 43, 44, 46, 47, 48, 49, 50, 51, 52, 54, 79, 81, 83, 84, 85, 86, 87, 105, 108, 109, 110, 126, 131

FN False Negative. 99, 104

FP False Positive. 99, 104

FR Frame Rate. 8

GAN Generative Adversarial Network. 63

GDS Gradient Descent Search. 64, 65, 66, 82, 116

GOP Group of Pictures. 21, 40, 97, 120

GPU Graphics Processing Unit. 126, 128

HD High Definition. 3, 15

HD Haute Définition. 129

HEVC High-Efficiency Video Coding. 3, 15, 16, 17, 18, 21, 23, 25, 29, 30, 31, 36, 81, 82, 125, 128, 129, 146

HM HEVC Test Model. 4, 126, 130

HMVP History-based MVP. 22

HT Hadamard Transform. 27

IPM Intra Prediction Mode. 5, 6, 18, 19, 20, 24, 25, 29, 30, 31, 33, 35, 36, 37, 39, 47, 50, 51, 52, 79, 81, 82, 84, 86, 87, 92, 93, 104, 105, 108, 110, 111, 113, 122, 123, 124, 125, 127, 131, 146

ISO International Organization for Standardization. 3, 15, 129

ISP Intra Sub-Partitions. 5, 20, 33, 39, 43, 46, 47, 49, 50, 51, 52, 79, 81, 87, 105, 110, 111, 120, 125, 128, 131

ISPD Intra Sub-Partitions Decision. 39, 50

IT Information Technology. 3, 4

ITU International Telecommunications Union. 3, 15, 16, 125, 129

JCT Joint Collaboration Team. 15

JVET Joint Video Experts Team. 3, 15, 16, 125, 129, 146

JVT Joint Video Team. 15

LFNST Low-Frequency Non-Separable Transform. 24, 25, 39, 126

Light-GBM Light-Gradient Boosting Machine. 5, 6, 77, 82, 83, 88, 89, 90, 97, 109, 110, 111, 113, 115, 116, 117, 118, 120, 121, 122, 123, 124, 126, 127, 128, 131, 132, 146

MIP Matrix weighted Intra Prediction. 5, 20, 34, 36, 37, 43, 47, 51, 52, 85, 86, 92, 96, 102, 104, 105, 110, 111, 113, 115, 118, 120, 121, 122, 123, 125, 126, 131

MIPD Matrix weighted Intra Prediction Decision. 36, 46, 49, 52

ML Machine Learning. 74, 79, 81

MPEG Motion Picture Experts Group. 15, 16, 125

MPM Most Probable Mode. 5, 31, 32, 37, 39, 51, 52, 54, 79, 81, 110, 111, 115, 119, 121, 122, 126, 131

MPMD Most Probable Modes Decision. 37, 46, 47, 50

MRL Multi-Reference Lines. 20, 32, 37, 43, 51, 52, 85, 86, 87, 88, 92, 96, 102, 104, 105, 111, 125

MRLD Multi-Reference Lines Decision. 36, 45, 46, 49

MSE Mean Squared Error. 13, 96

MT Multi-type Tree. 30, 50, 51, 80, 82, 127, 128

MTL Multi-Task Learning. 11, 13, 5, 6, 72, 73, 74, 82, 84, 85, 86, 88, 89, 91, 94, 95, 96, 100, 101, 102, 104, 105, 108, 126, 127, 146

MTS Multiple Transform Selection. 39, 126

MVD Motion Vector Difference. 21, 22

NLP Natural Language Processing. 62

NST Non-Separable Transform. 24

PDPC Position-Dependent intra Prediction Combination. 36

PMVP Pairwise Average MVP. 22

PSNR Peak Signal to Noise Ratio. 12, 13

PU Prediction Unit. 81

QP Quantization Parameter. 25, 40, 43, 50, 92, 96, 115, 120

QT Quadtree. 30, 80, 82, 91

QTMT QuadTree with nested Multi-type Tree. 4, 18, 29, 30, 51, 80, 82, 86, 91, 106, 125, 127, 130, 146

RD Rate-Distortion. 4, 79, 146

RDO Rate-Distortion Optimization. 4, 5, 6, 16, 26, 27, 28, 29, 36, 39, 45, 47, 78, 79, 80, 81, 84, 91, 130, 131, 132

ReLu Rectified Linear Unit. 67, 69

RFC Random Forest Classifier. 58, 81, 82

RGB Red, Green, Blue. 9

RMD Rough Mode Decision. 36, 39, 45, 46, 49, 79, 81, 82

RNN Recurrent Neural Network. 63

SAD Sum of Absolute Difference. 27, 37

SAO Sample Adaptive Offset. 26

SATD Sum of Absolute Transform Difference. 27, 36, 37, 79, 81

SL Statistical Learning. 79

SSD Sum of Squared Difference. 27, 39

STMVP Subblock-based Temporal MVP. 22

SVM Support Vector Machine. 58

TBC Truncated Binary Coding. 36, 37

TC Texture Complexity. 81

TI Technologie d'Information. 129, 130

TN True Negative. 99

TP True Positive. 99, 100

TT Ternary Tree. 30, 51, 80

UHD Ultra-High Definition. 3, 16, 92, 125

UHD Ultra Haute Définition. 129

VTM VVC Test Model. 4, 24, 40, 42, 45, 46, 47, 49, 84, 87, 92, 93, 96, 97, 104, 105, 106, 107, 111, 120, 121, 122, 123, 126, 130

VVC Versatile Video Coding. 3, 4, 5, 6, 8, 15, 16, 17, 18, 20, 21, 22, 23, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 39, 40, 48, 54, 79, 81, 82, 84, 108, 110, 125, 126, 127, 128, 129, 130, 131, 132, 146

WAIP Wide Angular Intra Prediction. 35

Chapter 1

Introduction

1.1 Preamble

Video consumption is thriving. with the remote work and the new immersive Information Technology (IT) trends, video traffic over the internet seems to account for more than 3 quarter of the global internet traffic [6]. Apparently, improving user experience delivery has become more and more vital today. Since the late 1980s, several video coding algorithms have been introduced to tackle efficient video storage and transmission [7]. For instance, a series of H.26x was released in the early 2000s, and soon after the High-Efficiency Video Coding (HEVC) was published for the purpose of enabling High Definition (HD) video transmission [8]. At the same time, the video industry is recognizing a remarkable breakthrough in visual communication, especially with the introduction of immersive and data-intensive video applications.

So far, new advances in IT have made it possible to blend digital content on top of the life environment and stream Ultra-High Definition (UHD) media services, which is undoubtedly out of the scope of HEVC. Due to this, International Telecommunications Union (ITU) and International Organization for Standardization (ISO) issued jointly a new video coding standard, so-called Versatile Video Coding (VVC). This standard was developed by the Joint Video Experts Team (JVET) to address forthcoming video applications.

Whilst intended to increase compression ratio for larger video resolutions, standardization bodies have also provided VVC with a set of features to support a wide scope of applications, which they refer to as versatility. Indeed, the VVC is almost twice as bit rate-efficient as its predecessor HEVC [9] for the same objective visual quality. Nevertheless, this outstanding coding efficiency comes at the expense of an increased computational complexity, which remains a key challenge towards the real-time or hardware implementation of the VVC codec. In fact, real-time streaming or embedded devices imply several constraints such as low latency, limited processing capabilities, and affordable prices. Thus,

studying the complexity reduction opportunities and introducing fast encoding decision algorithms for VVC is the playground of this thesis.

1.2 Context and motivation

The baseline codec of VVC is based on the HEVC Test Model (HM), where a vast number of coding tools, such as QuadTree with nested Multi-type Tree (QTMT) and finer-granularity angular intra prediction [10] have been added to substantially increase coding efficiency. Furthermore, the block-based hybrid video coding scheme remains in use in the new video coding standard, where the minimization of the Lagrangian cost function, also known as Rate-Distortion Optimization (RDO) [11], is carried out to control Rate-Distortion (RD) constraint.

The impact of RDO on computational complexity has been visible throughout the different standardization efforts. Unsurprisingly, the VVC seems to suffer from the same issue due to the increase in parameter space. Actually, the encoder of the VVC Test Model (VTM) is reported to be 31 times [9] more complex than that of HM. Since the RDO seeks to select the best parameters to encode a bitstream, it is eventually overwhelming for an embedded system to test all the parameter space and even for real-time streaming to spend a lot of time resolving these parameters. Consequently, it is desirable to boost computational complexity and coding efficiency trade-off in order to ensure the commercial success of this standard.

More recently, artificial intelligence has become a growing part of IT due to its success in exploring large data sets and performing potential decisions without being explicitly programmed [12]. Associated with this, an increasing interest in machine learning-based video coding techniques is growing in order to tackle embedded systems and real-time streaming limitations.

1.3 Objectives and contributions

This Ph.D. thesis aims to study the potential of artificial intelligence algorithms in addressing the complexity reduction of VVC. It incorporates the following contributions:

Complexity assessment of the intra prediction This complexity assessment is conducted to study the complexity of the intra prediction in VVC. It analyzes the upper bound of complexity reduction in each step of the intra mode decision as well as its impact on the decision run-time. In addition, a statistical analysis of the most time-demanding step of the intra mode decision is performed to provide a clear insight into the different correlations, that may help in reducing the computational complexity of VVC.

Multi-task learning based intra mode decision framework This contribution deals with the complexity reduction of the intra mode decision of VVC. It establishes a large database for the intra prediction and proposes a Multi-Task Learning (MTL)-based intra mode decision framework. To this end, a shared Convolutional Neural Network (CNN) architecture is proposed for the new intra coding tools, including finer-granularity Intra Prediction Modes (IPMs), Matrix weighted Intra Prediction (MIP), etc...This framework leverages common knowledge among all the intra coding tools and limits the parameter search space in the intra mode decision to the top-2 inferred intra coding tool.

Light weight decision tree for low complexity intra mode decision In this contribution, we made full use of Light-Gradient Boosting Machine (Light-GBM) to assess the likelihood or the probability of using each intra coding tool at Coding Unit (CU) level. Consequently, a number of Light-GBM classifiers were fit using a large diversity of square blocks in order to output whether to skip MIP, Intra Sub-Partitions (ISP), Most Probable Mode (MPM), regular IPMs as well as angular, DC or planar. These predictions are then used to shrink the parameter search space, which also reduces the number of candidates for the RDO.

1.4 Organization of the manuscript

This Ph.D. thesis is divided into six chapters and an appendix listing all the scientific publications issued within it.

Chapter 2 reviews the basic foundations of video coding with a practical focus on the new intra coding tools of VVC. It also gives a brief history of previous video coding standards.

Chapter 3 studies the complexity reduction opportunities in the intra mode decision

of VVC. It analyzes the complexity of the new intra coding tools and the upper bound of complexity reduction in the intra prediction, while investigating the different correlations between the RDO, encoding parameters, and video content.

Chapter 4 provides a background on artificial intelligence with an emphasis on its most important sub-fields. In addition, this chapter gives an overview of previous proposals on fast encoding decisions.

Chapter 5 addresses the complexity reduction of the intra mode decision. It proposes a MTL-based intra mode decision framework, that limits the set of candidates in the intra mode decision to the top-2 inferred intra coding tools.

Chapter 6 proposes a light-weight decision tree for low-complexity intra mode decision. It involves using several Light-GBM classifiers to estimate the likelihood or the probability of using each intra coding tools at CU level for square blocks. The parameter search space is then shrunk using these probabilities and thus skipping unlikely IPMs.

Part I

Versatile Video coding

Chapter 2

Foundations of video coding

Introduction

Digital video coding is integral to the way we produce, share, and consume video content. It refers to the process of optimizing video content in terms of size, quality, and format in order to meet a certain requirement. In this chapter, the basic foundations of digital video coding, such as color space and chroma sub-sampling, are presented. It also details the specifications of the latest video coding standard, namely Versatile Video Coding (VVC).

2.1 Digital video

A digital video is comprised of a series of images, also called frames, displayed at a certain Frame Rate (FR), to which are added audio streams and metadata. The recording of a video is done using the transduction of light or brightness data into an electric current through a camera. The flow of a digital video is characterized by its FR, which is referred to as the number of frames displayed or taken per second. In order for the human eye to perceive a slight animation, the images must scroll at a minimum speed or FR of 10 frames per second. Below this speed, the effect produced is that of a jerky rhythm. Beyond that, the reading of the frame is even more fluent. A digital video is also characterized by its resolution, which defines the quantity of information contained in a frame. This latter is expressed in the form of the number of pixels used on the horizontal axis, namely the frame width, multiplied by the number of pixels used on the vertical axis, also known as the frame height [13].

2.1.1 Color space

Color spaces are standards for representing colors in a digital video [14]. Indeed, the digitization of a video frame consists on decomposing it into various elementary points, so-called pixels, to which one associates a numerical value forming its gray level or color information. In the Red, Green, Blue (RGB) color space representation, the color information consists of three basic signals: Red, Green, and Blue. The RGB is physically a juxtaposition of three types of signals that the eye mixes into a single colored pixel. A colored pixel in RGB constitutes binary information of 24 bits, for the computer, where every 8 bits represents one of the three types of signals (i.e. Red, Green, and Blue).

For the YUV color space, also denoted as YCbCr, the color representation takes advantage of the fact that the human eye is more sensitive to brightness or luma ($\mathbf{Y}$) information than to color or chroma information ($\mathbf{U}$ and $\mathbf{V}$). Hence, This system consists in separating the color's notation into two concepts, namely light intensity and light hue. Since the YUV color space separates the luma and the chroma information, it has been widely used in video compression. For instance, to convert from RGB to YUV, Equation. (2.1) can be used [15].

$$\begin{cases} \mathbf{Y} = (0.299 \times \mathbf{R}) + (0.587 \times \mathbf{G}) + (0.114 \times \mathbf{B}) \\ \mathbf{U} = 0.564 \times (\mathbf{B} - \mathbf{Y}) \\ \mathbf{V} = 0.713 \times (\mathbf{R} - \mathbf{Y}) \end{cases} \tag{2.1}$$

2.1.2 Chroma sub-sampling

Chroma sub-sampling is an elementary method of digital video compression, that works by manipulating the chroma information or encoding less chroma information than the luma information [16]. So, Chroma sub-sampling takes advantage of the fact that the human eye is actually more sensitive to brightness information than to color, and therefore the difference in the image quality is going to be indistinguishable.

As aforementioned, digital images or video frames are composed of pixels or square regions. With chroma sub-sampling, each pixel has its own luma information, but its chroma information will be shared among a group of pixels. The higher the degree of chroma sub-sampling, the greater the number of pixels sharing the same chroma information within a group. In Essence, the sub-sampling format is represented in the form of three numbers separated by ":". The first number indicates the size of the horizontal sampling block in pixels, which is typically equal to 4. The second number represents

the number of pixels sharing the same chroma information in the top row (within the sampling block). And, the last one gives the number of pixels sharing the same chroma information in the bottom row (below the sampling block). Figure. 2.1 shows the different degrees of chroma sub-sampling for digital videos.

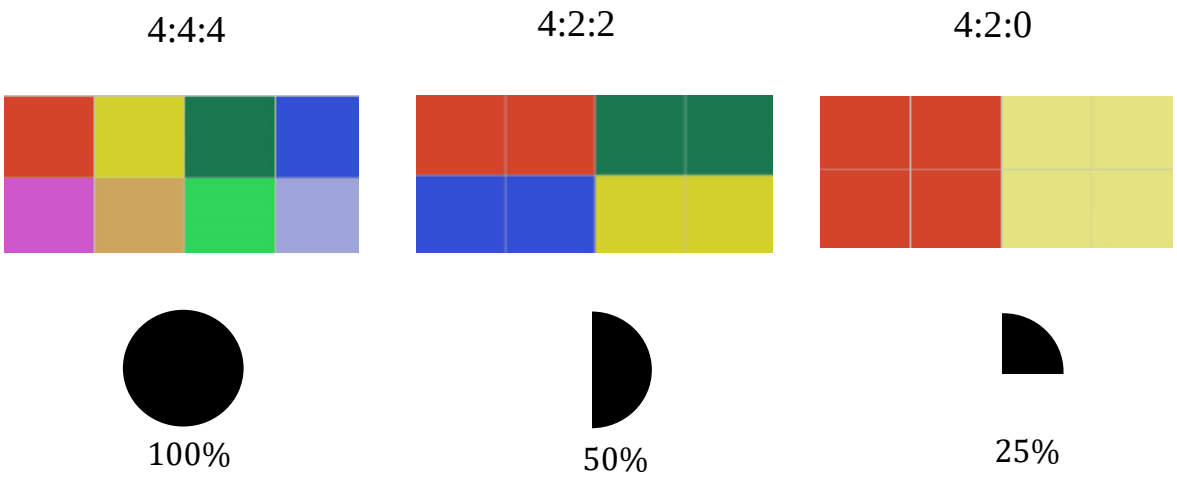


Figure 2.1 Illustration of chroma sub-sampling; the values 100%, 50%, and 25% give the amount of chroma samples compared to that of luma.

As we can see in this Figure, in the 4:4:4 chroma sub-sampling, each luma sample or pixel has its own chroma information. However, the amount of chroma samples is half that of the luma in the 4:2:2 chroma sub-sampling. Likewise, for the 4:2:0 chroma sub-sampling, we'll have only one chroma sample per four luma samples.

2.2 Video compression

Due to the large amount of raw data that represents a video file and the increasing need for its storage and transmission, several algorithms that we refer to as video compression algorithms have been proposed. The objective of these latter is to reduce the size of a video file, so that it takes less storage space and bandwidth for streaming. Regarding

videos, compression generally consists in reducing the amount of data while minimizing the impact on visual quality.

There are two main categories of video compression algorithms, namely lossless and lossy compression. These are terms that describe whether all the raw data of a video, can be restored when decoding it [17]. In lossless compression, every bit of data that was originally in the video before it was encoded or compressed is restored after it is decoded or decompressed, which means that the integrity of all the information in that video is preserved. However, lossy compression reduces the size of a video file by permanently eliminating some information, particularly redundant information. Hence, compared to lossless compression, a much higher compression ratio can be achieved with lossy compression [18].

2.2.1 Basic structure of video codecs

A video codec is a software, that incorporates an encoder, to encode or compress raw video data, and a decoder, to decode or decompress encoded video data. One of the core concepts of most video codecs is block-based hybrid video coding [19]. This latter combines prediction, namely spatial (intra) and temporal (inter) prediction, with residual transform and entropy coding [20]. Such an approach has proven to be very efficient in compressing a video into a reduced bitstream size. By prediction and transformation of the prediction error or the residual signal, redundant information in the raw video data can be eliminated. With the application of quantization after the transformation step, irrelevant parts of the information are also removed.

The block-based hybrid video scheme is illustrated in Figure 2.2. As seen in this Figure, the frames of an input video are first partitioned into coding blocks, also known as Coding Units (CUs). To each CU the encoder assigns a prediction mode, either intra or inter prediction, in order to generate the prediction signal. This latter, which has been generated from the information available to both the encoder and the decoder (depending on the selected prediction mode), is subtracted from the input signal to form the residual signal. The residual is then transformed, quantized, and coded using entropy coding with all the needed parameters to reproduce the input signal at the decoder side. For instance, the first encoded frame in a raw video can only use intra prediction, since no previous frame is available in the decoded frames buffer. For the following frames, the encoder may decide between the two prediction modes based on a decision criterion, which we are going to cover in much more detail in Section 2.3.7.

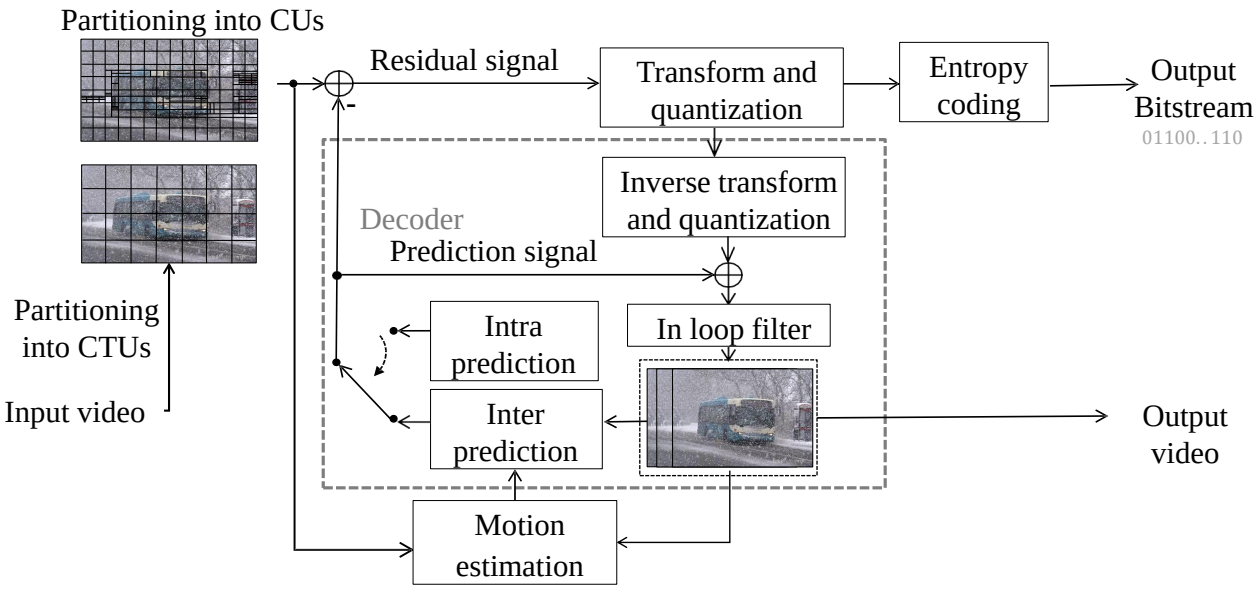


Figure 2.2 Block-based hybrid video coding scheme

During the reconstruction step, the transformed and quantified residual is added to the available prediction signal to reproduce the input signal at the decoder side. The signal is then processed by in-loop filters, to reduce block artifacts. Once the complete frame has been processed accordingly, its reconstruction becomes available and can be stored in the decoded frames buffer to participate in future temporal predictions.

2.2.2 Performance comparison metrics for video coding

To evaluate the performance of a video codec, several performance metrics can be used. These latter include:

— Objective measures of quality such as the Peak Signal to Noise Ratio (PSNR).

— bit rate.

— Bjontegaard delta metrics such as the Bjontegaard Delta Bitrate (BD-BR).

2.2.2.1 Objective measures of quality

The video quality is a difficult parameter to define and evaluate because it depends on the viewer or it is subject to viewer evaluation. However, there are quantifiable quality measures that could approximate this subjective score. These latter aim to define metrics

that are highly correlated with the quality scores given by a set of viewers. The Peak Signal to Noise Ratio (PSNR) is one of the most widely used objective measures of video quality [21]. It is measured on a logarithmic scale and depends on the Mean Squared Error (MSE) between the reconstructed video and the original one. For a single video frame, the PSNR can be calculated with Equation. (2.2).

$$PSNR(db) = 10 \log_{10} \frac{(2^n - 1)^2}{MSE} \tag{2.2}$$

where:

n is the number of bits representing a pixel.

MSE is the Mean Squared Error (MSE) computed by Equation (2.3).

$$\frac{1}{w \times h} \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} [\mathbf{Y}(x, y) - \hat{\mathbf{Y}}(x, y)] \tag{2.3}$$

$\mathbf{Y}$ is the original pixels' matrix of the current block.

$\hat{\mathbf{Y}}(x, y)$ is the predicted pixels' matrix of the current block.

h denotes the height of the current block and w is its width.

x and y are the pixel positions in the current block.

2.2.2.2 Bit rate

The bit rate is another important parameter to consider when compressing raw video data. It measures the required number of bits to be transmitted over a certain period of time. As shown in Equation. (2.4), the bit rate is usually calculated using the size of the encoded bitstream divided by the duration of the decoded video in seconds (s). This means that the bit rate is often measured in $kbit/s$. For instance, a low bit rate would reduce the required bandwidth. However, its reduction will generally come at the expense of increased computational complexity or low visual quality.

$$bitrate(kbit/s) = \frac{s_b}{d} \tag{2.4}$$

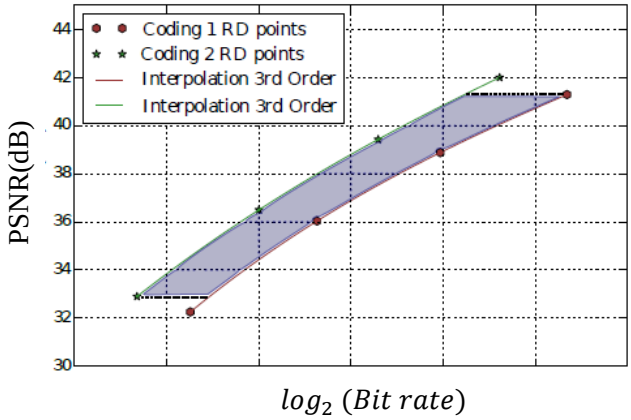
where:

s_b is the size of bitstream in $kbits$.

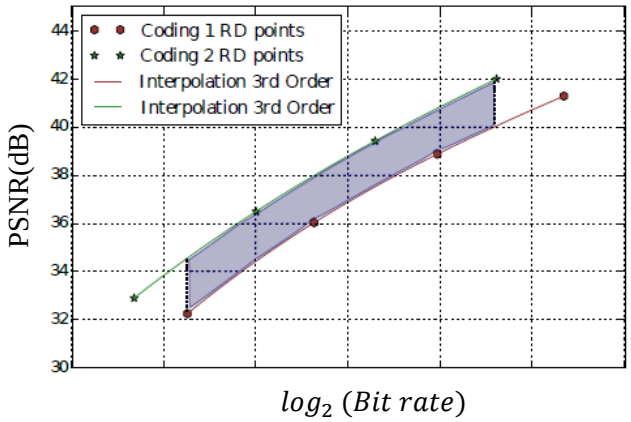
d is the duration of the decoded video in seconds (s).

2.2.2.3 Bjontegaard delta metrics

The Bjontegaard delta metrics were developed in 2001 [22] by Bjontegaard. Since then, these metrics have been widely used to compare the coding efficiency of a given video codec to that of a reference codec or setting. This is achieved based on interpolating curves



(a)



(b)

Figure 2.3 Calculation of delta Bjontegaard metrics [1];(a) Bjontegaard delta bit rate; (b) Bjontegaard Delta PSNR

from multiple data points of quality and bit rate, also known as test data points [23]. The calculation of Bjontegaard delta metrics, as illustrated in Figure. 2.3, usually involves two parts, namely Bjontegaard Delta Bitrate (BD-BR) and Bjontegaard Delta Peak Signal to Noise Ratio (BD-PSNR). For the BD-BR, it gives the average bit rate difference, in

percent, between two video codecs or two settings of the same codec, when considering the same video quality. whereas, the BD-PSNR reports the average quality difference in decibels (dB) when considering the same bit rate.

2.2.3 Brief history of video coding

Over the past two decades, video coding has evolved enormously. The first standardization effort began in the early 1990s, right after the arrival of digital video technology. Indeed, the first video coding standard, so-called H.261 [24], was issued by the two standardization bodies International Telecommunications Union (ITU) and International Organization for Standardization (ISO) to address video conferencing. Since then, ITU and ISO have collaborated on a series of video coding standards, including Motion Picture Experts Group (MPEG)-1 [25], which was published around 1992, H.263 [26], defined in 1995, and H.264 /Advanced Video Coding (AVC) [27], which was published by the Joint Video Team (JVT) in order to enable as twice as the bit rate saving of previous standards [18].

Aimed at enabling High Definition (HD) video encoding and decoding, meetings between Joint Collaboration Team (JCT), a group by ITU and ISO, started in 2010 to release the High-Efficiency Video Coding (HEVC) [28] by January 2013. Straightforward, the JCT continued the standardization of scalable and multi-view versions of HEVC [29]. For instance, the HEVC standard being commercialized today is able to produce files that are about 50% smaller than those encoded with H.264 for the same objective visual quality. Nevertheless, the future of new digital technologies lies in compression and the efficiency that this compression can provide. Therefore, if the resolution of our videos evolves faster than the storage capacities and the bandwidth of our network infrastructures, it is instrumental to introduce more compression on raw video data. To this end, Joint Video Experts Team (JVET) worked since 2015, to finalize the new video coding standard, so-called Versatile Video Coding (VVC) [10]. The performance promised by this new standard will make it possible to transmit and store 4k to 8k videos as well as stream immersive video applications such as 360° videos. The timeline of video coding standards developed by ITU and ISO over the last 25 years is depicted in Figure. 2.4.

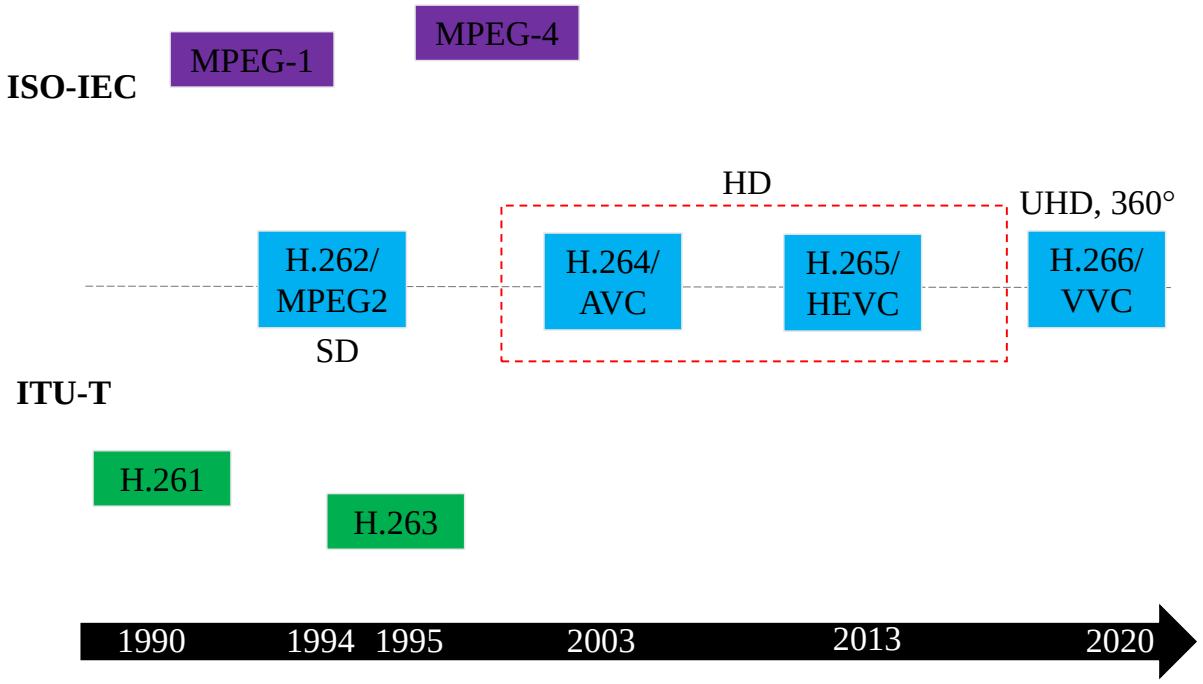


Figure 2.4 Timeline of international video coding standards

2.3 Versatile Video Coding

The JVET, a group by ITU and MPEG, released in July 2020 the new video coding standard, namely Versatile Video Coding (VVC). As with all other ITU and MPEG standards, since H.261, the VVC is based on the block-based hybrid video coding scheme [10]. Typically, versatile refers to a set of coding tools that enables VVC to handle a wide range of media services and offer high-quality video content at a low bit rate cost. In fact, the VVC can encode Ultra-High Definition (UHD) and immersive video applications at around 40% of bit rate saving compared to its predecessor HEVC [9].

In order to achieve superior coding efficiency, the block-based hybrid video coding scheme has undergone significant improvements, that mostly extended existing tools in HEVC. As a result, this Section details these new improvements, while reviewing the Rate-Distortion Optimization (RDO) theory.

2.3.1 Block partitioning

The partitioning step divides a video frame into non-overlapping blocks in order to prepare it for the different encoding decisions, such as prediction, transformation, and quantization. In video coding standards such as HEVC [30], video frames undergo partitioning into square blocks. Similarly, VVC divides video frames into Coding Tree Units (CTUs) [10] formed by three types of Coding Tree Blocks (CTBs), namely the luma CTB (or Y) and the two chroma CTBs (or Cb and Cr). The CTUs are then recursively divided into smaller Coding Units (CUs). In essence, a CTB has always the same size as a CTU. However, it may be too large to decide whether we should perform inter or intra

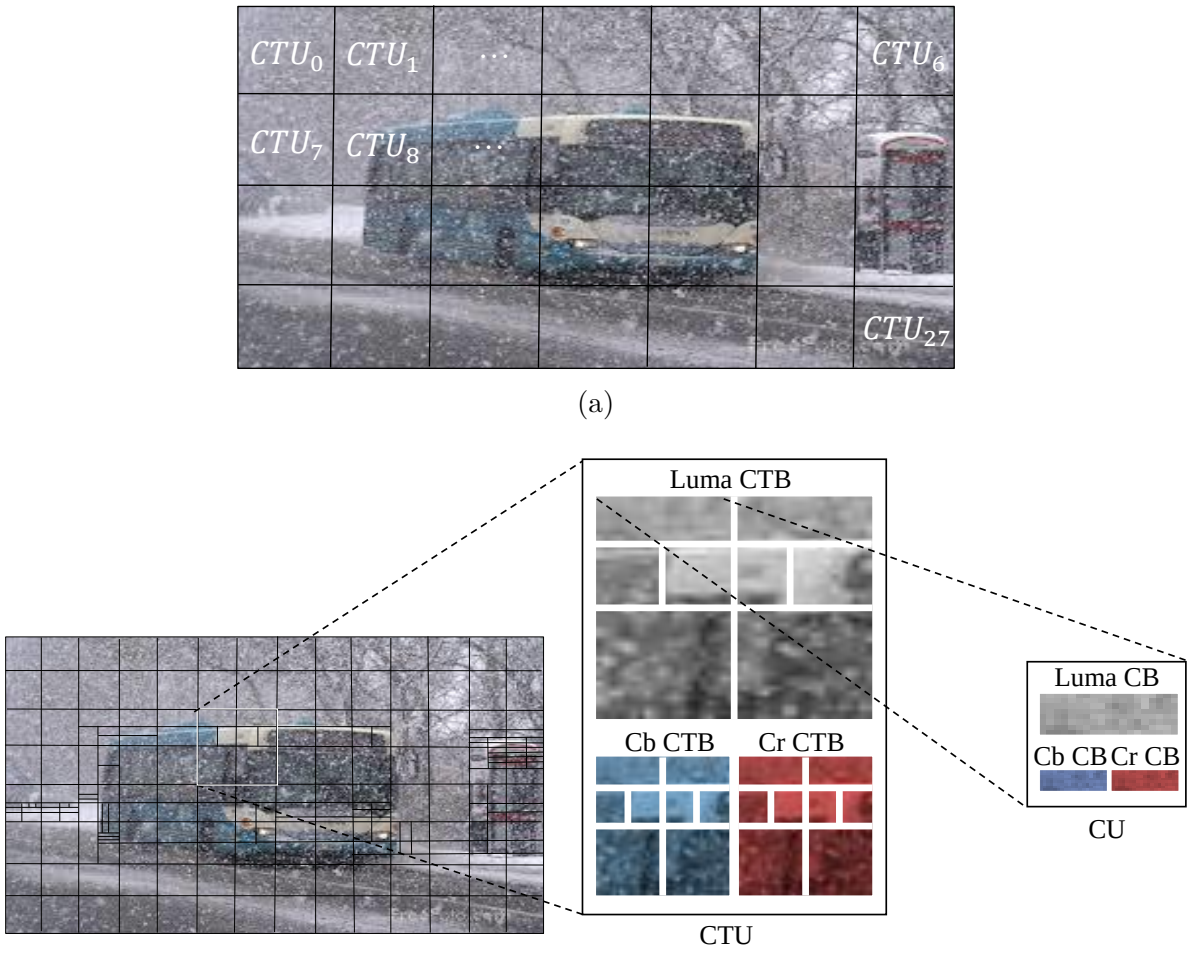


Figure 2.5 Block partitioning; (a): Frame partitioning into CTU; (b): Components of a CTU and partitioning into CUs

prediction. Thus, each CTB should be divided into several Coding Blocks (CBs) in order to decide the different coding parameters. Hence, each CB may have a size ranging from 4×4 to the CTU size [10]. More precisely, a CU, comprised of luma CB and two chroma CBs, may have a maximum size of 128×128 [31]. The partitioning step is detailed in Figure. 2.5.

In contrast to the HEVC, which involves the use of several processing units for partitioning, prediction, and transformation, the entire coding process in VVC requires only one processing unit, denoted as CU. This latter can have either a square or rectangular shape [20]. Thereby, the VVC introduced a new partitioning concept, so-called QuadTree with nested Multi-type Tree (QTMT).

2.3.2 Intra prediction

The intra prediction is the step, that addresses spatial redundancy [28]. It supposes that all the blocks in a video frame can be reproduced identically or, possibly, with very slight variations. At its core, intra prediction approximates the pixel values of a block using the pixels of its neighboring blocks, which are similar to it, or have identical color information. It is therefore sufficient to code a block, that will be defined as a reference block and then extrapolate the others using its reconstructed pixels or samples.

As shown in Figure. 2.6, the pixel $Y(x, y)$ can be generated by combining the reference samples, previously encoded, along a prediction direction, also known as the Intra Prediction Mode (IPM). Indeed, several types of IPMs can be used to derive the prediction of the current pixel from its adjacent pixels, which were previously encoded. These latter

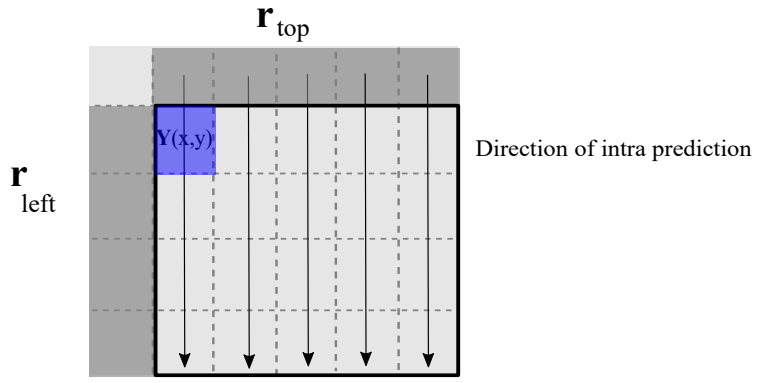


Figure 2.6 Illustration of intra prediction; r_{top} are the top reference samples, r_{left} are the left reference samples

include:

— **DC mode**: The pixel value of the current block is predicted as the average of all its reference samples (the boundary samples (horizontal and vertical) of its adjacent blocks) [32].

— **Planar mode**: The value of each pixel in the current block is calculated by presuming an amplitude surface or a plane with a horizontal and vertical slope derived from its reference samples. For instance, Equation. (2.5) can be used to compute the prediction signal at the pixel $\mathbf{Y}(x, y)$ when using the Planar mode.

$$\begin{cases} \mathbf{Y}(x,y) = \mathbf{Y}_v(x,y) + \mathbf{Y}_h(x,y) + n >> log_2(n+1) \\ \mathbf{Y}_v(x,y) = (n-y) \times \mathbf{r}(0,x) + y \times \mathbf{r}(0,n) \\ \mathbf{Y}_h(x,y) = (n-y) \times \mathbf{r}(0,y) + y \times \mathbf{r}(n+1,0) \end{cases} \tag{2.5}$$

where:

x and y represent the pixel position in the current block.

n is the total number of reference samples.

Y_v stands for the vertical slope and Y_h is for the horizontal slope.

r denotes the reference samples.

— **Angular modes**: The value of each sample in the current block is calculated by extrapolating the value of its reference samples along a prediction direction, namely the IPM [32]. Indeed, the reference pixels are combined according to Equation. (2.6) for the vertical directions and Equation. (2.7) is used for the horizontal directions.

$$\begin{cases} \mathbf{Y}(x,y) = (32 - w_y) \times \mathbf{r}(0,i) + w_y \times \mathbf{r}(0,i+1) + 16) >> 5 \\ c_y = (y \times \sigma) >> 5 \\ w_y = (y \times \sigma) \& 31 \\ i = y + c_y \end{cases} \tag{2.6}$$

$$\begin{cases} \mathbf{Y}(x,y) = (32 - w_y) \times \mathbf{r}(0,i) + w_y \times \mathbf{r}(0,i+1) + 16) >> 5 \\ c_x = (x \times \sigma) >> 5 \\ w_x = (x \times \sigma) \& 31 \\ i = x + c_x \end{cases} \tag{2.7}$$

where:

i is the index of the reference sample.

c_y and c_x are the pixel parameters corresponding to the pixel position.

w_x and w_y are the intra prediction weights.

σ is the displacement parameter (the prediction direction).

Besides the finer-granularity angular IPMs with DC and planar, several novelty intra coding tools were introduced in the VVC to enhance the intra prediction. This includes:

— Multi-Reference Lines (MRL) [33].

— Low-complexity neural-network based intra prediction, also known as Matrix weighted Intra Prediction (MIP).

— and Intra Sub-Partitions (ISP) [34].

We will cover all these intra coding tools in Chapter 3.

2.3.3 Inter prediction

The inter prediction takes advantage of the temporal redundancy, which comes from the correlation between the pixels over time. Indeed, in successive video frames, the same objects can be found at the same position or moved. Therefore, inter prediction refers to the prediction of a block using a reference block. As shown in Figure. 2.7, a motion vector,

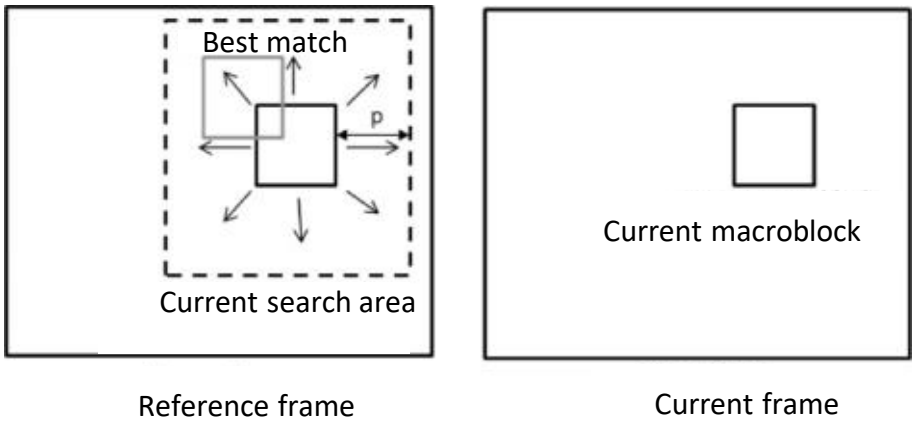


Figure 2.7 Illustration of motion estimation

indicating the displacement of the current block relative to the reference block [35] is estimated. The transmission of the reconstructed block is then done by sending the associated motion vector and the prediction error, also known as Motion Vector Difference (MVD), while minimizing the video size and keeping the compression as efficient as possible.

In video coding, a video is usually divided into a group of frames, so-called Group of Pictures (GOP) [36]. When using intra prediction, only I-frames, coded by intra prediction, can be distributed in the GOP. However, for inter prediction, P and B-frames are also included. Figure. 2.8 illustrates an example of GOP. For instance, the prediction of a P-frame, as shown in Figure. 2.9 (a), uses a unidirectional motion compensation, which means from a previous frame. This latter can be an I-frame or another P-frame. The prediction of a B-frame, as shown in Figure. 2.9 (b), is done using bidirectional motion compensation, which means from a previous or a future frame. This latter can be also either an I or a P-frame.

As in HEVC, VVC uses two methods to encode motion information, namely merge and Adaptive Motion Vector Prediction (AMVP). These latter predict the motion information of the current block by exploiting the spatial and temporal correlations between the current block and its reference block. In AMVP mode, the MVD is also signaled, which means that a residual coding is performed right after the motion compensation.

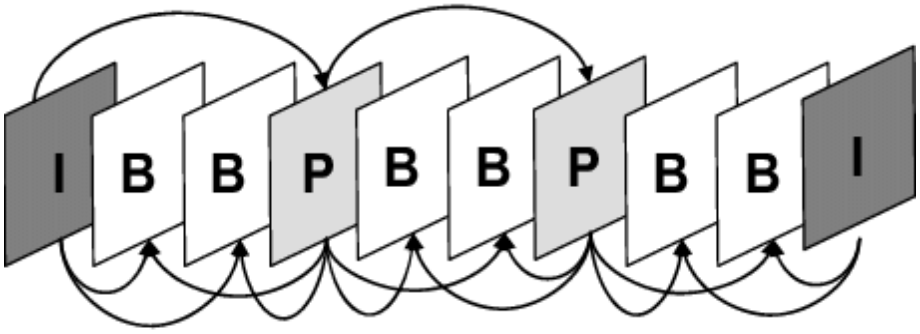


Figure 2.8 An example of GOP

In merge mode, only the motion candidate (the best-matching candidate) is signaled. In contrast, all motion information associated with the signaled candidate is inherited (MVD is assumed to be zero). In addition to these two methods, several techniques have been developed to improve merge and AMVP modes. For instance, the process of constructing the list of candidates was enhanced by including new motion candidates such as Pairwise Average MVP (PMVP), Subblock-based Temporal MVP (STMVP) [37], and History-based MVP (HMVP) from a First In First Out (FIFO) Table [38]. VVC also supports new motion compensation techniques like affine motion compensation [39] from different control points, as illustrated in Figure. 2.10. Furthermore, an improved motion vector signalization is introduced to send the motion information. Such signalization is enabled by symmetric MVD and Adaptive Motion Vector Resolution (AMVR) [40].

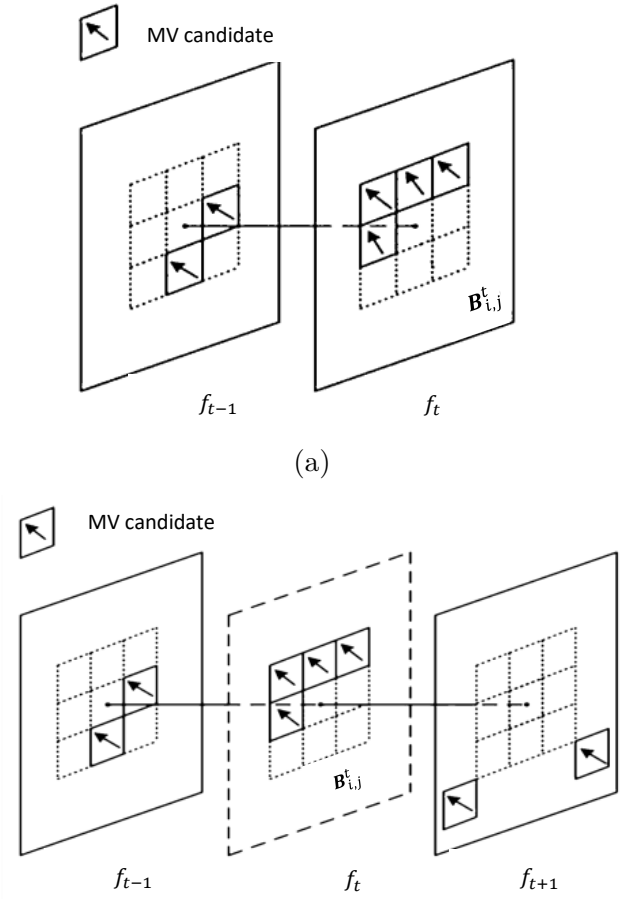


Figure 2.9 Inter prediction; (a): Inter prediction of P-frame ; (b): Inter prediction of B-frame

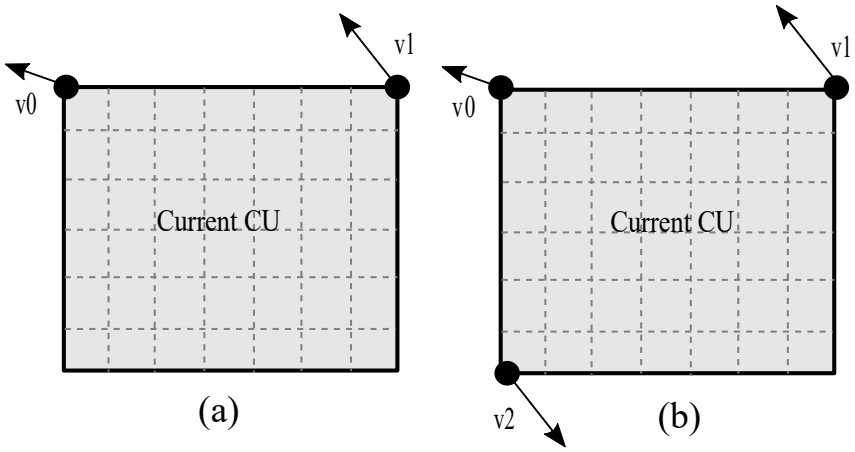


Figure 2.10 Affine motion compensation from different control points; (a): 4-parameter affine model (b): 6-parameter affine model

2.3.4 Transform

The transform step transforms the residual signal into a form that is easier to compress, where only a few coefficients have most of the energy. For instance, Most of the video's data tends to be concentrated in a few low-frequency coefficients, while higher-frequency coefficients are less relevant to the video reconstruction.

The transform step is then the domain in which the compression is actually performed. It reduces the remaining correlations in the residual block, calculated as the difference between the original block and the reconstructed block [41]. In VVC, the transform supports larger block sizes (up to 64×64), which is very useful, especially for high video resolutions. In addition to this, the following techniques have been developed to enhance the transform step:

Multiple Transform Selection: Besides the DCT-II transform, used in HEVC, VVC introduces a new set of transforms, namely DCT-VIII and DST-VII to transform the residual signal. For instance, Table. 2.1 illustrates the new set of transforms.

Table 2.1 Transform functions of DCT-II / VIII and DST-VII for n-point input

Transform type	Transform function $t_i(j)$, i, $j = 0, 1, \ldots, n-1$
DCT-II	$t_i(j) = \omega_0 \cdot \sqrt{\frac{2}{n}} \cos(\frac{\pi . i(2j+1)}{2n})$ avec $\omega_0 = \begin{cases} \sqrt{\frac{2}{n}} & \text{i} = 0 \\ 1 & i \neq 0 \end{cases}$
DCT-VIII	$t_i(j) = \sqrt{\frac{4}{2n+1}} \cos(\frac{\pi . (2i+1)(2j+1)}{4n+2})$
DST-VII	$t_i(j) = \sqrt{\frac{4}{2n+1}} \sin(\frac{\pi . (2i+1)(j+1)}{2n+1})$

Low-Frequency Non-Separable Transform: From VVC Test Model (VTM)5.0 onwards, a Low-Frequency Non-Separable Transform (LFNST) [42] is applied between the transform and the quantization step as well as between the transform and the inverse quantization step at the decoder side. For the LFNST, a 4×4 Non-Separable Transform (NST) matrix and an 8×8 NST matrix can be used depending on the block size. Thus,

Table 2.2 Selection of LFNST index based on the IPM

IPM	Transform index
IPM < 0	1
$0 \leq IPM \leq 1$	0 (LFNST is disabled)
$2 \leq IPM \leq 12$	1
$13 \leq IPM \leq 23$	2
$24 \leq IPM \leq 44$	3
$45 \leq IPM \leq 55$	2
$56 \leq IPM$	1

the LFNST index is chosen among 4 sets of transforms $\in \{0, 1, 2, 3\}$ depending on the IPM. This is illustrated in Table. 2.2

2.3.5 Quantization and entropy coding

The quantization, applied in the transform domain, is used to reduce redundant visual information. This refers to the mapping of a range of integer values into a single quantum [1]. The input of a quantizer is then the transformed residual signal and the output is always among a finite number of levels or values [18].

During the quantization step, the transformed residual signal is divided by a quantization step, denoted as q_{step}. this letter is controlled by a Quantization Parameter (QP), that varies between 0 and 51, as illustrated in Equation. (2.8). This parameter acts on the quality of the video and the achieved video compression. Hence, when the QP is low, some of the data can be lost but the quality remains high, which means that the achieved compression is also low. By increasing the QP, the elimination of the most significant details in the video may result in a greater loss of data and quality imperceptible to the human eye. Nevertheless, the compression increases significantly.

$$q_{step}(QP) = (2^{(1/6)})^{QP-4} \tag{2.8}$$

After the data has been quantized, it can be encoded using entropy coding to bring additional compression and reduce code redundancy. An entropy coding tries to encode a given set of symbols with the minimum number of bits required to represent them by taking advantage of their statistical features.

In the VVC, entropy coding uses Context Adaptive Binary Arithmetic Coding (CABAC) [43]. This is the same CABAC used by the HEVC. Still, it has undergone a number of improvements, particularly with regard to the probability estimation and the coding of the transform coefficients [44]. The core concept of CABAC is to efficiently represent data by adapting the encoding process to their statistical features and this is based on context modeling and binary arithmetic coding [45], where an encoder assigns shorter codes to the most common symbols and longer codes to those that are less frequent. This results in reducing the average code length and the bit rate needed to represent the data.

2.3.6 In loop filters

During the reconstruction step, the output of some decoded blocks may cause neighboring pixels to appear almost together and look like a larger block, which we call blocking artifacts. As televisions get larger, blocking and other artifacts would become more and more visible.

To reduce the impact of these artifacts and maintain a satisfactory quality, the VVC specifies 4 in-loop filters, namely Deblocking Filter (DBF) to minimize the blocking artifacts, Sample Adaptive Offset (SAO) to correct local average intensity changes and reduce ringing artifacts, Adaptive Loop Filtering (ALF) and Cross-Component Adaptive Loop Filtering (CC-ALF), which represent two methods of signal correction based on adaptive and linear filtering clipping [46].

2.3.7 Rate-Distortion Optimization

The implementation of an encoder often requires an encoding controller that decides the best prediction mode, transform set, filtering, and quantization parameters. Concretely, an encoding controller implements a recursive Rate-Distortion Optimization (RDO) to decide each of these encoding parameters in order to control the coding efficiency. Basically, the RDO is a theory, that measures the encoder's capability to reproduce the source information at a given bit rate [11]. As defined in Equation. (2.9), it checks for each encoding decision the distortion d and the bit rate (number of required bits) r, where λ is the Lagrangian multiplier.

$$j = d + \lambda \times r \tag{2.9}$$

The RDO is then the process of minimizing the bit rate under the assumption of minimal distortion. So, whenever the encoder has to make a decision, it should try to minimize the cost j in order to pick the parameter that gives the best quality and uses fewer encoding bits. This approach generally involves two sub-processes:

— Top-down verification: To determine the j cost of each combination explored by the encoder.

— Bottom-up comparison: To determine the best combination, means the one with the optimum j cost.

2.3.7.1 Distortion metrics

To select a certain encoding parameter, such as the prediction mode, the RDO needs to measure for each block its deviation from the original block. There are plenty of distortion metrics, such as the Sum of Absolute Difference (SAD), the Sum of Absolute Transform Difference (SATD), and the Sum of Squared Difference (SSD), that can be used to measure this deviation.

Sum of Absolute Difference: This metric is defined as the absolute difference between the original block and the reconstructed block, and it is computed by Equation. (2.10).

$$SAD = \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} \left| \mathbf{Y}(x,y) - \hat{\mathbf{Y}}(x,y) \right| \tag{2.10}$$

Sum of Square Difference: Is a commonly used metric in signal processing, and it is defined by Equation. (2.11).

$$SSD = \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} \left| \mathbf{Y}(x,y) - \hat{\mathbf{Y}}(x,y) \right|^2 \tag{2.11}$$

Sum of Absolute Transform Difference: This latter applies the Hadamard Transform before computing the SAD. To calculate the SATD, defined in Equation. (2.12), the residual or the difference between the original and the reconstructed block, is transformed by the Hadamard Transform (HT), given in Equation. (2.13):

$$SATD = \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} |\mathbf{R}_H(x,y)| \tag{2.12}$$

$$\mathbf{R}_H = \mathbf{H} \times (\mathbf{Y} - \hat{\mathbf{Y}}) \times \mathbf{H}^T \tag{2.13}$$

where:

$\mathbf{H}$ is the Hadamard Transform.

$\mathbf{R}_H$ is the transform of the residual using the HT.

$\mathbf{H}^T$ is the transpose of the HT.

2.3.7.2 Bit rate cost determination

Since the transmission channel's bandwidth, as well as the storage capacities, may be less than the information source's entropy, the fundamental idea behind the RDO is to find a way to encode raw video data using fewer bits, while maintaining an acceptable level of visual quality.

As indicated in Section 2.3.5, entropy coding can help in determining the minimum number of bits required to represent a given set of symbols or data, thereby reducing the overall bit rate required to transmit or store that data. However, in order to determine the cost of each encoding parameter the RDO may need at some point to pass each block of the video to be tested through the whole encoding process, which is comprised of prediction, transform, quantization, and entropy encoding, to finally get its actual bit rate cost and select the best combination.

Conclusion

In this chapter, video coding and its various steps were presented. For instance, The first part was devoted to a detailed description of the basic foundations of video coding and a brief presentation of prior video coding standards, such as H.261. The last part focused on the new video coding standard, namely VVC and it highlighted the improvements brought by this new standard. These improvements have allowed a significant gain in coding efficiency. But, they significantly increased the computational complexity, especially at the encoder side. Thus, in the next chapter, a complexity assessment of the intra prediction is conducted to underline the complexity reduction opportunities.

Chapter 3

Complexity assessment of the intra prediction

Introduction

Undoubtedly, the intra prediction of the Versatile Video Coding (VVC) has brought a significant coding gain compared to the High-Efficiency Video Coding (HEVC), offering finer granularity Intra Prediction Modes (IPMs) and new intra coding tools, which would help in adapting the intra prediction to various texture characteristics. Meanwhile, to solve the IPM decision, we need a metric that allows us to decide which IPM should be applied in a given situation. Such a metric is the Rate-Distortion Optimization (RDO). One of the major drawbacks of this process is the increased computational complexity. Hence, this chapter presents an in-depth complexity assessment of the intra prediction in the VVC, while reviewing its key intra coding tools.

3.1 Intra prediction

Intra prediction allows for reducing the spatial redundancy in a coding block by predicting the pixels that belong to that block using the previously encoded pixels of its neighboring blocks, also called reference samples. In VVC, this latter encountered several enhancements, such as introducing finer granularity IPMs and some other advanced tools. In this section, the key intra coding tools and the intra mode decision steps are reviewed.

3.1.1 Key intra coding tools of VVC

In response to the issued call for proposals, several contributions to the VVC have been investigated such as the QuadTree with nested Multi-type Tree (QTMT), finer granularity intra prediction, etc.

QuadTree with nested Multi-type Tree (QTMT): the QTMT [31] was introduced for block partitioning. It implies nesting a Multi-type Tree (MT) to the Quadtree (QT) [47] of the HEVC in order to make the coding process more flexible to forthcoming video resolutions and contents. Right after applying the QT partitioning to the Coding Tree Unit (CTU), the MT can further partition the QT leaf nodes into 2 or 3 partitions vertically or horizontally [31]. As illustrated in Figure. 3.1, this partitioning process is based on

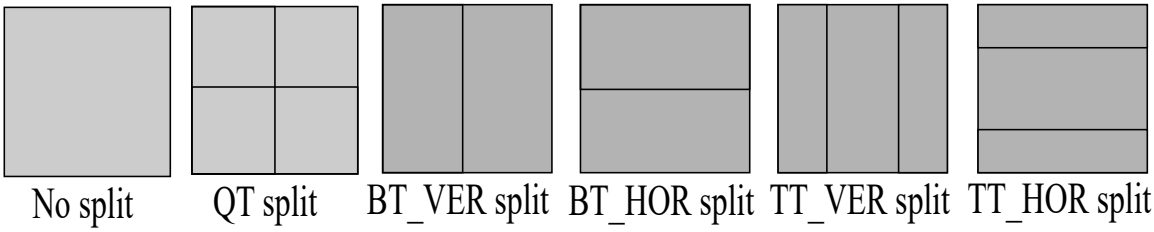


Figure 3.1 QTMT partitioning modes

the selected partitioning mode, either Binary Tree (BT) or Ternary Tree (TT). For the QTMT the following parameters are also defined:

- The size of the CTU: This represents the size of the root node of the QT.
- The MinQTSize: This is the size of the minimal QT leaf node.
- The MaxBTSize: Stands for the maximum size of the root node of the BT.
- The MaxTTSize: Gives the maximum size of the root node of the TT.
- The MaxMTDepth: This is the maximum depth of the MT.
- The MinBTSize: Refers to the minimal size of the binary leaf node.
- The MinTTSize: Defines the minimal size of the ternary leaf node.

In VVC, the partitioning tree can have separate partitioning structures for luma and chroma. Currently, for P and B-frames, the luma and chroma Coding Tree Blocks (CTBs) in a CTU must share the same partitioning structure. However, for I-frames, luma and chroma can have separate partitioning structures. This means that a CTU in an I-frame may consist of luma Coding Block (CB) or two chroma CBs, and a CTU in a P or B-frame always consists of CBs of all the three color components unless the video is monochrome (it has only one color component) [20].

Finer granularity intra prediction: Since the directional intra prediction plays the main role in efficiently predicting the different directional structures in video content, the number of angular Intra Prediction Modes (IPMs) was extended from 33 to 65. This new

set of IPMs still covers the same angular range as that of HEVC. It starts from 45° to -135°, where modes 2 and 66 indicate the angle 45° and -135°, respectively. In Figure. 3.2, the 35 original IPMs of HEVC are illustrated as black arrows. Whereas, the new IPMs in VVC are illustrated as green dotted arrows.

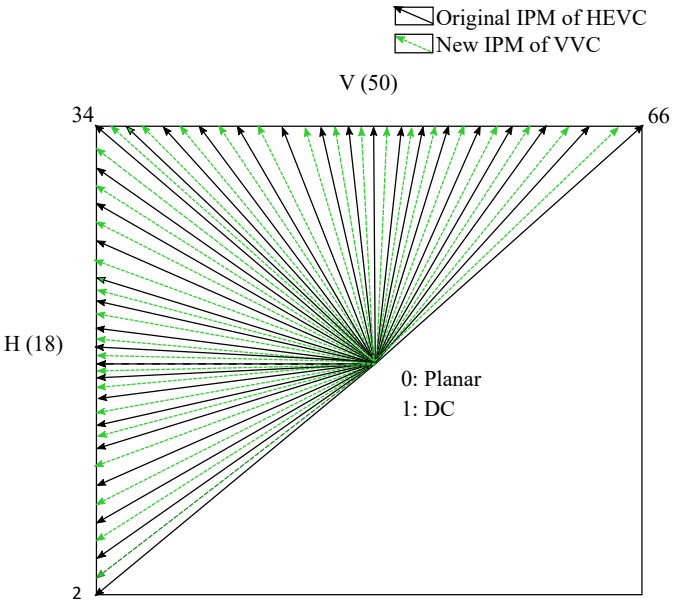


Figure 3.2 Illustration of the 67 IPMs in VVC. V and H are the vertical and horizontal modes, respectively

Intra coding with the 6 Most Probable Modes (MPMs) Due to the increased number of IPMs, an intra coding method with the 6 MPMs is used. These latter are created based on the modes of neighboring blocks [20], as detailed in Figure. 3.3. The modes m_L and m_A are the modes of left and above neighboring blocks, respectively. Hence, when m_L and m_A are non-angular, then DC, planar with horizontal (H), vertical (V) modes, and their derived modes are considered during the creation of the 6 MPMs. However, if these latter are both angular and the same, then only derived modes from m_L are considered. Otherwise, the maximum mode m_{max} and the minimum mode m_{min} between m_L and m_A, can be considered with their derived modes too. angular IPMs was extended from 33, in HEVC, to 65, in VVC. The angular range covered by this new set of IPMs is identical to that of HEVC. For instance, it starts from 45° to -135°, where modes 2 and 66 denote the angle 45° and -135°, respectively. In Figure. 3.2, the 35 original IPMs of HEVC are illustrated as black arrows. Whereas, the new IPMs in VVC are illustrated

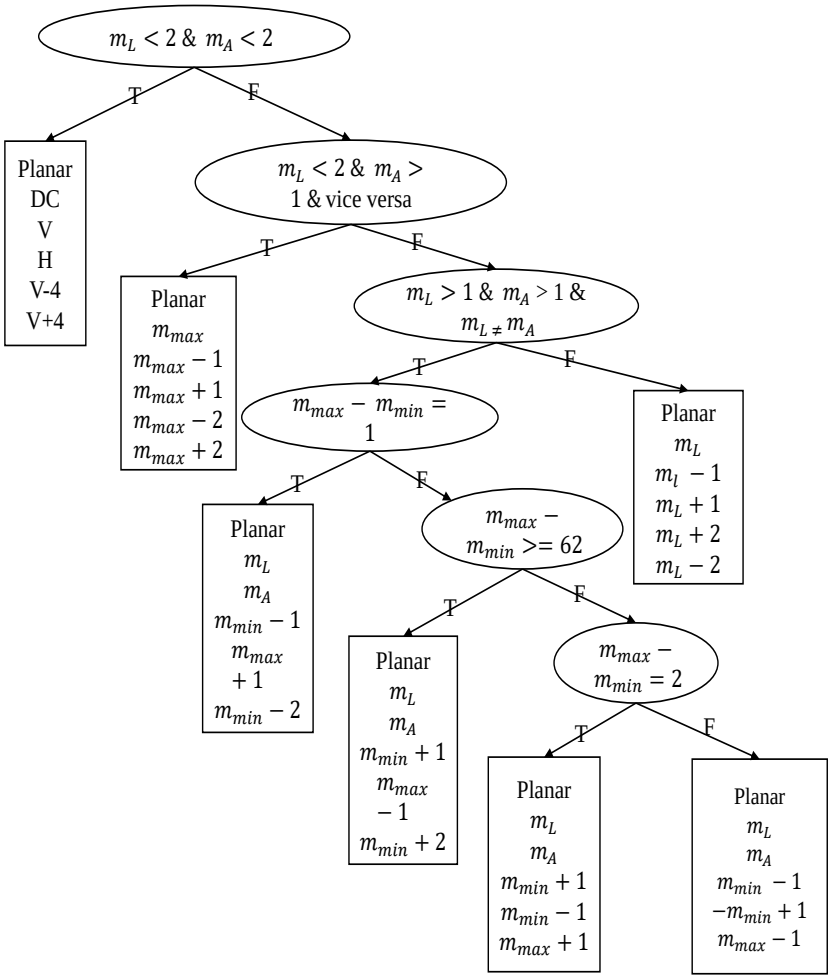


Figure 3.3 The 6 MPMs according to the modes of left and above neighboring blocks m_L and m_A; m_{max} and m_{min} are the maximum and minimum modes between m_L and m_A, respectively

as green dotted arrows.

Multi-Reference Lines (MRL): As detailed in Chapter 2, for the 67 regular modes, the intra prediction is carried out from neighboring pixels, which correspond to reference line 0, as depicted in Figure. 3.4. For instance, MRL allows VVC to use farther reference lines [34], as shown in the aforementioned Figure, to predict the current Coding Unit (CU). In this case, reference lines 1 or 3 can be considered. The index of reference lines is then signaled with the MRL flag. Furthermore, the MRL can be applied only on the 6 Most Probable Modes (MPMs).

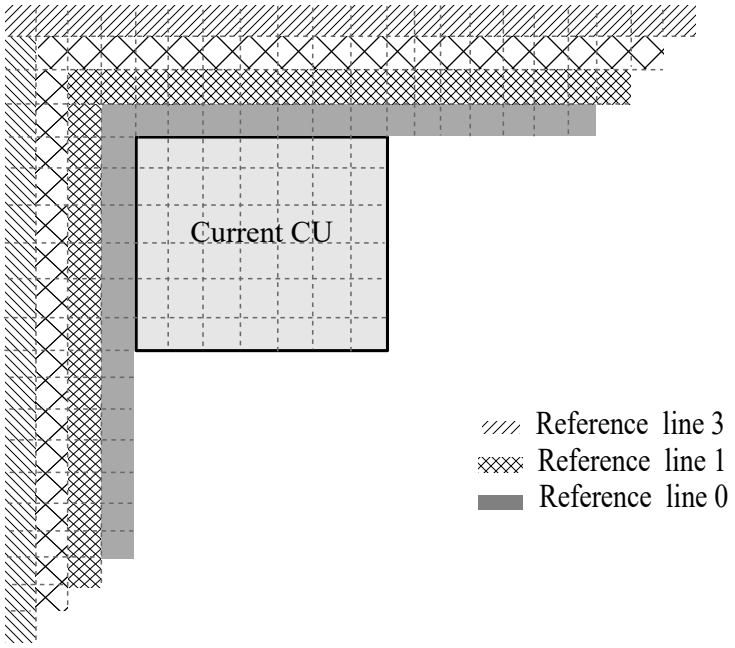


Figure 3.4 Reference lines neighboring to an intra-coded block

Intra Sub-Partitions (ISP): In order to adapt the intra prediction to various texture characteristics, the VVC introduced the ISP tool. This tool, as depicted in Figure. 3.5, enables the partitioning of an intra-coded block vertically or horizontally into 2 or 4 sub-partitions, in which the same IPM is used. Hence, for each mode, there are two options, when ISP is enabled, either partitioning the intra-coded block in a horizontal way (ISP = 1) or in a vertical way (ISP = 2). Furthermore, each sub-partition should be predicted using the reconstructed samples of the previously coded sub-partition [33].

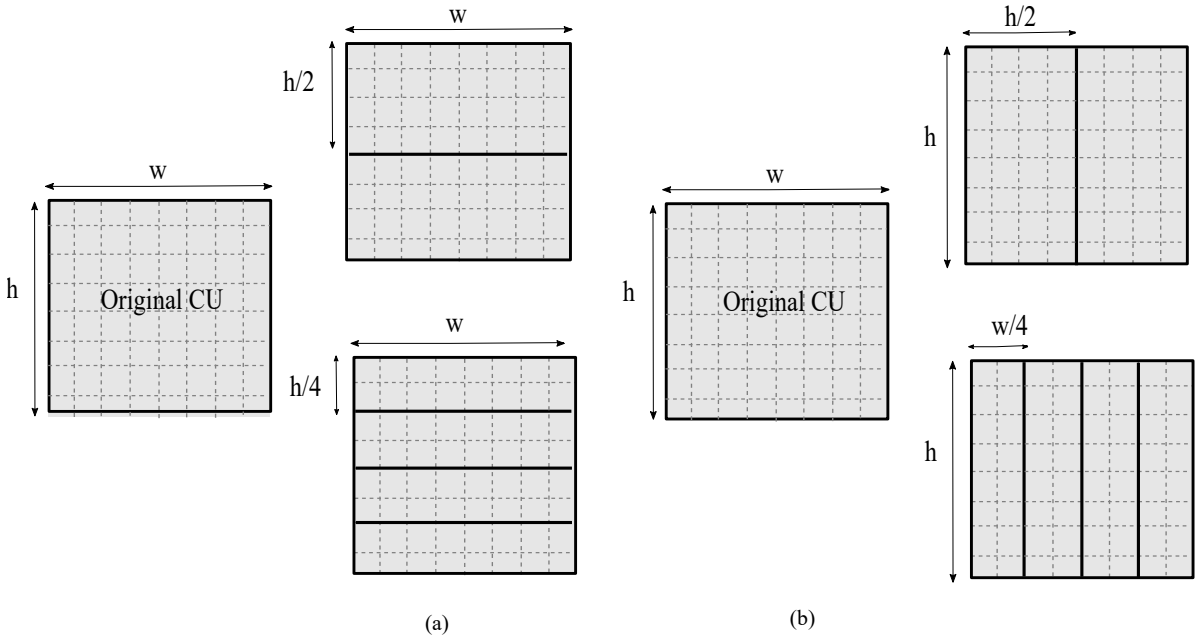


Figure 3.5 ISP splitting modes; (a) ISP=1; (b) ISP = 2

Matrix weighted Intra Prediction (MIP) Besides the simple weighted intra prediction [48], a three-step MIP [49] was integrated into VVC. As illustrated in Figure 3.6, when using MIP, the prediction signal is generated in three main steps. Firstly, reference

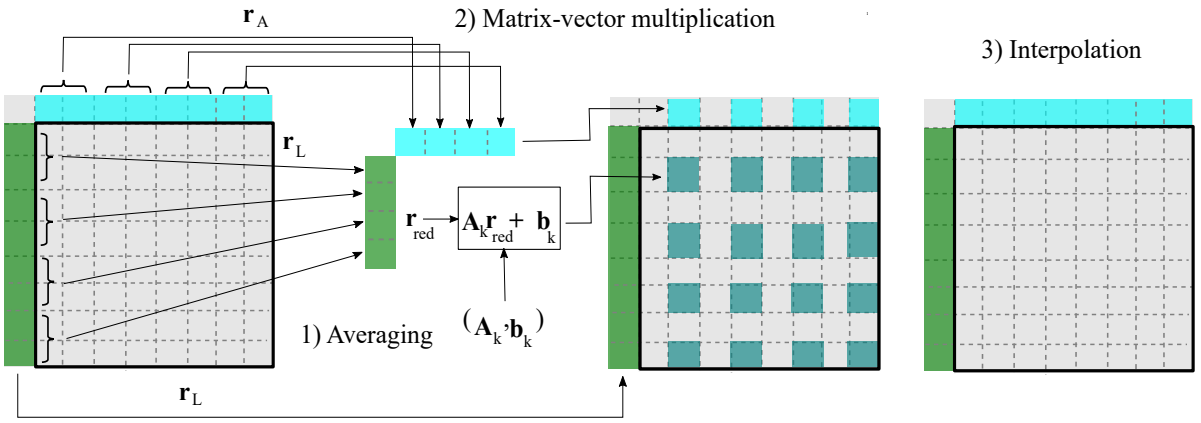


Figure 3.6 Steps of MIP; $\mathbf{A}_k$ is the matrix, $\mathbf{b}_k$ is the offset vector; $\mathbf{r}_A$ and $\mathbf{r}_L$ are the above and left reference samples, respectively

samples are reduced by averaging according to the block size. Then, these averaged samples, are used to generate a reduced prediction signal throw a matrix-vector multiplication,

followed by the addition of an offset. Finally, the samples at the remaining positions are interpolated from the reconstructed samples in the matrix-vector multiplication step. This tool uses up to 32 modes [20], which indicate the used matrix $\mathbf{A}_k$ and the offset vector $\mathbf{b}_k$. For instance, these latter are chosen from three sets, namely s_0, s_1, and s_2, where each set has a predefined number of matrices and offset vectors.
As shown in Equation. (3.1), the selected set's index i depends strongly on the block size.

$$i = \begin{cases} 0, & \text{for } w = h = 4 \\ 1, & \text{for } max(w,h) = 8 \\ 2, & \text{for } max(w,h) > 8 \end{cases} \tag{3.1}$$

Where:

h and w are the height and width of the current block, respectively.

Wide Angular Intra Prediction (WAIP): To accommodate the high diversity of block shapes, a WAIP [50] was also proposed in the VVC. The aim of this intra coding tool is to improve the intra prediction for rectangular blocks while maintaining the same number of conventional IPMs. Thus, some angular IPMs are replaced with WAIP modes, and the number of above and left reference samples, as shown in Figure. 3.7, is extended to adapt to these new directions.

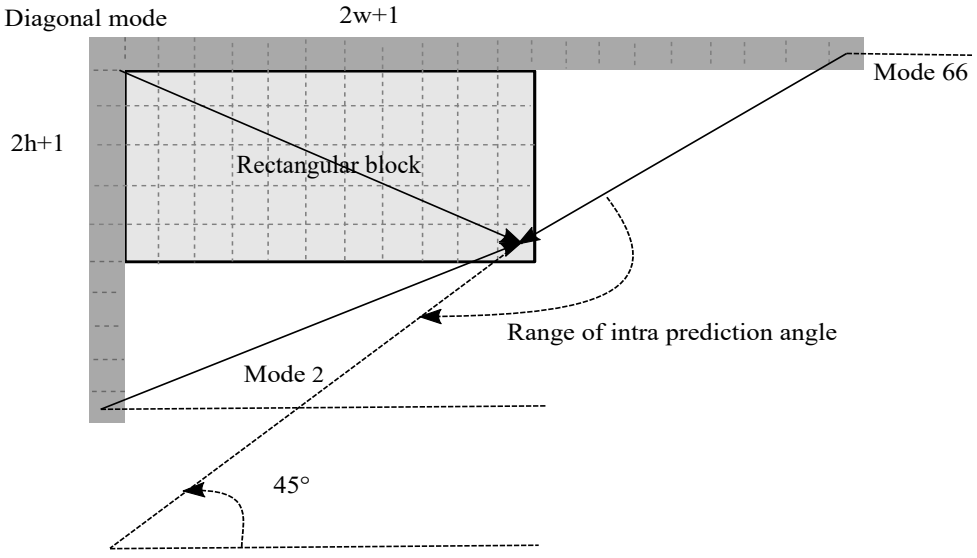


Figure 3.7 Illustration of WAIP

Interpolation filters: Regarding the interpolation step in the intra prediction, VVC uses a 4-tap Gaussian interpolation filter and smoothing filters [20] to generate the prediction signal from reference samples. Moreover, Position-Dependent intra Prediction Combination (PDPC) [51] can also be applied to the reference samples before carrying out the intra prediction. In fact, depending on the selected IPM, a combination of filtered and unfiltered reference samples may be considered.

3.1.2 The intra mode decision of VVC

As in HEVC, the intra mode decision in VVC relies on the RDO of the cost j, defined in Equation. 3.2. Due to the high density of IPMs, the best IPM is determined using three main steps, as illustrated in Figure. 3.8. These latter include Rough Mode Decision (RMD), intra coding tools decision, and Final Intra Prediction Mode Decision (FIPMD).

$$j = d + \lambda \times r \tag{3.2}$$

Where:

d refers to the distortion. r is the bit rate (number of required bits).

λ is the Lagrangian multiplier.

Rough Mode Decision (RMD): Represents the first step of the intra mode decision, which considers only the 67 regular IPMs. It generates a list of n best candidates (uiRdModeList), where $n \in \{2, \ldots, 8\}$. First, a test of DC, planar, and even angular modes (35 original IPMs of HEVC), is performed. Then, the direct angular neighbors (+1, -1) in the new set of IPMs in VVC are evaluated. The number of best candidates depends on the block size and whether the MIP is tested for the current block. Hence, n is usually between 2 and 3, unless the test of MIP is performed. In this case, the number of best candidates is refined by adding $log_2(min(h, w)) - 1$. Additionally, the test of IPMs in RMD is performed using Truncated Binary Coding (TBC) [20] to calculate the bit-rate metric r considered in Equation. (2.4) and the Sum of Absolute Transform Difference (SATD) as the distortion metric d. This latter is computed by Equation. (2.12).

Intra coding tools decision: The second step includes three main decisions, namely Multi-Reference Lines Decision (MRLD), Matrix weighted Intra Prediction Decision (MIPD),

and Most Probable Modes Decision (MPMD). It updates the list of n best candidates using different lists of IPMs. First, the two lists of 6 MPMs: MRL1, and MRL3, where 1 and 3 indicate the reference line indices, are tested using Context Adaptive Binary Arithmetic Coding (CABAC) to infer the number of required bits [20] and SATD as the distortion metric d. Second, the list of MIP modes is tested using TBC and the Sum of Absolute Difference (SAD), defined in Equation. (2.10). Finally, the list of the 6 regular MPMs is added to the best candidates if they are not already included [20].

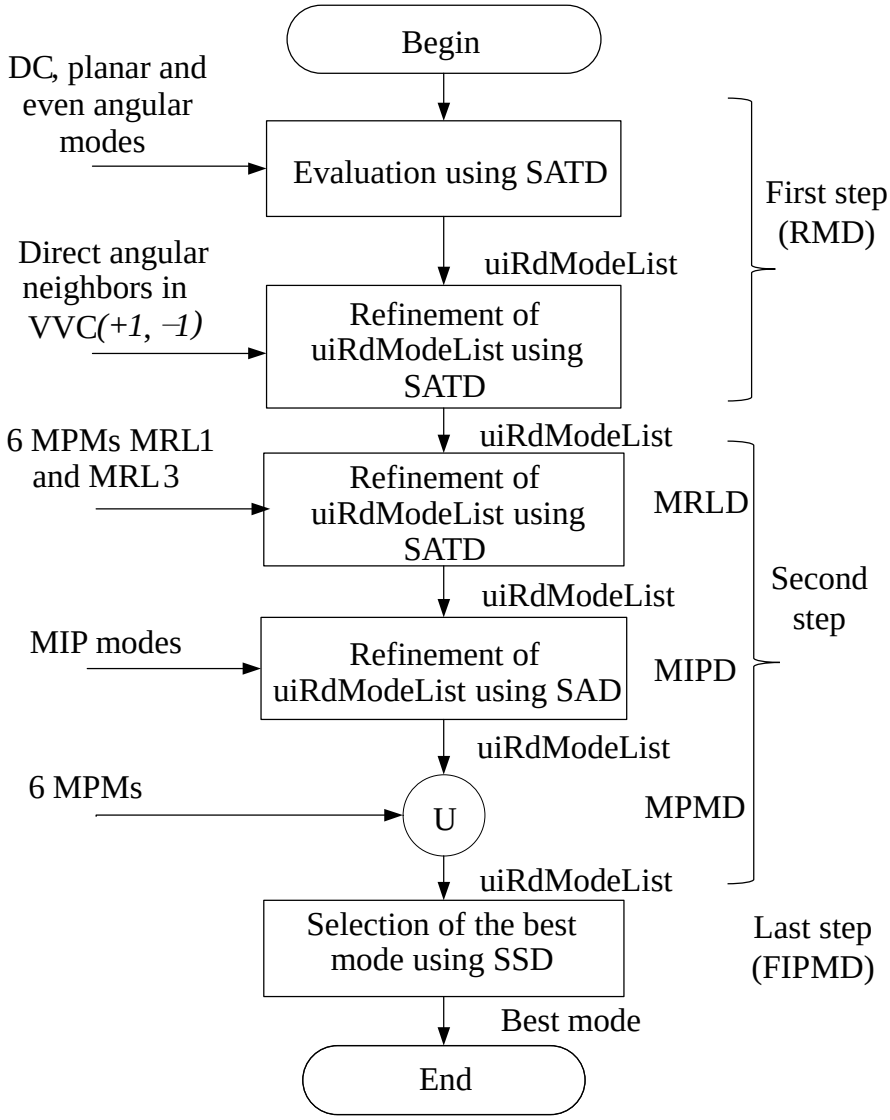


Figure 3.8 Intra mode decision of VVC; uiRdModeList is the list of the retained candidates for the FIMPD; 6 MPMs MRL 1 and 3 are the lists of the 6 MPMs of farther reference lines 1 and 3, respectively

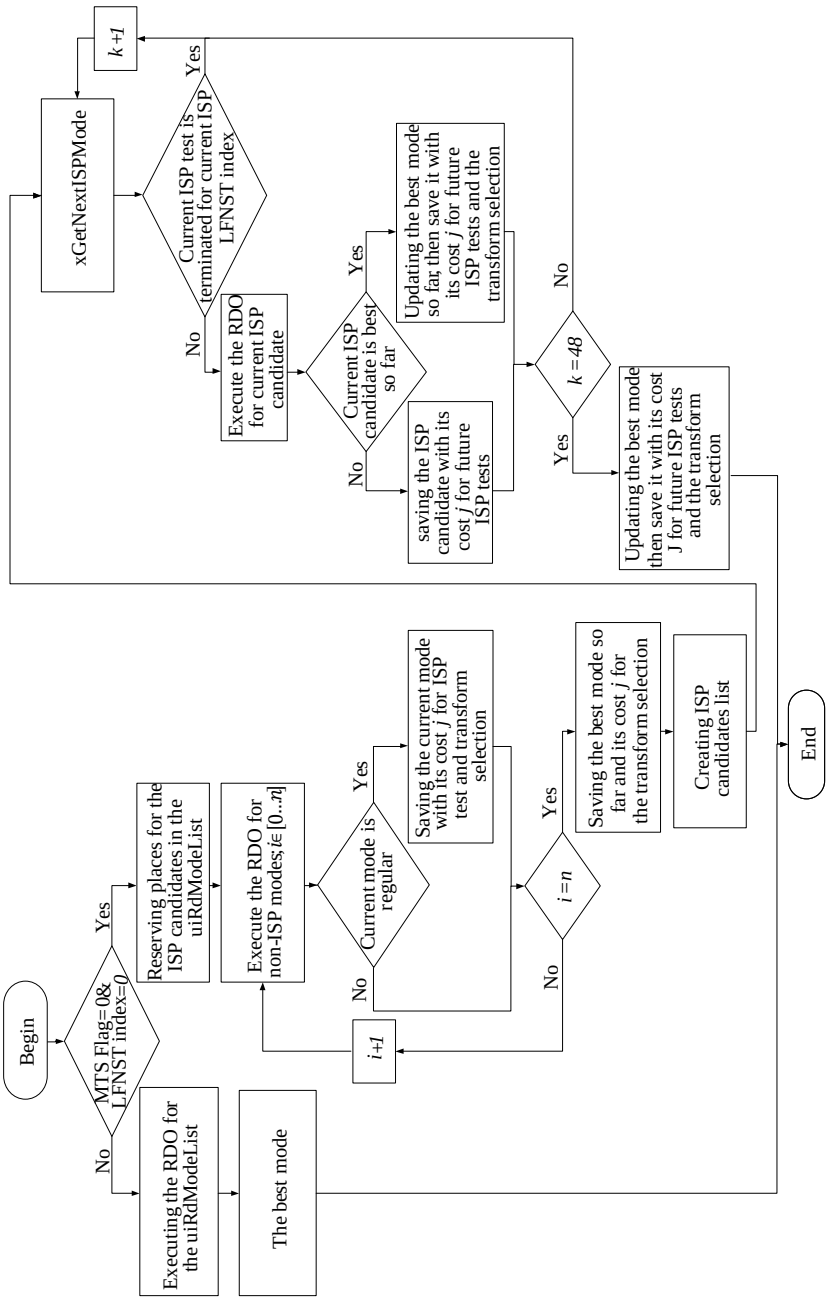


Figure 3.9 FIPMD; xGetNextISPMode decides whether to terminate the current ISP test or not

Final Intra Prediction Mode Decision (FIPMD): The last step is the FIPMD. As depicted in Figure. 3.9, this step may incorporate different tests in which the Sum of Squared Difference (SSD), computed by Equation. (2.11), is used as the distortion metric d and the bit-rate r is calculated using CABAC. Thus, the best IPM is updated progressively during the FIPMD.

According to the Multiple Transform Selection (MTS) flag and the Low-Frequency Non-Separable Transform (LFNST) [20] index, the encoder decides whether to test the ISP for the current block or not. For instance, the ISP cannot be combined with the LFNST or the MTS tools. Hence, when the MTS flag and the LFNST index are both different from zero, only the best IPMs, which were retained after the intra coding tools decision, are tested. Otherwise, the FIPMD can be performed on up to 62 IPMs, considering a maximum of 8 best candidates, 48 ISP modes, and 6 MPMs. In this case, the encoder firstly tests the non-ISP modes and saves the costs j of the regular ones. These last will be used later to construct the list of ISP candidates and select the transform core as well. In other words, if the ISP is tested, the VVC encoder creates an ordered list of potential ISP candidates using the best regular mode so far, non-angular modes (Planar and DC), and the remaining regular modes that were tested during the test of non-ISP modes. It also adds the list of 6 MPMs and the remaining regular modes from the Rough Mode Decision (RMD) when the ISP is more likely to be selected. Then, the encoder triggers the Intra Sub-Partitions Decision (ISPD) on the different ISP candidates.

As a first step, the encoder checks whether the current ISP test can be skipped based on the performance of the previously tested ISP or non-ISP modes. Therefore, if the ISP test has proceeded, then the encoder tests the different ISP partitioning modes for each ISP candidate and save their costs for future ISP tests and transform selection as well. More specifically, the transform selection for blocks using ISP will be performed based on these results instead of using the RDO. Unlike, the other steps of intra mode decision, the computation of the cost j is done using the reconstructed block instead of the intra-coded block. Hence, $\hat{\mathbf{Y}}$ in Equation. (2.11) is computed following an overall encoding process. In addition, the number of required bits r is inferred from an entropy coding of the residual block ($\mathbf{Y} - \hat{\mathbf{Y}}$) instead of an entropy coding of the selected IPM.

3.2 Complexity assessment of the intra prediction in VVC

This section is devoted to a complexity assessment of the intra prediction in the VVC. It identifies the most time-demanding intra coding tools and intra mode decision steps.

3.2.1 Test conditions and framework

In this work, simulations were conducted on the reference software VVC Test Model (VTM)-8.0 [52] under the Common Test Conditions (CTC) [53]. The coding configuration is the All Intra (AI) and the Quantization Parameter (QP) is fixed to 4 values, which are 22,27,32, and 37. Considering that the luma component has the heaviest decision [54] process, all the simulations were carried out only on this component. For this complexity assessment, the number of encoded frames in each video is set to one Group of Pictures (GOP), which is equivalent to one second.

3.2.1.1 All Intra configuration:

The All Intra (AI) is the coding configuration used to test the intra coding tools. In this latter, all the frames are coded using only spatial or intra prediction techniques. Each frame of the video is then coded independently of the other frames and no temporal or inter prediction is present. Thus, the QP is constant for all the frames.
This is shown in Figure. 3.10. In addition, a temporal sub-sampling of the video can be applied by using a sub-sampling factor of 8. This means that only one frame is coded for every eight frames. For instance, the sub-sampling can be activated using the parameter *TemporalSubsampleRatio* in the configuration file of the VTM.

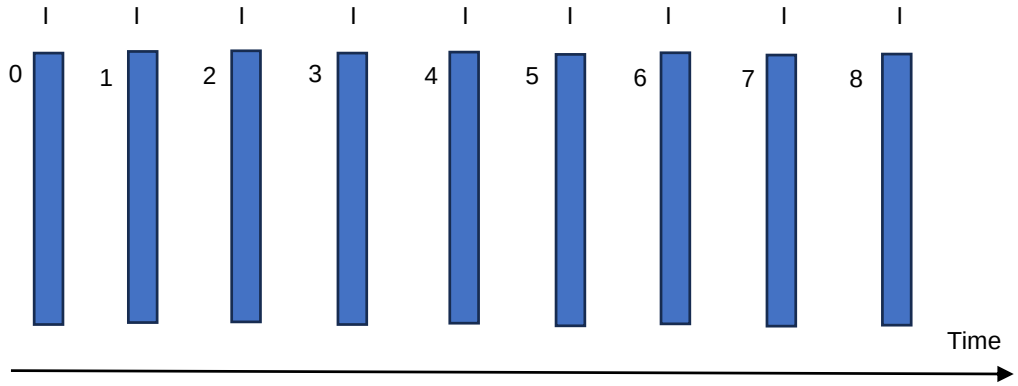


Figure 3.10 Illustration of the AI configuration without sub-sampling

3.2.1.2 Test videos:

The test videos, as defined in Table. 3.1, are divided into 6 classes, namely A1, A2, B, C, D, E, and F. The classes A1 and A2 correspond to the video with a resolution of 3840 × 2180 pixels, class B contains 1080p videos, class C contains videos with a resolution of 832 × 480 pixels, class D has a resolution of 416 × 240 pixels and class E contains 720p videos. The last class, which is class F, represents a combination of several video resolutions.

Table 3.1 Test videos

Class	Videos	Frame rate	Bit depth
A1-3840 × 2180	*Tango2*	60	10
	FoodMarket4	60	10
	Campfire	30	10
A2-3840 × 2180	*CatRobot1*	60	10
	DaylightRoad	60	10
	ParkRunning3	50	10
B-1920 × 1080	*MarketPlace*	60	10
	RitualDance	60	10
	Cactus	50	8
	BasketBallDrive	50	8

	BQTerrace	60	8
C-832 × 480	*RaceHorsesC*	30	8
	BasketBallDrill	50	8
	BQMall	60	8
	PartyScene	50	8
D-416 × 240	*RaceHorses*	30	8
	BQSquare	60	8
	BlowingBubbles	50	8
	BasketBallPass	50	8
E-1280 × 720	*FourPeople*	60	8
	Johnny	60	8
	KristenAndSara	60	8
F- different resolutions	*ArenaOfValor*	60	8
	BasketBallDrillText	50	8
	SlideEditing	30	8
	SlideShow	20	8

3.2.2 Profiling of the intra coding tools

To characterize the effect of the new intra coding tools on the encoder runtime and coding efficiency, the two averages of runtime difference Δt and Δt_{intra}, computed by Equation. (3.3) and Equation. (3.4), the Bjontegaard Delta Bitrate (BD-BR) [22] were measured for all the CTC videos when disabling some combinations of intra coding tools.

$$\Delta t = \frac{1}{4} \sum_{QP_i=22,27,32,37} \frac{t_o(QP_i) - t_r(QP_i)}{t_o(QP_i} \tag{3.3}$$

$$\Delta t_{intra} = \frac{1}{4} \sum_{QP_i=22,27,32,37} \frac{t_{oIntra}(QP_i) - t_{rIntra}(QP_i)}{t_{oIntra}(QP_i} \tag{3.4}$$

Where:

t_o and t_r represent the total encoder runtimes of the VTM-8.0 anchor and the VTM when disabling some combinations of intra coding tools, respectively.

t_{oIntra} and t_{rIntra} are their intra mode decision runtimes.

Table. 3.2 reports the obtained results. As shown in this Table, the ISP is the most time-demanding intra coding tool. Hence, when this last is disabled, the complexity can

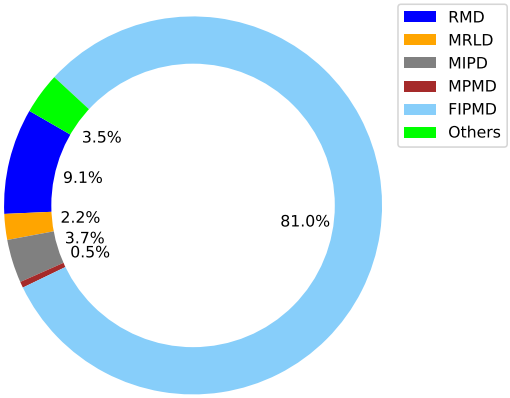
Table 3.2 The effect of the new intra coding tools on encoder runtimes and coding efficiency

MRL	MIP	ISP	BD-BR (%)	Δt (%)	Δt_{intra} (%)
	×	×	0.43	1.92	2.5
×		×	0.58	12.08	16.7
×	×		0,80	15.04	20.3
		×	1.05	14.4	19,4
	×		1.2	16.2	22.0
×			1.4	22.4	30.2
			1.91	28.4	39.3

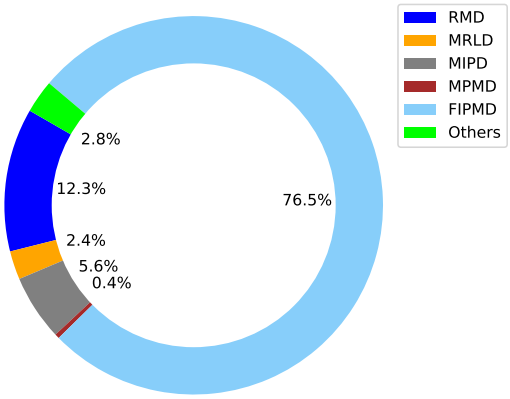
shift down by 15% and 20.3% on average for the total encoder runtime and the intra mode decision runtime, respectively. In addition, the ISP has the highest gain in coding efficiency among all the intra coding tools. This is due to the fact that ISP allows for further partitioning of the intra-predicted block into several sub-partitions, which would improve the prediction from reference samples. Furthermore, the MIP is less time-demanding than the ISP. Consequently, when these two tools are disabled, the encoder runtimes can be reduced significantly but the loss in coding efficiency would become more remarkable. In conclusion, the intra coding tools represent different trade-offs between complexity and coding efficiency. For example, the ISP considerably improves the coding efficiency, while introducing a huge increase in coding complexity. The MIP and MRL are contributing almost equally to the achieved coding efficiency. However, the MIP has the highest coding complexity.

3.2.3 Profiling of the intra mode decision runtime

To identify the most time-demanding steps of the intra mode decision, profiling of this decision runtime was performed on the video *Tango2* with all the intra coding tools enabled. Figure. 3.11 shows the results of this complexity assessment for two QPs. As depicted in this Figure, most of the intra mode decision runtime is related to the FIPMD, with almost 80%. Indeed, besides the additional ISP tests that generate about 3% of additional complexity in the FIPMD, this step uses an overall encoding process in order



(a) QP=22



(b) QP=37

Figure 3.11 Profiling of the intra mode decision runtime in the video *Tango2* for two QPs; (a) QP = 22, (b) QP = 37

to compute the reconstructed block. This latter may take nearly 70% of the intra mode decision runtime and between 80% to 90% of the FIPMD runtime.

3.3 Complexity reduction opportunities in the intra mode decision of VVC

The main focus of this Section is to identify the upper bound of complexity reduction in each step of the intra mode decision while analyzing their impact on the decision

runtime when they are disabled.

3.3.1 Analysis configurations

Since the upper bound of complexity reduction is obtained when the encoder can directly predict the different encoding decisions, an encoding process considering only the best decisions is performed. In this context, the VTM-8.0 encoder [52] is forced to encode only with the best mode(s). For this experiment, two encoding passes were performed as shown in Figure. 3.12. During the first pass, the RDO is used to determine the best

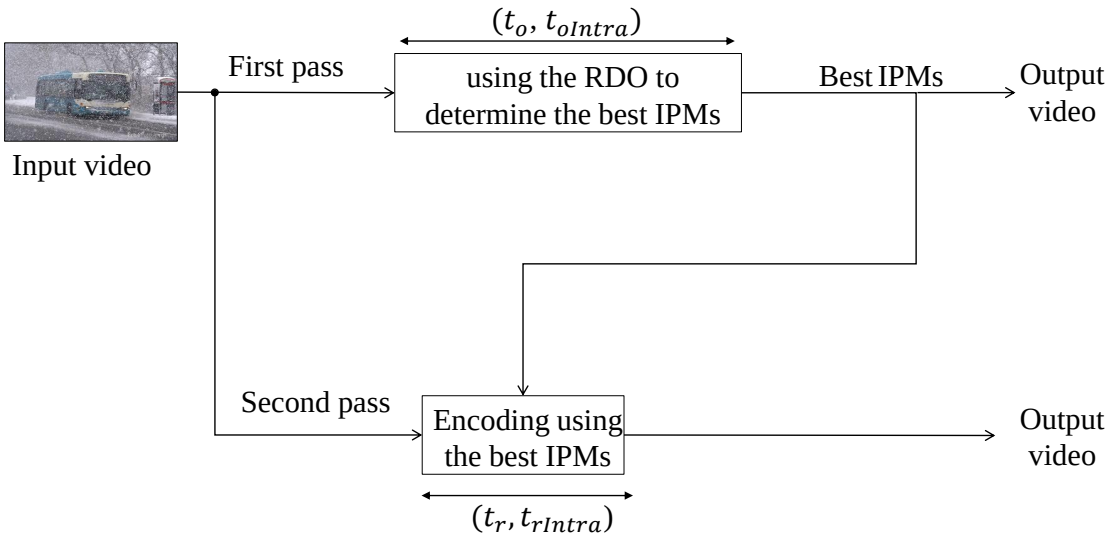


Figure 3.12 Encoding passes to assess the complexity of the different steps of the intra mode decision of VVC

mode(s). Then, this/these latter are extracted. In the second pass, the extracted mode(s) are fed back into the encoder in order to discard the complexity and assess the impact of disabling each step of the intra mode decision. To achieve this, five analysis configurations (C0 to C4), which represent different combinations of intra mode decision steps are defined in Table 3.3. For each configuration, the RDO can be either enabled (E) or disabled (D). Thus, when this last is disabled at an intra mode decision step, only the best mode(s) related to that decision step is fed back into the encoder. The Ref stands for the VTM-8.0 anchor or the first encoding pass. Whereas, the configurations C0, C1, C2, C3, and C4 refer to the RDO being disabled at RMD, at RMD and MRLD, at RMD, MRLD, and

Table 3.3 Analysis configurations

Intra coding tool	Ref	C0	C1	C2	C3	C4
RMD	E	D	D	D	D	D
MRLD	E	E	D	D	D	D
MIPD	E	E	E	D	D	D
MPMD	E	E	E	E	D	D
FIPMD	E	E	E	E	E	D

MIPD, at RMD, MRLD, MIPD, and MPMD, and finally at all the intra mode decision steps including the FIPMD, respectively. It is worth mentioning that the test of non-ISP modes in configuration C4 is maintained, as they intervene in the selection of ISP candidates and the transform core as well.

3.3.2 Analysis of complexity reduction opportunities

The upper bound of complexity reduction in each step of the intra mode decision was evaluated in the intra prediction and the overall encoding process. To this end, the two averages of runtime difference, computed by Equation. (3.3) and Equation. (3.4), were used for each analysis configuration. Thus, the average of total encoder runtime difference Δt between the VTM-8.0 anchor and the VTM when forced to encode only with the best mode(s) is employed for the overall encoding process. Whereas, the average of intra mode decision runtime difference Δt_{Intra} is used for the intra prediction. In this analysis, t_r and t_{rIntra} represent the encoder runtimes of the VTM when forced to encode only with the best mode(s).

After computation of these two averages for each CTC video, they are subsequently averaged for all the 26 CTC videos. Hence, Figure. 3.13 shows these latter as a function of the analysis configuration. The blue curve gives the upper bound of complexity reduction for the overall encoding process. Whereas, the orange curve illustrates it for the intra prediction. As illustrated in this Figure, being able to directly predict the best mode(s) at different steps of the intra mode decision could reduce progressively the complexity of both the intra prediction and the overall encoding process. For example, when disabling only the RMD and MRLD in configuration C1 the upper bound of complexity reduction can reach up to 12% on average in the overall encoding process and 16% in the intra

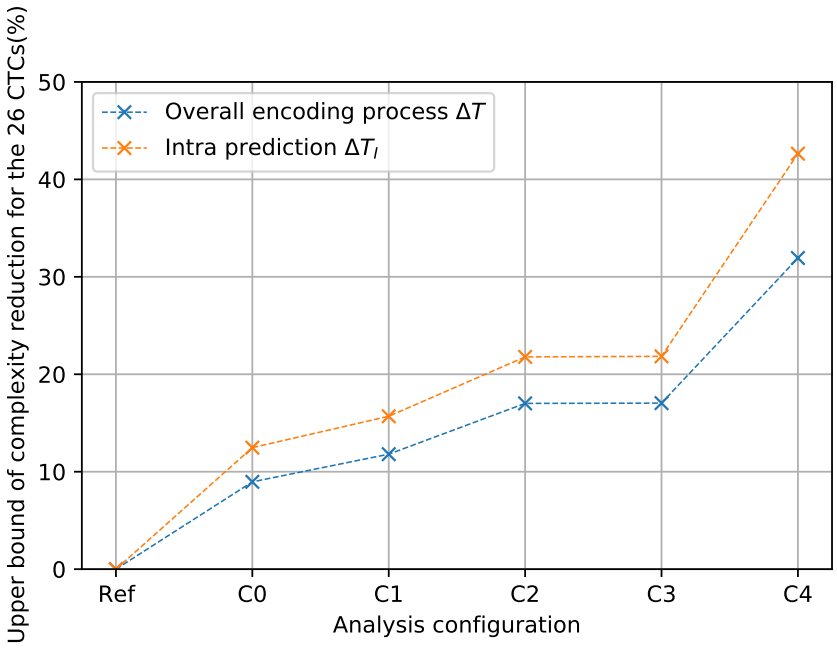


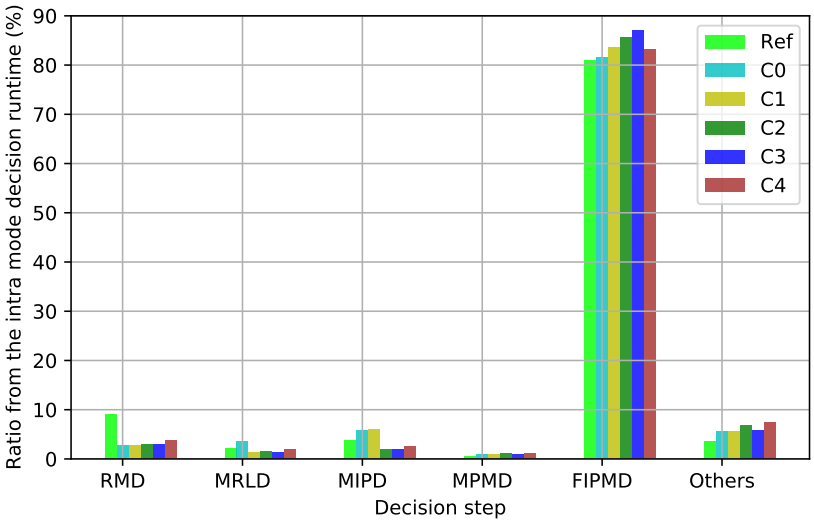
Figure 3.13 Upper bound of complexity reduction in the intra mode decision steps for all the 26 CTC videos

prediction. In addition, most of this reduction, about 43% on average, could be achieved in configuration C4. More precisely, when directly predicting the best IPM at FIPMD too. This fact is mainly due to the heaviest computation of the reconstructed block $\hat{Y}$, as explained in section 3.2. However, this upper bound of complexity reduction could be less significant compared to the VTM-3.0 [55], and this is directly related to the introduction of MIP and ISP tools which generated additional complexity, as shown in Section 3.2. Moreover, the MPMD introduced almost no difference in the total encoder runtime. Indeed, the MPMD consists in adding modes without any further computation. Thus, one can observe that on average only 0.03% of additional reduction can be introduced in the total encoder runtime of configuration C2 when compared to configuration C3.

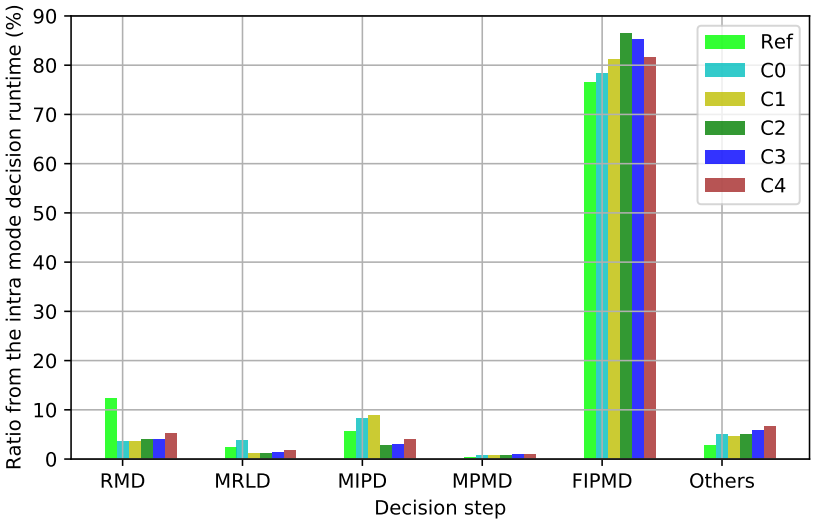
It can also be noticed, from Figure. 3.13, that all configurations showed a higher upper bound of complexity reduction in the intra prediction than the overall encoding process. This is obvious because the overall encoding process still uses the RDO at the partitioning level, which is very time-demanding compared to the intra prediction as explained in [54] and [55]. This fact may also influence the upper bound of complexity reduction in the intra prediction since this step is involved in each iteration of the RDO at the partitioning step. Thus, targeting simultaneously several decision levels in the RDO would lead to

higher complexity reduction.

To conclude, a direct prediction of the best mode at the FIPMD could contribute most to the complexity reduction of the VVC encoder. Hence, the upper bound of complexity reduction can reach up to 43% on average in the intra prediction and 32% in the



(a) QP =22



(b) QP=37

Figure 3.14 Profiling the intra mode decision runtime when disabling each decision step in the video *Tango2* for two QPs; (a) QP = 22, (b) QP = 37

overall encoding process. Moreover, the MIPD and MRLD could introduce additional reductions when they are disabled. These reductions may achieve on average 12% in the intra prediction and 9% in the overall encoding process.

3.3.3 Impact of disabling each step of the intra mode decision

In Figure. 3.14, a profiling of the intra mode decision runtime within the different analysis configurations is conducted on the video *Tango2*. Indeed, during the second encoding pass, the impact of disabling each intra mode decision step is studied.
When analyzing the results of this complexity assessment, it reveals that a direct prediction of the best mode(s) at any step of the intra mode decision could reduce its impact on this decision runtime. For example, when feeding back the best regular modes for the RMD in the configuration C0, its ratio from the intra mode decision runtime decreased significantly. Thus, about 8% decrease is observed for this decision step. The only exception is presented for the FIPMD as it represents the most time-demanding step of the intra mode decision. Hence, disabling it in configuration C4 could enable a significant upper bound of complexity reduction, and thus making its impact on the intra mode decision runtime more noticeable. Table 3.4 illustrates the profiling of the FIPMD runtime

Table 3.4 Profiling of FIPMD runtime when ISP is tested or not tested in the video *Tango2* for two QPs

QP	ISP	Ref		C4	
		intra mode decision runtime (%)	FIPMD runtime (%)	intra mode decision runtime (%)	FIPMD runtime (%)
22	Tested	43.2	53.4	57.0	68.6
	Not tested	37.7	46.6	26.1	31.4
37	Tested	43.1	56.3	60.2	73.8
	Not tested	33.4	43.7	21.4	26.2

when the ISP is tested or not tested in the VTM-8.0 anchor, denoted as Ref, and in the

configuration C4. It displays also its ratio from the intra mode decision runtime in order to provide a clear understanding of its impact on the intra mode decision runtime for both cases.
As it can be observed in this Table, when ISP is tested, the FIPMD impact is more visible in configuration C4 than in Ref. This is due to the fact that the test of non-ISP modes is maintained. More specifically, its ratio from the FIPMD runtime has increased. In this case, the ratio of the ISPD from the FIPMD runtime, even when testing only the best ISP mode, is going to be important in comparison with the achieved upper bound of complexity reduction in this configuration. The same behavior is observed for the enabled decision steps and the different initialization tasks, which are becoming heavier through the simulations. For the MPMD, its ratio from the intra mode decision runtime stabilized at around 1%. In fact, this decision step doesn't affect much the intra mode decision runtime, as explained in Section 3.2.

3.4 Statistical analysis of the selected IPMs

For a better understanding of the most time-demanding step of the intra mode decision, an in-depth statistical analysis [56] of the FIPMD according to the intra coding tool is provided in this section. It describes the ratio of selection for each intra coding tool and its most selected IPMs according to the MT depth and video texture characteristics. In order to represent only the most selected IPMs a threshold was set to 1.5%. Therefore, each IPM that has been selected less than this threshold is not represented.

Table 3.5 Default parameters of the QTMT

QTMT Parameter	Value
Size of the CTU	128×128
MinQTsize	16×16
MaxBTsize and maxTTsize	64×64
MinBTSize and MinTTsize	4
MaxMTDepth	4

In this section, the average ratio of selection of each IPMs for all CTC videos is represented. The results are illustrated only for the QP = 22. In addition, this statistical

analysis was conducted using the default parameters of the QTMT, given in Table. 3.5, where the MT depth is the sum of the two depths of the partitioning trees, namely BT and TT. Its maximum value is equal to 4 unless the size of the current MT leaf node exceeds the frame boundaries. Hence, MT0 represents 64×64 and the maximum MT depth may reach 5 in some cases.

Figure. 3.15 illustrates the average ratio of selection of each intra coding tool according to the MT depth. As it can be seen from this Figure, the FIPMD tends usually to select regular modes regardless of the MT depth. Moreover, the MRL and MIP tools are often chosen when regular modes are not selected. As a consequence, for each intra coding tool, the FIPMD is analyzed.

Figures 3.16, 3.17 and 3.18 display the most selected IPMs when regular, ISP, and MIP tools are selected, respectively. From Figures. 3.16, and 3.17, it reveals that when regular or ISP are used, several modes from the list of the 6 MPMs, as depicted in Figure. 3.3, are the most selected for all MT depths. In fact, the use of the 6 MPMs helps in reducing the signaling overhead of the IPMs in the bitstream [57]. Besides, the IPMs around the bottom-left direction (mode 2) are not commonly used. However, they are more likely to be selected if the MIP is used, as described in Figure. 3.18. This is a result of the shifting

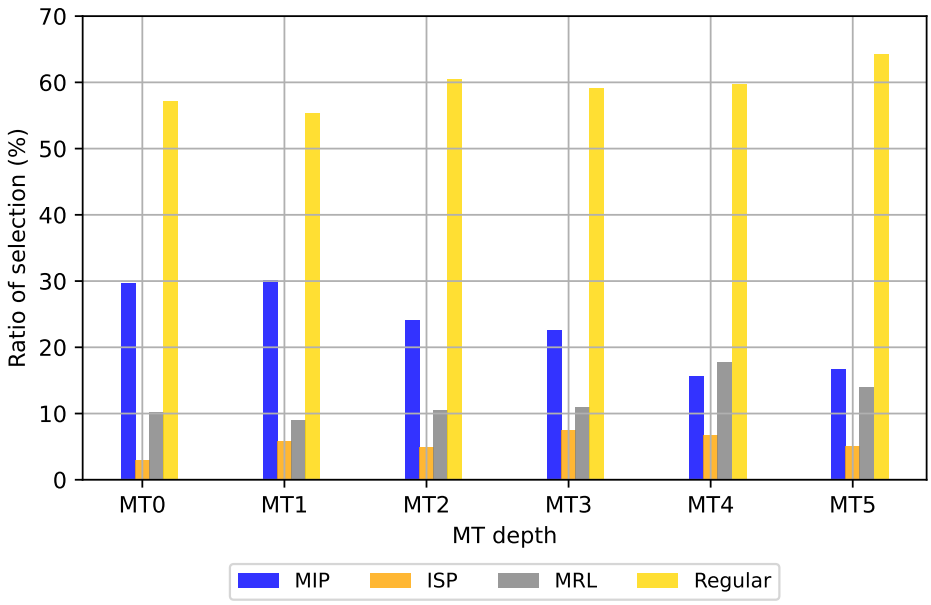


Figure 3.15 Average ratio of selection of each intra coding tool according to the MT depth for QP= 22

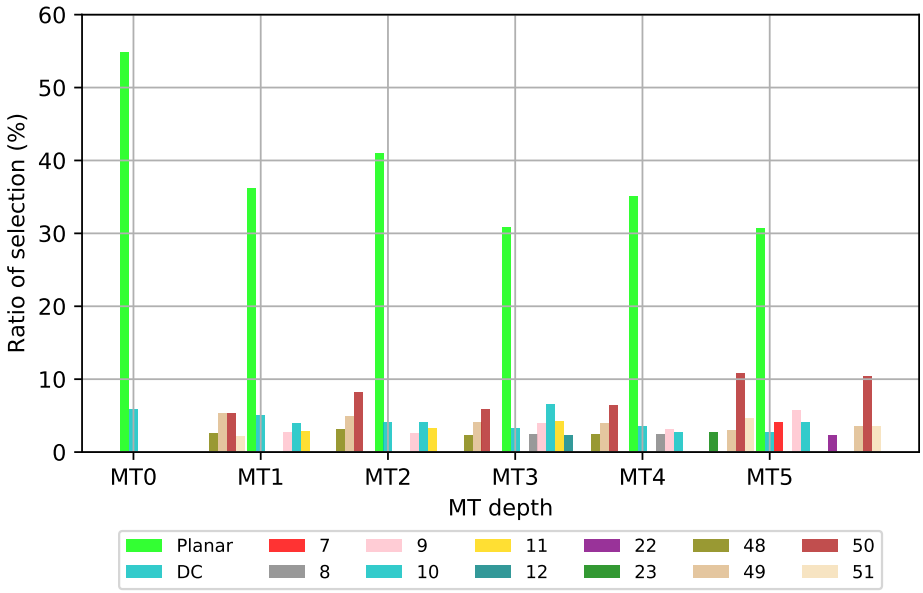


Figure 3.16 Most selected IPMs according to the MT depth when regular modes are selected and QP = 22

operation that has been applied to the other half of the MIP modes during the MIPD. As such, a high likelihood of selection would be granted to these directions. Furthermore, the ISP allows the selection of various directions of intra prediction and this is due to the use of the 6 MPMs along with regular modes. This behavior may result in improving objective visual quality and coding efficiency. Figure. 3.19 highlights the most selected IPMs when the MRL is used. As shown in this Figure, the 6 MPMs other than the Planar are often used for this tool. This fact is explained by the incompatibility of Planar mode with the MRL [20].

All in all, though, the new intra coding tools have shown different FIPMD behavior as they are using various sets of IPMs. As an example, the MIP represents a simplified neural network-based intra prediction, where the best mode is selected among a set of 32 trained modes. The MRL is also a good example since it considers only the 6 MPMs modes. The only exception is shown for the ISP as it considers a list of IPMs created based on the regular modes. As such the FIPMD, when ISP is used, may depend strongly on these modes. When studying also the results of the statistical analysis for all the intra coding tools it can be noticed that additional IPMs, such as 22 and 23, are slightly used along with the most selected IPMs and this is due to the texture richness of some CTC videos

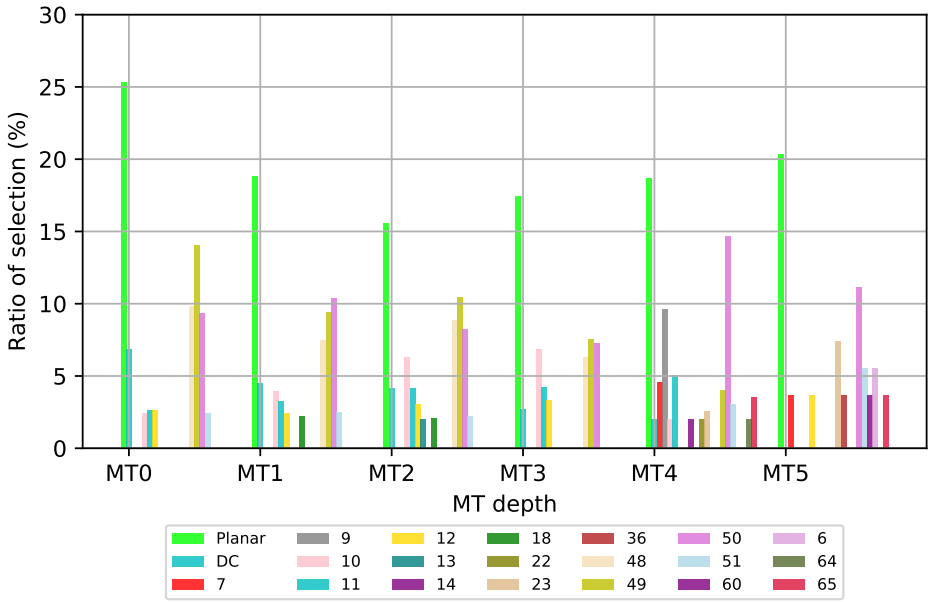


Figure 3.17 Most selected IPMs according to the MT depth when ISP is selected and QP = 22

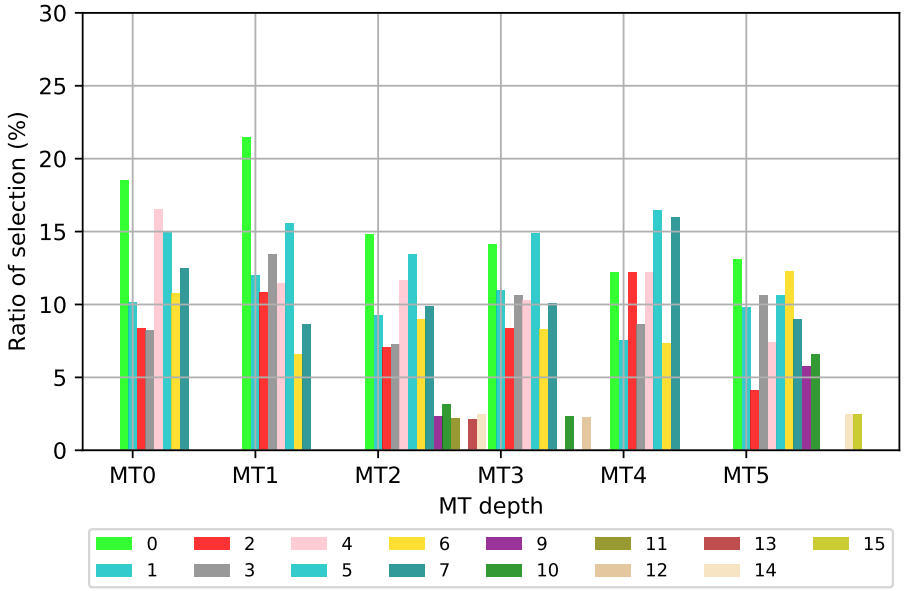


Figure 3.18 Most selected IPMs according to the MT depth when MIP is selected and QP = 22

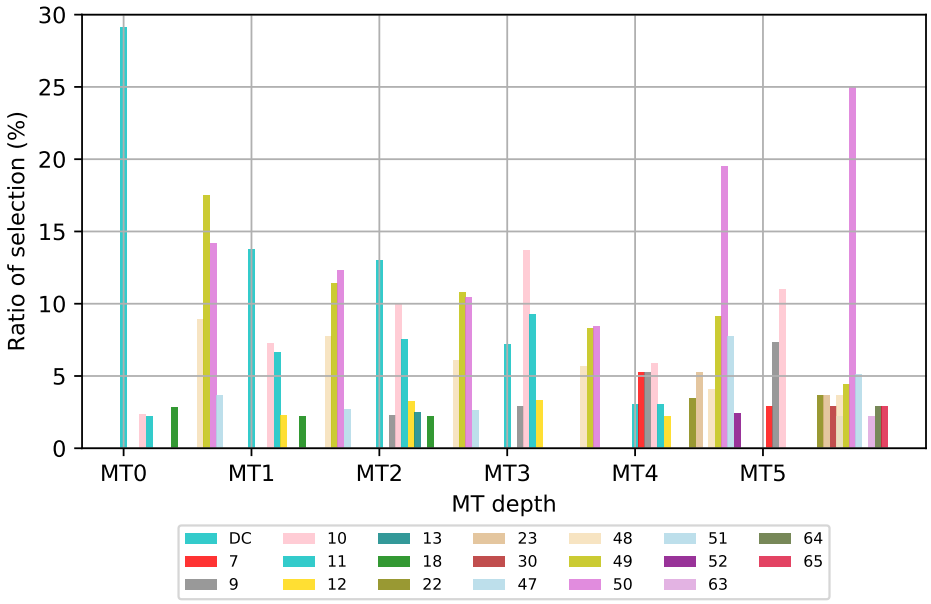


Figure 3.19 Most selected mode according to the MT depth when MRL is selected and QP = 22

and the diversity of their contents.

Conclusion

In this chapter, a complexity assessment of the intra prediction was conducted. This latter first identifies the most time-demanding intra coding tools and intra mode decision steps. Then, the upper bound of complexity reduction in each of these steps and tools is analyzed. Finally, an in-depth statistical analysis of the most time-demanding step of the intra mode decision, which is the FIPMD, is given. Results showed that these latter represent varying degrees of complexity. Thus, revealing several complexity reduction opportunities. For instance, the complexity of the VVC can be reduced by up to 43%. Also, the FIPMD tends usually to select the 6 MPMs. However, this selection may depend on the used intra coding tool and video texture characteristics.

Part II

Fast intra modes decision frameworks

Chapter 4

Machine learning-based techniques for complexity reduction

Introduction

Artificial intelligence emerged in 1956 as the field of computer vision, which enables machines to make intelligent decisions or take intelligent actions just like a human being does. For instance, artificial intelligence allows an intelligent agent (hardware, software, robot, or an application) to cognitively perceive its environment and, therefore, maximize its probability of success in a target task. Over the course of this chapter, the most important sub-fields of artificial intelligence are discussed while reviewing the state-of-the-art techniques in fast encoding decisions.

4.1 Machine learning

Machine learning is the subset of artificial intelligence in which machines can learn and adapt to a task automatically without being explicitly programmed. This process is referred to as learning through experience. The difference between machine learning and deep learning, which will be covered in Section 4.2, is that machine learning algorithms or models need minimal human interference, also known as feature engineering, to handle the input data more efficiently.

4.1.1 Life cycle

The machine learning life cycle refers to the entire workflow through which a machine learning system is developed [58]. It's comprised, as shown in Figure 4.1, of a series of

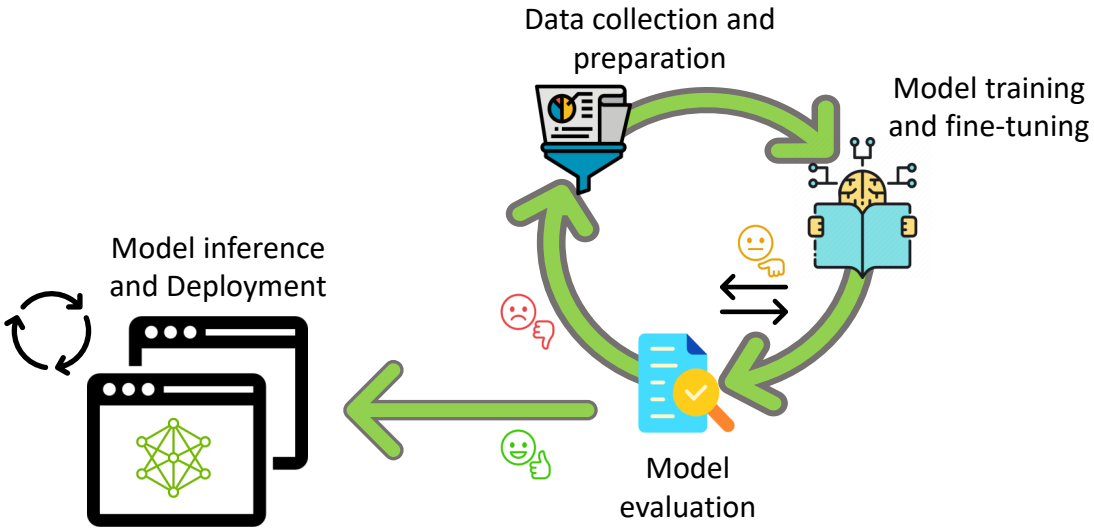


Figure 4.1 Machine learning life cycle; first training, validation, and test samples are collected. Then, the selected model is trained, fine-tuned and evaluated on the collected data set in order to be deployed in the real-world environment

steps, starting from data collection to inference or model deployment in a real-world environment.

Data collection: Data collection is a fundamental step in machine learning. It involves identifying which sort of data can be relevant for a model to draw reliable conclusions in a target task. This can be achieved through the use of existing data sets or the creation of new ones. Since the learning process in machine learning is data-driven, data may undergo several data preprocessing or transformations [59] before being used to train the desired model. For instance, the quality of the data fed into a model can strongly impact its performance. Thus, almost 90% [60] of the machine learning life cycle is devoted to data collection and preparation, especially when creating new data sets.

Model training: After collecting and preparing the selected data set, the next step would be to train a model on the target task. Typically, the learning process starts by defining a baseline algorithm or model architecture that could address the target task. Then, this baseline algorithm is trained to learn a function, also known as a hypothesis, that maps an output (e.g. label, cluster, or action) to an input data [61]. More preciously, as new training samples flow through the model, this latter tries to improve its perfor-

mance in the target task by learning the data representation and minimizing its error rate on the target task. Finally, one can fine-tune the hyper-parameters of the baseline model in order to create more powerful algorithms and enhance the overall performance.

Model evaluation: As stated previously fine-tuning may lead to the creation of several model architectures, that we have to choose from. To some extent, model selection may be addressed during training by evaluating the performance of each model architecture on a subset of the training data set, also called the validation data. Aside from choosing the best model architecture, model evaluation can help in spotting the true model performance on new data, also known as test data [62]. This latter would help in ensuring the model's generalization and reliability before deploying it into the real-world environment.

Model inference: The last step of the machine learning life cycle is the model inference or deployment. It encloses all the needed requirements to integrate the trained model in the real-world environment [59]. This includes releasing the model, so it can provide business value, as well as monitoring its performance on real-world data.

4.1.2 Learning settings

Machine learning uses a variety of algorithms to address learning tasks. These latter may be supervised, unsupervised, semi-supervised, or reinforcement learning depending on the task to be solved [12].

Supervised learning: In supervised learning, the model learns a function that maps an output label to the input data according to several input-output pairs or samples. More preciously, the learning process of a supervised algorithm needs external assistance to figure out the prediction class or value for each training sample. In the last decade, several supervised learning algorithms have been introduced, such as Decision Tree (DT), Support Vector Machine (SVM) [63], Random Forest Classifier (RFC) [64], etc...

Unsupervised learning: Unlike supervised learning, an unsupervised learning algorithm does not have access to the true labels or the ground truth of the training data. Consequently, it analyzes non-labeled data and drowns inferences in order to find distinct clusters for each sample's group. Thus, the training samples are grouped into clusters in such a way that the samples within a group are as similar as possible, while the samples

in different groups are as different as possible. One of the most well-known unsupervised learning algorithms is K-means [65], where k centers or clusters are defined. Then, each training sample is associated with the nearest center.

Semi-supervised learning: As its name indicates, semi-supervised learning lies between supervised and unsupervised learning. First, a limited set of labeled data is used to train a weighted model to predict the label of similar data that have not yet been labeled, also known as unlabeled data. These latter are then feedback into that model as new training samples. Hence, semi-supervised learning uses both labeled and unlabeled data to fit a model, which is very relevant for scenarios where labeled data is insufficient or hard to get [66]. In these cases, adding the unlabeled data may improve the accuracy of the model and prevent model failure due to the lack of sufficient labeled data.

Reinforcement learning: Reinforcement learning is a type of unsupervised learning that uses a reward function to predict efficiently its action at each state, denoted as s. So, given a reward function r_s a reinforcement learning algorithm tries to predict its next action a, which maximizes the reward function r_s of its next state s_{next} until reaching a terminal state, referred to as s_t. Reinforcement learning is usually used in playing chess games, building intelligent robots, or automated vehicles [67].

4.1.3 Supervised learning

As detailed in Section 4.1.2, supervised learning is one of the most popular learning paradigms in machine learning. This latter allows machines to draw inferences from labeled input data in order to predict the output value more accurately. In supervised learning, two types of problems or tasks to be solved can be distinguished: regression and classification.

— Classification: In a classification task the learning algorithm tries to fit the input data into different classes. In other words, the algorithm analyses the input data and tries to predict its class membership. Depending on the number of classes, the classification task can be binary or multi-class. In binary classification, the outcome of the algorithm is either "*Yes*" or "*No*". Whereas, in multi-class classification, the input data can be categorized into three or more classes [68].

— Regression: In a regression task, the learning algorithm deals with continuous output values. Unlike classification, regression tries typically to predict future values depending on the provided input samples. The most basic form of regression is

known as linear regression and it attempts to fit a straight line to the data when the relationship between them is linear [69].

There are plenty of supervised machine learning algorithms. So in this Section, we'll go through some of them.

4.1.3.1 Linear regression

Linear regression is a widely used machine learning algorithm that allows for studying the relationship between two types of variables, namely independent and dependent variables [70]. The independent variables also called the inputs, are all variables that may affect the dependent variable, also known as the output. As shown in Figure 4.2, the purpose of linear regression is to make predictions about the future by finding the best-fitting line through the input data. It can be presented as a linear function of the input data, as

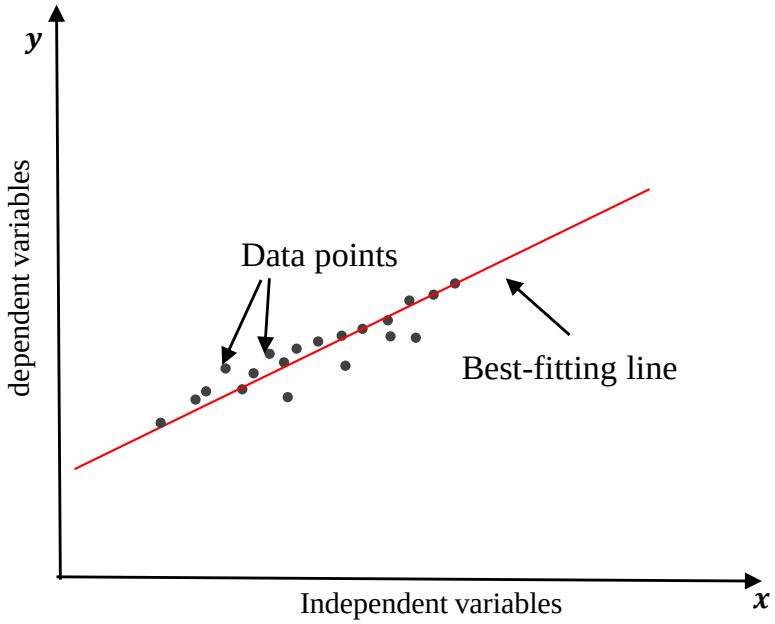


Figure 4.2 Illustration of linear regression

given in Equation. (4.1).

$$f_{\boldsymbol{\theta}} = \sum_{i=0}^{n} \boldsymbol{\theta}_i \times \mathbf{x}_i \tag{4.1}$$

where:

$\mathbf{x}$ refers to the independent variables.

$\boldsymbol{\theta}$ denoted the learned weights.

Hence, based on the number of independent variables, linear regression can be extended to multiple linear regression, where several independent variables are used to predict the output.

4.1.3.2 Decision Tree

The Decision Tree (DT) is a rule-based supervised machine learning algorithm. It categorizes samples in the data set into different classes by posing several questions about the provided features related to it. One of the simplest forms of a DT, as shown in Figure 4.3, would be to ask yes/no questions about the current sample. Depending on the answers to these questions, a DT may split the data hierarchically into "*yes*" and "*no*" samples until all the samples in a class are as homogeneous as possible. Each question is thereby an internal node in the DT and the answer to it are the children. The uppermost node in a DT is called the "*root*". While the last nodes are referred to as the "*leaves*". These latter usually represent the classes to which a sample may belong. Hence, following the path from the root to a leaf node, a sample can be classified based on the answers that apply to it [71].

Since the goal of a DT is to split the data set into small subsets with nearly homogeneous samples, several measures can be applied in order to ensure the purity of the resulting subsets. Assuming that $\mathbf{p}_i$ is the proportion of samples that actually belong to the class i, entropy, calculated by Equation. (4.2), for example, is lowest when a subset is impure means it contains samples of different classes while it's maximized when a subset is pure, which means it contains samples from a single class.

$$Entropy = -\sum_{i=1}^{n} \mathbf{p}_i \log(\mathbf{p}_i) \tag{4.2}$$

Another measure of impurity is the Gini coefficient or the Gini index. This latter is determined using the Equation. (4.3) and it varies usually between 0 and 1, where 0 indicates a pure classification and 1 implies a random distribution of the training samples across the different classes.

$$Gini \quad index = 1 - \sum_{i=1}^{n} (\mathbf{p}_i)^2 \tag{4.3}$$

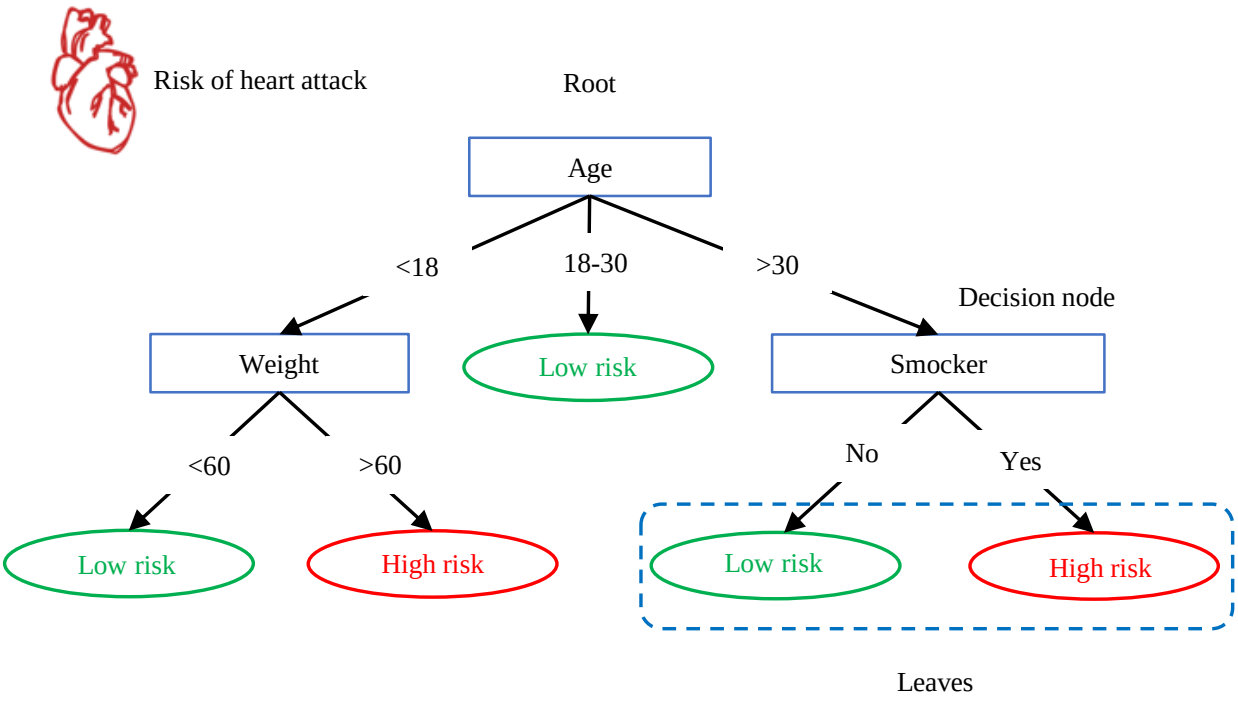


Figure 4.3 Data splitting in a DT

4.2 Deep learning

Deep learning is one of the widely used sub-fields of artificial intelligence, that appeared around 2010 [72]. In contrast to machine learning, deep learning implies more standalone algorithms, that can handle large-scale data sets efficiently. This latter is one of the main reasons for its popularity, especially for tasks like image classification, and computer vision, as well as Natural Language Processing (NLP) [73].

4.2.1 Deep Neural Network

Deep learning is usually referred to as Deep Neural Network (DNN). In fact, it allows computational models, made up of multiple layers of non-linear processing units, also known as neurons, to learn data representations with various levels of abstraction [74].

As shown in Figure 4.4, neurons are stacked together into several layers, each of which provides the next one with relevant information about the input data, also known as the features map. This latter got multiplied with the learned weights $\mathbf{\Theta}_i$ and adds to a bias

$\boldsymbol{a}_i$ at each layer. Then, an activation function is applied to decide which neuron should be fired to predict the output label. Typically, a DNN is comprised of three types of layers :

- Input layer: This layer is the first layer in a DNN. It feeds the model with the needed input data and passes it to the hidden layers.
- Hidden layers: The hidden layers are the most important type of layers in a DNN. They can be either one or more layers and they are used to draw inferences or learn features from the input data.
- Output layer: This layer is the last layer in a DNN. It looks at the features extracted by the hidden layers and determines the output label.

An image, for example, takes the form of an array of pixels, and the features learned in the first hidden layer typically represent the presence or absence of edges at particular orientations and positions in the image. The second layer often recognizes patterns by spotting particular arrangements of edges, regardless of small variations in edge positions. The third layer can assemble these patterns into larger combinations, that correspond to the parts of each object, and subsequent layers would detect objects as combinations of these parts. Deep learning, therefore, relies on the fact that these feature maps are not designed by human engineers. But, they are learned directly from raw data.

There is a multitude of successful deep learning algorithms, that have been introduced in the field of artificial intelligence such as Convolutional Neural Networks (CNNs) [75], Recurrent Neural Networks (RNNs) [76], auto-encoders [77], transformers [78] and Generative Adversarial Network (GAN) [79].

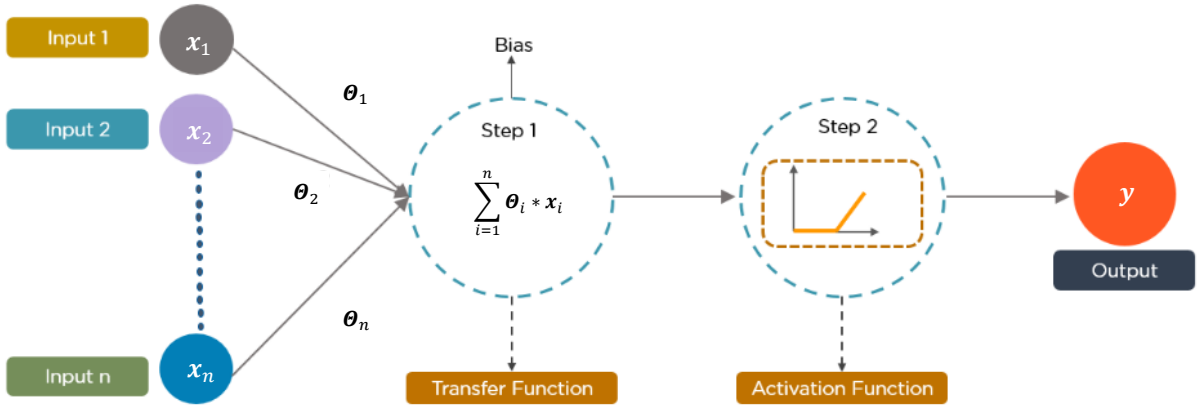


Figure 4.4 Neural network architecture

4.2.2 Back propagation

As explained in Section 4.2.1, a DNN consists of a high number of interconnected neurons, that learn data representation in a feed-forward basis. This means that each layer of neurons sends relevant information or signals about the input data to the next one. Then, the error rate propagates backward in order to improve the stimulation of each neuron in response to the input data. This process is known as backpropagation and it appeared around 1986 [80]. The intuition behind backpropagation is to define a cost function, also known as loss function l, and use an optimizer, such as Adam [81] or Gradient Descent Search (GDS) [82], to adjust the set of learned weights $\mathbf{\Theta}_i$ and optimize the model performance in the target task [83]. The first iteration of the backpropagation usually starts with random weights then these weights are fine-tuned over and over again based on the training samples until the error rate is minimized.

4.2.2.1 Gradient Descent Search

The Gradient Descent Search (GDS) is one of the most well-known optimization algorithms used in deep learning [82]. It tries to find the optimum weights $\mathbf{\Theta}_i$ subject to the constraint of a minimum error rate. The primary function of a gradient is to measure the change in each weight against the change in the error rate or the loss function. Mathematically speaking, gradients are defined as the slope of a curve at a given point in a specified direction. So, this slope will be steeper the higher the gradient, which will improve the model's training because it can learn quickly. However, the model will stop learning when the slope becomes zero.

As shown in Equation. (4.4), the gradient is usually described as a partial derivative with respect to its inputs.

$$\nabla \mathbf{f}(y) = \begin{bmatrix} \frac{\partial \mathbf{f}(y)}{\partial \mathbf{x}_1} \\ \frac{\partial \mathbf{f}(y)}{\partial \mathbf{x}_2} \\ \vdots \\ \frac{\partial \mathbf{f}(y)}{\partial \mathbf{x}_n} \end{bmatrix} \tag{4.4}$$

As illustrated in Figure 4.5, the GDS repetitively updates the learning weight $\mathbf{\Theta}_i$ using the gradient of the loss function with respect to the training data set and scales it by the learning rate α. Because we aim to minimize the loss function, this latter subtracts the result from the learning weight in order to take a new step toward the direction of the steepest decrease of the loss function l. This process can be typically written following

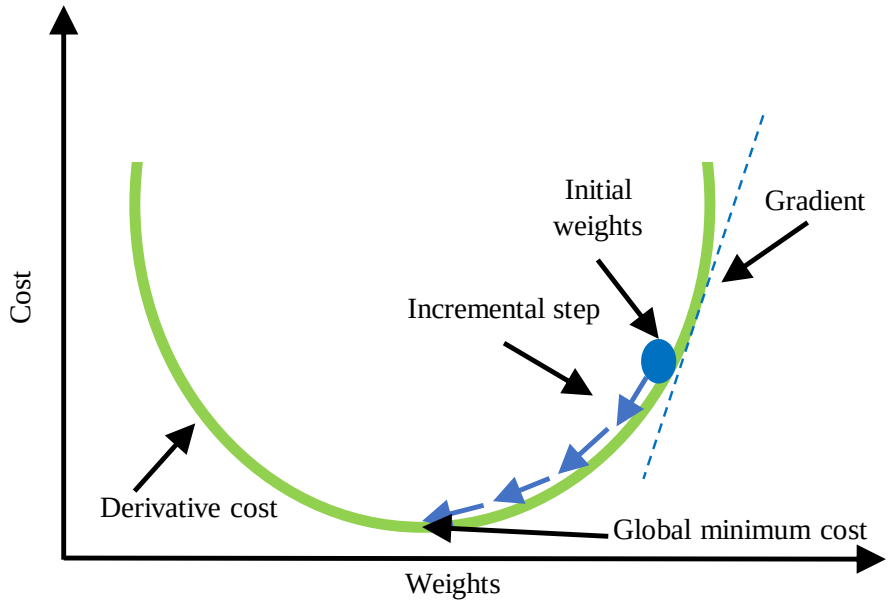


Figure 4.5 Illustration of GDS

the Equation. (4.5).

$$\boldsymbol{\Theta}_i = \boldsymbol{\Theta}_{i-1} - \alpha \times \frac{\partial l(\boldsymbol{\Theta}_{i-1})}{\partial \boldsymbol{\Theta}_i} \tag{4.5}$$

Gradient descent indeed has three popular variants that differ in how much data is used to compute the gradient of the loss function $\nabla l(\boldsymbol{\Theta})$. For instance, the variant presented in Equation. (4.5) is called batch GDS and it performs a parameter update over the entire training data set. The second variant, defined in Equation. (4.6), is the stochastic GDS and it performs a parameter update for every single training sample in the data set.

$$\boldsymbol{\Theta}_i = \boldsymbol{\Theta}_{i-1} - \alpha \times \frac{\partial l(\boldsymbol{\Theta}_{i-1}, \mathbf{x}_j, \mathbf{y}_j)}{\partial \boldsymbol{\Theta}_i} \tag{4.6}$$

The last variant is called mini-batch GDS. This latter performs a parameter update over m training samples in the data set and is defined by Equation. (4.7).

$$\boldsymbol{\Theta}_i = \boldsymbol{\Theta}_{i-i} - \alpha \times \frac{\partial l(\boldsymbol{\Theta}_{i-1}, \mathbf{x}_{j\in 0..m-1}, \mathbf{y}_{j\in 0..m-1})}{\partial \boldsymbol{\Theta}_i} \tag{4.7}$$

4.2.2.2 Adam optimizer

Adam optimizer or Adaptive Moment Estimation (ADAM) [81] is an improved version of GDS that helps in ensuring the convergence of a deep neural network. Since many computer vision problems may represent a very noisy or non-convex loss function, the simple GDS may fall into a local minimum instead of a global one. For this reason, the ADAM optimizer has been introduced to adapt the GDS to non-stationary problems with very noisy and/or sparse gradients.

So, instead of having a fixed learning rate or gradient step size for all the training samples, ADAM adaptively adjusts the learning rate using the first and the second moments of the loss function $l(\mathbf{\Theta})$. Typically, the first moment is the mean of the probability distribution of $l(\mathbf{\Theta})$, and the second moment is the central point or the median value. The weights update is then performed as in Equation. (4.8).

$$\mathbf{\Theta}_i = \mathbf{\Theta}_i - \alpha \times \frac{\hat{\mathbf{m}}_i}{\sqrt{\hat{\mathbf{v}}_i} + \epsilon} \tag{4.8}$$

where:

$\hat{\mathbf{m}}_i$ is the first moment and it is defined by Equation. (4.9).

$$\beta_1 \times \hat{\mathbf{m}}_{i-1} + \frac{1-\beta_1}{1-\beta_1^i} \times \frac{\partial l(\mathbf{\Theta}_{i-1})}{\partial \mathbf{\Theta}_i} \tag{4.9}$$

$\hat{\mathbf{v}}_i$ is the second moment and it is computed by Equation. (4.10).

$$\beta_2 \times \hat{\mathbf{v}}_{i-1} + \frac{(1-\beta_2)}{1-\beta_2^i} \times (\frac{\partial l(\mathbf{\Theta}_{i-1})}{\partial \mathbf{\Theta}_i})^2 \tag{4.10}$$

β_1 and $\beta_2 \in [0..1]$ are hyper-parameters to the ADAM optimizer.

ϵ is a very small number, typically between $1e^{-8}$ and $1e^{-10}$.

4.2.3 Convolutional Neural Network

CNN is a commonly used architecture for deep learning. It is often used to recognize objects and scenes or perform object detection or segmentation. They could learn directly from image data, which eliminates the need for feature engineering. The CNN may have dozens or hundreds of layers, each of which is used to learn or detect different features in an image. For instance, filters are applied to each training image at different resolutions, and the output of this operation is used as the input to the next layer. These filters can

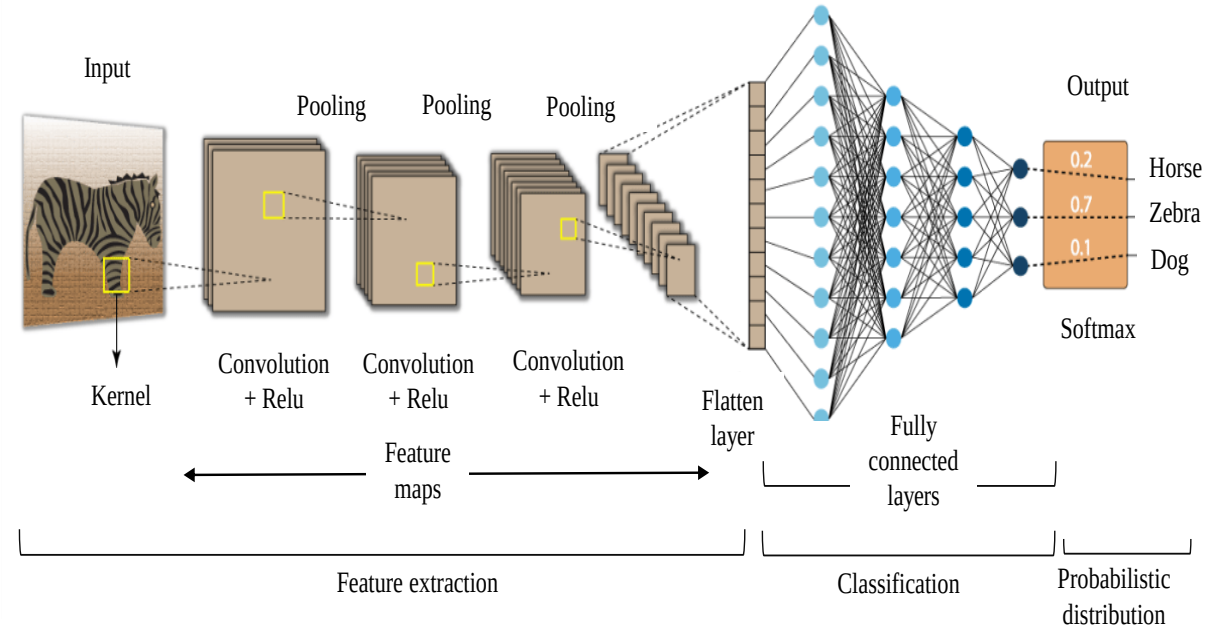


Figure 4.6 Overall architecture of deep CNN [2]

start as very simple features, such as brightness and edges, and increase in complexity to cover features that uniquely define an object as the layers progress. A typical architecture of a CNN, as shown in Figure 4.6, consists of four different types of layers [75].

— The convolution layers with Rectified Linear Unit (ReLu).
— Pooling layers.
— Flatten layers.
— Fully connected layers followed by Softmax.

4.2.3.1 Convolutional layer

The convolution layer is the core layer of a CNN. It represents its first hidden layer. Indeed, this layer allows a CNN to extract from an input, usually an image, a set of low-level features (e.g. simple edges) or high-level features (e.g. complex regions and shapes), which are called feature representations or feature maps.

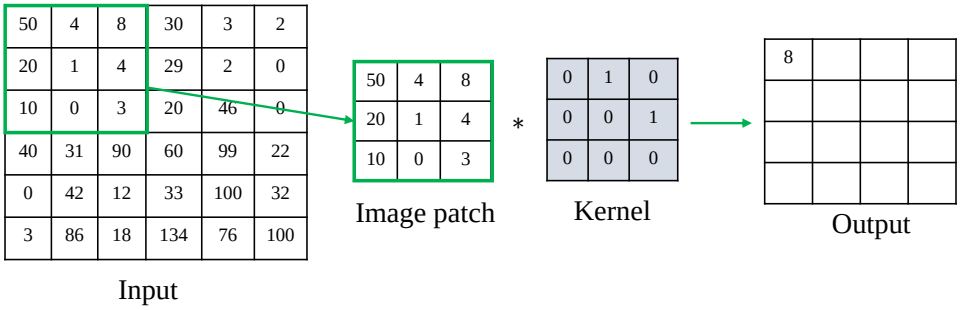
To extract a particular feature, a convolutional layer uses a "*filter*", also known as a

kernel. The objective of this filter is to identify the presence or not of a set of features in the input image. Hence, each filter has a size of $n \times n \times d$ and a "*stride*" denoted as s.

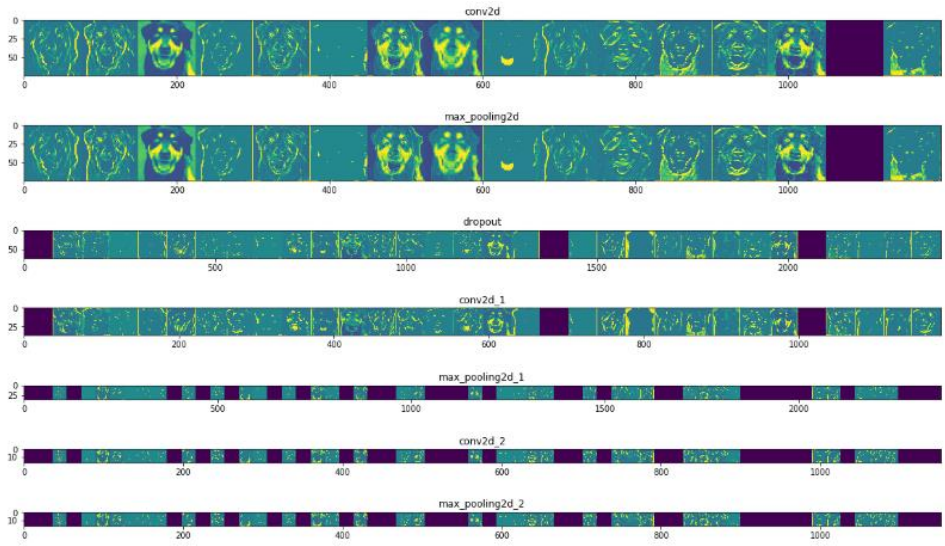
where:

d is the dimension of the image or the number of channels.

And s defines the number of columns or rows of pixels to shift when sliding the filter around the input image.



(a) Convolutional operation



(b) Feature maps

Figure 4.7 Illustration of the convolution operation; (a) convolutional operation, (b) an example of extracted feature maps of dogs vs. cats CNN

As shown in Figure 4.7 (a), as the filter slides around the input image it multiplies the values in the filter with the original pixel values in the image to give each convolution a unique number. This number can be too large which means that the feature to be extracted by the filter is present in that target region, or it can be too small (i.e. zero) otherwise.

4.2.3.2 Rectified Linear Unit

ReLu defines the activation function of a neuron in a hidden layer. By the term "*activation*" we mean the response of a hidden layer i to the input of a previously hidden layer or an input layer (if it is the first layer of the neural network). It allows, as shown

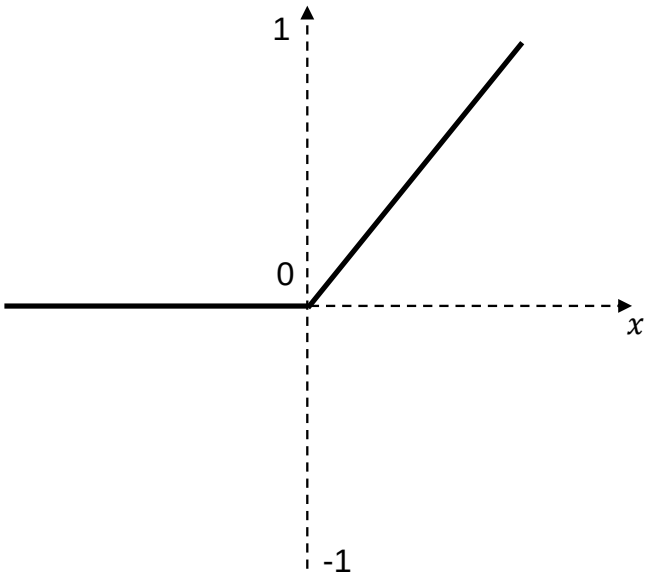


Figure 4.8 Illustration of ReLu

in Figure 4.8, to cancel all the negative values received in the input and it is defined by the Equation. 4.11 :

$$f(x) = max(0, x) \tag{4.11}$$

4.2.3.3 Pooling layer

This layer is used to reduce the spatial dimension of the feature map. It takes as input the output of the convolutional layer and applies the pooling operation on it.

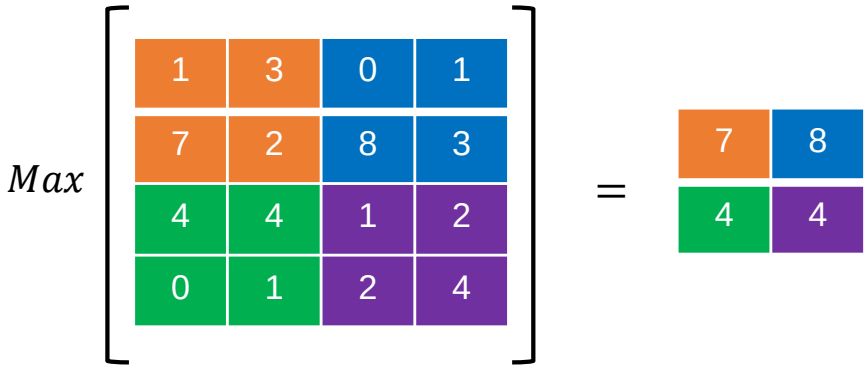


Figure 4.9 Illustration of max pooling; n is equal to 2 and s is set to 0

So, the objective of this layer is to reduce the dimension of the feature map, which means reducing the number of parameters and accelerating the training of the model while preserving the important features. One of the most common forms of pooling is "*max pooling*", which consists, as illustrated in Figure 4.9, in choosing the maximum pixel value according to a stride s in each $n \times n$ pixel region.

4.2.3.4 Flatten layers

After performing the pooling operation, a reduced multidimensional feature map is generated. Since the final classification layers in a CNN, also known as fully connected layers, accept only a one-dimensional vector, the flatten layers will turn the reduced multidimensional feature map into a one-dimensional vector in order to feed it to the fully connected layers. So the intuition behind flattening is to help us link the output of the convolutional layers/pooling layers to the fully connected layers.

4.2.3.5 Fully connected layers

The fully connected layers are the last layers of a CNN. As shown, in Figure 4.10, these layers refer to the layers in which each neuron is interconnected to every neuron in the

next layer. Indeed, the fully connected layers look at the output of the previous layer and determine the most significant features for each prediction class. So, each neuron in a fully connected layer corresponds to a single feature that may be present in the image. The value that the neuron sends to the next layer represents the likelihood that the feature is present in the image.

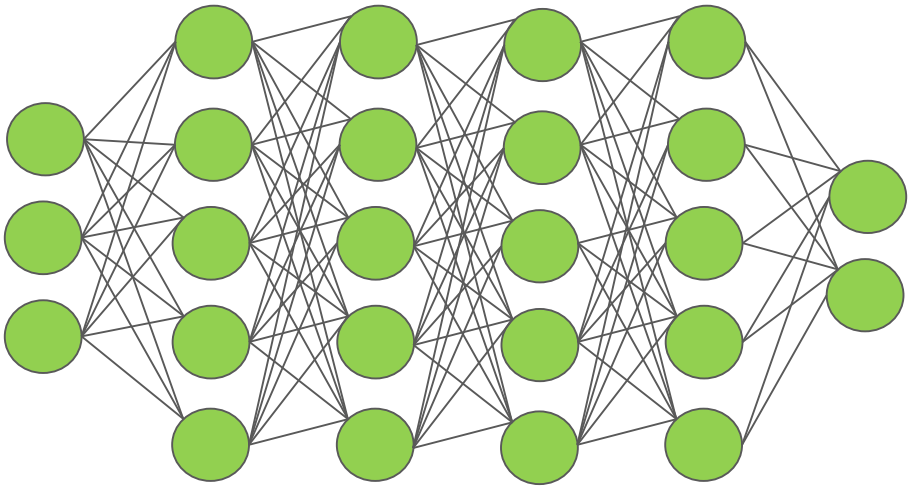


Figure 4.10 Fully connected layers

4.2.3.6 Softmax

The output layer of a CNN typically produces real-valued scores that are not easily scaled and may be challenging to deal with, so the softmax function, appended to the output layer, takes these real-valued scores and normalizes them between 0 and 1 so that the user can interpret them as a valid probability distribution. This latter tells us the likelihood that one prediction class is selected among all the prediction classes for a training sample and it is defined by the Equation. 4.12.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_k}} \quad for \quad i, j \in [1..k] \tag{4.12}$$

where:

k is the number of prediction classes.

$\mathbf{z} \in \mathbb{R}$ denotes the real-valued score.

4.3 Multi-Task Learning

Multi-Task Learning (MTL) is the sub-field of artificial intelligence, in which a single model is trained on a set of multiple tasks simultaneously [84]. This is done through the use of a shared feature presentation, that exploits similarities or common knowledge among all tasks.

4.3.1 Background and motivation

In order to perform MTL there are two important questions to ask, "*what to share?*" and "*how to share it?*". The "*What to share?*" specifies the form in which the common knowledge is shared among all tasks. This latter includes features, instances, and weight sharing. Instances sharing consists in identifying training instances or samples in a task that can be useful to share knowledge with other tasks. This approach is the least used in the state-of-the-art. Features sharing on the other hand is commonly used and it consists in learning the common features presentation among all tasks. More precisely, a new presentation of common features among all tasks is elaborated from the original feature presentation of every single task. Then, this latter is used to learn the different tasks. The last sharing form is weight sharing, which consists on using the model weights in one task in order to learn the model weights for the other related tasks.

After identifying the form in which the learning is shared, the "*how to share?*" refers to the concrete methods or ways in which the similarities or the common knowledge among all tasks could be shared [84]. as shown in Figure 4.11, the sharing of common knowledge among all tasks can be achieved using two main ways:

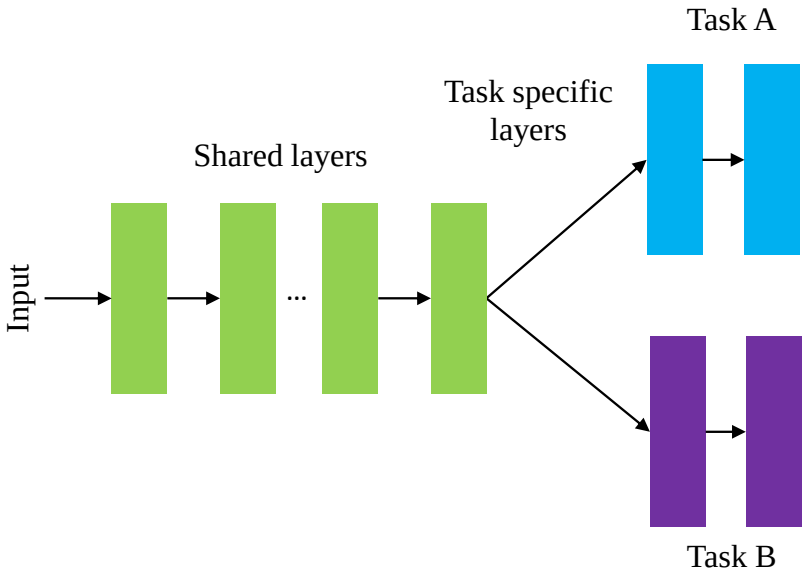
— Hard sharing: This type of sharing is the most used in MTL. It is applied throughout the sharing of model weights among all tasks while keeping for each task its own output. This method has the advantage of avoiding the risk of over-fitting.

— Soft sharing: This type of sharing is the least used in MTL. It consists in designing a model for each task, which will have its own weight. Then, the distance between the weights of all models is regularized to encourage similarity between tasks.

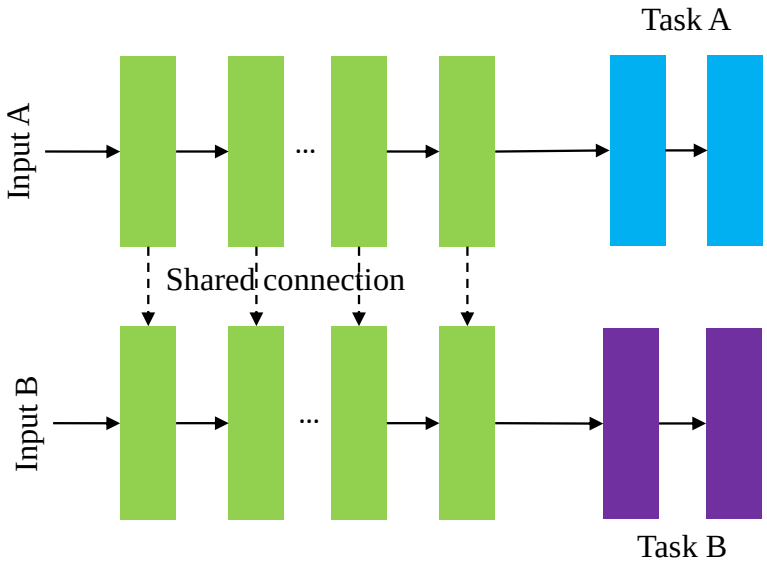
The idea of MTL has been motivated by the learning process of human beings, who learns new tasks by exploiting their prior knowledge of other related tasks. Indeed, using this learning paradigm could offer several advantages such as:

— Achieving better generalization, which reduces the risk of over-fitting through implicit data augmentation.

— Highlighting the importance of a feature in learning a task through the use of additional information provided by learning other tasks.
— Learning a task with the help of another.
— Being able to learn new tasks in relation to prior tasks.



(a)



(b)

Figure 4.11 Forms of sharing in MTL; (a) Hard sharing; (b) Soft sharing

MTL shares the same settings as Machine Learning (ML) in general. These are respectively classification and linear regression, which can be supervised, unsupervised, semi-supervised or reinforcement learning, etc. However, MTL can be grouped into two main categories depending on the type of tasks to be learned by the model [85]. The first category is called homogeneous MTL, in which the model considers several tasks of the same setting. More precisely, the tasks learned simultaneously can be either classifications or regressions, not both of them. These classifications or regressions should not be of different types too. In other words, they should be either supervised, unsupervised, semi-supervised, etc. The second category is known as heterogeneous MTL, in which the model may consider several tasks of different settings. Literally, the learned tasks can be both classifications and regressions, which means that they can be supervised, unsupervised, semi-supervised, etc.

4.3.2 Hard sharing

As aforementioned, the hard sharing of common knowledge among all tasks is the most used to ensure multi-task learning. It uses, a single model to learn all tasks simultaneously, and it is based on several approaches depending on the similarity to be shared. These latter include hard features and weight sharing.

4.3.2.1 Feature sharing

Because learned tasks are related, it makes sense to suppose that each task uses a common feature representation that is based on the original feature representations. Another reason to learn common feature representation instead of the original ones is that these latter might not have sufficient expressive power for multi-task when it's used directly. Thus, a more significant representation can be learned for all tasks using the training data for all of them, and this representation can help in enhancing the model performance. For instance, feature sharing can be done in two main ways:

— Feature Transformation: This approach involves transforming the feature matrix of all tasks into a sparse matrix. This means that the features matrix A formed by the feature vectors a_i of each of the tasks t_i presents a majority of zero values in the same rows, which encourages, on one hand, the tasks to share fewer features and, on the other hand, the model to learn a common features presentation among all tasks. In machine learning, this is ensured through two main techniques: the

regularization [86] [87] that penalizes the matrix norm A across all tasks, and the transformation matrix U [88] that the model will learn to determine a common presentation across all tasks. On the other hand, in deep learning, the transformation of features is ensured through the sharing of hidden layers, as in [89], which can ensure more complex transformations than regularization, also known as nonlinear transformations.

— Feature selection: This approach is somehow similar to feature transformation [90]. Yet, in this case, there is no transformation matrix. Therefore, only the common features in all tasks are selected from the most important original features of all tasks using the regularization of the norm of the feature matrix A. This approach is not used in deep learning due to the automatic feature extraction provided by the hidden layers, which do not only select the most relevant features but also transform them into a more complex common presentation.

4.3.2.2 Weight sharing

Weight sharing is often done through a reduced rank regularization approach [91]. Recall that the rank of a matrix is equivalent to the number of linearly independent vectors formed by its columns. Consequently, the weight sharing consists in approximating a reduced ranked weight matrix from the original model weight matrix, which has as columns the model weight vectors for each task. This approximation implies a strong dependency between tasks and therefore encourages the model to learn common weights for all tasks.

In machine learning, the general form of this approximation is given by Eq. (4.5). On the other hand, in deep learning, this concept is extended to tensor decomposition [92], which presents a matrix of the weight matrices of all the hidden layers shared among all tasks. In this case, the regularization, in Eq. (4.13), is provided through the tensor trace norm instead of the matrix trace norm.

$$\min_{\mathbf{\Theta},\mathbf{b}} l(\mathbf{\Theta},\mathbf{b}) + \lambda \sum_{i=1}^{r} \|\mathbf{\Theta}\|_{s(1)} \tag{4.13}$$

where:

l is the loss function of the model.

$\mathbf{\Theta}$ is the matrix of model weights.

$\mathbf{b}$ is an offset vector for all tasks.

λ is the regularization parameter.
$\|\|\|$ is the trace norm of the model weights matrix.

4.4 Ensemble Learning

Ensemble learning is the sub-field of artificial intelligence, in which several base learners, so-called weak learners are combined in order to create a strong model [93]. The process of ensemble learning consists in generating weak learners, which can be a DT, a neural network, or any other kind of machine learning algorithm. These latter are then combined to form a single yet stronger model than the weak learners. There are several effective ensemble approaches such as bagging [94] and boosting [95] [96]. It's worth mentioning that most of these approaches employ a single weak learning algorithm to generate homogeneous weak learners, but some may employ multiple learning algorithms to generate heterogeneous learners.

4.4.1 Bagging approach

Bagging approach, also known as bootstrap aggregation, was introduced in 1994 by Breiman [94]. This latter is considered as the earliest and simplest approach of ensemble learning [97]. It uses bootstrap sampling [98] to generate random equally-sized sub-sets, also called bootstraps, from the original training data set by replacement. We mean by "*replacement*" that some samples in the generated bootstraps can be replicated. Each of these bootstraps is then used to train one of the weak learners. After training all the weak learners, these latter are combined using either averaging or majority voting. For instance, the determination of the final prediction varies depending on the task to be solved. For regression, the predictions of weak learners are averaged to determine the final prediction. Yet, for classification, a majority vote is used. This means that the most frequently predicted class represents the final prediction. Figure. 4.12 illustrates the process of the bagging approach.
The bagging approach can be used to reduce the risk of over-fitting, which makes it well-suitable for problems with small training data sets [97]. One of the well-known applications of bagging is Random Forest [64], which uses a series of DTs as weak learners.

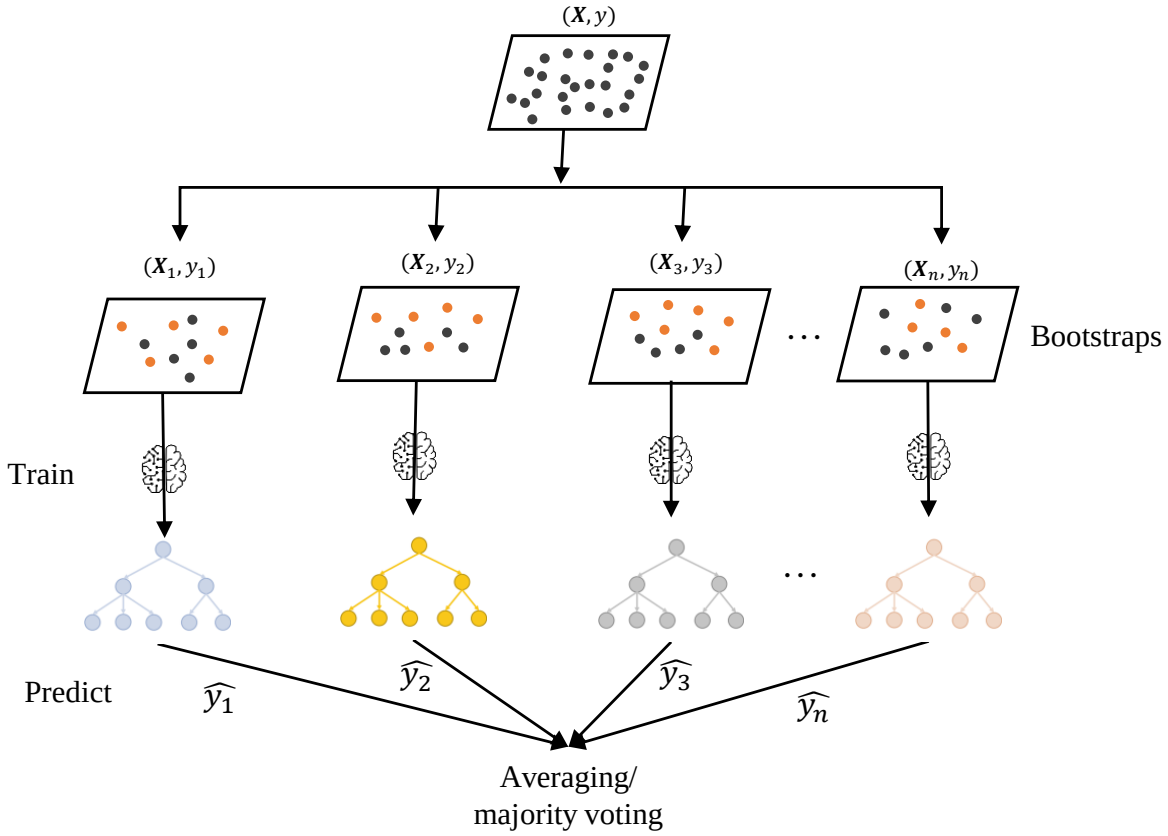


Figure 4.12 Illustration of the bagging approach; Orange data points represent the replicated samples in the bootstrap

4.4.2 Boosting approach

Boosting approach is actually a family of ensemble algorithms [99] such as Light-Gradient Boosting Machine (Light-GBM) [5], which we are going to cover in much more detail in Chapter 6. Similar to the bagging approach, boosting uses bootstrap sampling [98] to create several subsets or bootstraps from the original training data set. However, instead of training independent weak learners as it's the case in bagging, boosting trains each new weak learner by attempting to correct the previous one. For example, in AdaBoosting [95] [96], which represents the first ensemble algorithm based on the boosting approach, a first bootstrap from the original data set is created, and then a weak learner is trained based on this generated subset or bootstrap. Afterward, the original training data set is used to test the trained weak learner and the incorrectly classified samples from the original data set are identified. These latter are assigned with larger weights compared

to the correct ones, and then a new bootstrap is created to train another weak learner, which will attempt to give a better prediction on the incorrectly classified samples. This process is repeated based on the number of weak learners. Finally, the final prediction is determined using a weighted average of all the weak learners' predictions, where more weights are given to stronger learners. Figure. 4.13 shows the process of the boosting approach.

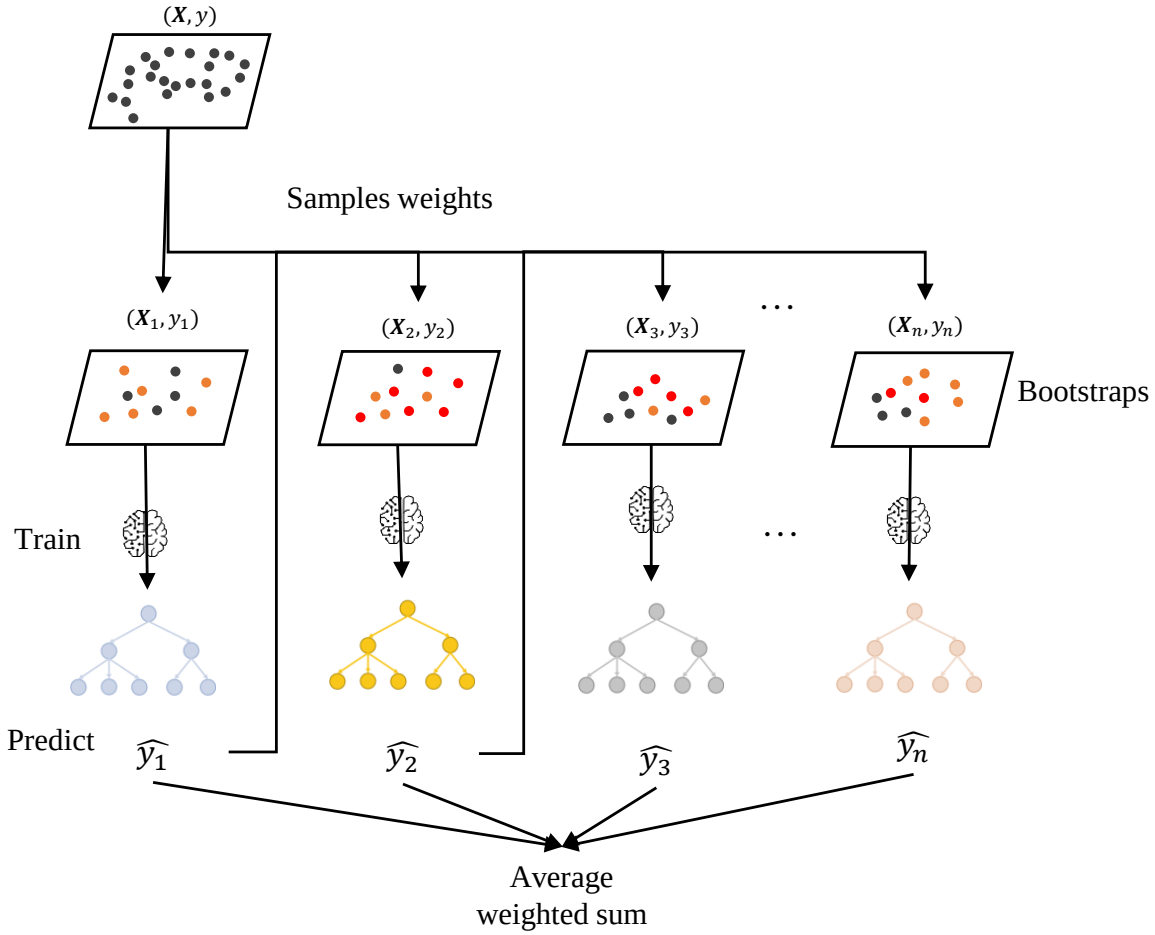


Figure 4.13 Illustration of the boosting approach; Orange data points represent the replicated samples in the bootstrap; Red data points are the incorrectly predicted samples

4.5 Previous proposals on fast decision

Since H.264/Advanced Video Coding (AVC), Rate-Distortion Optimization (RDO) [11] has been a core building block of video encoders. It managed to provide significant coding

capabilities through minimization of the cost j, expressed by Equation. (4.14).

$$j = d + \lambda \times r \tag{4.14}$$

Where:

d refers to the distortion. r is the bit rate (number of required bits).

λ is the Lagrangian multiplier.

Considering that the addition of new coding tools would necessarily incorporate new decisions to be made, the RDO should have become much slower and more complicated. Consequently, several works have been proposed to reduce the RDO overhead. Some were relying on video texture characteristics or Statistical Learning (SL) to decrease the decision set, whereas others have taken advantage of the recent advances on Machine Learning (ML) to terminate the decision process early. This Section presents complexity reduction techniques from the state-of-the-art focusing on the intra mode decision and the Coding Tree Unit (CTU) partitioning. The reviewed works are summarized in Table 4.1.

In [100], Y. Chen et al. investigated several correlations in the intra mode decision of Versatile Video Coding (VVC). Consequently, two strategies were proposed. The first strategy uses the correlation between the 6 Most Probable Modes (MPMs) and the best Intra Prediction Mode (IPM) in Rough Mode Decision (RMD) in order to reduce the decision set. It decides whether to skip IPMs other than the 6 MPMs by assuming that the best IPM is more likely to be MPM when the IPM with the least Sum of Absolute Transform Difference (SATD) cost in RMD is also MPM. Hence, when this condition is met, only MPMs are checked during the Final Intra Prediction Mode Decision (FIPMD). Otherwise, the best IPM is decided using the exhaustive Rate-Distortion (RD) search.

For the second strategy, early termination of the intra mode decision is evoked based on the difference between the best cost in RMD and that of the best IPM. Firstly, the RMD IPMs are ordered in ascending order. Then, the FIPMD is early terminated according to the difference between the cost of the best IPM so far and the IPM with the least SATD cost in RMD. Simulations show that the combination of these two strategies can reduce, on average, 30.59% of computational complexity with a slight increase in bit rate.

J. Park et al. [101] also made full use of some encoding information, such as the block shape, to infer a preprunable range of IPMs, that can be skipped during the test of Intra Sub-Partitions (ISP). Experimental results show that this method can save up to 12% of

encoding time with almost no effect in the coding efficiency.

The prediction distortion also plays a major role in reducing the RDO overhead. It has been exploited in [102] to terminate the Multi-type Tree (MT) partitioning early while providing a tunable decision framework with different trade-offs between the computational complexity and the coding efficiency. Basically, the MT partitioning modes are skipped based on the variance of the prediction distortion of all the possible sub-Coding Units (CUs), in which a CU, can be divided when applying a certain MT partitioning mode. Then, different thresholds are proposed to tune the trade-off between the computational complexity and the coding efficiency. Simulations showed that the speed-up may vary between 22.6% and 67.6% with birate loss ranging from 0.56% to 2.61%.

Texture characteristics were also as useful as the correlations in video content. It has been explored in many ways in order to accelerate the encoding decisions. For example, in [103] the number of candidates for the QuadTree with nested Multi-type Tree (QTMT) structure and the intra mode decision is reduced according to texture characteristics. Firstly, the authors analyze the relationship between the CTU texture and the partitioning

Table 4.1 Previous proposals.

Proposal	SL	ML/DL	Intra	Partitioning	Achieved Results	
					Δt (%)	BD-BR (%)
[100]	×		×		30.59	0.86
[101]	×		×		12.00	0.40
[102]	×			×	22.60-67.60	0.56-2.61
[103]	×		×	×	46.00	0.91
[104]	×		×		7.00	0.09
[105]		×	×		18.00-30.00	0.70
[106]	×	×	×	×	54.91	0.93
[107]	×	×	×	×	70.00	1.93
[3]		×		×	46.60-69.80	0.86-2.57

aspect. Then, based on different observations, which demonstrate the highest correlation between the CU texture and the selection of partition size, the authors proposed a fast CU size decision algorithm, where some MT partitioning modes are ignored. Hence, gradient difference is calculated between two or three sub-partitions, according to the partitioning mode, to decide whether to ignore Quadtree (QT), Binary Tree (BT) or Ternary Tree (TT).

For the intra prediction, a fast intra mode decision was also elaborated. Initially, texture characteristics are calculated using different gradients, including vertical, horizontal,

45° and 135° gradients. Then, the number of IPMs is reduced according to CU Texture Complexity (TC). More preciously, when the CU has a simple texture, the number of candidates for the FIPMD is reduced to 2. Otherwise, some IPMs are ignored based on the texture direction of the CU. This method can offer nearly 49% of speedup with a negligible loss in coding efficiency.

Lui et al. [104] also proposed a fast decision algorithm for the ISP by using the CU TC. They first classify the CUs into homogeneous and textured categories. Then, the authors terminate the ISP tests for homogeneous CUs. This solution can accelerate RDO by 7% with minimal loss on coding efficiency.

As for ML algorithms, they were able to achieve a significant tradeoff between computational complexity and coding efficiency, especially when trained on massive video data. In [105], for example, a ML approach based on Random Forest Classifier (RFC) was used to reduce the computational complexity of the intra mode decision for both the High-Efficiency Video Coding (HEVC) and VVC. Basically, this proposal uses the inferred IPM by the RFC to reduce the number of IPMs before carrying out the FIPMD. To train the RFC, training samples were generated by the encoder and collected from the decoder. Thus, Ryu et al. randomly selected from the Prediction Unit (PU)/CU pixel matrix, a vector of four pixels co-located in each quadrant to derive the PU/CU directional features and estimate the split function of the classifier.

To implement the trained classifier in the HEVC, the RMD is maintained to select a set of n best IPMs. Then, if this set contains only angular IPMs, then the IPM with the highest SATD cost is replaced with the inferred IPM, estimated by the RFC. Otherwise, non-angular IPMs are added with the 6 MPMs to the list of the best IPMs and the decision of the best IPM is made through the FIPMD. for the VVC, the authors use the same strategy, except that the second RMD is maintained since VVC uses two stages of RMD. Experimental results show that this algorithm allows 18% to 30% complexity reduction at the cost of 0.70% of bit rate loss.

Another fast decision algorithm based on RFC was designed in [106]. It includes two algorithms, namely, RFC-based CU size decision and texture-based intra mode decision. The first algorithm relies on the CU TC to decide whether to split the current CU or not. Hence, the authors used several measures such as the standard deviation and the relative pixel complexity in order to characterize the TC in the current CU. Then, a RFC is trained on these features to early terminate the CU size decision. For the second algorithm, Zhang, Q, et al established a texture-based intra mode decision. Consequently,

the original set of IPMs in HEVC is classified into 4 subsets according to the different texture directions. Then, the sub-set of IPMs to be tested during the first stage of RMD is selected according to the texture direction of the current CU. Furthermore, due to the frequent use of non-angular modes, these latter are added to the different subsets in order to improve the intra prediction accuracy. These two algorithms offered nearly 54.91% complexity reduction.

As RFC presents a combination of several dependent Decision Tree (DT), it reveals that DT could also be convenient in speeding up the encoding process. Consequently, H. Yang, et al. proposed in [107] two main strategies to select the CU size and the best IPM. The first strategy consists in ignoring some partitioning modes based on the encoder's decision for the current depth. At this point, if the QT partition is preceded for the current CU, then the MT tests are ignored. Otherwise, the splitting mode is decided using a DT.

The second approach is introduced to search for the best IPM. This latter uses the well-known Gradient Descent Search (GDS). Hence if all neighboring blocks of the current CU exist, the GDS is performed to find the best IPM. Thus, several experiments were conducted to set a good initial search point and check if the gradient descends in the right direction. Moreover, for these two strategies, the correlation between texture and encoder decisions as well as the spatial correlations in the video content were explored thoroughly to train the DT and improve the GDS. Hence, in [107] a combination of both approaches is performed. Experimental results show that these two proposals can accomplish about 70% of time-saving while slightly affecting the coding efficiency.

Previous work [3] introduces a two-stage learning-based QTMT framework. This latter includes a Convolutional Neural Network (CNN) to predict the spatial features of an entire 64×64 luma block and a Light-Gradient Boosting Machine (Light-GBM) to infer the most likely splits at each sub-block. Simulations show that this framework can achieve up to 69.8% of complexity reduction with a small decrease in coding efficiency.

Conclusion

In this chapter, the most important sub-fields of artificial intelligence as well as the state-of-the-art techniques on fast encoding decision were reviewed. Aimed towards the optimization of the VVC encoder, machine learning algorithms seem to be well-suited to introduce robust decision algorithms. Consequently, the next two chapters propose two intra mode decision frameworks. The First is based on a single Multi-Task Learning

(MTL) CNN, which limits the parameter search space of the FIPMD to the top-2 inferred intra coding tools. However, the second takes advantage of an ensemble model also known as Light-GBM to shrink the decision set as well while trying to improve the prediction accuracy by using a CU size adaptive training technique.

Chapter 5

Multi-task learning based intra mode decision framework

Introduction

Since AlexNet has shown superior performance over all the traditional methods of image classification, Convolutional Neural Network (CNN) has become one of the most impressive techniques for computer vision. The potential highlighted by CNNs in image classification was motivated by their deep structures and the convolution operation adapted to the image representation. As explained in Chapter 3, in Versatile Video Coding (VVC), the intra coding tools undergo Rate-Distortion Optimization (RDO) in order to decide the best intra coding tool and the Intra Prediction Mode (IPM) for each Coding Unit (CU). Taking into account that this decision can be seen as several binary classification tasks, we aim in this Chapter to replace it with a Multi-Task Learning (MTL) based intra mode decision framework, which could be used alongside a previous work [3]. This Chapter is then devoted to a detailed description of the proposed framework and an evaluation of its performance under the VVC Test Model (VTM).

5.1 Configurable fast intra coding for VVC

Fundamentally, our proposal deals with deciding the best IPM during the Final Intra Prediction Mode Decision (FIPMD), which represents the most time-consuming part of the intra mode decision [100]. However, given that the CU size decision could enable up to 91% of complexity reduction [55], we combined our proposal with a previous work [3] based on two configurations namely C1 and C2. As illustrated in Figure. 5.1, the configuration C1 uses the two proposals to form a single shared MTL CNN, where common knowledge among all tasks is leveraged to skip unlikely IPMs as well as some partitioning modes. Yet, in configuration C2, a MTL CNN is devoted to the intra mode

decision and the CU feature extraction for the CU size decision is processed separately using the adopted CNN in [3].

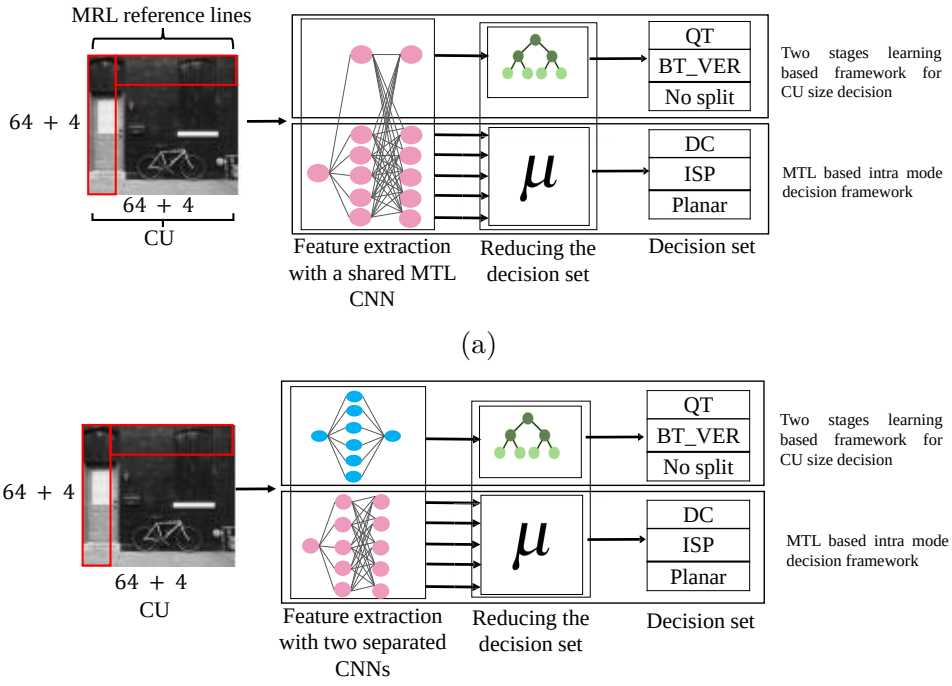


Figure 5.1 Used configurations to combine the MTL based intra mode decision framework with previous work in [3]; (a) configuration C1: a single shared MTL CNN is used to leverage common knowledge among the two proposals; (b) configuration C2: a MTL CNN is devoted for the intra mode decision and the CNN adopted in [3] is used to perform CU feature extraction for the CU size decision

5.1.1 Multi-Task Learning based intra mode decision framework

Multi-Task Learning (MTL) has recently made it possible to predict several related tasks simultaneously using a shared model, that leverages common knowledge among all tasks [84]. As we can see in Figure. 5.2, the proposed framework takes advantage of this learning paradigm regardless of the used configuration. It feeds the original luma block, denoted as $\mathbf{Y}$, to a MTL CNN in order to predict simultaneously, whether to skip Matrix weighted Intra Prediction (MIP), regular, Multi-Reference Lines (MRL) and/or planar, DC, by giving as an output a number of probability vectors $\hat{\mathbf{p}}_\mathbf{t}$. Using the predicted probability vectors the number of candidates for the FIPMD is reduced according to the inferred top-2 intra coding tools.

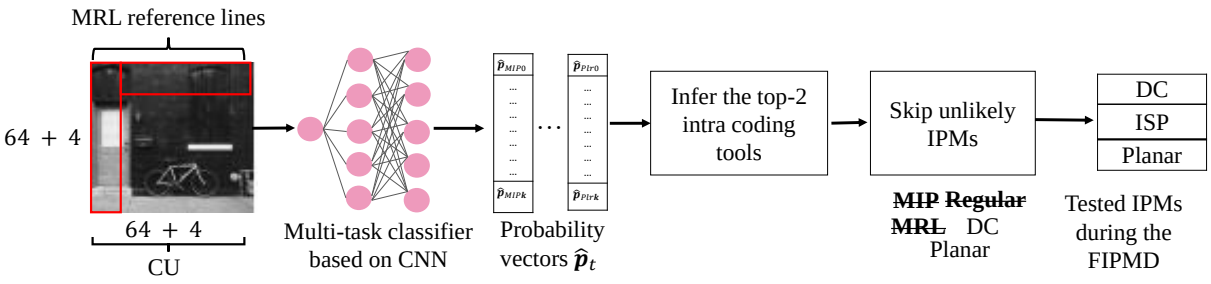


Figure 5.2 Workflow of MTL-based intra-mode decision framework. First, the luma block is fed to the MTL CNN to predict the probability vectors $\hat{\mathbf{p}}_{MIP}$, $\hat{\mathbf{p}}_{Reg}$, $\hat{\mathbf{p}}_{MRL}$, $\hat{\mathbf{p}}_{DC}$ and $\hat{\mathbf{p}}_{Plr}$. Then, the top-2 most likely intra-coding tools are inferred with the maximums of mean probabilities at each block. Finally, the FIPMD is performed only on IPMs of the top-2 intra-coding tools in order to decide the best IPM

5.1.1.1 Probability vectors representation

To accommodate the high diversity of block shapes in the QuadTree with nested Multi-type Tree (QTMT), the MTL CNN processes an entire 64×64 CU, padded with the four MRL references lines, to output a probability vector $\hat{\mathbf{p}}_t$ of usage at the 4×4 sub-blocks, for each intra coding tool $t \in \{\text{MIP, regular, MRL, DC, planar}\}$.

$$\hat{\mathbf{p}}_t = f_{\boldsymbol{\Theta}}(\mathbf{Y}) \tag{5.1}$$

where $f_{\boldsymbol{\Theta}}$ is a parametric function, with learnable parameters $\boldsymbol{\Theta}$, used to approximate the output vector $\hat{\mathbf{p}}_t$ for each coding tool t. The length of the output vector $\hat{\mathbf{p}}_t$ is defined as indicated in Equation. 5.2

$$k = \left(\frac{s_{\mathbf{Y}}}{4}\right)^2 \tag{5.2}$$

where, $s_{\mathbf{Y}}$ refers to the width of the input luma block, which has a square shape. Hence, for 64×64 block, k would be equal to 256. Figure. 5.3 explains the representation of output vectors $\hat{\mathbf{p}}_t$. As shown in this Figure., $\hat{\mathbf{p}}_{t0}$ is the probability of using an intra coding tool t at the first 4×4 sub-block of the 64×64 luma block and $\hat{p}_{tk}$ is the probability for the last 4×4 sub-block.

5.1.1.2 Integration under VTM based on the top-2 inferred intra coding tools

In order to skip unlikely IPMs and reduce the number of candidates for the FIPMD, the predicted probability vectors are used as follows: First, the probability of using MIP,

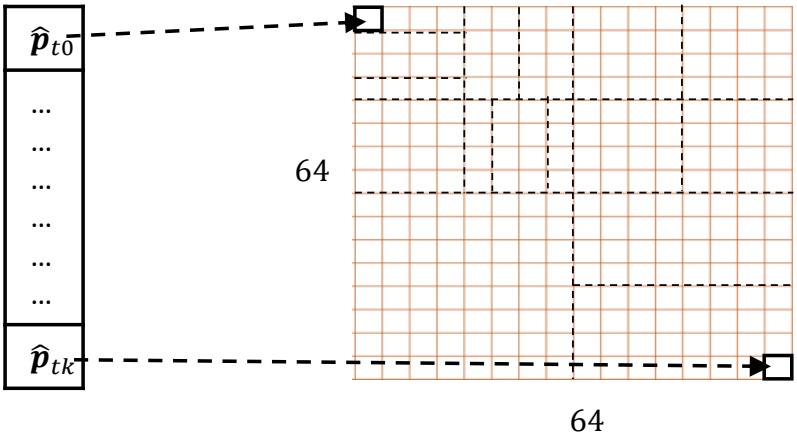


Figure 5.3 Representation of the output vector of the MTL CNN; k is the length of the probability vector $\hat{\mathbf{p}}_t$; t is the intra coding tool.

regular IPMs, MRL, DC, and planar are computed as the mean probabilities at each CU, denoted as μ_{MIP}, μ_{Reg}, μ_{MRL}, μ_{DC} and μ_{Plr}, respectively. Then, these latter are used to infer the top-2 intra coding tools in each CU. As illustrated in Equation. (5.3), the top-1 intra coding tool is inferred as the maximum of mean probabilities in each CU. Then, a second best intra coding tool is deduced also as the maximum of mean probabilities when excluding the top-1 intra coding tool.

$$p_{top1} = max(\mu_{MIP}, \mu_{Reg}, \mu_{MRL}, \mu_{DC}, \mu_{Plr}) \tag{5.3}$$

Finally, the FIPMD is performed only on IPMs of the inferred top-2 intra coding tools in order to decide the best IPM. For instance, the model was integrated under the VTM10.2 throughout the process illustrated in Figure. 5.4.
First, the model prediction is considered only when 2 top intra coding tools are inferred at the current CU. Therefore, if these tools are regular IPMs and DC or planar, the number of tested IPMs for the FIPMD is reduced to 2. Therefore, only DC and planar with the Intra Sub-Partitions (ISP) candidates are tested. Second, if DC and planar are not among the inferred top-2 intra coding tools, then ISP tests can be skipped. In fact, the ISP usually reproduce 2 or 4 homogeneous regions, which will probably be coded with non-angular modes as demonstrated in [56]. Finally, the IPMs other than the top-2

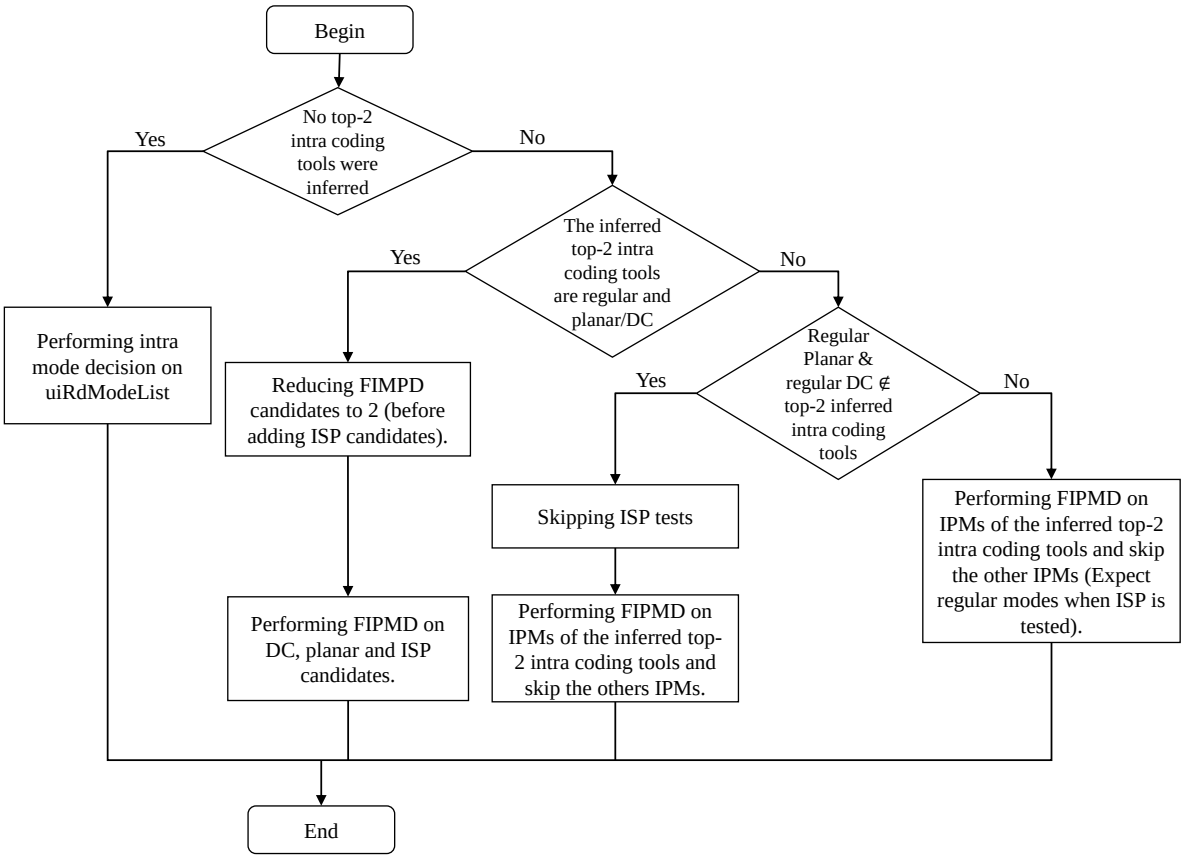


Figure 5.4 Process of integrating the MTL CNN under the VTM10.2 to skip unlikely IPMs

inferred intra tools are skipped. It is worth mentioning that the MRL was excluded from the model prediction since it does not allow a significant complexity reduction [54].

5.1.2 Two stages-learning-based framework for the CU size decision

As explained in [3], this proposal works quite similarly to the MTL based intra mode decision framework. It introduces a two stages-learning-based framework for the CU size decision based on two components, namely a CNN and a number of Light-Gradient Boosting Machine (Light-GBM) classifiers. The used CNN has also the advantage of predicting all possible partitioning of an entire 64×64 luma block by providing as an output a probability vector, denoted as $\hat{\mathbf{p}}_{part}$. This latter gives the probability of a split at the edges of each 4×4 sub-block of the entire 64×64 luma block. However, instead of skipping

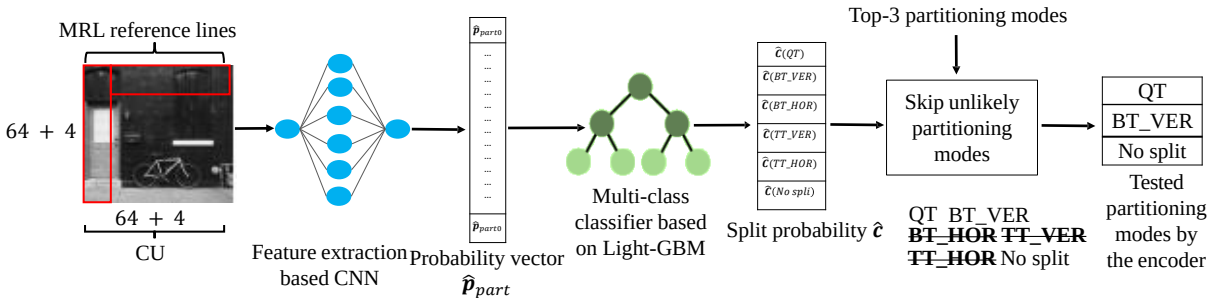


Figure 5.5 Workflow of the two-stages-learning-based framework for the CU size decision. First, the luma block is fed to the CNN to predict the probability vector $\hat{\mathbf{p}}_{part}$. Then, this latter is processed using several Light-GBM classifiers to get the probability vector $\hat{\mathbf{c}}_{part}$ of all the possible partitioning modes in the current CU. Finally, the RDO is performed only on the top-3 most likely partitioning modes in order to decide the best CU size

unlikely partitioning modes based on the mean of the probabilities at their edges, the Light-GBM classifiers are used to process the probability vector in order to select the top-3 partitioning modes, that have the highest probabilities. These latter are then used, as illustrated in Figure. 5.5, to shrink the decision set, thus reducing the computational complexity of the CU size decision.

5.1.2.1 Feature extraction based on CNN

The two-stages-learning-based framework for the CU size decision [3] relies on a CNN in order to extract the probability vector $\hat{\mathbf{p}}_{part}$ for an entire 64×64 luma block. This latter is optimized differently in the two configurations. For instance, in configuration C1, the used CNN is jointly optimized with the intra coding tools, forming together a single shared MTL CNN. On the other hand, the CU size feature extraction is solved independently from the intra coding tools in configuration C2. Taking into account that each 4×4 sub-block has two edges (i.e. bottom and left), as shown in Figure. 5.6, the length of the probability vector $\hat{\mathbf{p}}_{part}$ would be twice the length of the probability vectors in our proposal $\hat{\mathbf{p}}_t$. However, since the bottom-most and the left-most edges of the entire 64×64 luma block are already presenting a split, the Equation. 5.4 can be used to compute the length of the probability vector.

$$m = \frac{s_{\mathbf{Y}}}{2}\left(\frac{s_{\mathbf{Y}}}{4} - 1\right) \tag{5.4}$$

So, for the case of a 64×64 block, the length m of the probability vector should be equal to 480. Furthermore, the probabilities are distributed in the probability vector so that the

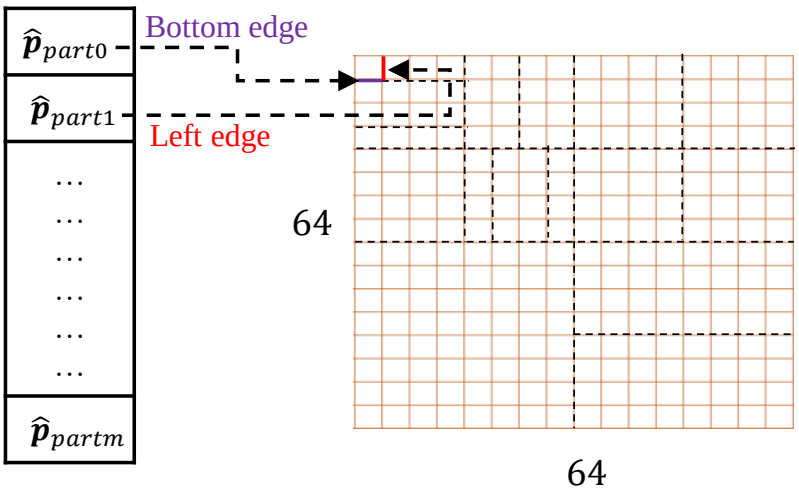


Figure 5.6 Representation of the output vector of the CNN; m is the length of probability vector $\hat{\mathbf{p}}_{part}$; *part* stands for the partitioning task.

probability of the bottom edge of each 4×4 sub-block is followed by the probability of the left edge of that same sub-block. For example, the first $\hat{\mathbf{p}}_{part0}$ and the second $\hat{\mathbf{p}}_{part1}$ corresponds to the probability of the bottom and the left edges of the first 4×4 sub-block, respectively.

5.1.2.2 Integration under VTM with multi-class classifiers based on Light-GBM

After extracting the probability vector $\hat{\mathbf{p}}_{part}$, a Light-GBM classifier, which we are going to cover in much more detail in Chapter 6, is used to process it and skip unlikely partitioning modes. More preciously, the extracted probability vector is cropped into several probability vectors, that include only the probabilities of the sub-block edges of a given CU size. Consequently, several Light-GBM classifiers were trained based on this cropped version of the probability vector $\hat{\mathbf{p}}_{part}$ as spatial features. Indeed, given the diversity of the input size, it is necessary to have a Light-GBM per CU size.

The output probability vector $\hat{\mathbf{c}}_{part}$, predicted at the stage of the Light-GBM classifier has six entries, representing the probabilities of all the possible partitioning modes in the

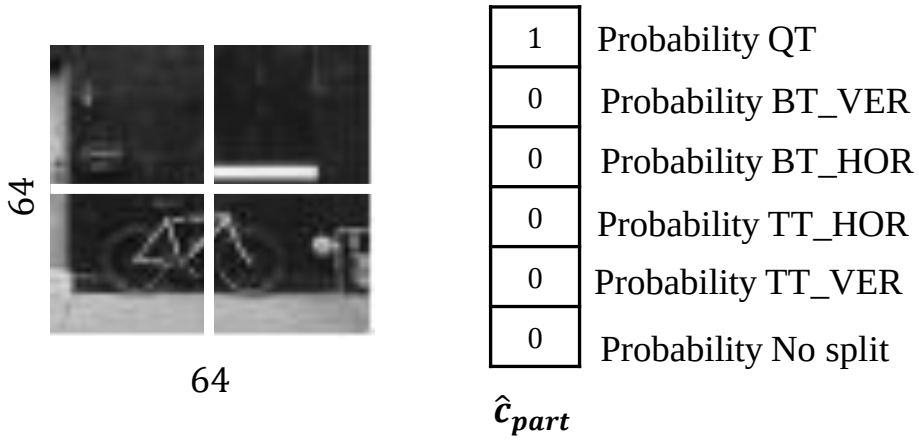


Figure 5.7 Representation of the output vectors $\hat{\mathbf{c}}_{part}$ of the six probabilities, that define all the possible partitioning modes for each of all the CUs in the tree

QTMT. For instance, based on the CU size, it's possible to further partition a CU into two to six different partitions, including the no split mode [3]. Figure. 5.7 illustrates an example of the probability vector $\hat{\mathbf{c}}_{part}$, that specifies the best partitioning mode for a given CU size. For example, a 64×64 CU can have only two possible splits: Quadtree (QT) and no split, therefore the remaining four entries in the vector are always set to zero. Instead, all partitioning modes in the QTMT can be available for a CU size of 32×32, therefore the optimum length of the output probability vector would be 6.

After finding the probability of all the possible partitioning modes at a CU, the top-3 most likely partitioning modes, that have the highest probabilities, are used to shrink the decision set of the RDO in order to decide the best CU size.

5.2 Data set and training process

In this Section, our data set representation is given. Then, the proposed MTL CNN with its training process are detailed.

5.2.1 Training data set

Due to the lack of public data sets for our classification tasks, a data set, that yields encoded CUs with their best intra mode is established. Since the proposed method focuses

on All Intra (AI) configuration, the image public data set Div2k [4] was selected to derive training samples. As illustrated in Figure. 5.8, the Div2k data set is a large collection of Ultra-High Definition (UHD) images with different scene contents, including people, homemade items, urban and rural cities, as well as underwater and low-light nature scenes. The aforementioned data set contains about 1000 images, that have 2k pixels on at least one of its axes (either height or width).

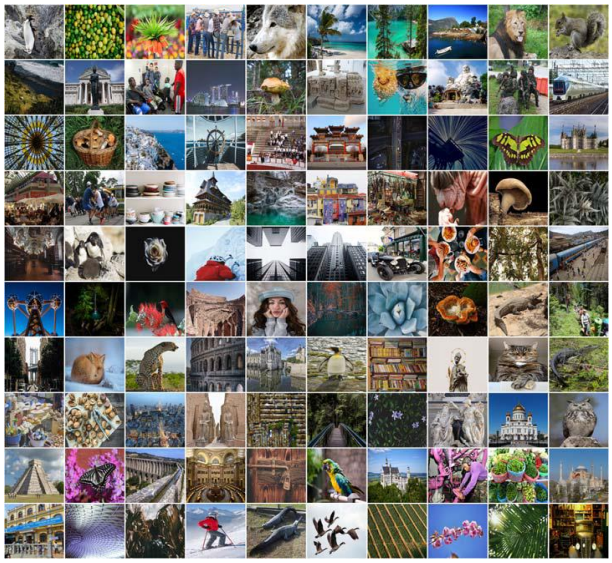


Figure 5.8 Div2K data set [4]

In order to be encoded using VTM [108], the images of this latter were concatenated as a pseudo video. Hence, they have been encoded under the AI configuration for 4 Quantization Parameter (QP) values (eg. 22, 27, 32, and 37). Considering that the VTM includes several fast decision techniques for the partitioning [109] and the intra coding tools as well [110] [111], these latter were disabled in order to build our training data set. Disabling these techniques would ensure sufficiently diverse data for more accurate predictions.

The best intra modes were first extracted from VTM encoder as a tree, where each node has its best intra mode information, including the used intra coding tool and IPM index. Then, these latter were converted, as illustrated in Figure. 5.3, to several 256 probabilities vectors $\hat{\mathbf{p}}_t$, that depicts whether MIP, MRL, regular and/or DC, planar are used for each 4×4 sub-block in the 64×64 luma block. Our data set has 1.2 million samples distributed as given in Figure. 5.9. As shown in this Figure, the regular IPMs are representing the majority of training samples, with at least 77%. Whereas MRL samples

are so under-represented, which is due to the fact that the VTM encoder tends usually to select regular IPMs [56].

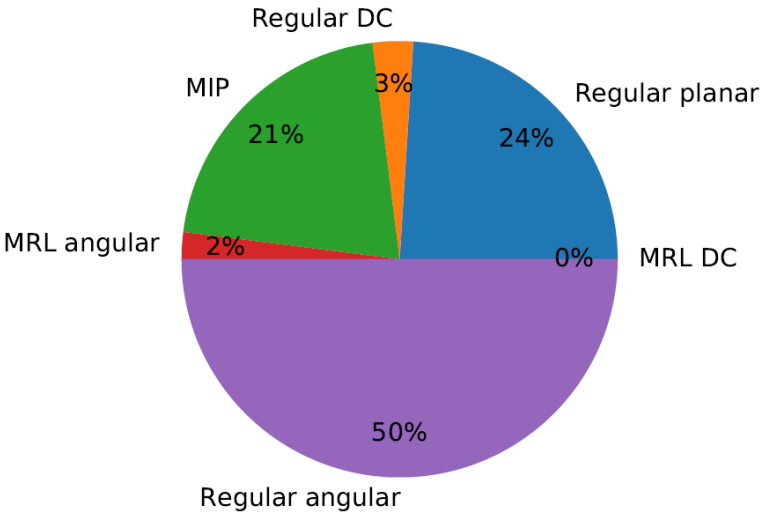
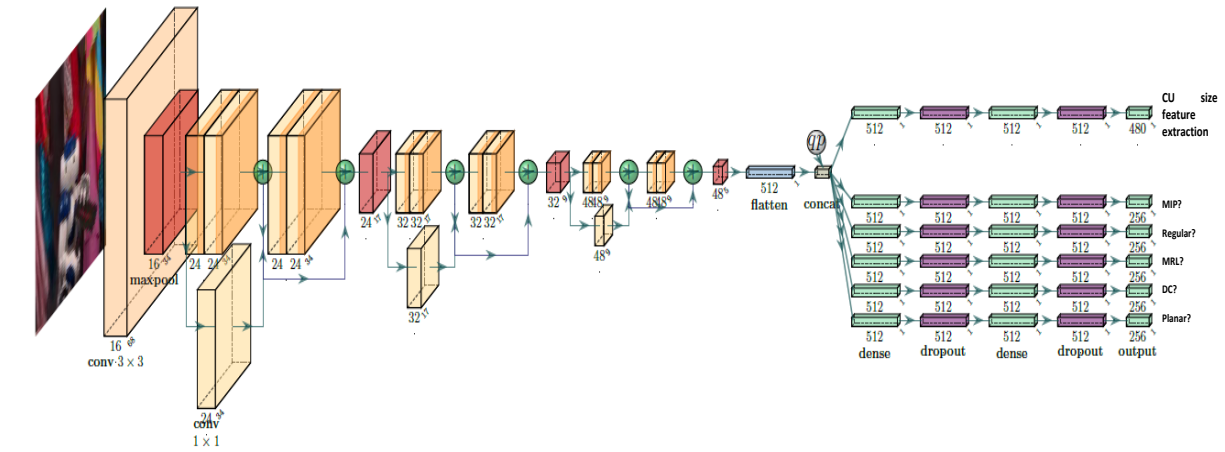


Figure 5.9 Distribution of training data set

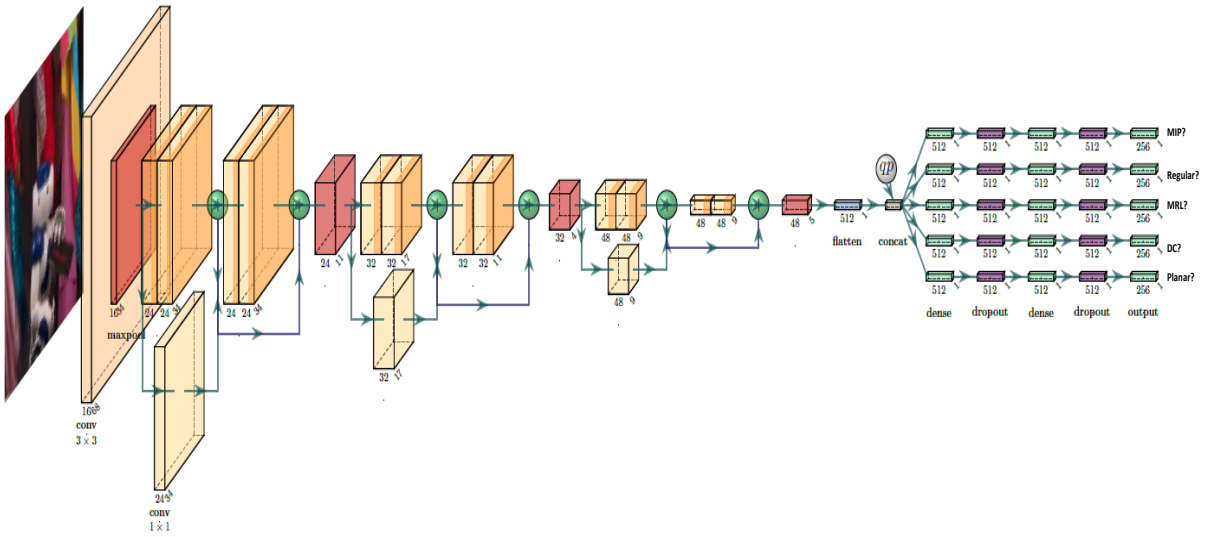
For the CU feature extraction in configuration C1, we used the soft representation of the data set created in [3] in order to jointly optimize the model for all the target tasks. Consequently, only the samples from the div2K data set were taken into consideration.

5.2.2 Training process

As indicated in Figure. 5.10, we trained our model differently based on the two used



(a)



(b)

Figure 5.10 The architecture of the MTL CNN; convolution layers are in orange and yellow, max-pooling layers in red; dropout in purple, and fully connected layers in cyan; (a): MTL CNN with CU size feature extraction as related task; (b) MTL CNN for intra coding tools only

configurations C1 and C2. In configuration C1, a single shared MTL CNN is trained for all tasks. Whereas, in the second configuration, also called C2, we had to separate the

training by using the proposed CNN in [3] for the CU size feature extraction and devoting a MTL CNN for the intra coding tools.

The proposed MTL CNN, as illustrated in the aforementioned Figure is inspired by the well-known residual neural network or ResNet for short [112]. Residual models first appeared in 2015 to ease the training of very deep CNNs. Unlike a plain model, a residual model includes shortcut connections or residual blocks to reduce the risk of overfitting or vanishing gradients. As shown in Figure. 5.11, a residual block is comprised of a few stacked convolutional layers where an identity mapping, denoted as $\mathbf{X}$, is added to the output of the few ahead convolutional layers.

Since convolutional layers tend usually to shrink the size of the input image, a linear projection, as defined in Equation. (5.5), can be used to ensure matching the two input dimensions. This latter is usually done using a 1×1 convolution layer.

$$y = f(\mathbf{X}) + \mathbf{W}_s \times \mathbf{X} \tag{5.5}$$

Where:

$\mathbf{W}_s$ is a square matrix used to shrink the identity mapping to the desired dimension.

In addition, our model adopts hard sharing of feature extraction layers [113] to ensure feature space sharing and performance across all tasks. Thus, instead of having their own feature extraction layers, all the tasks are jointly optimized through sharing the same feature extraction layers. Then, task-specific layers are used separately to predict the probability vector for each task t.

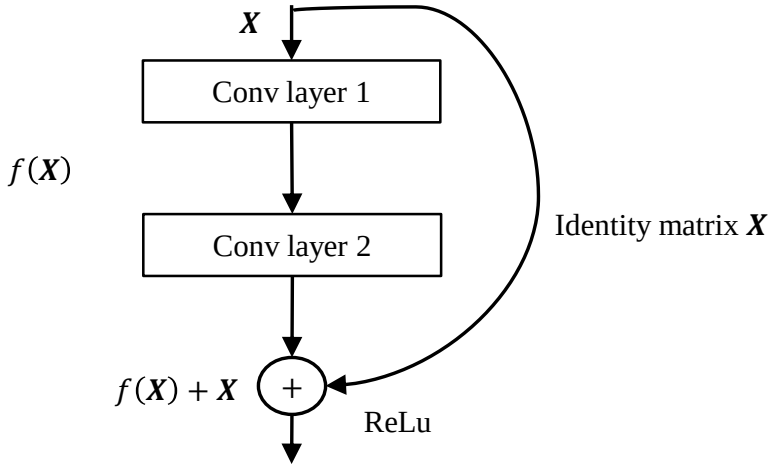


Figure 5.11 Residual block in ResNet

The proposed model was trained from scratch on the training data set using Adam optimizer [81] to learn the optimum weights $\mathbf{\Theta}$ and minimize the sum of the loss functions l_t defined in Equation. (5.6).

$$l = \sum_{t} l_t \tag{5.6}$$

Where:

l_t is the task-specific loss function. For each intra coding tool t, the loss function l_t is defined by Equation. (5.7),

$$l_t = \frac{1}{n} \sum_{i=1}^{i=n} \mathbf{p}_t(\log(\hat{\mathbf{p}}_t) + \alpha(1 - \mathbf{p}_t)(-\log(1 - \hat{\mathbf{p}}_t)) \tag{5.7}$$

with n gives the number of training samples in the current batch and α is the weight assigned to the positive class. This latter varies slightly between the two configurations. For instance, in configuration C1, we selected α as 5 for MIP and regular, 1 for MRL and DC, and 4 for planar. However, in configuration C2, the following values are used: 3.5 for MIP, 1 for MRL, and regular, 11.5 for DC, and 4.5 for planar. On the other hand, the Mean Squared Error (MSE), computed by Equation. (5.8) is used to train the CNN for the CU size feature extraction in both configurations.

$$MSE = \frac{1}{n} \sum_{i=1}^{i=n} (\mathbf{p}_{part} - \hat{\mathbf{p}}_{part})^2 \tag{5.8}$$

As explained in Section 5.2.1, the task-specific loss function used to train the MTL CNN on the intra coding tools was chosen due to the highly skewed data set we are dealing with in this task. Early stopping regularization was also used to ensure the model convergence in configuration C2. Thus, our MTL CNN was only trained for 14 epochs with a batch size of 128 and a learning rate of $10^{-}3$. Nevertheless, we let the model train for 100 epochs in configuration C1 as recommended in [3].

5.3 Experimental setup

To asset the performance of our method, simulations are conducted on the reference software VTM10.2 [108] with the Common Test Conditions (CTC) [114] video sequences. Indeed, a set of 26 videos from classes A1, A2, B, C, D, E, and F are encoded under the AI configuration for 4 QP values, which are 22,27,32, and 37. For the number of encoded

frames in each video, it is set to one Group of Pictures (GOP), which is equivalent to one second.

The effectiveness of our method is assessed using both complexity reduction and coding efficiency. Hence, the Bjontegaard Delta Bitrate (BD-BR) [22] is used to measure the coding efficiency, then the average of the run-time difference Δt, defined by Equation. (5.9) is used to give the achieved complexity reduction.

$$\Delta t = \frac{1}{4} \sum_{QP_i=22,27,32,37} \frac{t_o(QP_i) - t_r(QP_i)}{t_o} \tag{5.9}$$

Where:

t_o and t_r represent the total encoder runtimes of the VTM anchor and the VTM when using the proposed framework, respectively.

All the CNNs were built and trained under the Keras framework, then converted with the frugally deep library [115] to C++, in order to be used in the VTM encoder. For the different Light-GBM classifiers, the Light-GBM framework, developed by Microsoft Corporation [5], was used to build, train, and convert them to C++ code.

5.4 Performance of the MTL CNN

One of the core steps of building an efficient machine learning model is to test how robust it is in predicting the correct output. Claiming the effectiveness of a model is usually introduced as a constructive feedback process, where engineers train a model, get feedback from metrics, and improve the model until they achieve the desired performance for their target task. There are plenty of ways to evaluate the performance of machine learning models. Consequently, we considered in our work the three widely used evaluation metrics, which are cross-validation, confusion matrix, and recall/precision analysis.

5.4.1 Cross-validation

Cross-validation is the most intuitive evaluation method in machine learning. It considers leaving a small population of the training data to test the model performance before testing it on the new population, also known as the test set [116]. So, considering a data set of 1000 samples, one can split it into 800 samples for training and 200 samples for validation purposes. The validation score is usually measured using the accuracy metric,

computed in Equation. (5.10).

$$accuracy = \frac{n_c \times 100}{n} \tag{5.10}$$

Where:

n_c gives the total number of correct predictions.

n is the total number of samples in the training/test data set. The validation process

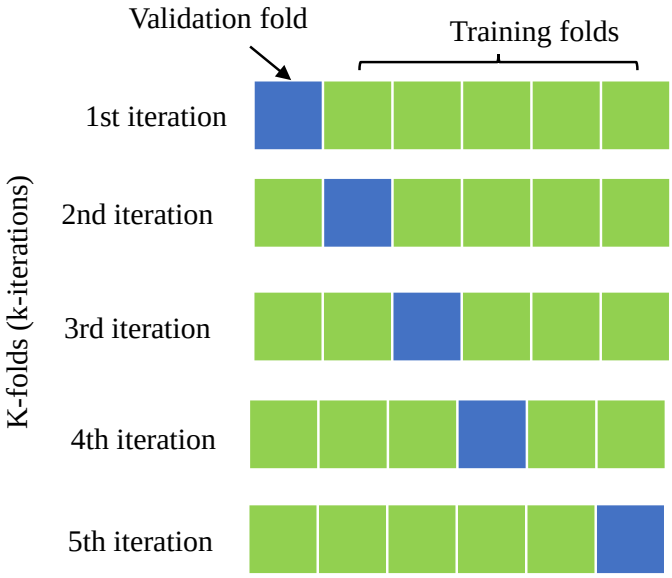


Figure 5.12 K-folds cross-validation

would yield a reliable estimate of the model performance and encourage generalization over new samples. However, in some applications, only a limited amount of data could be available. Thus, splitting the data once may implies risks for under/overfitting the data, also known as bias/variance respectively, due to the limited number of training and validation samples. For this reason, a k-fold validation strategy [117] can be applied. It consists in splitting the available data k-times or folds, as shown in Figure. 5.12. Each time, a different set of samples is assigned to the training set and to the validation in order to enhance the model performance and achieve a good bias/variance trade-off.

5.4.2 Confusion matrix

The confusion matrix [118] is one of the essential evaluation metrics for understanding the performance of a classification model. It reports the number of correct and incorrect predictions, in comparison to the ground truth. In a binary classification task, a confusion

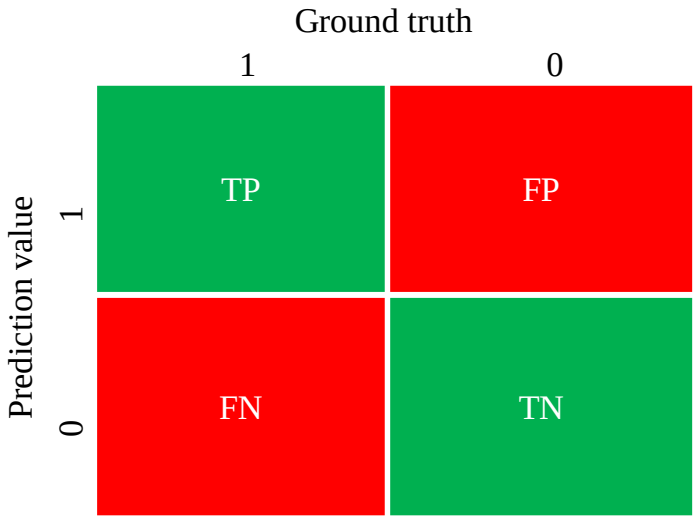


Figure 5.13 Confusion matrix

matrix is usually represented by a 2×2 matrix, where each column compares the ground truth with the prediction values. To interpret the confusion matrix, we simply examine the intersection of the rows and columns. As shown in Figure. 5.13, the top left corner shows the number of True Positive (TP), while the top right corner shows the False Positives (FPs). On the other hand, the lower left corner shows the False Negatives (FNs) while the lower right corner shows the True Negatives (TNs).

— TP vs FP : the TP gives the number of correct positive predictions. While FP represents the number of incorrect positive predictions.

— TN vs FN: similar to TP and FP, the TN and FN yield the number of correct and incorrect negative predictions, respectively.

The knowledge of these four values would help in identifying the type of error and computing several other evaluation metrics such as precision/recall and F1-score.

5.4.3 Recall/precision analysis

Finding a good bias/variance trade-off is crucial. Meanwhile, when it comes to real data sets, imbalanced classes may bring difficulties in spotting the true performance of a model. In binary classification, an overabundance of negative samples is very common. In such a situation, the model may overlook the negative samples in favor of the positive ones, which makes its accuracy very high. A good solution here would be to ignore the model accuracy altogether and use a more convenient trade-off, so called precision/recall trade-off or F1-score [119].

Precision is defined, in Equation. (5.11), as the ratio of TP among all the positive samples in the data set.

$$Precision = \frac{TP}{TP + FP} \tag{5.11}$$

As for the recall, it measures, as illustrated in Equation. (5.12), the proportion of TP out of all the positive predictions.

$$Recall = \frac{TP}{TP + FN} \tag{5.12}$$

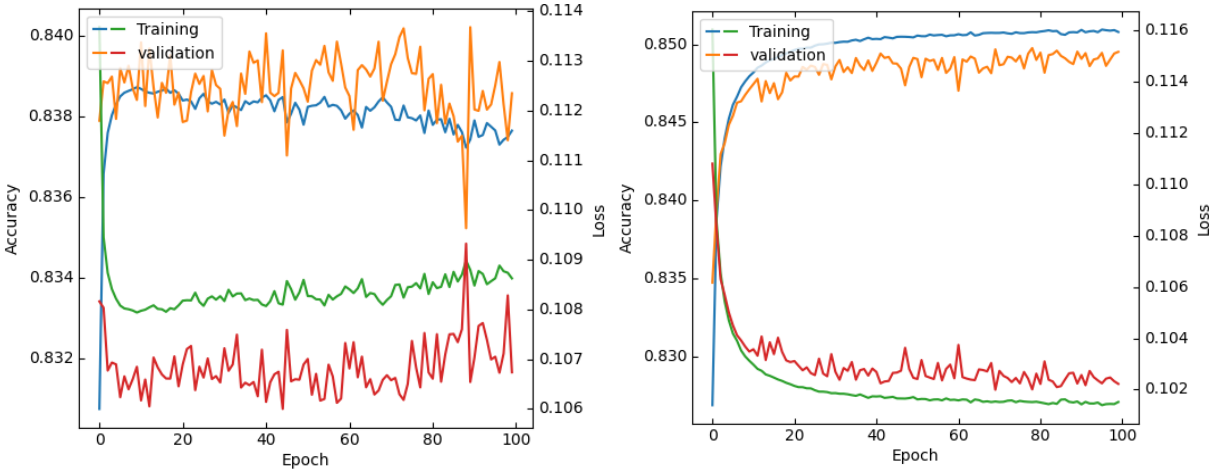
More preciously, precision gives the portion of relevant predictions. Whereas, recall reports how robust is the model in identifying positive samples. Regardless of the classification task, some situations may suggest that precision and recall are equally important. Unfortunately, high precision may come at the cost of a low recall and vice versa. For this reason, researchers have developed a trade-off metric, called F1-score. As defined in Equation. (5.13), the F1-score is usually a harmonic mean of precision and recall and it's more convenient for problems where both precision and recall are important. Its value ranges between 0 to 1. So, a very poor prediction where the number of incorrect predictions is very high would get an F1-score of zero. A prediction with some correspondence between the ground truth and prediction would be between 0 and 1. A very good prediction where the number of incorrect predictions is nearly zero would be very close to one.

$$F1 - score = \frac{2 \times (precision \times recall)}{precision + recall} = \frac{TP}{TP + (FP + FN)/2} \tag{5.13}$$

5.4.4 Model evaluation

To evaluate the MTL CNN for the CU size feature extraction we used the accuracy and loss curves, illustrated in Figure. 5.14. Indeed, the different partitioning classes are

quite equally distributed in the data set adopted in [3]. However, given that our data set for the intra coding tools is highly skewed, our model performance is evaluated with recall, precision, and F1-score, instead of using accuracy and loss metrics. These latter won't help in spotting the true model performance since the negative and the positive classes are not equally distributed.



(a) (b)

Figure 5.14 Decreasing loss (green and red curves) and increasing accuracy (blue and orange curves) as a function of the learning epoch for the training and validation data sets, respectively

Looking at Figure. 5.14, we can notice that the model performance for CU size feature extraction has shifted down a bit in configuration C1 when compared to configuration C2. Indeed, intra coding tools seem to introduce a little noise to the CU size feature extraction, which may cause some loss in the overall performance. In fact, the partitioning of CU is integrally linked to the texture complexity, where CUs with complex textures are more likely to be partitioned, whereas those with very simple textures are less likely to undergo further splitting. As indicated in Chapter 4, MTL tries to learn a common feature representation among all tasks in order to maximize its performance. However, given that the intra coding tools may strongly depend on the texture direction as well as several other texture-based features, serving to reduce the impact on coding efficiency by measuring the maximum variation between the pixels within a CU, this may introduce a certain amount of noise into the task of CU size feature extraction, leading to a decrease in the overall model performance.

For the intra coding tools, we first evaluate our model under the default classification

threshold, which is equal to 0.5. Then recall-precision curves, as shown in Figure. 5.15, are used to infer the best classification thresholds, that maximize the F1-score.

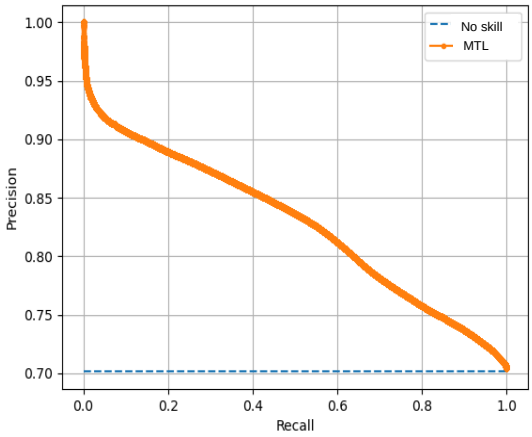
Table 5.1 Performance of the MTL CNN when the CU size selection is introduced as a related task

Coding tool	Default threshold			Best threshold		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Regular	0.39	0.71	0.50	0.40	0.68	0.50
DC	1.00	0.13	0.23	0.35	0.61	0.44
planar	0.43	0.53	0.48	0.39	0.58	0.47
MIP	0.37	0.55	0.44	0.35	0.65	0.46
MRL	0.57	0.24	0.33	0.36	0.71	0.48

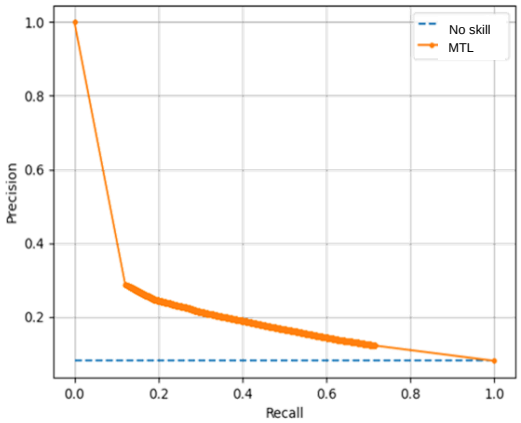
Since the task of our model, is to guess the class membership of each sample by predicting the probability of being positive or not, one can convert these probabilities into positive and negative labels by taking into account a classification threshold at which we may consider that the predicted probability is relevant. A default classifier may use the rule $\hat{p}_t > 0.5$ to decide whether to classify the current samples as positive or not. However, there is no apparent reason that this threshold would yield the optimum classification performance [120]. As shown in Figure. 5.15, one approach may be to adjust the classification threshold and plot all the possible pairs of recall and precision. For instance, the precision vs. recall curves of our model are illustrated as orange curves. Whereas,

Table 5.2 Performance of the MTL CNN for intra coding tools only

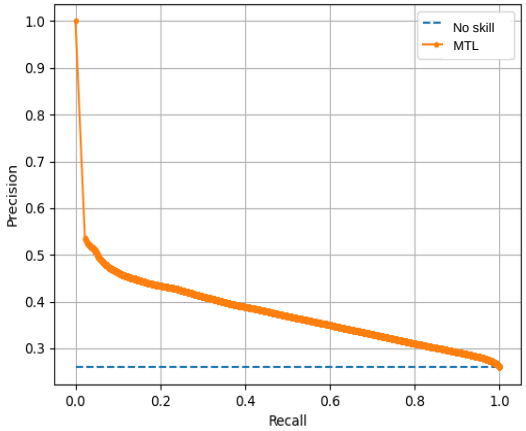
Coding tool	Default threshold			Best threshold		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Regular	0.72	0.97	0.82	0.71	0.99	0.83
DC	0.19	0.39	0.26	0.19	0.40	0.26
planar	0.35	0.61	0.45	0.33	0.73	0.45
MIP	0.41	0.56	0.48	0.38	0.67	0.49
MRL	0.64	0.00	0.01	0.22	0.39	0.28



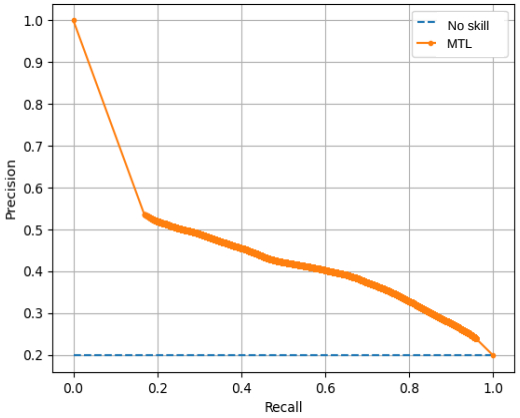
(a)



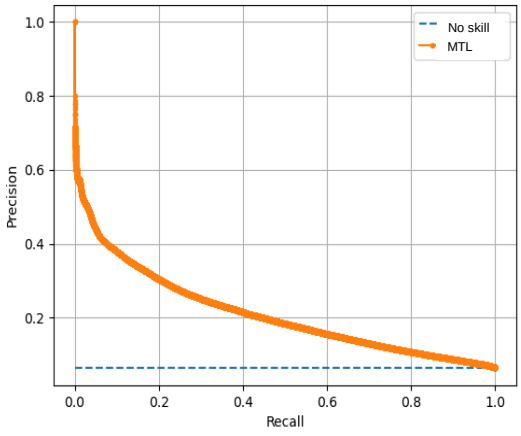
(b)



(c)



(d)



(e)

Figure 5.15 Precision vs recall curves: (a): for Regular; (b): for DC; (c): for Planar; (d): for MIP; (e): for MRL

the dashed blue curves give the precision vs. recall of a no-skill model, that outputs random guesses or predictions. Hence, the higher the area under the curve of precision vs. recall, the higher the trade-off between precision and recall (i.e. F1-score).

Table 5.1 and Table 5.2 report the aforementioned metrics for the default and the best thresholds. These latter are inferred from the precision vs. recall curves for both configurations. Thus, configuration C1 uses the classification threshold equal to 0.25 for MIP, 0,67 for regular, 0,30 for DC, and 0,10 for planar. However, in configuration C2, we selected 0.23 for MIP, 0,41 for regular, 0,46 for DC, and 0,42 for planar.

As shown in these Tables, regular IPMs represent the highest F1-score, which indicates that the model can generalize well in this classification task. Yet, for the remaining intra coding tools, the MTL CNN prediction is relatively closer to the random guess, which is due to the remaining tools being under-represented in the training data set. For instance, under the default threshold, the model may have a low precision for DC, planar with a good recall. While, for MRL, it tends to have good precision and very low recall. This latter can be improved by varying the classification thresholds. Although, the precision would become lower, which may introduce some loss in the coding efficiency. For this reason, the flowchart introduced in Figure. 5.4 has been adopted to integrate the model under VTM10.0 with minimum coding loss.

As intended, we may also notice a slight decrease in the model performance for the intra coding tools, when introducing the CU size feature extraction as a related task. Henceforth, the number of FN as well as FP, in some cases, are slightly important in configuration C1 when compared to configuration C2, which lowers the F1-score for more than half of the intra coding tools. Also, because regular IPM represents one of the most powerful tools used in the VTM to predict the pixels within a CU, it is critical to ensure a higher model performance for this latter compared to the remaining intra coding tools, in order to limit the loss in the coding efficiency.

In light of all these findings, we decided to use configuration C2 when integrating our MTL CNN under the VTM encoder. As stated before, this would ensure a good trade-off between coding efficiency and complexity reduction for all the target tasks.

5.5 Complexity reduction under VTM

Table 5.3 gives the Performance of our MTL based intra mode decision framework in comparison with the state-of-the-art techniques. Indeed, all the native speed-up techniques

included in VTM10.2 are activated to ensure a fair comparison. As reported in Table 5.3, our MTL intra mode decision framework can achieve on average 25.21% of complexity reduction for only 1.33% of BD-BR increase. Compared to the state-of-the-art our method outperforms [101] and [104] with about 13% and 19% of complexity reduction on average for a BD-BR increase of 0.9% and 1.02, which is considered as tolerated. The closest performance to our method is [100], which enables 30.10% of complexity reduction on average. Indeed, this method picks in advance the best IPM and it is implemented under the VTM2.0, which does not use either MIP, MRL, or ISP. However, our method is able to achieve closer performance for a more complex encoder when only predicting the set of IPMs to be tested during the FIPMD. For instance, the use of MIP and ISP enhanced the coding efficiency at the expense of 10% and 15% [121] of complexity overhead, respectively.

The best performance of our method is presented for the video RaceHorsesC with about 30% of complexity reduction for only 1.03% of BD-BR increase. While, the worst case is presented for the video SlideEditting, which is screen content. Indeed, the performance of our method, especially in term of BD-BR, decrease with the video resolution and this is directly related to the training data set.

Table 5.3 Performance of the MTL based intra mode decision framework in comparison with the state-of-the-art techniques

Class	video	Chen, Y, et al. [100], VTM2.0		Park, J, et al. [101], VTM9.0		Li, Y, et al. [104], VTM8.0		Our proposal, VTM10.2	
		Δt (%)	BD-BR (%)	Δt (%)	BD-BR (%)	Δt (%)	BD-BR (%)	Δt (%)	BD-BR (%)
A1	Campfire	28.06	0.92	12.00	0.09	-	-	24.54	0.78
	Tango2	23.39	0.93	11.00	0.09	-	-	23.13	0.98
	FoodMarket4	20.13	0.64	10.00	0.09	-	-	22.43	0.91
	Average	**23.86**	**0.83**	**11.00**	**0.09**	**-**	**-**	**23.37**	**0.89**
A2	CatRobot1	26.89	0.94	12.00	0.30	-	-	23.43	1.13
	DaylighRoad2	32.99	0.98	11.00	0.49	-	-	24.64	1.59
	ParkRunning3	20.32	0.67	9.00	0.07	-	-	20.63	0.59
	Average	**26.73**	**0.86**	**10.67**	**0.29**	**-**	**-**	**22.89**	**1.10**
B	MarketPlace	-	-	12.00	0.13	-	-	25.99	1.02
	RitualDance	-	-	12.00	0.32	-	-	22.98	1.25
	Cactus	29.47	0.54	13.00	0.49	6.00	0.14	27.11	1.36
	BasketBallDrive	34.69	0.51	11.00	0.64	9.00	0.24	24.70	1.82

	BQTerrace	37.17	0.44	12.00	0.48	4.00	0.01	26.94	0.49
	Average	**33.78**	**0.50**	**12.00**	**0.41**	**6.33**	**0.13**	**25.54**	**1.19**
C	RaceHorses	43.69	0.56	12.00	0.37	6.00	0.07	29.87	1.03
	BasketBallDrill	41.28	0.36	16.00	1.02	11.00	0.30	28.12	1.52
	BQMall	27.64	0.61	12.00	0.88	6.00	0.10	26.13	1.74
	PartyScene	43.69	0.56	13.00	0.49	4.00	0.01	27.97	1.24
	Average	**39.34**	**0.50**	**13.25**	**0.69**	**6.75**	**0.48**	**28.02**	**1.38**
D	RaceHorses	28.05	0.73	14.00	0.39	5.00	0.12	28.74	2.04
	BQSquare	30.08	0.61	12.00	0.57	8.00	0.18	27.82	1.45
	BlowingBubbles	29.09	0.70	14.00	0.65	6.00	0.00	26.53	1.56
	BasketBallPass	26.49	0.49	12.00	0.66	8.00	0.04	22.96	1.42
	Average	**28.43**	**0.90**	**13.00**	**0.57**	**6.75**	**0.085**	**26.51**	**1.62**
E	FourPeople	26.32	0.66	-	-	7.00	0.17	23.63	1.73
	Johny	25.85	0.59	-	-	8.00	0.22	22.95	1.72
	KristenAndSara	26.77	0.59	-	-	8.00	0.10	23.50	1.95
	Average	**26.31**	**0.61**	**-**	**-**	**7.67**	**0.16**	**23.36**	**1.80**
Average		**30.10**	**0.65**	**12.10**	**0.43**	**6.86**	**0.12**	**25.21**	**1.33**
F	ArenaOfValor	-	-	-	-	-	-	24.09	1.61
	BasketBallDrillText	24.96	0.44	-	-	-	-	24.24	1.66
	SlideEditting	32.33	0.84	-	-	-	-	17.67	1.89
	SlideShow	32.50	0.66	-	-	-	-	20.78	1.92
	Average	**29.93**	**0.65**		**-**	**-**	**-**	**21.69**	**1.77**

As aforementioned, the model is trained only with 2K fixed images, which reduces its performance for lower video resolution. Also, no screen content was used to train the model, which makes it not optimized for this type of content. Even though, our method allows a good trade-off between complexity reduction and coding efficiency. Hence, varying video resolution and content may improve further our model performance under the VTM10.2.

After evaluating the effectiveness of our framework for the intra prediction, this latter was combined with the two-stages-learning-based QTMT framework for the CU size decision [3] based on configuration two. Table 5.4 shows the performance of the overall proposal in comparison with the state-of-the-art techniques when separating the two tasks. Compared to the VTM anchor, the overall proposal can enable up to 63% of complexity reduction with a bite rate increase of 3.39%, which is relatively negligible.

Table 5.4 Performance of the overall proposal in comparison with the state-of-the-art techniques

Class	video	Cao, J, et al. [103], VTM4.2		Zhang, Q, et al. [106], VTM4.0		Tissier, A, et al. [3], VTM10.2		Our proposal + Tissier, A, et al. [3], VTM10.2	
		Δt (%)	BD-BR (%)	Δt (%)	BD-BR (%)	Δt (%)	BD-BR (%)	Δt (%)	BD-BR (%)
A1	Campfire	-	-	48.77	0.78	48.30	0.74	55.37	1.39
	Tango2	-	-	56.95	0.97	46.70	0.54	55.08	1.43
	FoodMarket4	-	-	52.53	0.84	47.50	0.53	54.36	1.36
	Average	**-**	**-**	**52.75**	**0.86**	**47.50**	**0.60**	**54.94**	**1.39**
A2	CatRobot1	-	-	54.55	1.08	46.10	0.99	53.78	2.12
	DaylighRoad2	-	-	53.63	0.78	50.60	1.02	59.01	3.03
	ParkRunning3	-	-	52.67	0.81	42.60	0.38	47.25	1.15
	Average	**-**	**-**	**53.61**	**0.89**	**51.00**	**1.10**	**53.35**	**2.10**
B	MarketPlace	-	-	-	-	55.60	0.53	64.43	1.57
	RitualDance	-	-	-	-	51.60	0.87	61.03	2.28
	Cactus	49.29	0.89	-	-	49.40	0.89	61.82	2.42
	BasketBallDrive	59.62	1.07	-	-	51.60	0.92	63.00	2.93
	BQTerrace	43.18	1.00	55.21	1.07	46.50	1.06	59.56	1.80
	Average	**50.70**	**0.99**	**55.21**	**1.07**	**51.00**	**0.85**	**61.97**	**2.20**
C	RaceHorses	0.59	39.05	56.71	0.92	46.20	0.66	63.04	1.88
	BasketBallDrill	46.61	1.87	59.65	0.97	47.00	1.01	61.26	3.21
	BQMall	42.94	0.96	-	-	44.10	0.58	62.85	3.04
	PartyScene	39.03	0.67	56.65	0.95	45.40	1.60	60.96	2.10
	Average	**41.61**	**0.95**	**57.67**	**0.73**	**45.70**	**0.96**	**62.03**	**2.56**
D	RaceHorses	38.98	0.70	54.67	0.76	43.60	0.67	62.15	4.13
	BQSquare	35.43	1.02	56.12	0.73	44.40	0.61	61.02	2.18
	BlowingBubbles	43.97	0.92	58.61	0.69	40.50	0.64	58.09	2.43
	BasketBallPass	48.08	1.15	-	-	44.50	0.90	58.77	2.72
	Average	**46.16**	**0.91**	**56.46**	**0.73**	**43.3**	**0.70**	**60.01**	**2.86**
E	FourPeople	47.02	1.27	54.34	0.85	44.70	1.20	63.34	3.22
	Johny	53.31	1.43	51.46	0.85	42.6	1.42	63.77	3.55
	KristenAndSara	49.06	1.08	52.62	1.32	44.40	1.06	62.68	3.39
	Average	**49.80**	**1.26**	**52.81**	**1.01**	**43.90**	**1.23**	**63.26**	**3.39**
Average		**-**	**-**	**-**	**-**	**46.60**	**0.86**	**59.67**	**2.42**
F	ArenaOfValor	-	-	-	-	22.00	1.08	56.82	2.87
	BasketBallDrillText	-	-	-	-	22.50	1.59	58.15	3.26
	SlideEditting	-	-	-	-	27.30	1.21	58.01	3.13

SlideShow	-	-	-	-	26.00	1.50	56.91	4.07
Average	**-**	**-**	**-**	**-**	**24.60**	**1.34**	**57.47**	**3.33**

As illustrated in Table 5.4, the overall proposal achieved superior performance in terms of complexity reduction compared to the state-of-the-art techniques such as [3] and [103]. Nevertheless, it presents a significant penalty for coding efficiency. This is generally due to the fact that every coding tool in the hybrid video coding scheme may impact the remaining set of tools during the encoding process. For instance, choosing a larger block size for narrow texture blocs may influence their prediction and vice versa. The impact on any coding tool is therefore introduced by an increase in the residual error, which subsequently increases the loss in coding efficiency.

Conclusion

In this Chapter, a MTL based intra mode decision framework for VVC is proposed. It deals with deciding the best IPM by reducing the set of candidates in the FIPMD according to the top-2 inferred intra coding tools. Simulations proved the success of our method, which can achieve up to 30% of complexity reduction with a slight increase of BD-BR. Moreover, our method can provide between 8% to 20% of additional complexity reduction when combined with previous work [3] on fast CU size decision.

Chapter 6

Lightweight Decision Tree for low-complexity intra mode decision

Introduction

Ensemble models have been widely used to improve prediction accuracy since their first breakthrough in the early 2000s. In contrast to single models, ensemble models use a collection of weak learners in order to form a strong one. The reliability of this kind of learning arises from the presumption that the subsequent model combined with prior models' predictions or hypotheses can perform way better than a single model with random initialization. Hence, this chapter proposes a lightweight Decision Tree (DT) for low complexity intra mode decision, which takes advantage of this ensemble learning to improve the performance and reduce further the inference time.

6.1 Light-weight DT for low-complexity intra mode decision

To tackle the intra mode decision and improve the prediction accuracy, several binary classifiers based on Light-Gradient Boosting Machine (Light-GBM) were fit to predict the intra coding tools at Coding Unit (CU) level. To limit the development time, the proposed method deals only with square blocks to shrink the decision set of the Final Intra Prediction Mode Decision (FIPMD).

6.2 Overall presentation of the proposed framework

Ensemble models are one of the most powerful models in machine learning. It learns by adding weak learners in a way that minimizes the overall error rate or the loss function [93]. As illustrated Figure. 6.1, our proposed method adopts this ensemble learning to select the

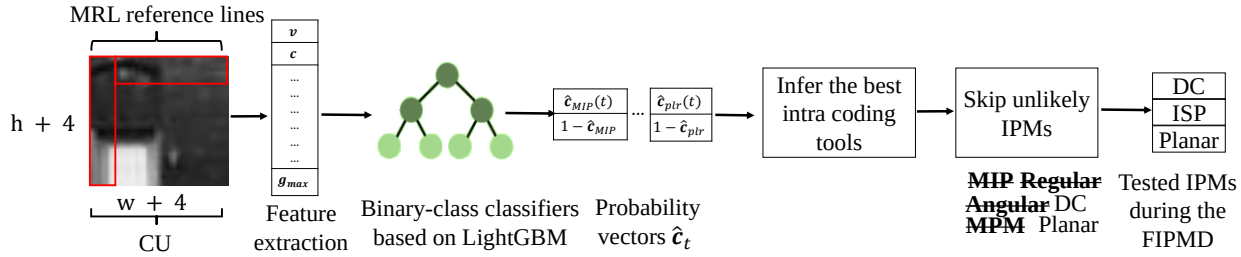


Figure 6.1 Workflow of the light-weight DT for low complexity intra mode decision. First, several texture-based as well as encoding features are extracted from the current CU, where h and w represent the CU dimensions. Then, these latter are processed using several binary-class classifiers based on Light-GBM [5] to get the probability vector $\hat{\mathbf{c}}_t$, indicating whether to skip an intra coding tool t or not. Finally, the best intra coding tools, which have the highest probabilities are used to shrink the decision set in the FIMPD

top intra coding tools for each CU, which also reduces the number of candidates for the FIPMD. It takes as an input several texture-based and encoding features, that are strongly related to the selection of the best intra coding tool, in order to output a probability vector $\hat{\mathbf{c}}_\mathbf{t}$ indicating whether to use Matrix weighted Intra Prediction (MIP), Intra Sub-Partitions (ISP), regular, angular Intra Prediction Modes (IPMs), Most Probable Modes (MPMs), DC or planar in the current CU size or not.

6.2.1 Output vector representation

Instead of using the Coding Tree Unit (CTU) level representation as in the previous proposal, we used a CU size adaptive technique to fit the different Light-GBM classifiers. As it can be seen in Figure. 6.2. the proposed Light-GBM classifiers operate by CU size in order to output a probability vector $\hat{\mathbf{c}}_t$ of two entries, indicating whether to skip an intra coding tool $t \in \{\text{MIP, ISP, regular, MPM, angular, DC, planar}\}$ or not.

In Versatile Video Coding (VVC), square blocks can have a size ranging from 64×64 to 4×4. Therefore, five Light-GBM classifiers are used per intra coding tool t in order to predict whether this latter should be tested during the FIPMD or not.

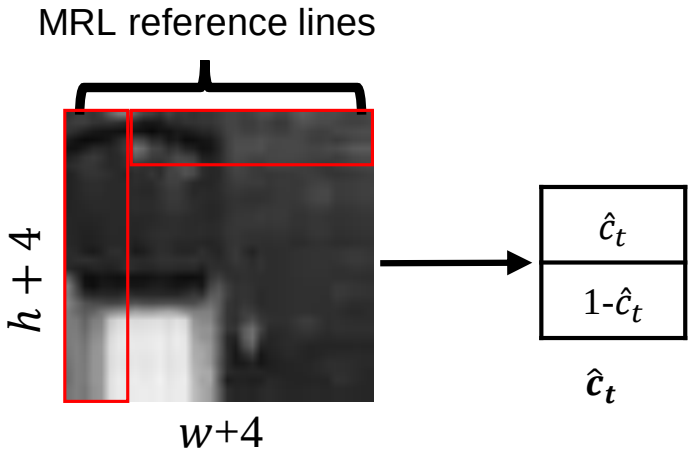


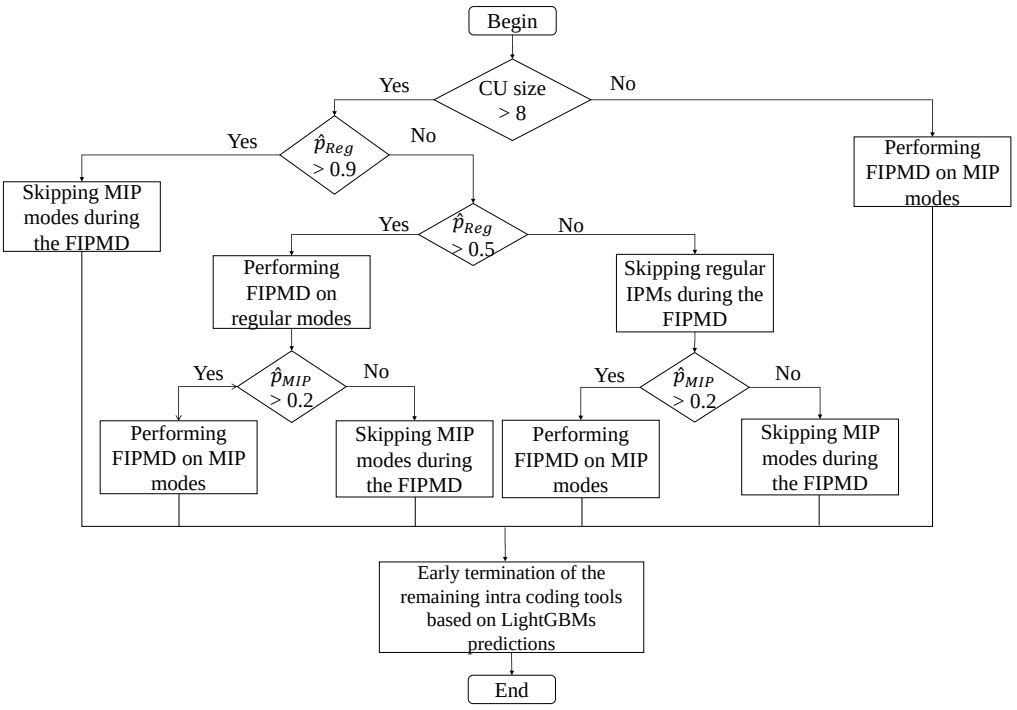
Figure 6.2 Representation of the output vectors $\hat{\mathbf{c}}_\mathbf{t}$ of the two probabilities, that indicate whether an intra coding tool t should be tested for the current CU or not

6.2.2 Integration under VTM

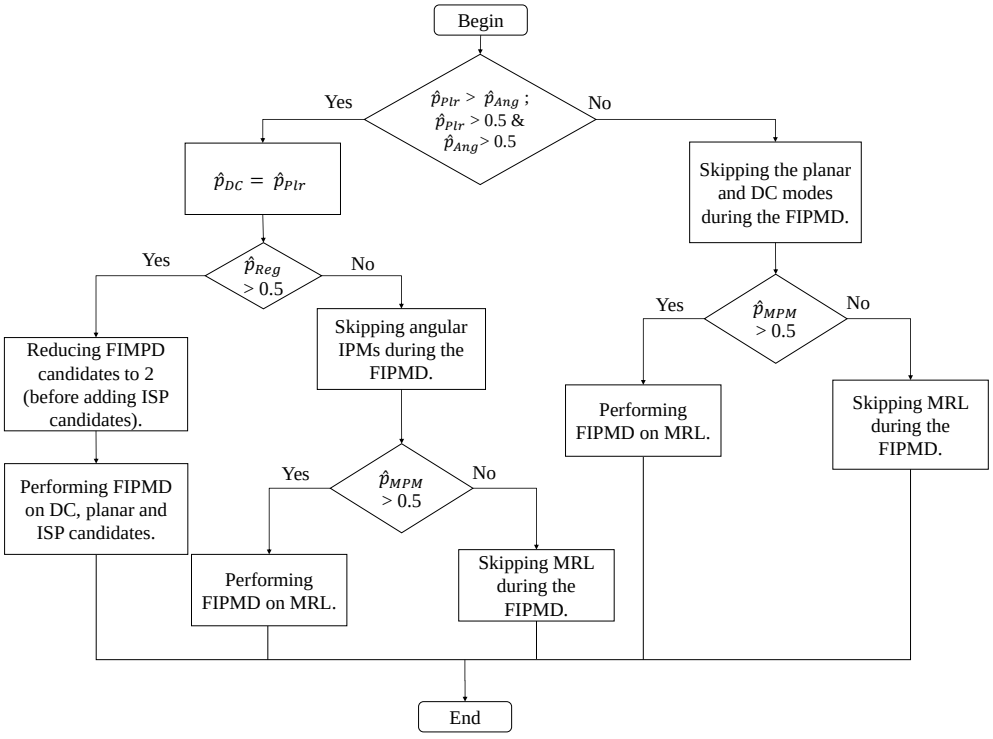
As shown in Figure. 6.3, the different Light-GBM classifiers were integrated into the VVC Test Model (VTM)10.2 using almost the same proposed algorithm as in our previous proposal. Yet, the best intra coding tools are determined at each CU level based on several conditions instead of using the top-2 strategy.

First, if the probability of regular IPMs is very high (i.e. > 0.9) then the test of MIP modes is turned off immediately for the current CU, unless the CU size is less than 8×8. In this case, the test of MIP modes remains activated. Second, taking into account that planar and DC modes are usually used for homogeneous regions we can infer the likelihood of testing DC from the likelihood of testing planar. Thus, when the test of planar mode is skipped, DC mode can be skipped as well. Third, if the probability of planar is greater than angular, then this latter is also skipped immediately without performing the Light-GBM prediction and vice versa. Finally, if the best IPM is inferred as non MPM, then Multi-Reference Lines (MRL) are skipped too.

It's worth mentioning that the ISP was excluded from the proposed framework since it does not introduce a significant complexity reduction for square blocks. Also, we used the default threshold (i.e 0.5) to decide whether to skip an intra coding tool t or not, except for MIP several experiments, under the VTM10.2, were conducted to decide the



(a)



(b)

Figure 6.3 Process of integrating the different light-GBM classifiers under the VTM10.2 to skip unlikely IPMs; (a): early termination of MIP based on regular IPMs; (b): early termination of the remaining intra coding tools

best classification threshold, which reduces the loss on the coding efficiency.

6.3 Training data set and training process

Throughout this section, the training data set is provided. Then the different LightGBM classifiers with their training process are described in detail.

6.3.1 Training data set

To derive the training samples, the same image public data set Div2K [4] is considered. But, this time several texture-based features as well as encoding features, that could have an impact on the selection of best intra coding tool, are selected to fit the different LightGBM classifiers instead of using the encoded CU immediately.

As shown in Chapter 3, the intra coding tool selection is closely related to the CU texture. For essence, a CU with vertical texture direction should be encoded using angular IPMs around the vertical direction rather than using DC or planar. Also, complex textures can reflect a tendency to select additional intra coding tools such as MIP, which would enhance the coding efficiency. Based on these observations, we choose the following texture-based features:

— v: refers to the variance on the whole CU defined as shown in Equation. (6.1).

$$v = \sum_{x=1}^{w} \sum_{y=1}^{h} (\mathbf{Y}(x, y) - m)^2 \tag{6.1}$$

where:

$\mathbf{Y}$ denotes the pixels' matrix of the whole CU.

x, y represents the pixel coordinates at the current CU.

m is the mean of the pixels' intensity in the current CU.

w, h are the width and height of the current CU, respectively.

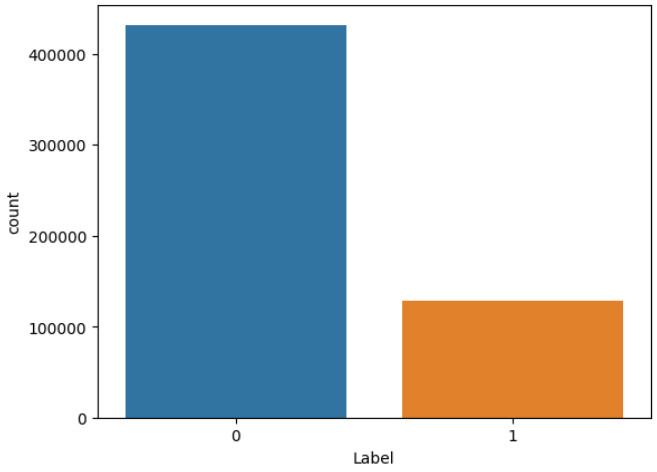
— $\mathbf{G}_x$ and $\mathbf{G}_y$: Stand for the gradients in horizontal and vertical directions on the whole CTU, where only the gradients of the four co-located pixels of each CU, namely top-left, top-right, bottom-left, and bottom-right pixels are used. The gradients are calculated using the Sobel operator, defined in Equation. (6.3), as

$\mathbf{G}_x = \mathbf{Y} \times \mathbf{S}_x$ and $\mathbf{G}_y = \mathbf{Y} \times \mathbf{S}_y$

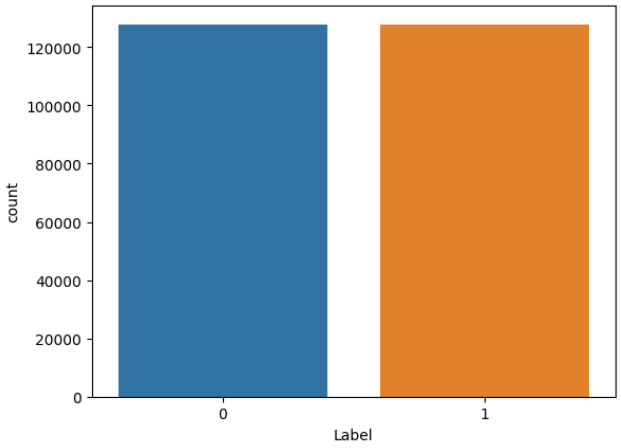
$$\mathbf{S}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}. \quad \mathbf{S}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}. \tag{6.2}$$

Where:

$\mathbf{Y}$ here refers to the pixels' matrix of the whole CTU.



(a)



(b)

Figure 6.4 Data set distribution for planar mode size 32×32; (a): Before under-sampling; (b): After undersampling

— c: The contrast on the whole CU given in Equation. (6.3).

$$c = \sum_{x=1}^{w} \sum_{y=1}^{h} (x - y)\mathbf{Y}(x, y) \tag{6.3}$$

— g_{max}: defines the maximum gradient among $\mathbf{G}_x$ and $\mathbf{G}_y$. This latter is computed by Equation. (6.4).

$$g_{max} = \max(\sum_{x=1}^{w} \sum_{y=1}^{h} \mathbf{G}_y(x, y), \sum_{x=1}^{w} \sum_{y=1}^{h} \mathbf{G}_x(x, y)) \tag{6.4}$$

As for encoding features the Quantization Parameter (QP), as well as neighboring reference lines are also selected to train the different Light-GBM classifiers. Because a Light-GBM classifier learns from annotated data, a binary label was used to show the model whether MIP, regular, angular, MPM, DC, and planar are used for the current CU.

As seen in Chapter 5, our training data set has a skewed distribution or class imbalance issue. Class imbalance, as aforementioned, occurs when nearly all samples of a data set are assigned to one class, whereas only a small portion of that data set is assigned to the other class [122]. One straightforward solution is to evaluate the model using more convenient performance metrics like recall/precision analysis in conjunction with an effective resampling of the training data set. For instance, we choose to use the under-sampling technique, in order to balance our data set distribution except for some CU sizes in regular, angular, and MPM data sets, which shows quite a balanced distribution.

Generally speaking, under-sampling consists in deleting samples from the majority class in order to ensure a uniform distribution of all classes in the training data set. As shown in Figure. 6.4, samples from the negative class were chosen and eliminated in order to deal with the overabundance of negative samples, which could miss laid the classification of the training as well as test samples.

6.3.2 Training process

Light-GBM [5] is a tree-based ensemble model, That uses the gradient Boosting strategy [123]. Gradient boosting is a very powerful algorithm that stands out for its prediction speed and accuracy, especially for big and complex data sets. It attempts, as shown in Figure. 6.5 to reduce the overall error rate or loss function by consecutively adding weak learners. This means that the gradient boosting algorithm starts with learning a base

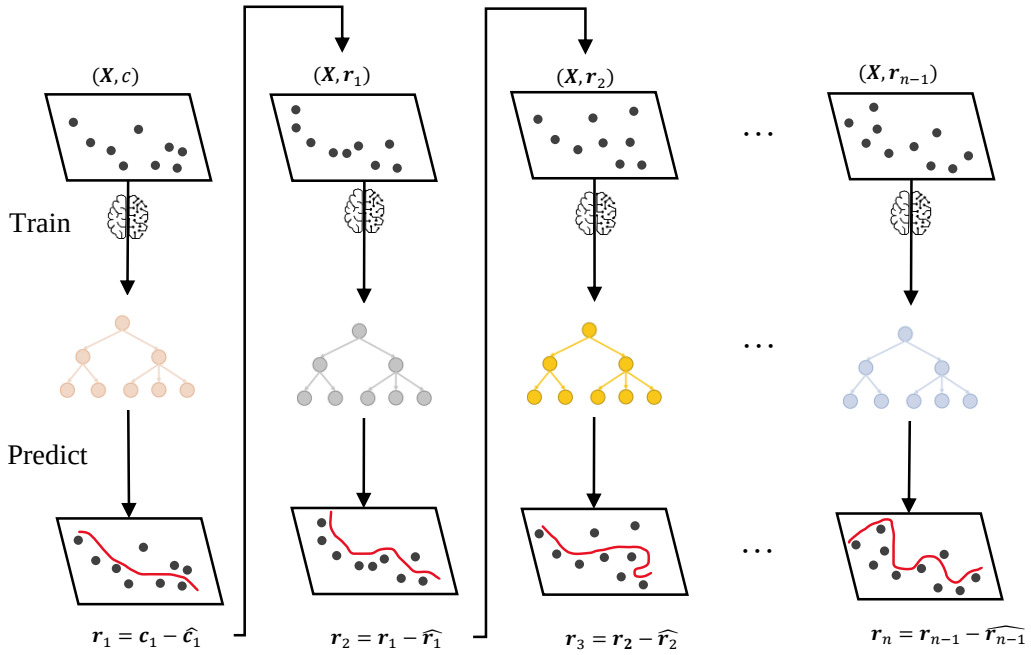


Figure 6.5 Gradient boosting strategy

learner, denoted as f_{m-1}, then at each iteration, it adds a new estimator or weak learner h_m to it, usually a DT, in order to improve the residual error of the previous model so far. This is introduced in Equation. (6.5).

$$f_m(x) = f_{m-1}(x) + \min(\sum_{i=1}^{n} L(\mathbf{c}_i, f_{m-1}(x_i) + h_m(x_i))) \tag{6.5}$$

Since gradient boosting uses the well-known Gradient Descent Search (GDS), it aims to minimize a loss function l, where l in our problem is the binary cross-entropy loss, defined in the Equation. (6.6).

$$l = l_t = \mathbf{c}_t(\log(\hat{\mathbf{c}}_t) + (1 - \mathbf{c}_t)(-\log(1 - \hat{\mathbf{c}}_t) \tag{6.6}$$

In order to train the different Light-GBM classifiers we used the training data set in a 4-folds cross-validation. As explained in Section 5.4.1, k-folds cross-validation is very useful for achieving a good bias/variance trade-off, which may improve further the model performance.

6.3.3 Fine-tuning with Grid search

As we may all know, most machine learning algorithms need to be well-parameterized in order to give the desired performance. For example, to train a Light-GBM classifier, we have to fix several hyper-parameters such as the number of weak DTs and the maximum number of leave nodes to be used at each of these DTs. Obviously, if we set these hyper-parameters manually, this can quickly become very time-consuming -and not necessarily very efficient. This is where fine-tuning algorithms can be very helpful and time-saving.

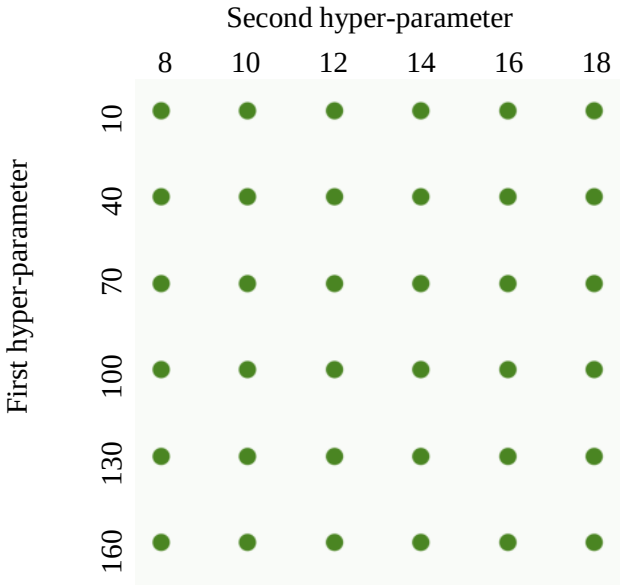


Figure 6.6 Illustration of Grid search for two hyper-parameters fine-tuning

Grid search is one of the most popular fine-tuning algorithms, that allows us to test a series of hyper-parameters and compare their performance in order to infer the best possible combination [124]. It creates, as shown in Figure. 6.6, a grid of hyper-parameter combinations and trains a model for each of them in order to select the best possible combination of hyper-parameters. The only drawback of this algorithm is that it's very time-consuming when a huge range of hyper-parameters is given. So we focused on optimizing the number of weak DTs, the maximum depth as well as the maximum number of leaf nodes [125] and we kept the remaining hyper-parameters in their default value. the values used to fine-tune the different Light-GBM classifiers are then given as follows:

— The number of weak DTs: $\{100, 150, 200, 300, 1000, 1500, 2000\}$

— The maximum number of leaf nodes: $\{31, 120, 160, 200, 220\}$

— The maximum depth: $\{-1, 10, 20, 40\}$

After running the Grid search on the different Light-GBM classifiers, the best combination of hyper-parameters is selected, as indicated in Table 6.1.

Table 6.1 Selected hyper-parameters by intra coding tool

Coding tool	Size	Hyper-parameter	Value
MIP	64×64	Number of weak DTs	300
		Maximum number of leaf nodes	160
		Maximum depth	20
	32×32	Number of weak DTs	1500
		Maximum number of leaf nodes	160
		Maximum depth	40
	16×16	Number of weak DTs	2000
		Maximum number of leaf nodes	220
		Maximum depth	20
Regular	64×64	Number of weak DTs	100
		Maximum number of leaf nodes	31
		Maximum depth	-1
	32×32	Number of weak DTs	100
		Maximum number of leaf nodes	31
		Maximum depth	-1
	16×16	Number of weak DTs	100
		Maximum number of leaf nodes	31
		Maximum depth	-1
	8×8	Number of weak DTs	100
		Maximum number of leaf nodes	31
		Maximum depth	-1
	4×4	Number of weak DTs	100
		Maximum number of leaf nodes	31
		Maximum depth	-1
	64×64	Number of weak DTs	100
		Maximum number of leaf nodes	31
		Maximum depth	-1

MPM	32×32	Number of weak DTs	1000
		Maximum number of leaf nodes	200
		Maximum depth	40
	16×16	Number of weak DTs	1500
		Maximum number of leaf nodes	220
		Maximum depth	-1
	8×8	Number of weak DTs	100
		Maximum number of leaf nodes	31
		Maximum depth	-1
	4×4	Number of weak DTs	100
		Maximum number of leaf nodes	31
		Maximum depth	-1
Planar	64×64	Number of weak DTs	300
		Maximum number of leaf nodes	200
		Maximum depth	20
	32×32	Number of weak DTs	300
		Maximum number of leaf nodes	-1
		Maximum depth	200
	16×16	Number of weak DTs	150
		Maximum number of leaf nodes	120
		Maximum depth	-1
	8×8	Number of weak DTs	200
		Maximum number of leaf nodes	120
		Maximum depth	20
	4×4	Number of weak DTs	val
		Maximum number of leaf nodes	val
		Maximum depth	val
	64×64	Number of weak DTs	300
		Maximum number of leaf nodes	220
		Maximum depth	-1
	32×32	Number of weak DTs	1000
		Maximum number of leaf nodes	200
		Maximum depth	40
	16×16	Number of weak DTs	2000

Angular		Maximum number of leaf nodes	220
		Maximum depth	20
	8×8	Number of weak DTs	100
		Maximum number of leaf nodes	31
		Maximum depth	-1
	4×4	Number of weak DTs	100
		Maximum number of leaf nodes	31
		Maximum depth	-1

It is worth mentioning that the ISP, DC, as well as small CU sizes for MIP, were excluded from the training process due to their insufficient training samples, especially after performing the under-sampling technique. For instance, ensemble models, such as Light-GBM classifiers, are prone to over-fitting and can quickly over-fit small training data sets.

6.4 Experimental results

In order to evaluate the performance of our proposed method, the reference software VTM10.2 [108] is used to run the experiments based on the Common Test Conditions (CTC) [114] and 26 videos from classes A1, A2, B, C, D, E, and F to account for the variety of video content. The coding configuration is set to All Intra (AI) and 4 QP values, namely 22,27,32, and 37 are used to encode all the video sequences. As for the number of encoded frames, one Group of Pictures (GOP) is used as aforementioned in Chapter 5.

The evaluation metrics include both complexity reduction and coding efficiency. Consequently, coding efficiency is assessed using the Bjontegaard Delta Bitrate (BD-BR) [22], then the complexity reduction is given by the average of the run-time difference ΔT, defined by Equation. (6.5).

$$\Delta t = \frac{1}{4} \sum_{QP_i=22,27,32,37} \frac{t_o(QP_i) - t_r(QP_i)}{t_o} \tag{6.7}$$

Where:

t_o and t_r denote the total encoder runtimes of the VTM anchor and the VTM when using the Light-GBM classifiers, respectively.

The Light-GBM classifiers were built, trained, and converted to C++ with the Light-GBM framework, introduced by Microsoft Corporation [5] in order to be used to skip unlikely intra coding tool in the VTM encoder.

6.5 Performance of the light-GBM classifiers

The performance of the different Light-GBM classifiers was evaluated in the first place using accuracy. Then the precision/recall analysis was used in order the ensure the relevance of their predictions. The accuracy is reported on the training data set, as well as the test data set in order to check for over-fitting. whereas, only the test data set is used to measure the generalization of the different Light-GBM classifiers.

Table 6.2 Performance of the different light-GBM classifiers

Coding tool	Size	Train accuracy	Test accuracy	Precision	Recall	F1-score
MIP	64×64	0.81	0,61	0,49	0,64	0,56
	32×32	0,84	0,61	0,55	0,63	0,59
	16×16	0,79	0,56	0,46	0,59	0,51
Regular	64×64	0,63	0,63	0,64	0,89	0,75
	32×32	0,60	0,58	0,58	0,66	0,62
	16×16	0,56	0,56	0,55	0,95	0,70
	8×8	0,66	0,68	0,91	0,71	0,80
	4×4	0,63	0,41	0,98	0,38	0,54
MPM	64×64	0,59	0,57	0,57	0,58	0,58
	32×32	0.82	0,60	0,53	0,63	0,58
	16×16	0,74	0,55	0,47	0,59	0,52
	8×8	0,52	0,57	0,71	0,65	0,68
	4×4	0,53	0,54	0,73	0,55	0,63
Planar	64×64	0.86	0,62	0,42	0,64	0,51
	32×32	0,78	0,55	0,28	0,60	0,38
	16×16	0,61	0,47	0,19	0,61	0,30
	8×8	0,55	0,46	0,36	0,72	0,48

	4×4	0,57	0,52	0,36	0,57	0,44
Angular	64×64	0,88	0,64	0,38	0,65	0,48
	32×32	0,87	0,62	0,38	0,63	0,47
	16×16	0,80	0,57	0,44	0,60	0,51
	8×8	0,53	0,53	0,53	0,99	0,69
	4×4	0,53	0,56	0,64	0,67	0,65

As illustrated in Table 6.2, the training and the test accuracies are quite closer to each other, which means that the different Light-GBM classifiers are not over-fitting the training data set. Furthermore, regular, MPMs and MIP represent the highest F1-score. This means that the trained Light-GBM classifiers on these classification tasks can perform well on new data, regardless of the CU size. Yet, for planar, the performance of the trained Light-GBM classifiers is deemed insufficient. For instance, the trained Light-GBM classifiers on planar have relatively poor precision, which could result in some loss in the coding efficiency. Also, the trained Light-GBM classifiers on angular IPMs for high CU sizes are not well optimized due to the fact that shallow partitions are usually encoded using non-angular IPMs (i.e planar and DC modes). For this reason, the flowchart illustrated in Figure. 6.3 is adopted to integrate the different Light-GBM classifiers under the VTM10.2 with minimum coding loss.

6.6 Complexity reduction under VTM

The light-weight DT framework is integrated into the VTM10.2. In order to ensure a fair comparison, all of the native speed-up techniques, available in the VTM10.2 are enabled. Also, the VTM is forced to encode only with square blocks in order to provide a clear insight into the achieved complexity reduction.

Table 6.3 shows the performance of our proposal compared to the VTM10.2 anchors. As given in this Table, two configurations are presented based on whether the trained Light-GBM classifiers on MIP are used or not. On average, our proposal can achieve up to 15.16% of complexity with 1.26 of BD-BR increase when all the trained Light-GBM

classifiers are used to infer the best intra coding tools and 9.07% for only 0.49 of loss in BD-BR when MIP tests remain activated. This achieved complexity reduction is deemed as insignificant when contrasted to the increase in BD-BR, especially for very textured video sequences, like Tango2 and FoodMarket4. As already stated in Section 6.3.1, complex textures may lead to the use of additional intra coding tools such as MIP as well as angular IPMs. Unfortunately, the trained Light-GBM classifiers on MIP and angular IPMs are not well optimized and may cause a significant loss in the coding efficiency. Furthermore, small CU sizes present a major limitation of our proposal since there is no Light-GBM classifier trained on MIP for these sizes. These latter are well-suited to encode textured regions, which might result in a significant time saving because partitioning would be as deep as the texture complexity. So adapting the training data set as well the ensemble learning algorithm to the desired CU size and intra coding tool may be a solution to overcome this limitation.

Table 6.3 Performance of the proposed method in comparison with the VTM anchor

Class	video	Our proposal without MIP, VTM10.2		Our proposal with MIP, VTM10.2	
		Δt (%)	BD-BR (%)	Δt (%)	BD-BR (%)
A1	Campfire	8.24	0.20	14.21	1.45
	Tango2	9.22	2.16	15.88	5.48
	FoodMarket4	8.20	1.03	15.45	4.63
	Average	**8.55**	**1.13**	**15.18**	**3.85**
A2	CatRobot1	10.22	0.35	16.05	1.49
	DaylighRoad2	9.22	0.64	14.69	1.55
	ParkRunning3	8.25	0.11	17.00	0.47
	Average	**9.14**	**0.37**	**15.92**	**1.17**
B	MarketPlace	9,28	0.17	16.62	1.95
	RitualDance	8.52	0.46	15.34	1.14
	Cactus	8,62	0.24	15.22	0.92
	BasketBallDrive	8.72	0.56	14,48	0.50
	BQTerrace	7.41	0.34	13,12	0.38
	Average	**8.51**	**0.35**	**14,96**	**0.98**
C	RaceHorses	10.38	0.13	17.28	0.67
	BasketBallDrill	8.87	0.49	15.53	1.09
	BQMall	8.97	0.08	12.96	0.42
	PartyScene	-	-	15.59	0.46
	Average	**9.69**	**0.23**	**15,34**	**0.66**

D	RaceHorses	8,63	0.37	15.94	0.13
	BQSquare	9.95	0.25	15.05	0.57
	BlowingBubbles	12,11	0.85	18,63	1.29
	BasketBallPass	11,14	0.21	14.20	0.24
	Average	**10.46**	**0.42**	**15.96**	**0.56**
E	FourPeople	8.44	0,35	14.42	0.80
	Johny	7.87	0.46	13.25	0.92
	KristenAndSara	7.52	0.36	12.55	1.26
	Average	**7.94**	**0.39**	**13.41**	**0.99**
	Average	**9.09**	**0.49**	**15.16**	**1.26**
F	ArenaOfValor	7.81	0.17	14.65	0.41
	BasketBallDrillText	8,53	0.33	14,30	0.72
	SlideEditting	9,28	3.76	14.16	3.90
	SlideShow	10.69	0.88	15.87	1.22
	Average	**9.07**	**1.29**	**14,75**	**1.56**

For Class F our technique yields 14.75% of complexity reduction for a 1.56 % increase in BD-BR. Indeed, specific videos featuring screen content, such as slides or gaming content, are included in this class. Because these contents were not evaluated during the training process, our technique may not be well-suited for them. Adding screen content to the training data set could therefore improve the achieved results.

Conclusion

This Chapter proposes a light-weight DT for low complexity intra mode decision. It considers estimating the likelihood or the probability of using each intra coding tool t at the CU level for square blocks using a number of Light-GBM classifiers. Using these probabilities, the parameter search space is then reduced, which allows to skip unlikely IPMs. The main goal of this contribution was to enhance further the performance and reduce the inference time. But, due to several limitations such as the CU size as well as the training data sets, this proposal should undergo a number of improvements in order to achieve the intended results.

Chapter 7

Conclusion

7.1 Summary of Ph.D. Achievements

It is undeniable that video technology is moving into a new era. From the streaming of digitized content to the use of sophisticated Augmented Reality (AR), the video industry is evolving quickly [6]. In fact, many key concepts have been introduced into the market, such as overlaying of digital content in the life environment, shooting, sharing, and streaming 360° videos, and, certainly, inclusive access to streaming platforms. Amid to these revolutionary changes, video traffic over the internet has quadrupled in only a few years, reaching roughly 82% of global IP traffic [6]. This rapid growth in video demand has made it crucial for organizations, such as International Telecommunications Union (ITU) and ISO/IEC Motion Picture Experts Group (MPEG), to urgently tackle the potential need for a more efficient video coding standard than High-Efficiency Video Coding (HEVC). Thus, in July 2020, the new video coding standard, namely Versatile Video Coding (VVC), was released by the Joint Video Experts Team (JVET).

Typically, versatile stands for various coding tools, that allow VVC to deliver high-quality videos at low bit rate cost and support a wide variety of media services. In fact, the VVC is able to encode Ultra-High Definition (UHD) and immersive video content at nearly 40% of the bit rate saving compared to its predecessor, HEVC [9]. This outstanding compression performance, as aforementioned, is essentially based on several improvements of the block-based hybrid video coding scheme. The new QuadTree with nested Multi-type Tree (QTMT) [10], for example, supports wide homogeneous regions in high spatial resolutions as well as rectangular narrow texture blocks. As for intra-prediction, 65 finer-granularity angular Intra Prediction Modes (IPMs) [10] with DC and planar modes were introduced alongside several novelty coding tools. These latter include Multi-Reference Lines (MRL), low-complexity neural network-based intra-prediction, also known as Matrix weighted Intra Prediction (MIP) and Intra Sub-Partitions (ISP). In addition, VVC supports new motion compensation techniques, such as affine motion compensation [39],

from different control points. It also enhances transformation, quantization, and entropy coding with several tools, such as Low-Frequency Non-Separable Transform (LFNST), Multiple Transform Selection (MTS), dependent scalar quantization, and Context Adaptive Binary Arithmetic Coding (CABAC) [20].
Despite achieving substantial coding efficiency and wide coding support, the computational complexity remains a key challenge, especially when looking toward real-time implementation of the VVC codec on streaming or embedded devices. According to [9], the VVC Test Model (VTM), in All Intra (AI) configuration, is 31 times more complex than the HEVC Test Model (HM), which is technically beyond most streaming or embedded device capabilities.
Under these circumstances, this thesis showed that the use of artificial intelligence algorithms such as Multi-Task Learning (MTL) and Light-Gradient Boosting Machine (LightGBM) would help in reducing the computational complexity of the VVC encoder and may open up additional hardware optimizations, such as parallelism or Graphics Processing Unit (GPU) acceleration. So, the key contributions of this thesis are summarized as follows:

An in-depth complexity assessment of the intra prediction: This latter is conducted to underline the complexity reduction opportunities in the intra coding tools of VVC, which could yield up to 43%. This latter also includes a statistical analysis of the most time-demanding step of the intra mode decision, which is the Final Intra Prediction Mode Decision (FIPMD). It provides a clear understanding of the various correlations, that may aid the development of fast encoding decision algorithms for the VVC encoder. For instance, VVC tends usually to select the 6 Most Probable Modes (MPMs). However, this selection may depend on the used intra coding tool and video texture characteristics.

Multi-task Learning based intra mode decision framework: This contribution establishes a large database for the intra prediction and proposes a novel MTL based intra mode decision framework. For this purpose, a shared Convolutional Neural Network (CNN) was trained on the new intra coding tools, including finer-granularity angular intra prediction, MIP, etc...This framework actually leverages common knowledge among all the intra coding tools and limits the parameter search space to the top-2 inferred intra coding tools in order to reduce the computational complexity of the VVC encoder. Experimental results show that this framework could enable up to 30% of complexity reduction while

slightly increasing the Bjontegaard Delta Bitrate (BD-BR). It also introduces additional complexity reduction when combined with previous work [3] and would open up a number of hardware optimizations such as parallelism or GPU acceleration.

Light-weight Decision Tree (DT) for low-complexity intra mode decision : This latter involves using several binary classifiers based on Light-GBM in order to estimate the likelihood or the probability of using each intra coding tool at Coding Unit (CU) level for square blocks. The parameter search space is then shrunk using these probabilities and thus skipping unlikely IPMs. Simulations reveal a number of limitations due to the use of this technique only for square blocks. Indeed the achieved complexity reduction is deemed insignificant when contrasted to the increase in BD-BR, especially for very textured video sequences. However, this latter may be enhanced by introducing some improvements in the learning pipeline such as extending the technique to support Multi-type Tree (MT) CUs and adapting the ensemble learning algorithm as well as the training data sets to the intra coding tool.

7.2 Perspectives for future works

The perspectives related to the proposed frameworks and the subject tackled at the scope of this thesis are multiple.
In Part I, an in-depth complexity assessment of the intra prediction was conducted in order to evaluate the upper bound of complexity reduction in the intra coding tools of VVC. Also, several statistical analyses were provided to give a better understanding of the most time-demanding step of the intra mode decision and analyze the range of selected IPMs by intra coding tool. However, the video content as well as the texture complexity were not discussed roughly to provide further information about the correlation between the selected intra coding tool and the coded area. As a result, further statistical analyses may be required to improve the proposed frameworks.
In Part II, we presented two novel intra mode decision frameworks using artificial intelligence. The first framework uses a MTL CNN that leverages common knowledge among all the intra coding tools in order to predict the probability vectors for all of them simultaneously. In addition, our proposed CNN supports all CU sizes throughout the use of a Coding Tree Unit (CTU) level representation. These latter would help in accommodating the high diversity of block shapes in the QTMT partitioning and reduce the

model inference time. However, our proposed method lacks disparity in video resolution and does not support screen content. One major factor in maximizing the performance of deep learning is to ensure the diversity in training samples, particularly with regard to its ability to generalize on new data. So, for future work, we can consider including low video resolutions and screen content in the training data set. Furthermore, this contribution can open up additional hardware optimizations, such as parallelism or GPU acceleration. For instance, by dedicating a GPU for prediction or implementing the proposed method on a multi-core Central Processing Unit (CPU) architecture, the CNN's inference time can be decreased significantly.

The second framework is a light-weight DT for low complexity intra mode decision. This latter consists in using a number of Light-GBM classifiers to infer the likelihood or probability of using each intra coding tool at CU level. Indeed, the proposed framework deals only with square blocks. As future work, we can consider extending it to support MT CUs, which may offer up to 91% of complexity reduction [126]. Also, small CU sizes as well as some intra coding tools such as ISP and DC have shown low performance due to their insufficient training samples. Hence, we can also consider adapting the training data set to these intra coding tools and small CU sizes or using a much more suitable ensemble learning such as bagging. As explained in [127], the ISP, for example, is typically used at the texture's edge. More preciously, CUs that use this tool should have separation lines that divide them into various sub-partitions.

So far, we have only considered the intra coding mode, where the predicted samples are extrapolated from neighboring reference samples. So, we think it is also important to extend the work to support inter coding mode. Inter coding is typically used in conjunction with intra coding in order to predict video areas that include previously occluded regions. In VVC, inter prediction is also more complex than that of HEVC. For instance, the number of motion candidates has been increased and several new motion estimation techniques such as affine motion compensation and Adaptive Motion Vector Resolution (AMVR) were introduced to predict and signal the motion information. While these additional techniques have little effect on a decoder, an encoder may need to test several choices in order to maximize the coding efficiency. Thus, one of the potential perspectives to extend the current work is to exploit artificial intelligence algorithms in order to speed up the inter mode decision.

Chapter 8

Résumé en Français

8.1 Préamble

La consommation de la vidéo est en plein essor. Dans un contexte de travail à distance et de nouvelles tendances immersives, le trafic vidéo sur internet semble représenter plus des trois quarts du trafic internet mondial [6]. Apparemment, l'amélioration de l'expérience de l'utilisateur est devenue de plus en plus vitale aujourd'hui. Depuis la fin des années 1980, plusieurs algorithmes de codage vidéo ont été introduits afin de permettre le stockage et la transmission efficaces de la vidéo [7]. Par exemple, une série de H.26x a été publiée au début des années 2000 et peu après la norme High-Efficiency Video Coding (HEVC) a été publié dans le but de permettre la transmission de la vidéo en Haute Définition (HD). Toutefois, l'industrie de la vidéo reconnaît aujourd'hui une révolution remarquable en communication visuelle, notamment avec l'introduction d'applications vidéo immersives tel que la vidéo 360°.

Jusqu'à ici, les nouvelles avancées en Technologie d'Information (TI) ont permis de projeter des contenus numériques sur l'environnement analogique et de diffuser des services média en Ultra Haute Définition (UHD), ce qui est sans aucun doute hors des capacites de la norme HEVC. Pour cela, l'International Telecommunications Union (ITU) et l'International Organization for Standardization (ISO) ont publié conjointement une nouvelle norme de codage vidéo, appelée Versatile Video Coding (VVC). Cette norme a été développée par le groupe Joint Video Experts Team (JVET) afin de répondre aux exigence des nouvelles applications vidéos.

Bien qu'elle soit destinée à augmenter le taux de compression pour les résolutions vidéo trés importantes, les organismes de standardisation ont également doté la nouvelle norme d'un ensemble des techniques permettant de prendre en charge un large éventail des applications vidéo [8]. En effet, la norme VVC est presque deux fois plus efficace, en termes de débit binaire, que son prédécesseur HEVC [9] pour la même qualité visuelle. Néanmoins, cette efficacité de codage exceptionnelle se fait au détriment d'une complexité

de calcul croissante, qui represente un défi majeur vis à vis de l'implémentation matérielle ou la mise en oeuvre en temps réel du codec VVC. En effet, le streaming en temps réel ou les systéme embarqués impliquent plusieurs contraintes telles qu'une faible latence, des capacités de traitement limitées et des prix abordables. Ainsi, l'étude des opportunités de réduction de la complexité et l'introduction d'algorithmes de décision rapide pour la norme VVC sont les objectifs principales de cette thèse.

8.2 Contexte et motivation

Le codec de base du VVC s'appui sur le HEVC Test Model (HM), où un grand nombre d'outils de codage, tels que le QuadTree with nested Multi-type Tree (QTMT) et la prédiction intra angulaire à granularité plus fine [10], ont été ajoutés dans le but d'augmenter l'efficacité du codage. En outre, le schéma de codage hybride basé sur les blocs reste toujours utilisé dans la nouvelle norme de codage vidéo, où la minimisation de la fonction de coût lagrangienne, également appelée Rate-Distortion Optimization (RDO) [11], est effectuée pour contrôler la contrainte de distorsion et du débit. L'impact de RDO sur la complexité de calcul a été visible à travers les différents efforts de standardisation. Sans surprise, la norme VVC semble souffrir du même problème en raison de l'augmentation de l'espace des paramètres. En fait, l'encodeur du VVC Test Model (VTM) est 31 fois [9] plus complexe que celui du HM. Puisque le RDO cherche à sélectionner les meilleurs paramètres pour encoder un flux binaire, il est éventuellement écrasant pour un système embarqué de tester tout l'espace de paramètres et même pour un service de streaming en temps réel de consacrer beaucoup de temps à résoudre ces paramètres. Par conséquent, il est souhaitable d'améliorer le compromis entre la complexité de calcul et l'efficacité du codage afin d'assurer le succès commercial de cette norme.

Plus récemment, l'intelligence artificielle est devenue une part croissante des TI en raison de son succès dans l'exploration de grosses bases de données et dans la prise de décisions potentielles sans être explicitement programmée [12]. Associé à cela, un intérêt croissant pour les techniques de codage vidéo basées sur l'intelligence artificielle se développe pour répondre aux limitations des systèmes embarqués et du streaming en temps réel.

8.3 Objectifs et contributions

L'objectif de cette thèse est d'étudier le potentiel des algorithmes d'intelligence artificielle dans la réduction de la complexité de la norme VVC. De fait, elle intègre les contributions suivantes:

Évaluation de la complexité de la prédiction intra cette évaluation de la complexité est menée pour étudier la complexité de la prédiction intra dans la norme VVC. Elle analyse la limite supérieure de la réduction de la complexité dans chaque étape de la décision de mode intra, y compris son impact sur le temps d'exécution de la décision. En outre, une analyse statistique de l'étape la plus gourmande en terme de temps d'exécution est étudiée pour fournir un aperçu plus clair des différentes corrélations, qui peuvent aider à réduire la complexité de calcul de la norme VVC.

Algorithme de décision de mode intra basé sur l'apprentissage multi-tâche cette contribution traite la réduction de la complexité de la décision de mode intra de la norme VVC. Elle établit une base de données pour la prédiction intra et propose un algorithme de décision de mode intra basé sur l'apprentissage multi-tâche. Par cela, un réseau de neurone convolutif partagé est proposé pour les nouveaux outils de codage intra, y compris la prédiction intra angulaire à granularité plus fine, le Matrix weighted Intra Prediction (MIP), etc... En effet, cet algorithm exploite les caractéristiques communes entre tous les outils de codage intra afin de limiter l'espace de la recherche RDO aux deux meilleurs outils de codage intra.

Arbre de décision léger pour une décision de mode intra peu complexe Dans cette contribution, nous avons utilisé le Light-Gradient Boosting Machine (Light-GBM) pour évaluer la probabilité d'utilisation de chaque outil de codage intra au niveau Coding Unit (CU). Par conséquent, un certain nombre de classifiers Light-GBM ont été entrainés sur une grande diversité de blocs carrés afin de déterminer s'il faut ignorer les outils de prediction intra MIP, Intra Sub-Partitions (ISP), Most Probable Mode (MPM), Intra Prediction Modes (IPMs) régulier ainsi que les modes angulaires, DC ou planar. Ces prédictions sont ensuite utilisées pour réduire l'espace de la recherche RDO, ce qui réduit également le nombre de candidats pour Final Intra Prediction Mode Decision (FIPMD).

8.4 Organisation du manuscrit

Cette thèse de doctorat est divisée en six chapitres et une annexe reprenant l'ensemble des publications scientifiques parues au cours de cette thèse de doctorat.

Le Chapitre 2 donne les fondements de base du codage vidéo en mettant l'accent sur les nouveaux outils de codage intra de la norme VVC. Il présente également un bref historique des normes de codage vidéo antérieurs.

Le Chapitre 3 étudie les opportunités de réduction de la complexité dans la décision du mode intra de la norme VVC. Il analyse la complexité des nouveaux outils de codage intra et la limite supérieure de réduction de la complexité dans la prédiction intra, tout en étudiant les différentes corrélations entre le RDO, les paramètres de codage et le contenu de la vidéo.

Le Chapitre 4 présente l'intelligence artificielle tout en introduisant ses sous-domaines les plus importants. En outre, il discute les traveaux antérieurs sur la décision rapide de paramètres d'encodage.

Le Chapitre 5 traite la réduction de la complexité de la décision du mode intra. Il propose un algorithme rapide de décision de mode intra basé sur l'apprentissage multi-tâche. Celui-ci réduit l'ensemble des candidats de selection en se basant sur les deux meilleurs outils intra prédits par le réseau de neurone convolutif multi-tâche.

Le Chapitre 6 propose un arbre de décision léger pour une décision de mode intra peu complexe. Il s'agit d'utiliser plusieurs classifiers Light-GBM pour estimer la probabilité d'utiliser chaque outil de codage intra au niveau CU pour les blocs carrés. L'espace de recherche des paramètres est ensuite réduit à l'aide de ces probabilités, ce qui permet d'ignorer les modes de prédiction intra peu probables.

Part III

References and Appendices

BIBLIOGRAPHY

[1] T. Amestoy, *Optimisation du Codec VVC basé sur la Réduction de Complexité et le Traitement Parallèle*. PhD thesis, INSA Rennes, 03 2021.

[2] S. K. E, "Convolutional neural network," 2023. [online] available at `https://developersbreach.com/convolution-neural-network-deep-learning/`.

[3] A. Tissier, W. Hamidouche, S. B. D. Mdalsi, J. Vanne, F. Galpin, and D. Menard, "Machine learning based efficient qt-mtt partitioning scheme for vvc intra encoders," *IEEE Transactions on Circuits and Systems for Video Technology*, 2023.

[4] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 126–135, 2017.

[5] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[6] cisco, "Cisco visual networking index: Forecast and trends 2017–2022," 2018. [online] available at `https://cloud.report/whitepapers/`.

[7] S. Akramullah, *Digital video concepts, methods, and metrics: quality, compression, performance, and power trade-off analysis*. Springer Nature, 2014.

[8] T. Zhang and S. Mao, "An overview of emerging video coding standards," *GetMobile: Mobile Computing and Communications*, vol. 22, no. 4, pp. 13–20, 2019.

[9] JVET, *AHG report: Test model software development (AHG3)*. Joint Video Experts Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-B1010, 20th Meeting, teleconference, OCTOBER, JVET-T0003-v1 (2020).

[10] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G. J. Sullivan, and J.-R. Ohm, "Overview of the versatile video coding (vvc) standard and its applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3736–3764, 2021.

[11] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE signal processing magazine*, vol. 15, no. 6, pp. 74–90, 1998.

[12] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, pp. 381–386, 2020.

[13] *Video coding standards: AVS China, H.264/MPEG-4 PART 10, HEVC, VP6, DIRAC and VC-1.* Signals and Communication Technology, Springer Dordrecht, 2014.

[14] I. Pitas, *Digital image processing algorithms and applications.* John Wiley & Sons, 2000.

[15] R. Westwater and B. Furht, *Real-time video compression: techniques and algorithms*, vol. 376. Springer, 2007.

[16] C. Poynton, "Chroma subsampling notation," *Retrieved June*, vol. 19, p. 2004, 2002.

[17] T. Sikora, "Trends and perspectives in image and video coding," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 6–17, 2005.

[18] *High Efficiency Video Coding (HEVC): Algorithms and Architectures.* Integrated Circuits and Systems, Springer International Publishing Switzerland, 2014.

[19] J.-W. Chen, C.-Y. Kao, and Y.-L. Lin, "Introduction to h. 264 advanced video coding," in *Proceedings of the 2006 Asia and South Pacific Design Automation Conference*, pp. 736–741, 2006.

[20] JVET, *Algorithm description for Versatile Video Coding and Test Model 8.* ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 17th Meeting: Brussels, BE, 17–17 January 2020.

[21] Q. Huynh-Thu and M. Ghanbari, "The accuracy of psnr in predicting video quality for different video scenes and frame rates," *Telecommunication Systems*, vol. 49, no. 1, pp. 35–48, 2012.

[22] G. Bjontegaard, "Document vceg-m33: Calculation of average psnr differences between rd-curves," in *ITU-T VCEG Meeting, Austin, Texas, Technical Report*, 2001.

[23] T. K. Tan, R. Weerakkody, M. Mrak, N. Ramzan, V. Baroncini, J.-R. Ohm, and G. J. Sullivan, "Video quality evaluation methodology and verification testing of hevc compression performance," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 76–90, 2015.

[24] C. S. G. XV, *Video Codec for Audiovisual Services at p x 64 kbids*, 1990.

[25] ISO and MPEG, *Coding for Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s*, 1992.

[26] ISO and MPEG, *Video Coding for Low Bit-rate Communications*, 1995.

[27] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h. 264/avc video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.

[28] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[29] M. M. Hannuksela, Y. Yan, X. Huang, and H. Li, "Overview of the multiview high efficiency video coding (mv-hevc) standard," in *2015 IEEE International Conference on Image Processing (ICIP)*, pp. 2154–2158, IEEE, 2015.

[30] JCT-VC, *High Efficiency Video Coding (HEVC) Test Model 9 Encoder Description*. Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 11th Meeting: Shanghai, CN, 10–19 October 2012.

[31] X. e. a. Li, *Multi-Type-Tree,document JVET-D0117*. 4th JVET meeting,Chengdu, CN, October 2016.

[32] D. Patel, T. Lad, and D. Shah, "Review on intra-prediction in high efficiency video coding (hevc) standard," *International Journal of Computer Applications (0975 8887)*, vol. 12, 2015.

[33] J. Li, B. Li, J. Xu, and R. Xiong, "Intra prediction using multiple reference lines for video coding," in *2017 Data Compression Conference (DCC)*, pp. 221–230, IEEE, 2017.

[34] S. De-Luxán-Hernández, V. George, J. Ma, T. Nguyen, H. Schwarz, D. Marpe, and T. Wiegand, "An intra subpartition coding mode for vvc," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 1203–1207, IEEE, 2019.

[35] *High Efficiency Video Coding, Signals and Communication Technology,*, livre Video Coding Fundamentals. Springer, 2015.

[36] W.-Y. Wei, "Digital video compression fundamentals and standards," *National Taiwan University, Taipei, Taiwan, ROC*, 2009.

[37] W.-J. Chien, Y. Chen, J. Chen, L. Zhang, M. Karczewicz, and X. Li, "Sub-block motion derivation for merge mode in hevc," in *Applications of Digital Image Processing XXXIX*, vol. 9971, p. 99711K, International Society for Optics and Photonics, 2016.

[38] L. Zhang, K. Zhang, H. Liu, H. C. Chuang, Y. Wang, J. Xu, P. Zhao, and D. Hong, "History-based motion vector prediction in versatile video coding," in *2019 Data Compression Conference (DCC)*, pp. 43–52, IEEE, 2019.

[39] K. Zhang, Y.-W. Chen, L. Zhang, W.-J. Chien, and M. Karczewicz, "An improved framework of affine motion compensation in video coding," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1456–1469, 2018.

[40] Y. Zhang, C.-C. Han, Y; Chen, C.-H. Hung, W.-J. Chien, and M. Karczewicz, *Locally adaptive motion vector resolution and MVD coding, document JVET-K0357*. 11th JVETmeeting, Ljubljana, SI, July 2018.

[41] A. A. Batalla, *Multiple transforms for video coding*. PhD thesis, INSA, 2015.

[42] M. Koo, M. Salehifar, J. Lim, and S.-H. Kim, "Low frequency non-separable transform (lfnst)," in *2019 Picture Coding Symposium (PCS)*, pp. 1–5, IEEE, 2019.

[43] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the h. 264/avc video compression standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 620–636, 2003.

[44] A. D. L. R. GÓMEZAREVALILLO, "Investigating the adaptive loop filter in next generation video coding," Master's thesis, KTH ROYAL INSTITUTE OF TECHNOLOGY, 2017.

[45] K. Sayood, "Chapter 4 - arithmetic coding," in *Introduction to Data Compression (Fifth Edition)* (K. Sayood, ed.), The Morgan Kaufmann Series in Multimedia Information and Systems, pp. 89–130, Morgan Kaufmann, fifth edition ed., 2018.

[46] M. Karczewicz, N. Hu, J. Taquet, C.-Y. Chen, K. Misra, K. Andersson, P. Yin, T. Lu, E. François, and J. Chen, "Vvc in-loop filters," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3907–3925, 2021.

[47] T. W. Detlev Marpe, Heiko Schwarz, "Generic quadtree-based block partitioning in hevc." `https://www.hhi.fraunhofer.de/en/departments/vca/research-groups/image-video-coding/research-topics/generic-quadtree-based-block-partitioning-in-hevc/`. visited on 29-10-2018.

[48] J. Lainema, F. Bossen, W.-J. Han, J. Min, and K. Ugur, "Intra coding of the hevc standard," *IEEE transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1792–1801, 2012.

[49] M. Schäfer, B. Stallenberger, J. Pfaff, P. Helle, H. Schwarz, D. Marpe, and T. Wiegand, "An affine-linear intra prediction with complexity constraints," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 1089–1093, IEEE, 2019.

[50] L. Zhao, X. Zhao, S. Liu, X. Li, J. Lainema, G. Rath, F. Urban, and F. Racapé, "Wide angular intra prediction for versatile video coding," in *2019 Data Compression Conference (DCC)*, pp. 53–62, IEEE, 2019.

[51] A. Said, X. Zhao, M. Karczewicz, J. Chen, and F. Zou, "Position dependent prediction combination for intra-frame video coding," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 534–538, IEEE, 2016.

[52] JVET, "Vvcsoftware_vtm 8.0 reference software," 2018. [online] available at `https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/`.

[53] JVET, *JVET common test conditions and software reference configurations for SDR videos*. Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-B1010, 14th Meeting: Geneva, CH, 19 to 27 Mar. 2019.

[54] M. Saldanha, G. Sanchez, C. Marcon, and L. Agostini, "Complexity analysis of vvc intra coding," in *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 3119–3123, IEEE, 2020.

[55] A. Tissier, A. Mercat, T. Amestoy, W. Hamidouche, J. Vanne, and D. Menard, "Complexity reduction opportunities in the future vvc intra encoder," in *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, pp. 1–6, IEEE, 2019.

[56] N. Zouidi, A. Kessentini, F. Belghith, and N. Masmoudi, "Statistical analysis of the qtmt structure: intra mode decision," in *2020 IEEE 4th International Conference on Image Processing, Applications and Systems (IPAS)*, pp. 180–185, IEEE, 2020.

[57] K. REUZE, P. PHILIPPE, O. DEFORGES, and W. HAMIDOUCHE, "Enhanced intra prediction modes signalling in hevc," in *2016 Picture Coding Symposium, PCS 2016*, (Nuremberg, Germany), Dec. 2016.

[58] O. Spjuth, J. Frid, and A. Hellander, "The machine learning life cycle and the cloud: implications for drug discovery," *Expert opinion on drug discovery*, vol. 16, no. 9, pp. 1071–1079, 2021.

[59] M. Gaertler, V. Khaydarov, B. Klöpper, and L. Urbas, "The machine learning life cycle in chemical operations–status and open challenges," *Chemie Ingenieur Technik*, vol. 93, no. 12, pp. 2063–2080, 2021.

[60] M. Stonebraker, I. F. Ilyas, *et al.*, "Data integration: The current status and the way forward.," *IEEE Data Eng. Bull.*, vol. 41, no. 2, pp. 3–9, 2018.

[61] H. Suresh and J. Guttag, "A framework for understanding sources of harm throughout the machine learning life cycle," in *Equity and access in algorithms, mechanisms, and optimization*, pp. 1–9, 2021.

[62] E. Raj, *Engineering MLOps: Rapidly build, test, and manage production-ready machine learning life cycles at scale*. Packt Publishing Ltd, 2021.

[63] W. S. Noble, "What is a support vector machine?," *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.

[64] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[65] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[66] J. E. Van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine learning*, vol. 109, no. 2, pp. 373–440, 2020.

[67] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[68] A. A. Soofi and A. Awan, "Classification techniques in machine learning: applications and issues," *J. Basic Appl. Sci*, vol. 13, pp. 459–465, 2017.

[69] S. Ray, "A quick review of machine learning algorithms," in *2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon)*, pp. 35–39, IEEE, 2019.

[70] P. Ali and A. Younas, "Understanding and interpreting regression analysis," *Evidence-Based Nursing*, vol. 24, no. 4, pp. 116–118, 2021.

[71] C. Kingsford and S. L. Salzberg, "What are decision trees?," *Nature biotechnology*, vol. 26, no. 9, pp. 1011–1013, 2008.

[72] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE signal processing magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[73] P. P. Shinde and S. Shah, "A review of machine learning and deep learning applications," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp. 1–6, 2018.

[74] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[75] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[76] L. R. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, pp. 64–67, 2001.

[77] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv preprint arXiv:2003.05991*, 2020.

[78] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[79] A. Aggarwal, M. Mittal, and G. Battineni, "Generative adversarial network: An overview of theory and applications," *International Journal of Information Management Data Insights*, vol. 1, no. 1, p. 100004, 2021.

[80] D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin, "Backpropagation: The basic theory," *Backpropagation: Theory, architectures and applications*, pp. 1–34, 1995.

[81] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[82] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[83] N. Gupta *et al.*, "Artificial neural network," *Network and Complex Systems*, vol. 3, no. 1, pp. 24–28, 2013.

[84] M. Crawshaw, "Multi-task learning with deep neural networks: A survey," *arXiv preprint arXiv:2009.09796*, 2020.

[85] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[86] A. Argyriou, T. Evgeniou, and M. Pontil, "Multi-task feature learning," *Advances in neural information processing systems*, vol. 19, 2006.

[87] A. Argyriou, T. Evgeniou, and M. Pontil, "Convex multi-task feature learning," *Machine learning*, vol. 73, no. 3, pp. 243–272, 2008.

[88] A. Maurer, M. Pontil, and B. Romera-Paredes, "Sparse coding for multitask and transfer learning," in *International conference on machine learning*, pp. 343–351, PMLR, 2013.

[89] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[90] G. Obozinski, B. Taskar, and M. Jordan, "Multi-task feature selection," *Statistics Department, UC Berkeley, Tech. Rep*, vol. 2, no. 2.2, p. 2, 2006.

[91] J. Chen, L. Tang, J. Liu, and J. Ye, "A convex formulation for learning shared structures from multiple tasks," in *Proceedings of the 26th annual international conference on machine learning*, pp. 137–144, 2009.

[92] Y. Yang and T. M. Hospedales, "Trace norm regularised deep multi-task learning," *arXiv preprint arXiv:1606.04038*, 2016.

[93] Z.-H. Zhou, *Ensemble Learning*, pp. 270–273. Boston, MA: Springer US, 2009.

[94] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, pp. 123–140, 1996.

[95] R. E. Schapire, "The strength of weak learnability," *Machine learning*, vol. 5, pp. 197–227, 1990.

[96] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[97] M. Zounemat-Kermani, O. Batelaan, M. Fadaee, and R. Hinkelmann, "Ensemble machine learning paradigms in hydrology: A review," *Journal of Hydrology*, vol. 598, p. 126266, 2021.

[98] R. J. Tibshirani and B. Efron, "An introduction to the bootstrap," *Monographs on statistics and applied probability*, vol. 57, no. 1, 1993.

[99] Z.-H. Zhou and Z.-H. Zhou, *Ensemble learning*. Springer, 2021.

[100] Y. Chen, L. Yu, H. Wang, T. Li, and S. Wang, "A novel fast intra mode decision for versatile video coding," *Journal of Visual Communication and Image Representation*, vol. 71, p. 102849, 2020.

[101] J. Park, B. Kim, and B. Jeon, "Fast vvc intra prediction mode decision based on block shapes," in *Applications of Digital Image Processing XLIII*, vol. 11510, pp. 581–593, SPIE, 2020.

[102] Y. Li, G. Yang, Y. Song, H. Zhang, X. Ding, and D. Zhang, "Early intra cu size decision for versatile video coding based on a tunable decision model," *IEEE Transactions on Broadcasting*, vol. 67, no. 3, pp. 710–720, 2021.

[103] J. Cao, N. Tang, J. Wang, and F. Liang, "Texture-based fast cu size decision and intra mode decision algorithm for vvc," in *International Conference on Multimedia Modeling*, pp. 739–751, Springer, 2020.

[104] Z. Liu, M. Dong, X. H. Guan, M. Zhang, and R. Wang, "Fast isp coding mode optimization algorithm based on cu texture complexity for vvc," *EURASIP Journal on Image and Video Processing*, vol. 2021, no. 1, pp. 1–14, 2021.

[105] S. Ryu and J. Kang, "Machine learning-based fast angular prediction mode decision technique in video coding," *IEEE Transactions on Image Processing*, vol. 27, no. 11, pp. 5525–5538, 2018.

[106] Q. Zhang, Y. Wang, L. Huang, and B. Jiang, "Fast cu partition and intra mode decision method for h. 266/vvc," *IEEE Access*, vol. 8, pp. 117539–117550, 2020.

[107] H. Yang, L. Shen, X. Dong, Q. Ding, P. An, and G. Jiang, "Low complexity ctu partition structure decision and fast intra mode decision for versatile video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[108] JVET, "Vvcsoftware_vtm 10.2 reference software," 2020. [online] available at `https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/`.

[109] A. Wieckowski, J. Ma, H. Schwarz, D. Marpe, and T. Wiegand, "Fast partitioning decision strategies for the upcoming versatile video coding (vvc) standard," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 4130–4134, 2019.

[110] JVET, *CE3: Intra Sub-Partitions Coding Mode*. Joint Video Experts Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-B1010.

[111] JVET, *CE3: Affine linear weighted intra prediction*. Joint Video Experts Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-B1010.

[112] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[113] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, "Facial landmark detection by deep multi-task learning," in *European conference on computer vision*, pp. 94–108, Springer, 2014.

[114] JVET, *JVET common test conditions and software reference configurations for SDR videos*. Joint Video Experts Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-B1010, 14th Meeting: Geneva, CH, 19–27 Mar. 2019.

[115] T. Hermann, "Frugally deep," 2018. [online] available at `https://github.com/Dobiasd/frugally-deep/`.

[116] G. S. Handelman, H. K. Kok, R. V. Chandra, A. H. Razavi, S. Huang, M. Brooks, M. J. Lee, and H. Asadi, "Peering into the black box of artificial intelligence: evaluation metrics of machine learning methods," *American Journal of Roentgenology*, vol. 212, no. 1, pp. 38–43, 2019.

[117] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," 2010.

[118] N. Japkowicz, "Why question machine learning evaluation methods," in *AAAI workshop on evaluation methods for machine learning*, pp. 6–11, Citeseer, 2006.

[119] P. Flach and M. Kull, "Precision-recall-gain curves: Pr analysis done right," *Advances in neural information processing systems*, vol. 28, 2015.

[120] Y. Nan, K. M. Chai, W. S. Lee, and H. L. Chieu, "Optimizing f-measure: A tale of two approaches," *arXiv preprint arXiv:1206.4625*, 2012.

[121] M. Saldanha, G. Sanchez, C. Marcon, and L. Agostini, "Performance analysis of vvc intra coding," *Journal of Visual Communication and Image Representation*, vol. 79, p. 103202, 2021.

[122] R. Mohammed, J. Rawashdeh, and M. Abdullah, "Machine learning with oversampling and undersampling techniques: overview study and experimental results," in *2020 11th international conference on information and communication systems (ICICS)*, pp. 243–248, IEEE, 2020.

[123] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[124] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: a big comparison for nas," *arXiv preprint arXiv:1912.06059*, 2019.

[125] Microsoft Corporation, *LightGBM Release 3.3.5.99*, 2023.

[126] M. Saldanha, G. Sanchez, C. Marcon, and L. Agostini, "Complexity analysis of vvc intra coding," in *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 3119–3123, 2020.

[127] X. Dong, L. Shen, M. Yu, and H. Yang, "Fast intra mode decision algorithm for versatile video coding," *IEEE Transactions on Multimedia*, vol. 24, pp. 400–414, 2021.

Appendix A

Personnel Publications

A.1 Scientific journals

[**J1**] **Zouidi, N.**, Kessentini, A., Masmoudi, N. and D. Menard, Complexity assessment of the intra mode decision of Versatile Video Coding.Multimedia Tools and Applications, 2023, p. 1-20.

[**J2**] **Zouidi, N.**, Kessentini, Masmoudi, N. and D. Menard, Multitask Learning Based Intra-Mode Decision Framework for Versatile Video Coding. Electronics, 2022, vol. 11, no 23, p. 4001.

A.2 International Conference

[**C1**] **Zouidi, N.**, Kessentini, A., Belghith, F., and Masmoudi, N. (2020, December). Statistical analysis of the QTMT structure: intra mode decision. In 2020 IEEE 4th International Conference on Image Processing, Applications and Systems (IPAS) (pp. 180-185). IEEE.

COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Titre : Réduction de complexité de l'encodage VVC à l'aide de techniques d'apprentissage automatique : prediction intra

Mot clés : Compression vidéo, VVC, décision de mode intra, optimisation RD, intelligence artificielle

Résumé : En juillet 2020, la nouvelle norme de codage vidéo Versatile Video Coding (VVC), a été publiée par le groupe Joint Video Experts Team (JVET). Cette norme permet un niveau plus élevé de polyvalence avec une meilleure performance en compression vidéo par rapport à son prédécesseur, High-Efficiency Video Coding (HEVC). En effet, elle introduit plusieurs nouveaux outils de codage tels que les modes de prédiction intra à granularité plus fine (IPMs) et la division Quad Tree with nested Multi Type Tree (QTMT). Étant donné que la recherche des meilleures décisions de codage est généralement précédée par l'optimisation du coût en distorsion et débit binaire, l'introduction de nouveaux outils de codage ou l'amélioration des outils existants nécessite des calculs supplémentaires. En fait, la norme VVC est 31 fois plus complexe que le HEVC. Par conséquent, cette thèse vise à réduire la complexité de calcul de la norme VVC et plus particulièrement au niveau des outils intra. Elle étudie en premiers lieu les opportunités de réduction de la complexité dans la décision du mode intra de la norme VVC. Puis, deux algorithmes rapides de décision de mode de prédiction intra basé sur des modéles d'apprentissage automatique telles que les réseaux de neurones convolutifs multi-tachés et l'arbre de decision Light-Gradient Boosting Machine (Light-GBM) ont été proposés.

Title: Complexity reduction of VVC encoder using machine learning techniques: intra prediction

Keywords: Video compression, VVC, intra mode decision, RD optimization, artificial intelligence

Abstract: In July 2020, the new video coding standard Versatile Video Coding (VVC), was released by the Joint Video Experts Team (JVET). This standard enables a higher level of versatility with a better compression performance compared to its predecessor, High-Efficiency Video Coding (HEVC). Indeed, It introduces several new coding tools such as finer-granularity Intra Prediction Modes (IPMs) and QuadTree with nested Multi-type Tree (QTMT). Because finding the best encoding decisions is usually preceded by optimizing the Rate-Distortion (RD) cost, introducing new coding tools or enhancing existing ones would require additional computations. In fact, the VVC is 31 times more complex than the HEVC. Therefore, the aim of this thesis is to reduce the computational complexity of the VVC encoding. First, it studies the upper bound of complexity reduction in the intra mode decision of the VVC. Then, two fast decision algorithms for the intra mode decision based on machine learning algorithms such Multi-Task Learning (MTL) and Light-Gradient Boosting Machine (Light-GBM) were proposed.