Lightweight Machine Learning-Based Complexity Optimization for Real-Time VVC Encoding

Ibrahim Taabane

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, Vision*

Par

Ibrahim TAABANE

« Lightweight Machine Learning-Based Complexity Optimization for Real-Time VVC Encoding »

Thèse présentée et soutenue à « l'USMBA de Fès », le « 21 décembre 2024 »
Unité de recherche : « Institut d'Electronique et des Technologies du numéRique (IETR) - UMR CNRS 6164 »
Thèse N° : « 24ISAR 32 / D24 - 32 »

Rapporteurs avant soutenance :

François-Xavier COUDOUX	Professeur à l'Université Polytechnique Hauts-de-France, France
Mohamed Ali BEN AYED	Professeur à l'ENET'Com Sfax, Tunisie
El Habib NFAOUI	Professeur à la Faculté des Sciences Dhar El Mehraz de Fès, Maroc

Composition du Jury :

Président :	Sanaa EL FKIHI	Professeur à l'ENSIAS, Université Mohammed V de Rabat, Maroc
Examinateurs :	François-Xavier COUDOUX	Professeur à l'Université Polytechnique Hauts-de-France, France
	Mohamed Ali BEN AYED	Professeur à l'ENET'Com Sfax, Tunisie
	El Habib NFAOUI	Professeur à la Faculté des Sciences Dhar El Mehraz de Fès, Maroc
	Maher JRIDI	Professeur à l'ISEN Yncrea Ouest, France
Dir. de thèse :	Daniel MENARD	Professeur à l'INSA de Rennes, France
Co-dir. de thèse :	Ali AHAITOUF	Professeur à la FST de Fès, USMBA de Fès, Maroc
Co-encadrant de thèse :	Anass MANSOURI	Professeur à l'ENSA de Fès, USMBA de Fès, Maroc

Acknowledgement

First and foremost, I would like to thank the jury members for honoring me by reviewing this work.

I am very grateful to my thesis supervisors, Mr. Daniel Menard, Mr. Ali Ahaitouf, and Mr. Anass Mansouri, for their guidance and expertise. Their feedback and encouragement were essential throughout this journey.

I am deeply thankful to my friends and family for their unwavering love and support. To my dear parents, your sacrifices and unconditional love have been the foundation of all my efforts. I am truly grateful for everything you have done.

A special thanks to my colleagues and fellow researchers for the engaging discussions, shared experiences, and friendships. You made this journey more enjoyable.

Lastly, I thank everyone who contributed to this journey. This experience has been both challenging and rewarding, and I am grateful to all who helped make it a success.

Table of Contents

List of Figures

List of Tables

Acronyms

AI All Intra. ix, x, xi, xii, 3, 4, 6, 7, 24, 25, 27, 31, 32, 35, 36, 40, 44, 45, 46, 47, 48, 49, 50, 52, 53, 54, 55, 62, 63, 64, 65, 66, 70, 75, 80, 81, 88, 94, 106, 117, 123, 124, 126, 139, 140, 141, 142

ALF Adaptive Loop Filter. 14, 22, 23

AOM Alliance for Open Media. 13, 14

AR Augmented Reality. 119

ARM Advanced RISC Machines. x, 4, 6, 40, 56, 58, 59, 62, 117, 124, 126

AUC Area Under the Curve. 34

AV1 AOMedia Video 1. 13, 14

AVC Advanced Video Coding. 7, 12, 13, 14, 32, 34

B-frame Bi-predictive-Frame. 25, 26

BDBR Bjøntegaard Delta Bit Rate. x, xi, xii, xiii, 14, 29, 35, 36, 45, 46, 47, 54, 55, 64, 65, 66, 67, 68, 69, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 92, 93, 94, 95, 96, 97, 104, 107, 108, 109, 110, 111, 112, 114, 115, 118, 145, 146, 147

BT Binary-Tree. xii, 17, 18, 49, 52, 53, 54, 55, 91, 94, 102, 108, 139

BTH Binary Tree Horizontal. 17, 49, 70, 72, 75, 77, 78, 80, 102, 103, 105, 108

BTV Binary Tree Vertical. 17, 49, 70, 72, 75, 77, 78, 80, 102, 103, 105, 108

CABAC Context-Adaptive Binary Arithmetic Coding. 23

CCLM Cross Component Linear Model. 19

CNN Convolutional Neural Network. 35, 36, 65, 66, 68, 69, 95, 96, 114

CPU Central Processing Unit. 41, 56, 77

CTB Coding Tree Block. 17

CTC Common Test Conditions. xi, xii, 30, 31, 32, 44, 54, 66, 71, 81, 84, 90, 106, 109, 110, 111, 115, 139, 140

CTU Coding Tree Unit. ix, xiii, 16, 17, 18, 48, 49, 52, 65, 66, 68, 87, 90, 93, 94, 96, 104, 106, 142

CU Coding Unit. ix, x, xi, xii, 4, 5, 17, 18, 19, 20, 21, 22, 35, 37, 40, 48, 49, 50, 51, 52, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 81, 82, 83, 84, 86, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 116, 118, 124, 125

DBF Deblocking Filter. 22

DCT Discrete Cosine Transform. 20, 21, 36

DL Deep Learning. 5, 33, 34, 35, 38, 39, 63, 64, 65, 68, 69, 91, 96, 97, 118, 119

DQ Dependent Quantization. 22

DST Discrete Sine Transform. 20, 21, 36

DT Decision Tree. 36, 39, 68, 76, 95, 96

ETR Encoding Time Reduction. x, xi, xii, xiii, 30, 45, 46, 47, 54, 55, 69, 79, 80, 81, 82, 83, 84, 85, 87, 88, 94, 95, 96, 97, 107, 108, 109, 110, 111, 112, 114, 115, 145, 146, 147

FCN Fully Connected Network. 35

FN False Negative. 33, 34

FP False Positive. 33, 34

FPGA Field-Programmable Gate Array. 36, 118

FPS Frames per Second. x, xii, 10, 11, 56, 57, 58, 59, 60, 61, 62

GBDT Gradient Boosted Decision Trees. 77

GCN Global Convolutional Network. 36, 68

GOP Group of Pictures. ix, 27, 107

GPU Graphics Processing Unit. 56, 77, 118

HD High Definition. ix, x, 10, 11, 50, 51, 52, 53, 110, 139, 141, 142

HDR High Dynamic Range. 11, 31

HEVC High Efficiency Video Coding. xii, 1, 2, 4, 5, 7, 12, 13, 14, 18, 19, 22, 23, 24, 32, 35, 37, 40, 43, 64, 65, 90, 91, 92, 93, 122, 124, 126, 127, 140, 152

HM HEVC test Model. ix, 13, 36, 43, 65, 66, 92, 93

I-frame Intra-Frame. 24, 25, 26, 31

IEC International Electrotechnical Commission. 12, 13

ISO International Organization for Standardization. 12

ISP Intra Sub-Partitions. xi, 19, 35, 143, 144, 145, 146, 147, 148

ITU-T International Telecommunication Union Telecommunication Standardization Sector. 13

JCT-VC Joint Collaborative Team on Video Coding. 13

JEM Joint Exploration Model. 95, 97

JVET Joint Video Experts Team. 2, 13, 31, 41, 107, 122

JVT Joint Video Team. 13

LD Low Delay. ix, 7, 24, 25, 26, 31, 32, 36, 66, 94

LFNST Low-Frequency Non-Separable Transform. 21

LightGBM Light Gradient Boosting Machine. x, xii, xiii, 4, 5, 6, 35, 39, 63, 67, 69, 70, 71, 73, 75, 76, 77, 78, 89, 90, 98, 102, 106, 107, 116, 117, 118, 124, 125, 126, 152

LMCS Luma Mapping with Chroma Scaling. 22

LSTM Long-Short Term Memory. 66

MIP Matrix-Based Intra Prediction. 19, 143, 144, 145, 146, 147, 148

ML Machine Learning. x, 4, 5, 6, 29, 33, 34, 35, 36, 38, 39, 63, 64, 65, 67, 68, 69, 70, 72, 73, 76, 77, 79, 86, 90, 91, 93, 94, 96, 97, 98, 108, 119, 126, 152

MOS Mean Opinion Score. 29, 30

MPEG Moving Picture Experts Group. 12, 13

MRL Multiple Reference Line. xi, 19, 143, 144, 145, 146, 147, 148

MSE Mean-Squared Error. 23, 28

MTS Multiple Transform Selection. 20, 21, 36, 66

MTT Multi-Type Tree. 17, 48, 49, 52, 53, 72, 77, 78, 80, 86, 87, 91, 94, 100, 112, 119, 139, 142

MV Motion Vector. xi, 20, 27, 92, 93, 100, 101, 102

MVD Motion Vector Differences. 20

MVP Motion Vector Prediction. 19, 20

NS No Split. 17, 70, 72, 77, 78, 80, 105, 106

P-frame Predictive-Frame. 25, 26

PDPC Position Dependent Prediction Combination. 19

POC Picture Order Count. 95

PSNR Peak Signal-to-Noise Ratio. 14, 23, 28, 29, 82, 83, 140, 141, 142

QP Quantization Parameter. x, 22, 30, 48, 50, 51, 52, 53, 54, 71, 74, 81, 95, 99, 107, 139

QT Quadtree. xii, 14, 17, 18, 48, 49, 52, 53, 54, 55, 70, 72, 75, 77, 78, 80, 87, 91, 93, 96, 100, 102, 103, 105, 108, 119, 139, 140

QTBT Quadtree plus Binary-Tree. 94

QTMTT Quadtree with Multi-Type Tree. ix, 4, 6, 14, 17, 18, 37, 40, 48, 52, 63, 68, 69, 70, 87, 89, 90, 95, 99, 100, 101, 117, 118, 124, 125, 126, 127, 152

RA Random Access. ix, x, xi, xii, 3, 4, 6, 7, 24, 26, 27, 31, 32, 36, 40, 41, 44, 45, 47, 48, 49, 50, 52, 53, 54, 55, 62, 89, 90, 91, 92, 93, 94, 95, 96, 99, 103, 105, 106, 107, 116, 118, 123, 125, 127, 139, 140, 141

RAM Random Access Memory. 56, 77

RD Rate-Distortion. xi, 64, 69, 71, 73, 80, 92, 93, 96, 104, 112, 113, 140, 141, 142

RDO Rate-Distortion Optimization. 4, 5, 7, 22, 23, 24, 34, 35, 36, 37, 38, 44, 63, 65, 66, 67, 68, 69, 71, 72, 73, 78, 89, 92, 93, 116, 118, 124, 126

RF Random Forest. 35, 36, 39, 67, 69, 76, 95

RGB Red, Green, Blue. 8

RISC Reduced Instruction Set Computer. xiv

ROC Receiver Operating Characteristic. 34

SAO Sample Adaptive Offset. 14, 22

SBT Sub-Block Transform. ix, 21

SD Standard Definition. ix, 1, 10, 121

SDR Standard Dynamic Range. 31

SIMD Single Instruction Multiple Data. 41, 44

SSIM Structural SIMilarity. 14, 28, 29

SVM Support Vector Machine. 37, 39, 65, 67, 68, 69, 93

TN True Negative. 33

TP True Positive. 33, 34

TT Ternary-Tree. xii, 17, 18, 49, 52, 53, 54, 55, 91, 94, 102, 108, 139

TTH Ternary Tree Horizontal. 17, 49, 70, 72, 75, 77, 78, 80, 102, 103, 105, 106, 108

TTV Ternary Tree Vertical. 17, 18, 49, 70, 72, 75, 77, 78, 80, 102, 103, 105, 106, 108

UHD Ultra High Definition. 14, 121, 139, 141, 142

VCEG Video Coding Experts Group. 12, 13

VMAF Video Multimethod Assessment Fusion. 14, 29

VR Virtual Reality. 14, 119

VTM VVC Test Model. ix, 2, 13, 31, 36, 40, 41, 43, 44, 54, 66, 67, 68, 69, 87, 94, 95, 96, 97, 114, 117, 122, 123, 143

VVC Versatile Video Coding. ix, x, xii, xiii, 1, 2, 3, 4, 5, 6, 7, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 32, 35, 36, 37, 38, 39, 40, 41, 43, 44, 48, 49, 52, 62, 63, 64, 66, 67, 68, 69, 70, 89, 90, 91, 94, 95, 96, 97, 99, 110, 116, 117, 122, 123, 124, 125, 126, 127, 143, 147, 148, 152

VVenC Versatile Video Encoder. ix, x, xi, xii, xiii, 2, 3, 4, 5, 6, 13, 37, 40, 41, 42, 43, 44, 45, 46, 47, 48, 50, 51, 52, 54, 55, 56, 58, 59, 60, 61, 62, 63, 69, 70, 71, 76, 77, 80, 81, 82, 83, 84, 85, 86, 87, 89, 90, 91, 94, 95, 96, 97, 98, 105, 107, 109, 110, 111, 112, 114, 115, 116, 117, 118, 119, 123, 124, 125, 126, 127, 139, 140, 141, 142, 143, 145, 147, 148, 152

WAIP Wide-Angle Intra Prediction. 19

Introduction

General Context

In recent years, there has been a significant change in the way people consume media, mainly due to the rise of video-on-demand services like Netflix and YouTube, live-streaming platforms such as Twitch and Kick, and the widespread impact of social media platforms like Facebook, Instagram, and X (previously Twitter). Consequently, video content now dominates global internet traffic. In 2022, Cisco reported that video traffic constituted over 82% of global internet traffic, and this figure has continued to rise with the increasing adoption of high-resolution formats like 4K and 8K [1]. According to a 2023 report by Sandvine, video traffic saw a 24% increase in 2022, accounting for 65% of all internet traffic by the start of 2023 [2]. This trend is mirrored in mobile data usage, with the Ericsson Mobility Report estimating that video will represent 71% of all mobile data traffic by the end of 2023, a proportion expected to reach 80% by 2028 [3]. The global reach of online video is further highlighted by a Statista report, which found that by the end of 2023, 92% of internet users worldwide engaged with online video content [4], and the number of live video viewers was projected to reach 164.6 million [5].

This surge in video consumption has been facilitated by significant advancements in network infrastructure, including the widespread deployment of 4G, 5G, and fiber-optic technologies, which have substantially increased available bandwidth. However, the massive volume of video data, combined with the rising demand for higher-quality and immersive content, poses substantial challenges related to bandwidth, storage, and computational resources. For example, an 8K video frame comprises 33 million pixels, necessitating far greater bandwidth than Standard Definition (SD) content.

To address these challenges, the development of efficient video compression techniques has become essential. Video codecs, responsible for compressing and decompressing video files, are critical in balancing video quality with file size. Over the years, video coding standards have evolved significantly, from the early H.261 to the more advanced High Efficiency Video Coding (HEVC) [6] and the most recent Versatile Video Coding (VVC) [7]. Finalized in July 2020, VVC offers approximately 40% bitrate savings over HEVC while

maintaining comparable video quality, making it particularly well-suited for the transmission of high-resolution content.

However, the enhanced compression efficiency of VVC comes with a significant increase in computational complexity. The encoding process for VVC is estimated to be 6.6 to 25.8 times more time-consuming than that of HEVC, depending on the specific encoding configurations employed [7]. This increased complexity presents a major barrier to the widespread adoption of VVC, especially in real-time video streaming scenarios where low latency is crucial.

Motivations and Objectives

As video continues to play a pivotal role in both consumer and professional settings, the need for efficient video compression techniques has become increasingly critical. In particular, the VVC standard, while offering significant improvements in compression efficiency, introduces substantial computational complexity that poses challenges for real-time applications. Addressing this complexity without compromising on compression efficiency is essential for enabling seamless video experiences, especially in scenarios requiring low latency.

This thesis is dedicated to optimizing one of the most computationally demanding aspects of VVC: the partitioning process. According [8], this process shows a high potential for complexity reduction, accounting for 97% of the overall encoding complexity. The primary objective is to accelerate the VVC encoding process by developing and implementing fast partitioning algorithms based on machine learning models. These algorithms aim to reduce encoding time while preserving the high coding efficiency that VVC is known for. The ultimate goal is to make real-time encoding feasible, minimizing the trade-offs between speed and video quality.

Following the standardization of VVC, the Joint Video Experts Team (JVET) released the VVC Test Model (VTM) software as a reference implementation. While VTM is highly effective for benchmarking and evaluating codec performance, it suffers from significantly longer encoding times due to its unoptimized nature. In contrast, this research utilizes the Versatile Video Encoder (VVenC) [9], developed by the Fraunhofer Heinrich Hertz Institute (HHI). VVenC is an optimized implementation of the VVC standard, built upon VTM, and offers comparable performance while significantly reducing encoding time [9]. Its optimizations, designed for real-time and practical applications, make VVenC an ideal

choice for our research, providing both efficiency and performance.

Focusing on All Intra (AI) and Random Access (RA) configurations of the VVenC encoder, this thesis proposes novel partitioning strategies designed to enhance encoding efficiency. These optimizations have been integrated into the VVenC encoder, and extensive testing has demonstrated that the proposed methods outperform existing approaches. The results pave the way for real-time VVenC encoding, offering a practical solution to the challenges posed by the increasing demands of modern video consumption.

In addressing the critical balance between video quality, encoding speed, and computational complexity, this thesis contributes to the ongoing advancement of video compression technologies, ensuring they meet the evolving needs of current and future media landscapes.

Contributions

This thesis presents three primary contributions, each focusing on significant optimizations of the VVenC encoder. The statistical analysis serves as a foundational study, examining the frequency and complexity of different coding modes and partitioning block sizes in both intra and inter-coding within the VVenC encoder. This analysis identifies key areas where optimization can significantly reduce computational complexity. Given that the VVenC encoder is a relatively new video codec, there are limited studies on its complexity reduction, particularly concerning block partitioning optimizations. Addressing this gap, our research focuses on accelerating the encoding process in VVenC through targeted optimizations in both AI and RA configurations. Our contributions have been integrated into the VVenC, and experimental results demonstrate that our methods achieve superior performance compared to existing approaches, paving the way for real-time VVenC encoding. The three principal contributions are detailed below:

I. Statistical Analysis and Performance Evaluation of the VVenC Encoder The first contribution of this thesis is a comprehensive statistical analysis and performance evaluation of the VVenC encoder, an open-source and optimized implementation of the VVC standard [9]. This analysis begins with an introduction to the VVenC encoder, highlighting its unique features, innovations, and the configuration of its presets. The study employs a robust methodology, utilizing various test environments and datasets to ensure a thorough examination of the encoder's performance. Key performance metrics

are defined and used to compare the VVenC, offering insights into its efficiency and effectiveness across different encoding scenarios, resolutions, and bitrates. This chapter also examines the VVenC encoder's performance on hardware architectures, including ARM and x86, to assess how closely VVenC approaches real-time capabilities and to identify the gains achieved by increasing the number of threads. The results of this analysis provide a detailed understanding of the operational characteristics of VVenC and identify specific areas where additional optimizations could lead to significant improvements in encoding speed and quality. This foundational work sets the stage for the detailed optimization efforts in the following chapters, ensuring that the VVenC encoder can meet the demands of modern video applications.

II. Machine Learning Based Fast QTMTT Partitioning Strategy for VVenC Encoder in Intra Coding The second key contribution of this thesis is the development of an innovative Machine Learning (ML)-based strategy to optimize the Quadtree with Multi-Type Tree (QTMTT) partitioning process within the VVenC encoder, specifically for intra-coding under the AI configuration. As the VVC standard brings significant improvements in compression efficiency compared to its predecessor, HEVC, it also introduces substantial computational complexity, particularly due to its advanced partitioning block structure. To mitigate this complexity, we have proposed a fast Coding Unit (CU) partitioning algorithm that utilizes Light Gradient Boosting Machine (LightGBM) classifiers [10]. These classifiers are trained to predict the most probable partitioning mode, thereby reducing the need for the exhaustive Rate-Distortion Optimization (RDO) process typically required. By embedding these models into the VVenC encoder, our approach effectively reduces encoding time while maintaining a balance between time-saving and coding efficiency. The results of this work demonstrate significant time reductions in encoding processes with only a slight increase in bitrate. This contribution enhances the performance of the VVenC encoder and represents a substantial step forward in making real-time video encoding more feasible for high-resolution and complex video content.

III. Low Complexity Learning-Based QTMTT Partitioning Scheme for Inter Coding in VVC Encoder The third major contribution of this thesis involves the development of a low-complexity QTMTT partitioning scheme for inter-coding in the VVenC encoder, specifically under the RA configuration. The VVC standard incorporates significant advancements in video compression efficiency, notably through its complex QTMTT

structure. While this structure greatly enhances compression performance, it also introduces considerable computational challenges, particularly for real-time applications. To address these challenges, we propose an optimized partitioning strategy that leverages inter-prediction techniques, incorporating both coding and motion information from inter-frames to accelerate the encoding process. This approach utilizes a series of lightweight LightGBM classifiers to predict the optimal split mode for each CU, effectively reducing the need for exhaustive computations. Implemented within the VVenC encoder, our method has demonstrated a significant reduction in runtime with only a minimal increase in bitrate. This contribution enhances the efficiency of the VVenC encoder and offers a practical solution for reducing computational complexity in high-demand video encoding scenarios, further advancing the capabilities of the VVC standard.

Outline of the Manuscript

This section outlines the organization of the thesis manuscript, which is structured into a general introduction, a general conclusion, and four main chapters. Each chapter delves into a specific aspect of video compression, with a particular focus on optimizing the VVC standard using ML-based approaches.

The manuscript starts by a general introduction that introduces the thesis by providing the general context, discussing the motivations, and outlining the main contributions of this research.

Chapter 1 provides a comprehensive overview of video coding. The chapter begins by introducing the fundamental characteristics of video, including aspects like color space, spatial and temporal resolutions, and bit depth. Following this, it traces the evolution of video coding standards, with a specific focus on the HEVC and VVC standards. This chapter also provides a detailed analysis of the VVC coding process, covering key stages such as partitioning schemes, intra and inter-prediction coding modes, RDO, and other relevant techniques. Furthermore, particular attention is given to the role of Machine Learning (ML) and Deep Learning (DL) in video compression. The potential advantages of these approaches are highlighted, while also examining the challenges related to computational demands and significant data requirements.

Chapter 2 focuses on the statistical analysis and performance evaluation of the VVenC encoder. This chapter introduces the VVenC, detailing its features and configurations. It also presents a thorough comparative analysis of VVenC's performance under

various encoding scenarios, resolutions, and bitrates, establishing a baseline for understanding where further optimizations can be made. Additionally, this chapter examines the performance of the VVenC encoder on hardware architectures including ARM and x86 to assess how closely VVenC approaches real-time capabilities and to identify the gains achieved by increasing the number of threads.

Chapter 3 presents a ML-based strategy for fast QTMTT partitioning within the VVenC, specifically targeting intra-coding configuration. Building upon the findings from the previous chapter, this one offers a comprehensive overview of the current state-of-the-art methods focused on fast partitioning for VVC under AI configuration. Thus, it introduces a novel approach utilizing Light Gradient Boosting Machine (LightGBM) classifiers to predict partitioning decisions. By integrating this method into the VVenC encoder, significant reductions in computational complexity are achieved, as demonstrated by experimental results that validate the effectiveness of this approach.

Chapter 4 extends the optimization efforts to the inter-coding configuration of the VVenC encoder. This chapter provides a comprehensive overview of the current state-of-the-art methods focused on fast partitioning for the VVC standard under the RA configuration. It introduces a low-complexity QTMTT partitioning scheme that leverages texture and temporal features to accelerate the encoding process within the RA configuration. The proposed method is thoroughly evaluated against existing state-of-the-art techniques, demonstrating substantial improvements in encoding speed while maintaining high coding efficiency.

In the conclusion, the manuscript summarizes the key contributions of this thesis and discusses potential future research directions to build upon and extend the impact of our work.

Chapter 1

General Background on Video Coding

1.1 Introduction

In the general introduction, the general context of this thesis was presented, along with an overview of the importance and applications of video coding. This chapter delves into the background and fundamental concepts of video coding. Section 1.2 discusses the video signal format, including spatial and temporal resolutions, bit depth, and color sampling. Following this, Section 1.3 introduces the main video coding standards, such as Advanced Video Coding (AVC), High Efficiency Video Coding (HEVC), and the latest standard, Versatile Video Coding (VVC). This thesis focuses on VVC and its primary stages in the encoding process, including partitioning, inter-coding, intra-coding, and the Rate-Distortion Optimization (RDO) process, as outlined in Section 1.4. Next, Section 1.5 describes the concepts of lossy and lossless compression, as well as coding mode configurations such as All Intra (AI), Low Delay (LD), and Random Access (RA) coding. Moreover, Section 1.6 presents the metrics used in video coding to evaluate the quality of the implemented video compression methods. Finally, Section 1.8 concludes this chapter by reinforcing the key concepts and setting the stage for the subsequent chapters. This foundational knowledge is essential for understanding the developments and challenges in video coding.

1.2 Video Signal Format

A video is a multimedia file format that stores visual information as a sequence of images also known as frames. These frames themselves are then displayed in a fast succession to provide an illusion of motion. At a specific frame rate, these images continuously display motion due to the persistence of vision. Furthermore, video files may contain some

metadata such as encoding details, frame rate, and resolution, which are crucial for proper playback. Each frame within a video is a digital image composed of small elements called pixels. Each pixel corresponds to a single point in the image and carries a specific color value within a defined color space. The color space is an organized model that enables the accurate reproduction of colors in both digital and analog formats.

Color Space: Colors in the video are mainly represented in two formats: YUV (i.e. YCbCr) and Red, Green, Blue (RGB). YUV separates the image into one luminance, namely Y, and two chrominance components, U and V, thereby allowing more efficient compression and transmission since human eyes are more sensitive to changes in brightness than in color [11]. These constitute elements of different intensity levels to make up the color of a pixel. The RGB format represents the Red, Green, and Blue colors, where each of its components is represented by a single value whose combination gives the pixel color. Conversion of YUV to RGB or vice versa is done by the following equations 1.1 and 1.2. The conversion from YUV to RGB is described by equation 1.1, where R, G, and B denote the red, green, and blue components in the RGB color space, and Y, U, and V represent the luminance and chrominance components in the YUV color space.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.58060 \\ 1 & 2.03211 & 0 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix}. \tag{1.1}$$

On the other hand, the conversion from RGB to YUV is represented by equation 1.2.

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51498 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}, \tag{1.2}$$

These transformations ensure that it is possible to recover the original color information from the YUV values. The YUV color space is used in video compression and broadcasting because it allows greater compression ratios to be achieved by reducing the bandwidth required for chrominance components without significant degradation in perceived image quality.

Color Subsampling: It is an essential technique in video compression. It reduces the amount of color information being encoded, based on the fact that human eyes are more sensitive to differences in brightness (luma) than in color (chroma) [11]. This gives a big reduction in data with minimal loss perceived in image quality. Chroma subsampling retains all the luma information for every pixel but shares chroma data among groups of pixels. The amount of subsampling is indicated in ratio form, such as 4:4:4, 4:2:2, or 4:2:0 [11]. In this ratio, the first number represents the horizontal block size of sampling, usually 4 pixels; the second and third numbers refer to how many pixels share the same chroma information from top and bottom rows, respectively as depicted in Figure 1.1. This figure also shows the three common color sampling formats used in video compression: 4:4:4, 4:2:2, and 4:2:0. For example, the 4:4:4 format has a lot of chroma resolution, meaning there is no subsampling. The 4:2:2 format reduces chroma by half horizontally and is typically used in professional video formats. The 4:2:0 format, which is standard in most consumer video formats, from Blu-ray to streaming, reduces chroma information in horizontal and vertical dimensions, aiming at achieving an effective quality and compression balance.

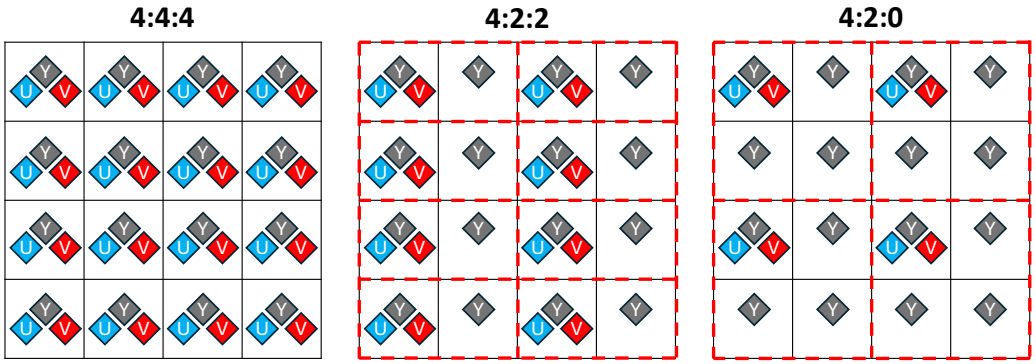


Figure 1.1 *Illustration of various YUV chroma subsampling formats*

Moreover, YUV color space is normally used in video compression, where luminance (Y) information is kept separate from chrominance components U and V [11]. This separation is advantageous because it allows for efficient compression, making it compatible with both monochrome and color displays. The different chroma subsampling ratios, such as 4:4:4, 4:2:2, and 4:2:0, show different levels of compression efficiency and image quality targeted toward the requirements of various applications in videos.

Spatial and Temporal Resolutions: Video resolution can be categorized into two types: spatial resolution, which refers to pixel density and image clarity, and temporal resolution, which is related to the frame rate and the smoothness of motion in a video.

Spatial resolution measures the number of pixels that constitute a video frame. This frame is an image that contains pixels making columns and rows, where columns define the frame width (W) and the rows define its height (H). Therefore, the resolution of the video is represented as W×H pixels. It also defines the level of detail that can be seen in the video. Higher spatial resolution results in more detailed images with an increased number of pixels, unlike low resolution, which contains fewer pixels, as shown in Figure 1.2. Common resolutions are Standard Definition (SD) (480p), High Definition (HD) (720p), Full-HD (1080p), 4K (2160p), and 8K (4320p) as illustrated in Figure 1.2. The spatial resolution is critical for big screen displays and for applications that require high resolutions.

Figure 1.2 *Various spatial resolutions from SD to 8K*

Temporal resolution is determined by the frame rate and is typically measured in Frames per Second (FPS). The frame rate tells how often a refresh rate for the video frames occurs. For instance, Figure 1.3 shows a video played at a temporal resolution of f FPS where each frame shows for $\frac{1}{f}$ seconds before refreshing. This refresh rate is important in achieving smooth and fluid motion for video play. Higher frame rates provide a smooth viewing experience, particularly in fast-action content like sports, while lower frame rates may show visible stuttering. Video codecs must effectively handle high

temporal resolutions to maintain smooth motion while simultaneously using compression techniques to keep data rates efficient. Common temporal resolutions include 24 FPS (cinematic productions), 30 FPS (standard for television), and 60 FPS (preferred for HD content and sports).

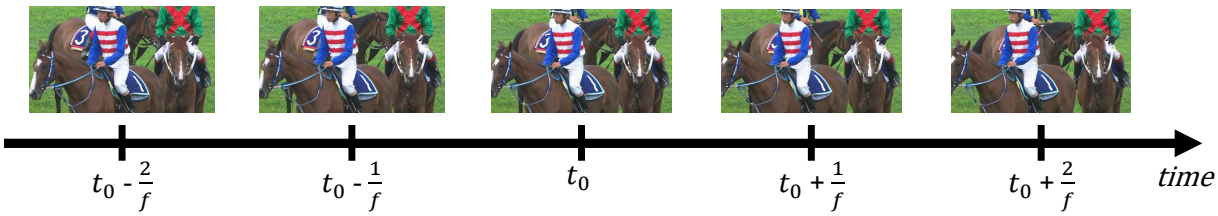


Figure 1.3 *Video frames displayed in a temporal resolution f*

Bit depth: It refers to the number of bits used to represent the color information of each pixel in an image or video. Higher bit depths allow for a more precise representation of luminance and chrominance values, enhancing pixel accuracy and improving overall video quality. During the coding process, both luminance and chrominance are typically maintained in 8-bit or 10-bit formats for input and output. In 8-bit format, there are 256 levels per channel, resulting in a total of $256^3 = 16,777,216$ possible colors. In 10-bit, this increases to 1024 levels per channel, leading to $1024^3 = 1,073,741,824$ colors. The greater the bit depth, the higher the number of available colors, which is crucial for delivering more vibrant and makes color shades more accurate in the images. In addition, the benefits of higher bit depths extend beyond color representation; they also significantly enhance the dynamic range, i.e., the spectrum between the darkest and brightest parts of the video. This is especially relevant for High Dynamic Range (HDR) content, where increased bit depth allows for more detailed and nuanced color gradations. However, Bit depths can extend to 12 bits per channel, increasing the number of colors to 4096 levels per channel, resulting in more than 68 billion colors. This capability supports more levels of color and brightness, which is crucial in post-production, display technology, and the creation of professional-grade content. The support of high bit depth for color and dynamic range precision significantly shapes the quality and realism of the modern video experience, particularly in applications that demand maximum visual fidelity.

1.3 Common Video Coding Standards

Video coding has changed over the past decades. The development history is illustrated in Figure 1.4, where some major organizations involved in the standards of video coding are presented. The most adopted standards among them are Advanced Video Coding (AVC) [12], High Efficiency Video Coding (HEVC) [6], and recently the Versatile Video Coding (VVC) [7]. These standards are crucial for efficient compression, transmission, and storage of video content. Developed by leading organizations, these standards undergo continuous refinement to ensure high-quality video output while maximizing bandwidth efficiency in response to the high requirements of different video applications.

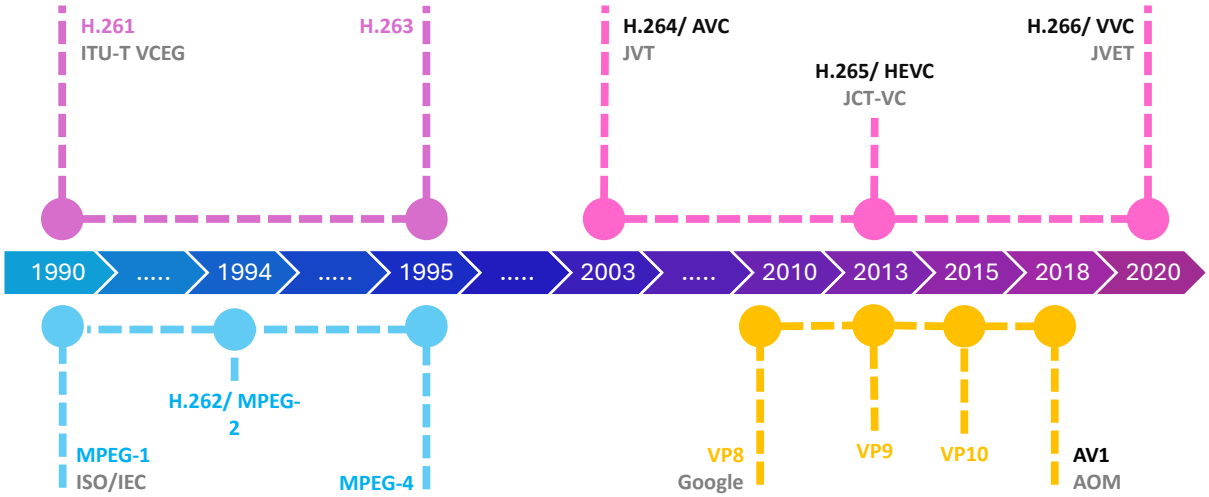


Figure 1.4 *Development of video coding standards over time*

The introduction was in the early 1990s with the H.261 proposed by the Video Coding Experts Group (VCEG) [13], with the first use of the classical hybrid video coding scheme, which is used in the combination of spatial and temporal prediction with the transform coding. Remarkably, this type of hybrid coding remains the base for different video coding standards. Simultaneously, the Moving Picture Experts Group (MPEG), a working group of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC), developed MPEG-1 [14]. The year 1993 saw the release of MPEG-1, hence setting a fundamental base standard for the compression and playback of moving images and audio. This marked a pivotal development that paved

the way for further advancements in digital video and audio technology.

Building on the success of its predecessors, the mid-1990s saw the emergence of H.262, also known by the commercial name MPEG-2 [15]. This standard, developed jointly by International Telecommunication Union Telecommunication Standardization Sector (ITU-T) and IEC, was further optimized for receiving broadcasting on television and storing them on Digital Video Discs (DVDs). In 1995, H.263 [16] was released by the VCEG as an improvement over H.261, offering better compression efficiency and supporting low-bitrate video communications, which significantly improved the quality and accessibility of video conferencing.

The early 2000s marked a significant advancement in video coding technology with the development of H.264, also known as AVC [12]. It was a joint work of the Joint Video Team (JVT), formed by experts from VCEG and MPEG. AVC exhibited a huge improvement in compression efficiency over the predecessor standards. Hence, it became very popular for a wide range of applications, including internet streaming, Blu-ray Discs, and HDTV broadcasting. After the successful adoption of H.264/AVC, the Joint Collaborative Team on Video Coding (JCT-VC), consisting of experts from VCEG and MPEG, developed H.265, also known as HEVC [6]. In 2013, HEVC was released, giving almost double the compression ratio of AVC at the same video quality level [6]. This standard came along with the introduction of the HEVC test Model (HM) as the reference codec software [17].

The H.266, also known as VVC [7], is the latest in that evolutionary timeline. Developed under Joint Video Experts Team (JVET), composed of members from VCEG and MPEG, VVC was released in 2020. It achieves about 50% more bitrate saving by providing better compression efficiency and flexibility compared to its predecessor, the HEVC [7]. In this standard, a special codec software called the VVC Test Model (VTM) was provided [18]. Another implementation of this standard was provided by the Fraunhofer Heinrich Hertz Institute (HHI), referred to as Versatile Video Encoder (VVenC) [9]. It offers an optimized and fast implementation of the VVC standard. Since its initial release in September 2020, each subsequent version has delivered improvements in performance and runtime efficiency [19]. Significantly, Google developed the VP8 [20] and the VP9 [21] codecs, which form the backbone of the WebM video format [22]. In 2010, VP8 was declared open and royalty-free for use in the H.26x standards, and in 2013, the same declaration was made for VP9. Finally, Alliance for Open Media (AOM), founded by leading technology companies including Google, Microsoft, and Apple, developed AOMedia Video 1 (AV1) [23], an open, royalty-free video coding format. Released in 2018,

AV1 aims to provide superior compression efficiency compared to existing formats and is widely used on the web.

Kerdranvat et al. [24] conducted a comparative study of HEVC, AOM, and VVC, focusing on objective quality metrics and processing time. Their results show that for Ultra High Definition (UHD) sequences, VVC outperforms HEVC with a 42% Bjøntegaard Delta Bit Rate (BDBR) gain at the same Peak Signal-to-Noise Ratio (PSNR) level. AV1 also improves over HEVC, offering an 18% BDBR gain, though this is less than the improvement seen with VVC. The BDBR gains for VVC and AV1 remain consistent when evaluated using Video Multimethod Assessment Fusion (VMAF) or MS-Structural SIMilarity (SSIM) instead of PSNR. In terms of processing time, VVC is the most complex, with encoding and decoding times 1308% and 192% longer than HEVC, respectively. AV1 is less complex, with 497% longer encoding time and 76% longer decoding time compared to HEVC. However, it's important to note that VVC software is primarily for standardization, whereas AV1 software is optimized for production. This optimization over AV1's three years of development contributes to its relatively faster performance despite being less complex than VVC.

Table 1.1 *Key features comparison of HEVC (H.265) vs VVC (H.266)*

Feature	**Standard**	
	HEVC (H.265)	**VVC (H.266)**
Standardization Year	2013	2020
Compression Efficiency	Up to 50% better than AVC	30-50% better than HEVC
Partition Structure	Quadtree (QT), max 64×64	QTMTT, max 128×128
Intra Prediction Modes	35 modes	67 modes with advanced tools
Transforms	Square Transform	Multiple Transform Set (square/rectangular)
In-Loop Filtering	Deblocking, SAO	Adaptive deblocking, SAO, ALF
Resolution Support	Up to 4K	Native 4K, 8K support
Application Focus	Streaming, broadcasting	Streaming, VR, 360-degree video, gaming
Complexity	Medium	High

Table 1.1 compares the widely used HEVC standard with the latest VVC standard, highlighting the key features of each. VVC demonstrates significant improvements over HEVC in compression efficiency, partitioning flexibility, prediction modes, and resolution support, making it well-suited for high-resolution applications such as 4K, 8K, VR, and gaming. However, these advancements bring higher computational demands due to the increased complexity of prediction tools, transform sets, and in-loop filters, making real-time encoding and decoding more challenging, especially for low-power devices. The next section breaks down each step in the VVC process, covering partition structures, predic-

tion modes, transform techniques, and in-loop filters. This overview will help identify areas where our optimization efforts can reduce complexity without sacrificing performance.

1.4 Overview of VVC Encoding Process

The most recent video compression standard, known as Versatile Video Coding (VVC), is designed to significantly improve compression efficiency across a wide range of video applications, including traditional broadcasting, 8K streaming, and immersive media [7]. VVC employs a classical hybrid video coding approach, incorporating both intra and inter-frame prediction alongside transform coding.

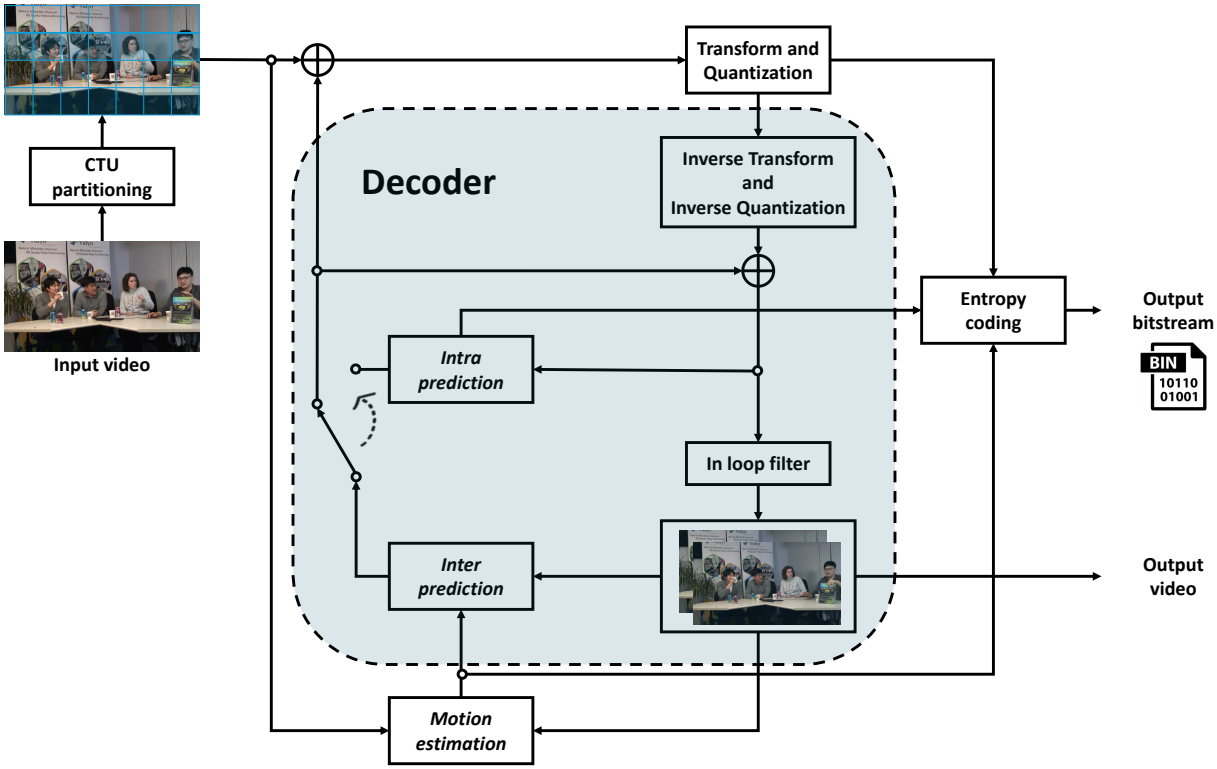


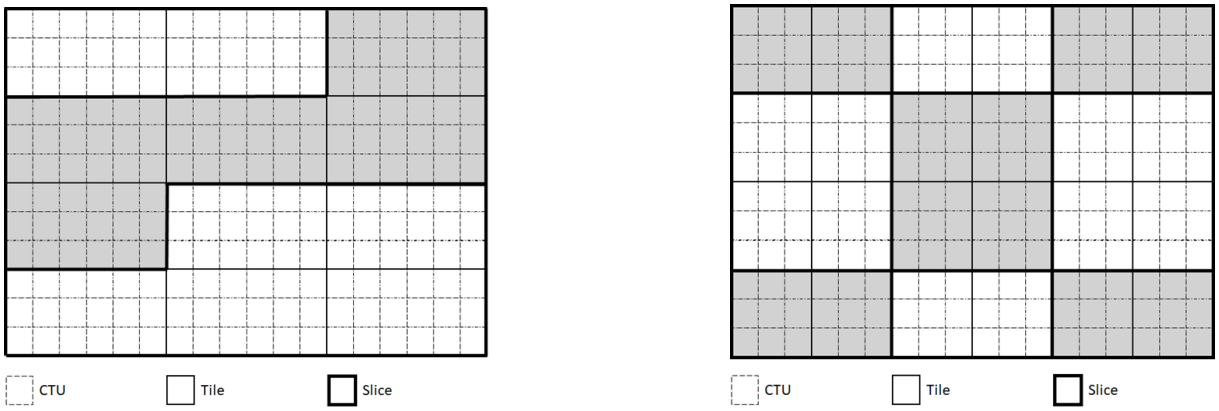
Figure 1.5 *Block-based hybrid video coding framework*

The encoding process, illustrated in Figure 1.5, involves several crucial steps: dividing the video frame into smaller blocks, performing either intra or inter-frame prediction, applying transform and quantization, and finally, conducting entropy coding [7]. This section focuses on the VVC standard, a key topic in this thesis. The principles discussed here are broadly applicable across various video coding standards. Figure 1.5 illustrates how the encoder converts video input into a compressed bitstream. This latter, consisting

of a sequence of bits, is processed by a decoder to reconstruct the video, albeit with some quality degradation. During the encoding process, several tools are employed to maximize the decoded video quality while minimizing the bitstream size.

1.4.1 Partitioning Block

In VVC, a picture is segmented into one or more tile rows and columns. A tile is defined as a sequence of Coding Tree Units (CTUs) that span a rectangular area within the picture. The CTUs within a tile are processed in a raster scan order specific to that tile. In this context, a slice consists of either an integer number of complete tiles or a series of consecutive complete CTU rows within a tile. As a result, every vertical slice boundary aligns with a vertical tile boundary. However, a horizontal slice boundary may not necessarily coincide with a tile boundary; it can instead comprise horizontal CTU boundaries within a tile. This occurs when a tile is divided into multiple rectangular slices, each consisting of some consecutive complete CTU rows within the tile, as shown in Figure 1.6a.



(a) *A picture with 18×12 luma CTUs, structured into 12 tiles and divided into 3 raster-scan slices*

(b) *A picture with 18×12 luma CTUs, structured into 24 tiles and 9 rectangular slices*

Figure 1.6 *Detailed illustration of picture partitioning into tiles, slices, and CTUs*

The VVC standard supports two slice modes: raster-scan slice mode and rectangular slice mode. In the raster-scan slice mode, illustrated in Figure 1.6a, a slice is composed of a sequence of complete tiles following the tile raster scan order of a picture. Conversely, in the rectangular slice mode shown in Figure 1.6b, a slice consists of either multiple complete tiles forming a rectangular region of the picture or consecutive complete CTU

rows from a single tile that together create a rectangular area. Within a rectangular slice, the tiles are processed in raster scan order within the defined rectangular region. This latter is made up of one or three Coding Tree Blocks (CTBs), depending on whether the video signal is a monochrome signal or has three color components [7]. In addition, VVC supports separated trees for both luma and chroma partitioning [7]. As a result, a

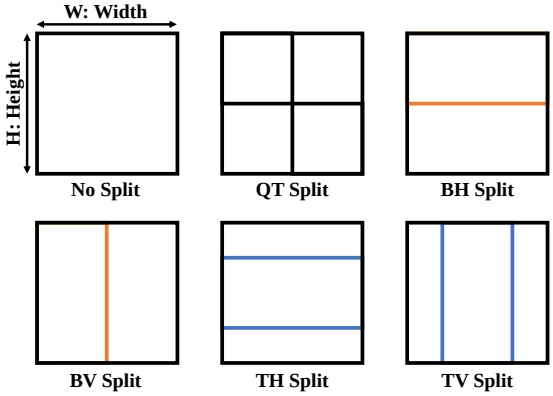


Figure 1.7 *Available CU split modes in VVC*

picture is segmented into equal-sized blocks called CTU, with a maximum size of 128×128 pixels. Then, each frame is processed from top left to bottom right in a raster scan order. Furthermore, in a low level, each CTU is recursively subdivided into smaller blocks, either square or rectangular shapes, called Coding Unit (CU), using the recursive partitioning scheme called Quadtree with Multi-Type Tree (QTMTT). This leads to six possible split modes, including No Split (NS), Quadtree (QT), Binary Tree Horizontal (BTH), Binary Tree Vertical (BTV), Ternary Tree Horizontal (TTH) and Ternary Tree Vertical (TTV), as illustrated in Figure 1.7.

In the QTMTT scheme, if a CU is assigned the NS mode, it maintains its original dimensions and the coding process begins. When a CU of size W×H (W: Width, H: Height) undergoes a QT split, it is divided into four equal sub-CUs, each with a size of W/2×H/2. Furthermore, Multi-Type Tree (MTT) enables the division of rectangular blocks within the CU into various shapes and sizes, offering flexibility in selecting the optimal partitioning mode during the encoding process.

For instance, in the Binary-Tree (BT) mode, the CU is split into two sub-CUs, either horizontally or vertically. Specifically, the BTH mode splits the CU into two sub-CUs with sizes of W×H/2, while the BTV mode splits it into sub-CUs of size W/2×H. For the Ternary-Tree (TT) mode, the CU is divided into three sub-CUs. In the TTH split mode, the first sub-CU has dimensions of H/4×W, the second sub-CU has dimensions of

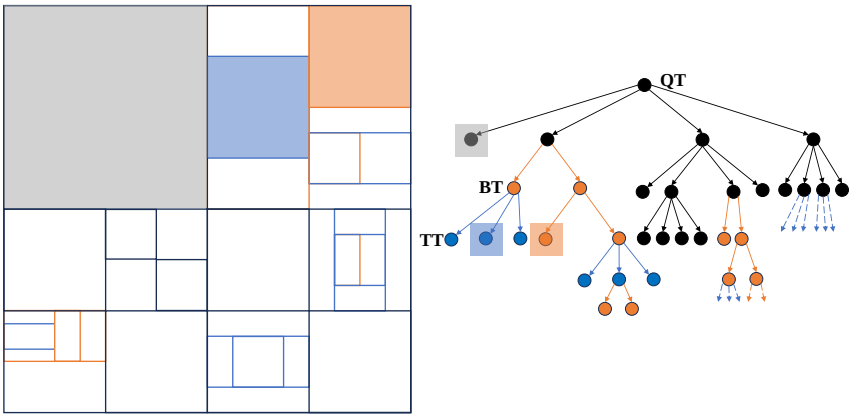


Figure 1.8 *Illustration of an example of QTMTT scheme division for a CTU next to its division tree*

H/2×W, and the third sub-CU has dimensions of H/4×W. Conversely, in the TTV split mode, the first sub-CU has dimensions of H×W/4, the second sub-CU has dimensions of H×W/2, and the third sub-CU has dimensions of H×W/4. These various split modes create a partitioning scheme with six possible configurations, as depicted in Figure 1.7, while Figure 1.8 illustrates an example of CTU partitioning using the QTMTT scheme, accompanied by its corresponding partition tree. In this figure, black nodes and arrows in this diagram represent QT partitions. The orange arrows and nodes indicate the BT partition modes, used in both horizontal and vertical directions. Similarly, TT partition modes are represented by blue arrows and nodes, which are also utilized in both directions. This color-coded visual representation simplifies the understanding of the hierarchical partitioning process within the VVC algorithm. Moreover, the gray box highlighting the first leaf corresponds to the gray-filled CU, similar to the orange and blue-filled CUs corresponding to the orange and blue leaves highlighted in the partitioning tree, respectively.

1.4.2 Intra Prediction

Intra-prediction exploits spatial redundancy within a single frame by predicting pixel values of a block based on the pixels of previously encoded blocks within the same frame. VVC introduces several new intra-prediction modes, significantly increasing the number of available directions and reference samples, as illustrated in Figure 1.9. While HEVC offers 33 prediction modes (represented by black arrows), VVC expands the number of angular modes to 65 angular and wide angular modes (represented by orange and black arrows), while still maintaining the DC and Planar modes [7]. Furthermore, new prediction

techniques for intra-frame coding were added in VVC. Among them, is the adoption of Intra Sub-Partitions (ISP) [25], which divides the CU luma component into two or four sub-partitions vertically or horizontally and shares the coding mode information while separating the prediction and transform processes [7]. Another intra-frame technique called Multiple Reference Line (MRL) [26], has been added to allow the prediction of luma samples from non-adjacent reference lines that can be two or three lines away from the current block. Other techniques such as Wide-Angle Intra Prediction (WAIP), and Matrix-Based Intra Prediction (MIP) have also been included in VVC [27]. Other techniques, such as Position Dependent Prediction Combination (PDPC), apply spatially varying weights to filter blocks that use Planar, DC, and certain angular prediction modes [28]. The Cross Component Linear Model (CCLM) predicts the chroma components of a block by employing linear models based on collocated reconstructed luma samples [28].

These enhancements allow for more accurate prediction of complex textures and patterns within a frame. Each block can be predicted using these modes, and the mode resulting in the least prediction error is selected for encoding.

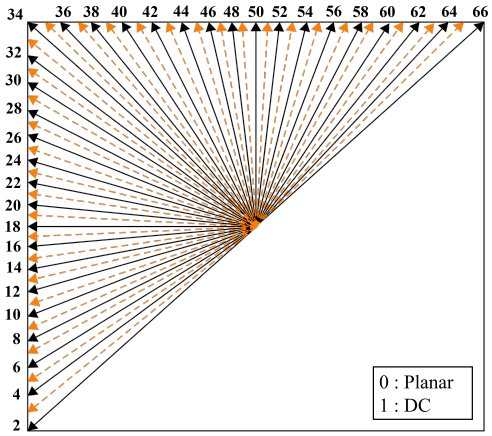


Figure 1.9 *Intra prediction modes*

1.4.3 Inter Prediction

Unlike intra mode, which focuses on spatial redundancies within a single frame, inter mode in VVC leverages temporal redundancies by predicting the content of a CU using previously encoded frames. VVC introduces several advanced inter-prediction tools that surpass the capabilities of its predecessor, HEVC. For instance, Extended Merge Prediction constructs the merge candidate list using spatial Motion Vector Prediction (MVP)

from neighboring CUs, temporal MVP from collocated CUs, history-based MVP, pairwise average MVP, and zero Motion Vectors (MVs). VVC also enhances motion compensation precision to $1/16^{\text{th}}$ of a luminance sample, which is particularly beneficial for affine motion modes. In Merge Mode with Motion Vector Differences (MVD), the prediction is refined using MV differences, allowing for more accurate motion depiction. Affine Motion Compensation utilizes two or three MVs to handle complex motions such as zooming, rotating, and perspective changes. Additionally, Sub-block Based Temporal Motion Vector Prediction refines MVs at a sub-CU level, improving prediction accuracy. The Adaptive Motion Vector Resolution (AMVR) allows varying levels of precision for MV differences, optimizing the trade-off between complexity and accuracy. VVC further introduces Bi-directional Optical Flow (BDOF), which enhances bi-prediction by refining MVs for every 4×4 sub-CU, and Decoder Side Motion Vector Refinement (DMVR), which refines MVs on the decoder side to improve prediction quality. Combined Inter and Intra Prediction (CIIP) merges inter and intra prediction signals for enhanced prediction accuracy. These sophisticated tools collectively contribute to the superior compression efficiency and video quality offered by VVC.

1.4.4 Transform and Quantization

Transform The transform process in video coding aims to condense the energy of the residuals produced by intra or inter-prediction into fewer, more significant components. This process involves converting the residuals into the frequency domain and concentrating the signal information in low-frequency components. In VVC, various transform types are used, such as Discrete Cosine Transform (DCT)-2, DCT-8, and Discrete Sine Transform (DST)-7, as part of the Multiple Transform Selection (MTS) framework. These transforms allow for more flexible and efficient encoding by adapting to the characteristics of the video content.

$$\text{DCT-2} \Rightarrow T_i(j) = \omega_0 \cdot \sqrt{\frac{2}{N}} \cdot \cos\left(\frac{\pi \cdot i \cdot (2j+1)}{2N}\right), \tag{1.3}$$

$$\text{DCT-8} \Rightarrow T_i(j) = \sqrt{\frac{4}{2N+1}} \cdot \cos\left(\frac{\pi \cdot (2i+1) \cdot (2j+1)}{4N+2}\right), \tag{1.4}$$

$$\text{DST-7} \Rightarrow T_i(j) = \sqrt{\frac{4}{2N+1}} \cdot \sin\left(\frac{\pi \cdot (2i+1) \cdot (j+1)}{2N+1}\right), \tag{1.5}$$

where ω_0 is defined as:

$$\omega_0 = \begin{cases} \sqrt{\frac{2}{N}} & \text{if } i = 0 \\ 1 & \text{if } i \neq 0 \end{cases} \tag{1.6}$$

In VVC, the MTS flag indicates whether these advanced transforms are used. If the flag is disabled, only the DCT-2 transform is applied both horizontally and vertically. When enabled, additional flags specify if DST-7 or DCT-8 transforms are performed in either direction. VVC supports large block sizes, which is advantageous for high-resolution video sequences. To manage the complexity of these large blocks, high-frequency coefficients are often set to zero. This approach helps in efficiently encoding large blocks while maintaining high video quality. VVC standard also includes two additional transform tools that are:

— **Low-Frequency Non-Separable Transform (LFNST)** applies an additional non-separable transform to the low-frequency coefficients of the primary transform. LFNST is particularly effective for blocks predicted with directional intra-prediction modes, where traditional transforms may not adequately compact the energy.

— **Sub-Block Transform (SBT)** This mode involves coding only part of the residual block for the CU. When sub-block transform is applied, the transform block is either half or one-quarter the size of the residual block. The transform kernel is selected according to the position of the block, with position-dependent core selection applied to the luma transform blocks. There are two modes, SBT Horizontal and SBT Vertical, which use different core transforms. For example, for SBT-V position 0, the horizontal transform is DCT-8, and the vertical transform is DST-7, as depicted in Figure 1.10. When one side of the residual transform unit (TU) is larger than 32, DCT-2 is used for both dimensions.

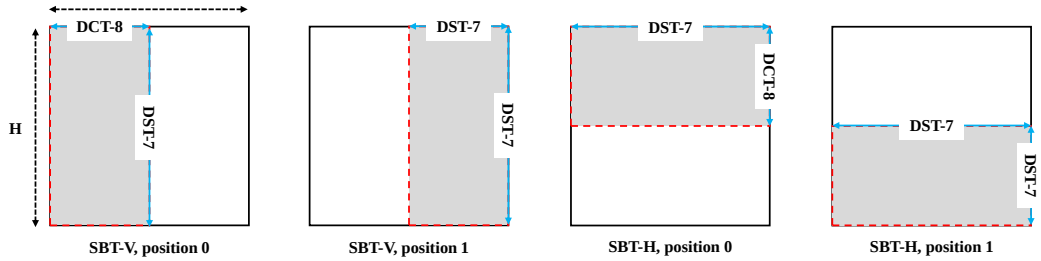


Figure 1.10 *SBT position and transform type*

Quantization in VVC is significantly improved through the introduction of Dependent Quantization (DQ). DQ enables the quantizer to adjust dynamically based on previously quantized coefficients, effectively implementing a form of vector quantization that enhances the coding efficiency of the residual signal [7]. While this process compresses the data, it introduces some loss of detail. VVC employs adaptive quantization, where the Quantization Parameter (QP) is adjusted according to the content of each frame to strike a balance between visual quality and compression efficiency. Moreover, VVC uses advanced quantization matrices tailored for different block sizes and transform types to optimize compression further. Rate-Distortion Optimization (RDO) is also applied to refine this process, ensuring the best trade-off between bitrate and visual distortion. These techniques allow VVC to deliver high-quality video at lower bitrates. The most commonly used QP values in VVC are 22, 27, 32, and 37.

1.4.5 In-loop Filters

In-loop filters are integrated into the decoding loop to mitigate artifacts such as blocking, ringing, or blurring introduced during quantization and other coding processes. Figure 1.5 (page 15) illustrates an image with such artifacts. The encoder's in-loop filters optimize filter parameters to enhance a frame's quality. These parameters are then transmitted to the decoder to ensure consistent quality improvement.

VVC utilizes four in-loop filters [29, 30]:

— **Luma Mapping with Chroma Scaling (LMCS)** [31]: This filter consists of converting input luma values to new code values and scaling chroma residue values based on luma. The process enhances coding efficiency for both standard and high dynamic range videos by using a piecewise linear model for luma mapping and neighboring luma samples for chroma scaling.

— **Deblocking Filter (DBF)** [32]: The DBF smooths the edges of CUs to eliminate blocking artifacts caused by block structures. It applies a smoothing filter based on the characteristics of surrounding blocks and the specific pixel values at block edges.

— **Sample Adaptive Offset (SAO)** [33]: SAO, retained from HEVC, further refines the reconstructed image by adjusting pixel values based on the characteristics of neighboring pixels. It operates after the DBF and before ALF, enhancing the overall image quality by reducing ringing and other artifacts that may be introduced during the encoding process.

— **Adaptive Loop Filter (ALF)** [30]: ALF minimizes the mean absolute error between the original and reconstructed images to reduce visible artifacts. It operates after the initial decoding steps, targeting the residual artifacts and distortions that may arise due to lossy compression.

1.4.6 Entropy Coding

Like HEVC, VVC uses Context-Adaptive Binary Arithmetic Coding (CABAC) for entropy coding [34], which is essential for compressing the quantized transform coefficients and other syntax elements into a compact bitstream. However, VVC introduces several improvements to CABAC to enhance coding efficiency [7]:

— **Improved Coefficient Coding:** VVC uses a more sophisticated coefficient scanning and coding method, which includes a reverse diagonal scan pattern and refined probability models for coefficient levels. These improvements reduce the number of bits required to encode the residuals, particularly for high-frequency coefficients [7].

— **High-Accuracy Multi-Hypothesis Probability Estimation:** This technique involves using multiple probability estimates for each context model, which are updated independently. By maintaining more accurate models of the data distribution, this method enhances the efficiency of the entropy coding process, leading to better overall compression performance [7].

1.4.7 Rate-Distortion Optimization

RDO is a fundamental technique in video coding that aims to balance the trade-off between the bitrate (rate) and the quality of the reconstructed video (distortion) [35]. The core idea is to minimize the distortion for a given bitrate or, equivalently, to minimize the bitrate for a given level of distortion. The optimization problem can be formulated as minimizing the Lagrangian cost function J, which is defined in Equation 1.7:

$$J = D + \lambda \times R \tag{1.7}$$

where D represents the distortion, R denotes the bitrate, and λ is the Lagrange multiplier that controls the trade-off between rate and distortion. The distortion D is typically measured using metrics such as Mean-Squared Error (MSE) or PSNR, while the rate R corresponds to the number of bits required to encode the video.

For example, consider a scenario where a video encoder must decide between two coding modes for a macroblock: Mode A and Mode B. Mode A might result in lower distortion but higher bitrate, whereas Mode B might result in higher distortion but lower bitrate. By computing the Lagrangian cost for both modes:

$$J_A = D_A + \lambda R_A \tag{1.8}$$

$$J_B = D_B + \lambda R_B \tag{1.9}$$

The encoder selects the mode with the lower cost J. Suppose $\lambda = 0.5$, $D_A = 10$, $R_A = 20$, $D_B = 15$, and $R_B = 10$. The costs are:

$$J_A = 10 + 0.5 \times 20 = 20 \tag{1.10}$$

$$J_B = 15 + 0.5 \times 10 = 20 \tag{1.11}$$

In this case, both modes have the same cost, and the encoder might use additional criteria to make the final decision. RDO is a crucial component in modern video coding standards, such as VVC and HEVC, ensuring efficient compression while maintaining high video quality.

1.5 Coding Modes

VVC provides different coding configurations for various applications, such as All Intra (AI), Low Delay (LD), and Random Access (RA) coding. Video compression reduces data by removing redundancy, with two main types: lossy and lossless. Lossy compression decreases file size with some quality loss, using methods like quantization and entropy coding. Lossless compression retains full quality by fully recovering data during decompression, using techniques like Huffman coding [36]. Lossy compression is ideal for streaming, while lossless is preferred for professional video editing due to its perfect data recovery.

1.5.1 All Intra Coding

AI coding is a configuration where every frame in the video sequence is encoded as an Intra-Frame (I-frame), which is self-contained and does not rely on any other frame for decoding as illustrated in Figure 1.11.

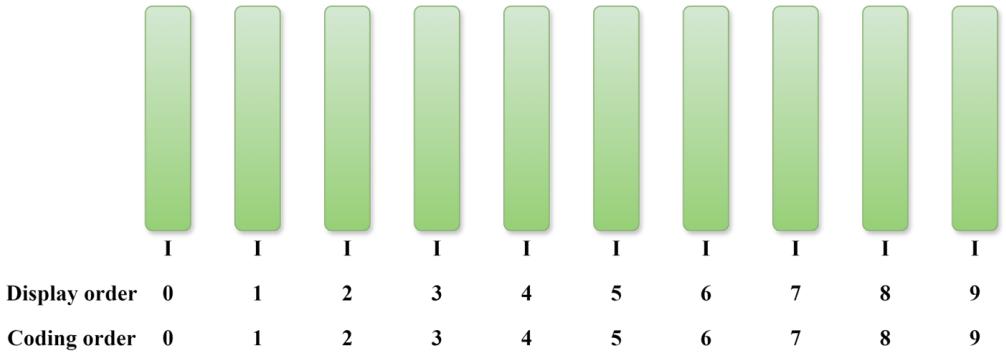


Figure 1.11 *AI coding configuration*

This method is typically employed in scenarios requiring low latency and high resilience to errors, such as video editing, broadcasting, and archival storage. In AI Coding, each frame is encoded independently without referencing other frames, leading to higher bitrates due to the absence of temporal redundancy reduction. This configuration is particularly suitable for applications needing immediate access to frames, providing high resilience as each frame is independently decodable, making the stream more robust to data loss or corruption.

1.5.2 Low Delay Coding

LD-B and LD-P configurations in VVC are designed to minimize latency, making them ideal for real-time communication applications such as video conferencing and live streaming. In LD-B configuration, the video sequence includes both I-frames and Bi-predictive-Frames (B-frames), with B-frames using both past and future frames for prediction, which improves compression efficiency while maintaining low delay.

Conversely, LD-P configuration uses I-frames and Predictive-Frames (P-frames), where P-frames reference only past frames as shown in Figure 1.12, further reducing the computational complexity and latency. Both configurations prioritize low delay to ensure smooth and immediate playback, with LD-P offering slightly faster processing at the cost of reduced compression efficiency compared to LD-B. These configurations are essential for applications requiring real-time interaction and responsiveness, balancing the trade-offs between compression efficiency and minimal latency.

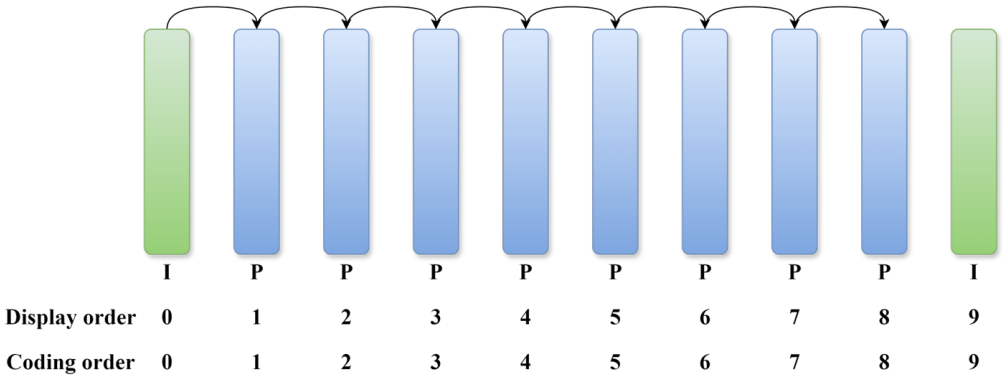


Figure 1.12 *LD-P coding configuration*

1.5.3 Random Access Coding

RA configuration in VVC is designed to facilitate efficient random access to different parts of a video sequence. In this configuration, the sequence includes both intra-frames (I-frames) and inter-frames (P-frames and B-frames), where I-frames are self-contained, and P-frames and B-frames utilize temporal redundancy by referencing other frames for encoding as shown in Figure 1.13.

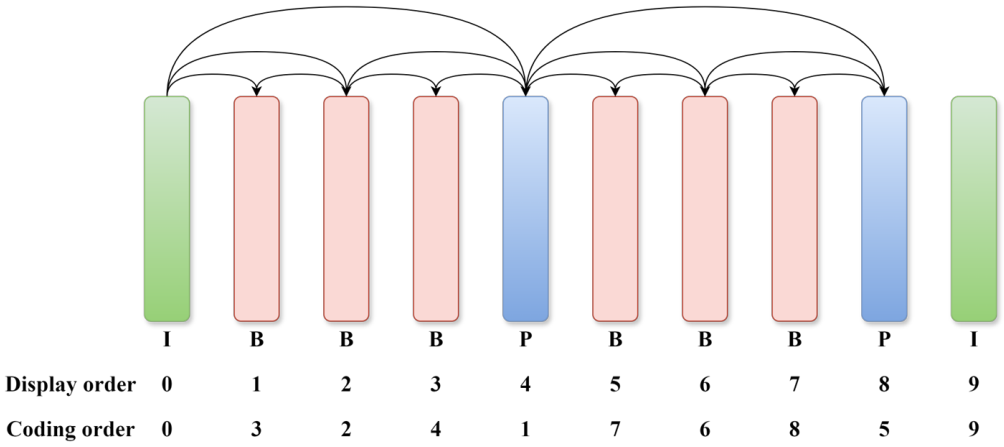


Figure 1.13 *RA coding configuration*

Figure 1.14 illustrates the hierarchical structure of frames with different Temporal Layer IDs (TIds), which are crucial for efficient inter-prediction. Orange squares represent I-frames with (TId: 0), these frames serve as primary reference points, while B-frames represented by blue squares have a TId ranging from 1 to 5, they use motion information from both I-frames and other B-frames to minimize redundancy. B-frames, also known as

bi-directional frames, enhance prediction accuracy by utilizing MV from both preceding and succeeding frames as illustrated in Figure 1.14 by blue arrows.

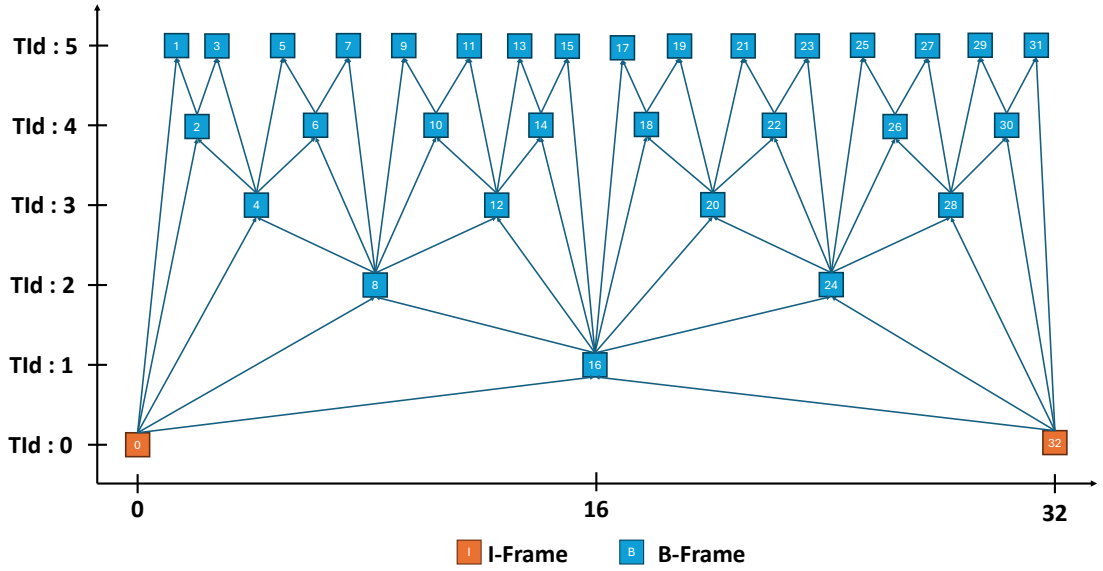


Figure 1.14 *Group of Pictures (GOP) of 32 frames in RA structure*

This structure allows for lower bitrates compared to AI Coding due to the efficient compression achieved through temporal prediction. RA Configuration is particularly beneficial for streaming and broadcasting applications where users might jump to different parts of the video, as it balances the need for random access with the benefits of compression efficiency. Key frames are periodically inserted to ensure that the video can be accessed at various points without significant delay, optimizing both storage and playback performance.

1.6 Video Quality Evaluation Metrics

Assessing video quality is essential for analyzing the effectiveness of video coding methods and ensuring a positive user experience. Video quality evaluation encompasses several metrics, which can be divided into two main categories: objective and subjective quality metrics. In addition to the complexity reduction metric, that measures the computational complexity between the baseline and proposed method. Figure 1.15 depicts the codec overview, illustrating the transition from input raw video to output decoded video. The evaluation metrics are derived by comparing coding data between the raw and decoded videos.

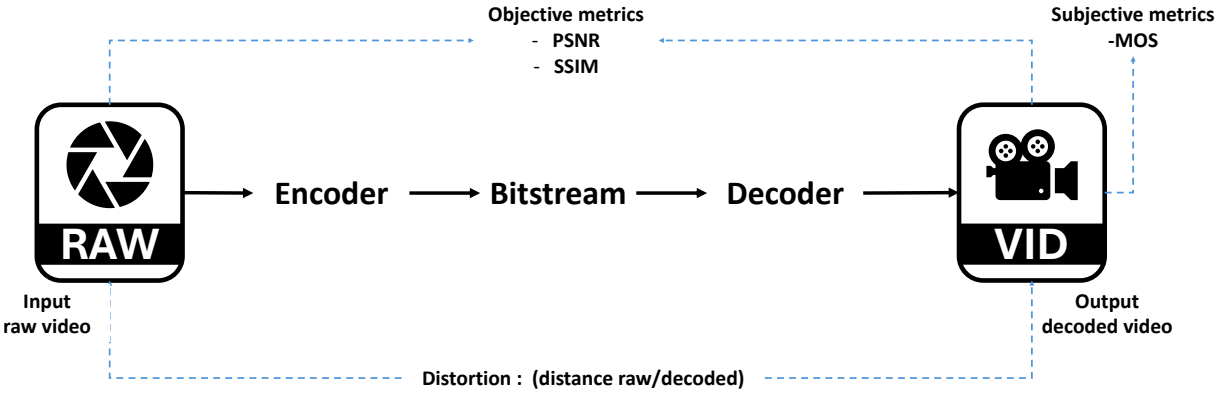


Figure 1.15 *Codec process overview*

1.6.1 Objective and Subjective Quality Metrics

Objective quality metrics are quantitative measures used to evaluate video quality based on mathematical comparisons between the original and compressed video. These metrics are reproducible and provide a consistent way to assess video quality.

Peak Signal-to-Noise Ratio (PSNR) is a classical objective metric that quantifies the difference between the original and compressed video frames. It is defined based on the MSE between corresponding pixels of the original and compressed images, denoted by I and K respectively. PSNR is expressed in decibels (dB) and is defined as:

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right), \tag{1.12}$$

with MAX_I is the maximum possible pixel value of the image (e.g., 255 for an 8-bit image) and MSE is the mean squared error between the original and compressed images, calculated as:

$$MSE = \frac{1}{W\,H} \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} [I(i,j) - K(i,j)]^2, \tag{1.13}$$

with W denoting the width of the image and H the height of the image, I represents the reference image, and K represents the evaluated image.

Structural SIMilarity (SSIM) is designed to improve upon traditional methods like PSNR and MSE by taking into account changes in structural information, luminance, and contrast [37]. Additionally, SSIM has recently been utilized in the rate-distortion optimization phase of [6] to improve the perceived video quality during the rate control

process. The SSIM index between two images I and K is defined as:

$$SSIM(I,K) = \frac{(2\mu_I\mu_K + c_1) + (2\sigma_{IK} + c_2)}{(\mu_I^2 + \mu_K^2 + c_1)(\sigma_I^2 + \sigma_K^2 + c_2)}, \tag{1.14}$$

with μ_I and μ_K are the average values of I and K, μ_I^2 and μ_K^2 the variance of I and K, σ_{IK} the covariance of I and K, c_1 and c_2 are two variables dependent of the dynamic range of pixels.

Video Multimethod Assessment Fusion (VMAF) developed by Netflix in collaboration with the Laboratory of Image and Video Engineering, is a perceptual video quality assessment algorithm designed to measure video quality degradation due to compression and rescaling [38]. As a full-reference metric, VMAF aims to closely approximate human perception of video quality and aligns strongly with subjective Mean Opinion Scores (MOSs). By leveraging Machine Learning (ML) techniques, a model was trained using a large sample of MOS scores as ground truth. VMAF calculates perceived quality by combining scores from various quality assessment algorithms using a support vector machine. The input features for the support vector machine include the detail loss metric, which accounts for visual defects that distract viewers; visual information fidelity, assessing information loss at different spatial scales; mean co-located pixel difference, measuring luminance differences between consecutive frames; and anti-noise signal-to-noise ratio. VMAF has demonstrated superior prediction accuracy compared to other metrics like SSIM and PSNR across multiple datasets, aligning closely with subjective ratings.

Bjøntegaard Delta Bit Rate (BDBR) is used to measure the improvement in bitrate efficiency achieved by using a certain video coding method or technology compared to a reference method. It is defined as mentioned in [39] and it is calculated as a weighted average of the BDBR with a ratio of 6:1:1, for the luma component Y, and the two chroma components U and V of a video signal by using the following Equation 1.15,

$$BDBR_{YUV} = \frac{(6.BDBR_Y + BDBR_U + BDBR_V)}{8} \tag{1.15}$$

Subjective quality metrics involve human observers to evaluate video quality. These metrics provide insights into how viewers perceive video quality, which is essential since human perception is the ultimate determinant of quality.

Mean Opinion Score (MOS) is a standard for subjective video quality assessment. It involves a group of viewers rating the quality of video sequences on a predefined scale, typically from 1 (bad) to 5 (excellent). The average of these scores provides the MOS value:

$$MOS = \frac{1}{N}\sum_{i=1}^{N} R_i, \tag{1.16}$$

where R_i is the rating given by the i-th viewer, and N is the total number of viewers.

1.6.2 Computational Performance Metrics

This refers to the metrics used to evaluate the efficiency of video coding algorithms in terms of computational resources and processing time, crucial for real-time applications and environments with limited computational power.

Computational Complexity measures the computational resources required to encode or decode a video sequence. In this thesis, complexity reduction is assessed using the Encoding Time Reduction (ETR) metric, symbolized as ΔETR. This metric evaluates performance improvements by calculating the difference in encoding complexity between a baseline reference encoding and the proposed encoding approach. The comparison is averaged across four distinct QPs to ensure robustness. The computation of ΔETR is detailed in Equation 1.17,

$$\Delta ETR = \frac{1}{4}\sum_{QP_i \in \{22,27,32,37\}} \frac{T_R\left(QP_i\right) - T_P\left(QP_i\right)}{T_R\left(QP_i\right)} \tag{1.17}$$

This equation signifies that the overall time savings are estimated by averaging the differences in encoding times between the reference encoder (T_R) and the proposed encoder (T_P) across the four specified QP values (22, 27, 32, and 37). Here, $T_R(QP_i)$ represents the encoding time taken by the reference encoder at the i^{th} QP, while $T_P(QP_i)$ denotes the encoding time for the proposed encoder at the same QP. By averaging these differences, ΔETR provides a comprehensive measure of the encoding time efficiency achieved by the proposed method relative to the baseline.

1.6.3 Common Test Condition (CTC)

The Common Test Conditions (CTC) have been established as a standardized environment to compare different innovations and advancements in video coding. This framework

is designed to meet the precise requirements of standardization committees. For instance, the CTC was defined by the JVET. Detailed guidelines for various types of video content, such as Standard Dynamic Range (SDR), HDR, 360-degree videos, high bit-depth, and high bit-rate coding [40]. Each type of video content, such as HDR or SDR, is represented by a broad set of sequences, which are categorized into various classes based on specific characteristics like resolution. There are seven different classes for SDR sequences: A1, A2, B, C, D, E, and F. Each class has predefined specific parameters in terms of resolution, frame rate, bit depth, and the number of frames that should be coded as detailed in Table 1.2. The number of I-frames in the sequence, referred to as the Intra Period, is also predefined based on the frame rate. This ensures consistent and reliable testing across different video coding contributions for various applications.

The CTC described in this document are defined for conducting experiments in a controlled environment where the outcomes of the experiments are comparable [40]. For all test scenarios, three temporal configurations are mandatory:

— All Intra (AI)
— Random Access (RA)
— Low Delay (LD)

The default encoder options consisting of the temporal and coding tools configuration are according to the JVET CTC (JVET-N1010), and some encoding parameters are different depending on the test scenario specified in this document. They are overwritten by the additional configuration file provided with the VTM software package.

1.6.3.1 Test Sets

According to the CTC in video coding, not all video sequences are required for every temporal configuration, as shown in Table 1.2. Indeed, the CTC specifies mandatory sequences based on temporal settings. The CTC video sequences are categorized into seven classes according to resolution: A1 (3840 × 2160), A2 (3840 × 2160), B (1920 × 1080), C (832 × 480), D (416 × 240), E (1280 × 720), and F (832 × 480 to 1920 × 1080). These classes exhibit varying frame rates, bit depths, motion characteristics, textures, and spatial resolutions. Specifically, Classes A through E consist of natural video sequences, while Class F includes specific screen content sequences [29].

Table 1.2 provides a summary of the CTC sequences, detailing their number of frames, frame rates, and bit depths. Additionally, it indicates whether each sequence is required for each temporal configuration. This structured approach ensures that testing is relevant

to the specific configuration being evaluated, allowing for more efficient and focused assessments. By following these standardized methods, the CTC guarantees reliable results across different video coding standards.

Table 1.2 *CTC video sequences characteristics*

Class	Sequence name	Frames	Rate	Bit-depth	AI	RA	LD
A1 (3840×2180)	Tango2	294	60	10	M	M	-
	FoodMarket4	300	60	10	M	M	-
	Campfire	300	30	10	M	M	-
A2 (3840×2180)	CatRobot	300	60	10	M	M	-
	DaylightRoad2	300	60	10	M	M	-
	ParkRunning3	300	50	10	M	M	-
B (1920×1080)	MarketPlace	600	60	10	M	M	M
	RitualDance	600	60	10	M	M	M
	Cactus	500	50	8	M	M	M
	BasketballDrive	500	50	8	M	M	M
	BQTerrace	600	60	8	M	M	M
C (832×480)	RaceHorses	300	30	8	M	M	M
	BQMall	600	60	8	M	M	M
	PartyScene	500	50	8	M	M	M
	BasketballDrill	500	50	8	M	M	M
D (416×240)	RaceHorses	300	30	8	M	M	M
	BQSquare	600	60	8	M	M	M
	BlowingBubbles	500	50	8	M	M	M
	BasketballPass	500	50	8	M	M	M
E (1280×720)	FourPeople	600	60	8	M	-	M
	Johnny	600	60	8	M	-	M
	KristenAndSara	600	60	8	M	-	M
F (various)	ArenaOfValor	600	60	8	M	M	M
	BasketballDrillText	500	50	8	M	M	M
	SlideEditing	300	30	8	M	M	M
	SlideShow	500	20	8	M	M	M

1.7 Machine Learning and Deep Learning in Video Compression

Video compression is an essential technology that enables the efficient storage and transmission of video content across diverse platforms and devices. Traditional video coding standards, including AVC [12], HEVC [6], and the more recent VVC [7], have achieved substantial compression efficiencies through advanced techniques refined over decades.

However, the rise of ML and Deep Learning (DL) offers new methods that can overcome the limitations of traditional approaches, providing better performance and adaptability in video compression.

1.7.1 Classification Metrics in ML

Assessing the performance of machine learning and deep learning models is a fundamental step to ensure their effectiveness and reliability. Evaluation metrics provide quantitative measures to determine how well a model generalizes to unseen data, enabling practitioners to identify strengths, weaknesses, and areas for improvement. Moreover, these metrics facilitate comparisons between models, aiding in the selection of the most suitable one for a given task. This section introduces key classification metrics that serve as the foundation for evaluating the performance of classification models:

— **Confusion Matrix:** A confusion matrix summarizes the counts of True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) predictions, as depicted in Figure 1.16:

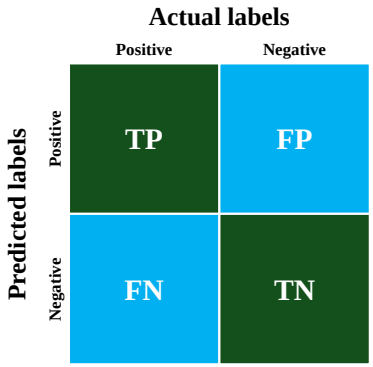


Figure 1.16 *Confusion matrix representation*

where :

— **True Positive (TP)**: Correctly predicted positive class.

— **False Positive (FP)**: Incorrectly predicted positive class.

— **True Negative (TN)**: Correctly predicted negative class.

— **False Negative (FN)**: Incorrectly predicted negative class.

— **Accuracy:** This metric measures the proportion of total correct predictions (both TP and TN) out of all predictions made, using the following equation 1.18:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{1.18}$$

— **Precision:** Precision is the ratio of TP predictions to the total positive predictions made. It indicates the classifier's exactness.

$$Precision = \frac{TP}{TP+FP} \tag{1.19}$$

— **Recall (Sensitivity):** Recall measures the classifier's ability to detect all relevant instances, highlighting its sensitivity.

$$Recall = \frac{TP}{TP+FN} \tag{1.20}$$

— **F1 Score:** The F1 Score is the harmonic mean of precision and recall, providing a balance between them.

$$F1 = \frac{2\times TP}{2\times TP+FP+FN} \tag{1.21}$$

— **Area Under the Curve (AUC):** AUC represents the area under the Receiver Operating Characteristic (ROC) curve, plotting the true positive rate (recall) against the false positive rate. A higher AUC indicates better performance.

1.7.2 ML and DL for Complexity Reduction of Video Coding Tools

The incorporation of Machine Learning (ML) and DL into the video compression pipeline encompasses various stages, such as intra and inter prediction, motion estimation and compensation, entropy coding, and post-processing. These techniques aim to optimize compression efficiency, reduce computational complexity, and improve the quality of the reconstructed video. Unlike traditional methods that rely heavily on manually crafted algorithms, ML and DL approaches utilize data-driven models to capture complex patterns and redundancies present in video content.

Since the development of AVC, Rate-Distortion Optimization (RDO) [35] has been an essential element in video encoder design. RDO contributes to coding efficiency by minimizing the overall cost, which is mathematically represented by Equation (1.7) in page 23. The introduction of new and advanced coding tools always brings in more decision points during the coding process. This increase in coding options results in greater computational complexity and slower processing times for RDO. This is because the optimizer needs to assess a larger set of potential coding options. To tackle the increasing complexity and computational demands of RDO, researchers have developed various approaches to reduce

its overhead by narrowing the search space. Some methods concentrate on video texture characteristics or employ heuristic techniques to limit the coding decisions considered by RDO. Meanwhile, other approaches leverage ML and DL to accelerate the process, enabling the encoder to make quicker decisions by predicting optimal coding choices in advance.

Intra Prediction Liu et al. [41] tackled the computational challenges in the VVC standard, specifically focusing on the ISP coding mode. While the ISP mode excels with frames rich in texture, it does not offer advantages for flat or uniform frames and can add unnecessary computational load. To address this, authors in [41] developed an algorithm based on Random Forest (RF) classifier that assesses the texture complexity of CUs to determine whether the ISP mode should be applied. By skipping the ISP mode for simple textures. Their experimental results showed a 7% reduction in coding time with only a 0.09% increase in the BDBR under the AI configuration. Similarly, researchers in [42] have focused on accelerating the ISP prediction in VVC by incorporating machine learning techniques. This method uses a simple Light Gradient Boosting Machine (LightGBM) model to make early skip decisions for the ISP mode. This approach evaluates features such as the mean absolute sum of transform coefficients to quickly determine whether testing the ISP mode is necessary for a given block. By making these early predictions, the encoding process becomes faster. Experiments have shown that this method achieves an average encoding time reduction of 7.2% under the AI coding configuration, with a 0.08% increase in BDBR.

Inter Prediction In addition to these advancements in intra prediction, ML has been applied to inter mode prediction to enhance video coding quality. Wang et al. [43] proposed a neural network-based inter prediction algorithm for HEVC that utilizes both spatial and temporal information to improve prediction accuracy. Their method employs a neural network architecture combining a Fully Connected Network (FCN) and a Convolutional Neural Network (CNN). The FCN processes spatial and temporal neighboring pixels to produce an output matrix matching the size of the current block. This output is then combined with the traditional inter prediction and fed into the CNN, which generates refined residuals. These residuals are added to the inter prediction to yield a more accurate prediction of the current block. This approach is integrated into HEVC after conventional inter prediction modes like inter, merge, or skip. It achieves an average BDBR reduction

of 1.7% under the LD-P test condition compared to HM 16.9. However, this improvement comes at the cost of increased encoding complexity.

Transform An advanced tool introduced in the VVC standard is The MTS which significantly increases the complexity of the transform module by requiring the selection of an optimal transform. Hamidouche et al. introduced a hardware-efficient approach to approximate the DST-7 and DCT-8 in the context of VVC [44]. The authors demonstrated that DCT-8 could be derived from DST-7 through the application of a simple vector reflection and sign-change matrices. The focus of this proposed approach was on approximating the forward and inverse DST-7 using optimized and fast DCT-2 implementations. To achieve this, they employed a genetic algorithm to identify the adjustment band-matrix which is essential for approximating DST-7 from DCT-2. This algorithm iteratively modifies elements of the matrix during the mutation process until convergence is achieved. Their approach resulted in minimal coding loss, achieving a 0.07% BDBR for the AI configuration while maintaining 96% encoding time and 85% decoding time. The RA and LD-B configurations exhibited similar coding loss and computational complexity during encoding and decoding. However, this method significantly reduced the number of operations and memory requirements, making it particularly advantageous for hardware implementations like Field-Programmable Gate Array (FPGA) platforms. Another work in [45] proposed a lightweight ML-based method using the RF algorithm to optimize the MTS process in the VVC encoder. The proposed method simplifies the selection of transform combinations, reducing the complexity introduced by MTS. Experimental results show a 39.44% reduction in transform coding time and a 9% decrease in overall encoder time, with only a 1.6% increase in bitrate compared to the standard approach.

Partitioning Process A notable example is the two-stage learning-based technique proposed by Tissier et al. [46], implemented in VTM 10.2. This method first uses a CNN to analyze the spatial features of input blocks, generating probability vectors that predict partitioning decisions along each 4×4 edge. Then, a Decision Tree (DT) model predicts the most probable splits for each block based on these features. By concentrating the RDO process on the most likely configurations, this approach significantly reduces encoding time by over 47.4% with corresponding BDBR increases of 0.79% under AI configuration. In another advancement, Zhang et al. [47] proposed a Global Convolutional Network (GCN) module to improve partitioning predictions in VVC encoding. This module uses large-

kernel convolutions to capture global information from CUs, enhancing decisions within the QTMTT structure. Another study in [48] treated the CU partitioning problem as a binary classification task using a fuzzy support vector machine (FSVM). This approach improved the accuracy of CU decisions and enhanced the overall performance of the HEVC encoder. By utilizing image features in combination with Support Vector Machines (SVMs), this approach created classification models that not only improved the accuracy of CU partitioning but also optimized the RDO process.

Finally, the main focus of this thesis work is to optimize the VVC encoder, particularly focusing on the VVenC to bring it closer to real-time application requirements. Thus, a detailed discussion of other ML and DL-based approaches in video compression will be covered in the subsequent chapters. We will discuss various state-of-the-art approaches related to each of our contributions.

1.7.3 End-to-end Video Coding Frameworks

End-to-end video coding frameworks are reshaping the landscape of video compression by replacing traditional, hand-crafted algorithms with fully optimized neural network architectures. These frameworks have demonstrated the ability to significantly improve compression efficiency, particularly in challenging conditions like low-bitrate scenarios.

For instance, a framework for learned video compression was proposed by Lu et al. [49]. Its primary contribution lies in its joint optimization of motion estimation, motion compensation, and residual compression using deep neural networks. This framework employs learning-based optical flow estimation to capture motion information between frames, which is then encoded using auto-encoder style networks. Both motion and residual information are jointly compressed, and all components in the framework are optimized via a single loss function that balances rate-distortion performance. This approach contrasts with traditional codecs, where each stage (motion estimation, compensation, quantization, etc.) is independently optimized.

Another study in [50] proposed a framework that shifts the compression process from the pixel space to the feature space. This innovation addresses some of the inherent limitations of pixel-domain compression, such as inaccuracies in motion estimation and compensation. In this framework, core operations such as motion estimation, motion compensation, and residual encoding are performed in the feature space, allowing for more abstract and effective representations of video content. The core contribution of this approach lies in its deformable compensation module, where motion estimation occurs in

feature space to generate motion offsets, which are then compressed using an auto-encoder network. The predicted feature maps are then used for motion compensation, leveraging deformable convolution to improve accuracy. Residuals between current and predicted frames are also processed in the feature domain. The framework incorporates a non-local attention mechanism for fusing information from multiple frames, resulting in superior frame reconstruction.

These end-to-end techniques are performed without the RDO process, which eliminates the exhaustive search required by conventional standards. The complexity issue is then shifted to the optimization of the neural networks.

1.7.4 Advantages and Challenges of ML and DL in Video Compression

The integration of ML and DL into video compression brings significant improvements and challenges. These approaches enhance compression by learning from data to accurately capture complex video patterns, leading to better prediction and fewer errors. However, despite these advantages, using ML and DL in video compression also presents several challenges. The computational demands of deep learning models can limit their use in real-time applications and on devices with limited power. Also, fitting these models into existing video compression standards can be difficult due to compatibility issues. Another challenge is the need for large and diverse training datasets to prevent models from overfitting to specific video types. Managing these complexities while keeping energy use low and maintaining fast processing speeds is a key focus in the development of video technology.

Ongoing research efforts are focused on overcoming the aforementioned challenges by developing more efficient neural network architectures, such as lightweight models that reduce computational demands without compromising performance. Additionally, strategies such as transfer learning and unsupervised learning are being investigated to reduce dependence on large labeled datasets, aiming to improve model generalization across various types of video content. Hybrid methods that merge traditional coding techniques with ML and DL enhancements are also under exploration to capitalize on the strengths of both approaches. The inclusion of ML and DL in new video coding standards like VVC is anticipated to significantly advance video compression technologies, facilitating their incorporation into real-time applications.

Overall, ML algorithms are generally more lightweight and computationally less demanding than DL algorithms. Traditional ML methods such as DT, RF, LightGBM, or SVM tend to have faster inference times and lower memory requirements, which makes them more suitable for real-time applications, especially in environments with limited computational resources. Building on this, the LightGBM model was selected for its superior balance of efficiency and performance, especially when dealing with large datasets and high-dimensional data. LightGBM employs a gradient-boosting framework optimized for both speed and memory efficiency, allowing for rapid training and prediction times while maintaining high accuracy [10].

1.8 Conclusion

This chapter provided a comprehensive overview of key video coding concepts, covering essential elements such as video signal formats, including color space, subsampling, spatial and temporal resolutions, and bit depth. Major video coding standards were discussed, with a particular focus on the VVC encoding process. Key aspects of this process, including partitioning, prediction methods, transform and quantization, in-loop filters, and entropy coding, were examined in detail. Additionally, the chapter reviewed both lossy and lossless compression techniques, coding modes, and video quality evaluation metrics. Moreover, the chapter explored the role of ML and DL in video compression, highlighting their potential benefits as well as challenges such as computational demands and data requirements. ML has demonstrated significant promise in reducing encoding complexity, which led to the decision to integrate a lightweight ML model in this thesis work.

This comprehensive foundation in this chapter paves the way for the subsequent chapters, which will delve into our contributions to optimize video encoding. These contributions, based on ML-driven approaches, aim to enhance encoding performance with a particular focus on real-time applications.

Chapter 2

Statistical Analysis and Performance Evaluation of the VVenC Encoder

2.1 Introduction

As discussed in Chapter 1, Versatile Video Coding (VVC) introduces several improvements over its predecessor, High Efficiency Video Coding (HEVC). These enhancements make its reference encoder, the VVC Test Model (VTM), more efficient but also computationally demanding. To create a practical and high-performance encoding solution, the Versatile Video Encoder (VVenC) encoder was developed as an open-source implementation of VVC, aiming to provide similar efficiency as VTM with reduced runtime [9]. This chapter provides an overview of the VVenC encoder. It begins by introducing the VVenC framework and its flexible preset configurations, which are tailored to meet a range of encoding needs. It then explores core components such as partitioning schemes, the different split modes and their impacts on encoding performance. To assess the encoder's performance, VVenC is systematically compared under configurations such as Random Access (RA) and All Intra (AI). Utilizing a custom-built application, *AnalyzeITV*, data on runtime efficiency and bitrate performance is gathered and analyzed for each configuration. A major focus of this analysis is on VVenC's partitioning process, with particular attention to how parameters like Coding Unit (CU) size distribution and Quadtree with Multi-Type Tree (QTMTT) partitioning influence encoding complexity and output quality. The *AnalyzeITV* tool enables detailed insights into the impact of various partitioning choices on encoding efficiency across presets. In addition to partitioning, this chapter examines VVenC's capability for real-time encoding across different hardware architectures, including ARM and x86. It also evaluates performance using thread parallelism and tiles parallelism to assess the encoder's effectiveness in these configurations.

The findings from this chapter provide a comprehensive understanding of VVenC's operational characteristics, laying the groundwork for the optimization strategies discussed in subsequent chapters. By identifying specific areas where VVenC can be further optimized, this analysis paves the way for achieving even greater encoding speed and quality, aligning the encoder's capabilities with the demands of modern video applications.

2.2 Overview of VVenC Encoder

Since the standardization of VVC, Joint Video Experts Team (JVET) supplies a reference software called VTM [18]. Furthermore, just after the finalization of the VVC standard, the Fraunhofer Heinrich-Hertz-Institut (HHI) presented a project to develop an encoder based on VTM software to provide a real-world encoder known as VVenC [9]. VVenC can be defined as an open optimized implementation of VVC including all VTM's performance at shorter runtimes. Moreover, it presents real-world encoder features like single-pass and two-pass rate control and efficient multi-threading for further speedup [9]. Based on the VTM and implemented in C++, the VVenC software is optimized through a new design that reduces performance bottlenecks. It includes enhanced Single Instruction Multiple Data (SIMD) optimizations, faster encoder search algorithms, and multithreading support to fully utilize parallel processing [9]. SIMD enables VVenC to perform repetitive tasks, like motion estimation and transform coding, on multiple data points in parallel. Multithreading further divides encoding tasks across multiple CPU cores, allowing concurrent processing of frames and frame regions (tiles). This approach boosts encoding speed by maximizing CPU utilization and supports both frame-level and tile-level parallelism [19]. VVenC also offers practical features such as frame-level bitrate control and visually optimized encoding. VVenC is particularly well-suited for the RA use case and high-resolution videos [51]. Two standalone encoder executables are available: *vvencapp* and *vvencFFapp*. The first is a simple application designed for ease of use with basic encoding options. The second is a more advanced encoder with a VTM-style interface, offering greater flexibility in configuring encoding parameters.

In the remainder of this thesis, *vvencFFapp* is used, as it closely aligns with VTM performance while providing reduced runtime [52].

2.2.1 Presets Configuration

VVenC provides five predefined presets ranging from slower to faster allowing users to balance between encoding speed and compression efficiency. These predefined presets simplify the encoder's use, making it accessible for different application scenarios without requiring extensive parameter tuning [52].

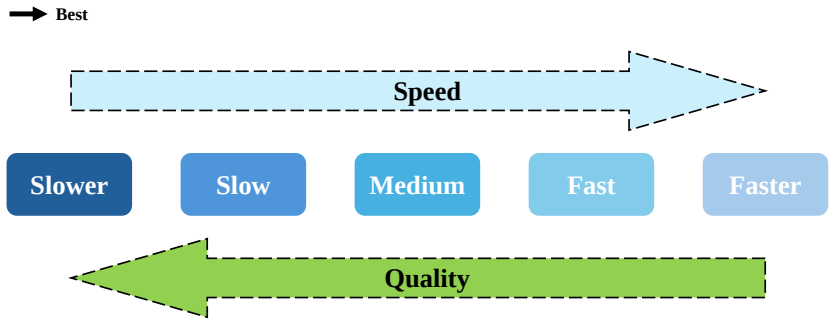


Figure 2.1 *Speed vs. quality of VVenC encoder presets*

Figure 2.1 depicts the presets and their corresponding speed-quality trade-off. Moving from "slower" to "faster" improves encoding speed but reduces quality, as "slower" achieves better encoding efficiency (lower BD-rate). Hence, users can select the most appropriate preset based on their specific requirements and the trade-off between quality and speed that is acceptable for their applications [19]. These improvements in each preset result from carefully selected values for each encoding parameter, as outlined in [19].

Table 2.1 *Parameter values for VVenC presets*

Parameter	Faster	Fast	Medium	Slow	Slower
Partitioning					
CTUSize	64	64	128	128	128
MaxMTTDepth	0	0	1	2	3
Motion Estimation					
SearchRange	128	128	384	384	384
BiPredSearchRange	1	1	4	4	4
Speedups					
qtbtt speedup	7	3	3	2	1
fastHad	1	0	0	0	0
Tools					
Affine	0	5	4	3	1
ALF	0	1	1	1	1
MIP	0	0	1	1	1
MMVD	0	3	3	3	1
RDOQ	2	1	1	1	1

Table 2.1 presents some parameter values for each preset, fine-tuning options like motion estimation, partitioning, and speedups to achieve an optimal balance between encoding speed and compression efficiency. A comprehensive table with all parameters and detailed explanations is available in [19].

2.2.2 VVenC Encoder Performance Evaluation

Figure 2.2, taken from the VVenC Wiki [19], compares different VVenC versions with the HEVC reference model (HEVC test Model (HM)-17.0) and the VVC reference model (VTM-19.2). This comparison illustrates how VVenC has evolved over time to enhance both compression efficiency and encoding speed [52]. HM-17.0 exhibits the slowest encoding and lowest compression, while VTM-19.2 achieves better compression at the cost of significant encoding time. Early versions of VVenC (e.g., 0.1.0.0) prioritize speed but with less compression efficiency. Over time, VVenC versions have improved drastically, with VVenC 1.12.0 achieving compression efficiency close to VTM but at much faster encoding speeds as shown Figure 2.2. This makes VVenC an increasingly competitive option for practical applications where both speed and compression are essential, outperforming HM significantly and approaching VTM's performance with far better speed.

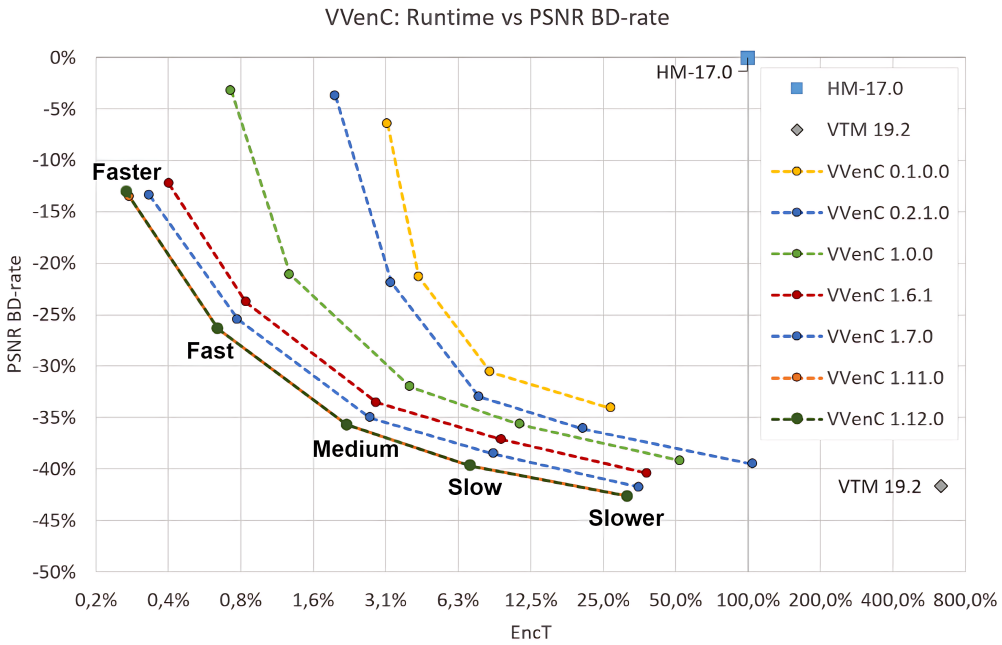


Figure 2.2 *Encoding efficiency (BD-rate) vs. encoding time (EncT) and relative encoder runtime for VVenC in comparison to HM-17.0 and VTM [from VVenC wiki]*

VVenC incorporates several innovative features and optimizations that enhance its performance and usability: it utilizes efficient parallel processing techniques, such as multi-threading and SIMD optimizations, to significantly accelerate encoding times without compromising video quality. Additionally, VVenC leverages advanced Rate-Distortion Optimization (RDO) strategies to improve compression efficiency, allowing it to achieve near-optimal quality at lower bitrates. The encoder also supports flexible presets and tuning options as previously shown in Table 2.1, enabling users to prioritize either speed or quality based on their specific use case. These features, combined with its alignment to the VVC standard, make VVenC highly adaptable to a wide range of video applications.

VVenC is open source and available on GitHub [52], which makes it accessible for academic research and development. We have provided detailed documentation to guide users through the setup and usage of the VVenC encoder, making it easier for the community to adopt and contribute [53]. Our comprehensive documentation includes detailed explanations of the encoder's files, classes, and functions, making it easier for other researchers in the field to explore and work with the code. Figure 2.3 illustrates the homepage of our documentation website [53]. This latter can be accessed using the following link: `https://vvenc.netlify.app/`.

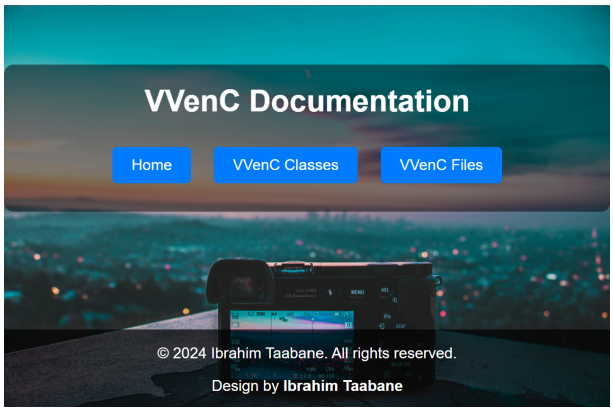


Figure 2.3 *VVenC source code documentation homepage overview*

2.2.3 Performance Analysis Under RA and AI Configurations

This section provides a detailed comparison of the VVenC encoder presets using the RA and AI configurations following the Common Test Conditions (CTC) guidelines for a thorough analysis. The slower preset was chosen as the reference point for comparisons, as it closely aligns with the performance of the VTM encoder. The other presets were

Table 2.2 *Trade-offs between ETR(%) and BDBR(%): performance evaluation of VVenC presets vs. slower under RA configuration*

	Video	Slow RA		Medium RA		Fast RA		Faster RA	
		BDBR(%)	ΔETR(%)	BDBR(%)	ΔETR(%)	BDBR(%)	ΔETR(%)	BDBR(%)	ΔETR(%)
A1	Campfire	3.10	70.70	8.14	91.11	22.39	97.79	35.70	99.00
	Tango2	5.01	73.38	11.56	91.41	27.73	97.16	43.96	98.66
	FoodMarket4	4.80	72.19	9.24	89.62	22.24	96.65	40.28	98.47
	Class average	4.30	72.09	9.65	90.71	24.12	97.20	39.98	98.71
A2	DaylightRoad2	8.71	73.40	16.79	91.88	34.71	97.17	57.55	98.79
	CatRobot1	9.13	73.32	16.38	91.83	33.79	97.07	60.19	98.75
	ParkRunning3	4.14	75.13	8.69	92.46	21.97	98.03	43.27	99.22
	Class average	7.33	73.95	13.95	92.06	30.16	97.42	53.67	98.92
B	MarketPlace	5.63	74.01	10.50	91.87	22.87	97.39	45.85	98.94
	RitualDance	4.77	72.29	11.08	91.51	24.37	97.43	37.63	98.77
	BQTerrace	5.54	76.68	11.10	92.25	23.68	96.78	47.39	98.73
	BasketBallDrive	5.80	75.45	13.70	92.84	30.65	97.76	49.83	99.05
	Cactus	5.46	76.01	11.48	92.37	23.01	97.49	50.55	98.93
	Class average	5.44	74.89	11.57	92.17	24.92	97.37	46.25	98.88
C	PartyScene	5.57	75.79	10.78	92.72	22.27	97.69	40.43	99.09
	BQMall	6.16	76.42	13.53	93.21	29.42	97.54	46.35	98.94
	BasketBallDrill	4.99	76.52	11.94	93.03	26.30	97.80	41.98	99.04
	RaceHorsesC	6.50	70.96	14.20	92.49	34.35	97.95	52.23	99.16
	Class average	5.81	74.92	12.61	92.86	28.09	97.75	45.25	99.06
D	BQSquare	5.99	80.55	11.25	92.69	19.97	97.36	47.26	98.95
	BlowingBubbles	5.09	78.99	10.77	93.59	21.94	97.76	37.49	99.09
	BasketBallPass	4.91	77.54	11.92	93.81	25.48	98.08	40.17	99.15
	RaceHorsesD	5.28	73.25	12.74	92.46	31.54	98.07	46.46	99.20
	Class average	5.32	77.58	11.67	93.14	24.73	97.82	42.85	99.10
E	FourPeople	4.36	77.72	9.50	92.06	16.35	96.14	30.76	98.31
	Johnny	4.56	77.91	9.81	90.73	17.98	95.16	34.28	97.74
	KristenAndSara	4.77	77.43	9.71	91.31	17.80	95.86	33.02	98.04
	Class average	4.56	77.69	9.67	91.37	17.38	95.72	32.69	98.03
F	ArenaOfValor	6.10	76.49	13.82	92.80	27.56	97.23	43.45	98.87
	SlideEditing	2.81	53.90	11.64	82.67	19.23	92.97	35.29	97.75
	BasketBallDrillText	6.12	79.19	13.71	93.92	29.88	97.78	48.31	99.00
	SlideShow	15.68	68.62	29.35	89.42	48.86	95.74	70.67	97.96
	Class average	7.68	69.55	17.13	89.70	31.38	95.93	49.43	98.40
	Total Average	5.78	74.38	12.32	91.72	25.82	97.03	44.30	98.73

evaluated against this reference, with a focus on their coding efficiency and the speed gains achieved relative to the slower preset under both RA and AI configurations. The results presented in Table 2.2 highlight the trade-offs between Encoding Time Reduction (ETR) and Bjøntegaard Delta Bit Rate (BDBR) across various presets of the VVenC encoder compared to the slower preset under the RA configuration. The slow preset maintains a strong compression efficiency with an average BDBR of 5.78% while achieving an ETR of 74.38%. In contrast, the medium preset exhibits a moderate increase in BDBR to 12.32% and an improved ETR of 91.72%, suggesting a balanced performance for scenarios where encoding speed is essential yet quality must be preserved. The fast preset significantly

increases the BDBR to 25.82%, paired with an impressive ETR of 97.03%, indicating a greater sacrifice in quality for reduced encoding times. The faster preset pushes this trade-off further, yielding the highest BDBR of 44.30% while achieving a time-saving of 98.73%, which may render it suitable for applications prioritizing speed over quality.

In the context of AI coding, Table 2.3 illustrates the trade-off between ETR and coding efficiency (BDBR) across various VVenC presets under the AI configuration.

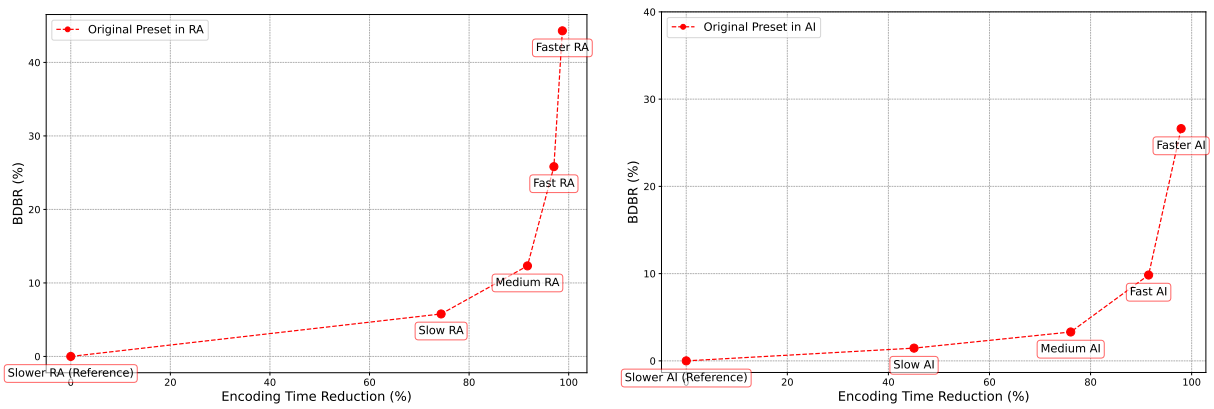
Table 2.3 *Trade-offs between ETR(%) and BDBR(%): performance evaluation of VVenC presets vs. slower under AI configuration*

	Video	Slow AI		Medium AI		Fast AI		Faster AI	
		BDBR(%)	ΔETR(%)	BDBR(%)	ΔETR(%)	BDBR(%)	ΔETR(%)	BDBR(%)	ΔETR(%)
A1	Campfire	1.07	43.95	2.10	73.99	6.77	89.74	26.41	97.20
	Tango2	0.64	38.48	1.21	60.26	7.39	79.64	31.38	93.25
	FoodMarket4	0.93	38.64	1.39	66.18	5.30	85.01	26.77	95.63
	Class average	0.88	40.36	1.57	66.81	6.49	84.80	28.19	95.36
A2	DaylightRoad2	1.49	45.26	3.13	74.06	11.50	90.88	32.31	97.73
	CatRobot1	1.81	43.60	3.09	72.63	10.39	89.56	28.98	97.33
	ParkRunning3	0.99	41.43	1.69	74.83	6.43	90.93	18.80	98.07
	Class average	1.43	43.43	2.64	73.84	9.44	90.46	26.70	97.71
B	MarketPlace	1.07	47.66	1.71	76.32	4.96	91.41	20.88	98.28
	RitualDance	1.16	45.74	2.74	77.64	7.29	91.83	21.12	98.27
	BQTerrace	1.74	48.96	3.88	79.57	12.20	93.98	30.13	98.80
	BasketBallDrive	1.48	45.23	3.21	73.30	11.42	91.48	36.99	97.96
	Cactus	1.11	45.99	2.71	77.68	8.81	92.95	24.96	98.56
	Class average	1.31	46.72	2.85	76.90	8.94	92.33	26.82	98.37
C	PartyScene	1.18	50.87	2.81	82.48	8.00	94.84	18.86	99.00
	BQMall	1.40	47.59	3.46	79.12	11.71	93.87	26.72	98.68
	BasketBallDrill	2.36	46.36	4.75	77.71	15.62	92.74	37.87	98.47
	RaceHorsesC	0.99	49.08	2.59	79.99	7.38	93.44	21.90	98.52
	Class average	1.48	48.48	3.40	79.83	10.68	93.72	26.34	98.67
D	BQSquare	1.44	51.43	3.19	81.96	9.59	94.70	20.48	98.85
	BlowingBubbles	0.87	50.87	2.53	82.16	7.60	94.71	18.88	98.98
	BasketBallPass	1.27	50.39	3.08	79.84	10.56	93.85	24.62	98.64
	RaceHorsesD	0.86	51.83	2.55	80.97	7.91	93.56	21.89	98.57
	Class average	1.11	51.13	2.84	81.23	8.92	94.21	21.47	98.76
E	FourPeople	1.30	44.84	3.21	77.42	9.42	92.87	24.33	98.44
	Johnny	1.31	46.21	3.35	76.10	11.00	91.91	27.61	98.05
	KristenAndSara	1.14	44.63	2.86	75.69	9.97	91.85	25.88	97.98
	Class average	1.25	45.23	3.14	76.40	10.13	92.21	25.94	98.16
F	ArenaOfValor	2.68	43.88	4.82	77.45	12.51	93.03	28.93	98.66
	SlideEditing	2.29	29.59	9.76	75.51	15.53	92.03	31.78	98.34
	BasketBallDrillText	3.20	45.60	5.65	77.97	17.13	93.02	39.44	98.52
	SlideShow	2.94	41.60	6.72	78.96	11.90	92.60	23.39	98.19
	Class average	2.78	40.17	6.74	77.47	14.27	92.67	30.89	98.43
	Total Average	1.46	45.07	3.31	76.07	9.84	91.48	26.62	97.92

This table shows that as the presets progress from slow to medium, fast, and faster, there are significant speed gains, with ETR values reaching up to 98% for many sequences, such as Campfire and BasketBallDrillText. However, these speed gains come with a no-

ticeable increase in BDBR, indicating a loss in coding efficiency. For example, the BDBR for BasketBallDrillText rises dramatically from 3.20% with slow to 39.44% with faster, highlighting how complex motion or text-heavy sequences experience the most quality degradation. In contrast, simpler sequences like MarketPlace and Tango2 maintain relatively lower BDBR values (20.88% and 31.38%, respectively) while achieving substantial time-saving improvements. On average, across all classes, BDBR increases from 1.46% to 26.62%, while ETR improves from 45.07% to 97.92%, indicating that the faster preset is best suited for applications where speed is prioritized, while slower presets offer better coding efficiency. This balance between speed and quality should be carefully considered depending on the content and use case, as some sequences tolerate faster presets better than others.

Figure 2.4 summarizes the results of Table 2.2 and Table 2.3 and depicts the trade-offs between the different presets in terms of encoding efficiency and time-saving of each preset under RA (Figure 2.4a) and AI (Figure 2.4b) configurations.



(a) *VVenC Original Presets Under RA* **(b)** *VVenC Original Presets Under AI*

Figure 2.4 *VVenC original presets trade-offs under RA and AI configurations*

Figure 2.5 illustrates the encoding speedup factors achieved by each preset relative to the reference slower preset. This data demonstrates that, in the RA configuration, the speedup increases progressively with each faster preset. For instance, the slow preset achieves a 3.72x speedup over slower, while medium achieves a 12.04x increase. This trend continues with the fast preset reaching a 38.97x speedup and faster achieving a remarkable 93.35x increase. In the AI configuration, speedup factors are generally lower, with the 'slow' preset achieving a 1.85x speedup and 'faster' reaching a maximum of 65.58x. RA mode encodes faster than AI by reusing data across frames, reducing computational load.

In contrast, AI encodes each frame independently, which requires more processing and slows down encoding. Further analysis results across various resolutions are available in Appendix A (see page 139).

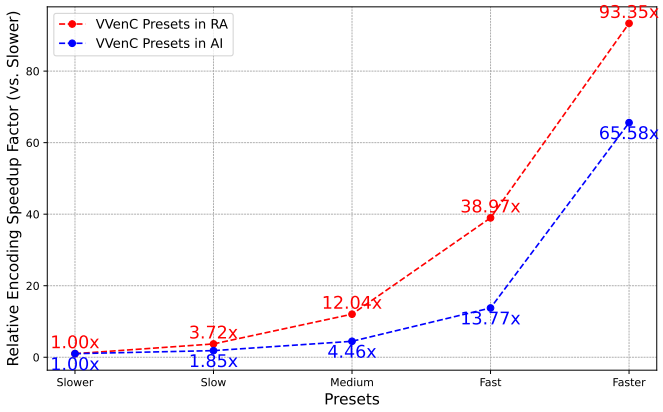


Figure 2.5 *Speedup factors across VVenC presets under RA and AI configurations*

2.3 Partitioning Across VVenC Presets

As outlined in Chapter 1 (section 1.4.1, page 16), a significant advancement in the VVC standard is the introduction of the QTMTT partitioning structure. This structure provides greater flexibility in how CUs are divided, allowing for more adaptive partitioning that enhances the quality of detailed areas, particularly in complex regions of the video. This section examines the implementation of the partitioning process in the VVenC encoder across its various presets. The analysis includes the impact of partitioning on the distribution of split modes and CU sizes, taking into account different resolutions, Quantization Parameter (QP) values, and coding modes. By varying these factors, it becomes possible to assess how the encoder's presets balance coding efficiency and complexity, as well as how partitioning choices influence overall video quality and encoding speed.

2.3.1 Partitioning Process in VVenC

In the VVC standard, frames are segmented into Coding Tree Units (CTUs) of up to 128×128 pixels. Each CTU is recursively divided into smaller CUs using a combination of Quadtree (QT) and Multi-Type Tree (MTT) partitioning strategies, which include

Binary-Tree (BT) and Ternary-Tree (TT) splits. These partitioning strategies result in five distinct modes: QT, Binary Tree Horizontal (BTH), Binary Tree Vertical (BTV), Ternary Tree Horizontal (TTH), and Ternary Tree Vertical (TTV). To enhance encoding efficiency, VVC limits the available partitioning modes based on the CU size, which is outlined in Table 2.4. This selective use of partition modes helps to optimize both compression performance and encoding speed by adapting the partitioning strategy to the content complexity of each CU. For AI coding, there is a mandatory initial division of each 128×128 CTU into four sub-CUs using the QT split mode. After this step, the remaining split modes are the same for both RA and AI configurations, allowing for flexibility while maintaining efficient encoding across both coding scenarios.

Table 2.4 *Possible split modes according to the CU size*

<table>
<tr><th>Height \ Width</th><th>128</th><th>64</th><th>32</th><th>16</th><th>8</th><th>4</th></tr>
<tr><td>128</td><td>QT, BT</td><td colspan="3">BT</td><td colspan="2">-</td></tr>
<tr><td>64</td><td rowspan="3">BT</td><td>All</td><td colspan="2">BT, TT</td><td rowspan="2">BT, TTH</td><td rowspan="2">BTH, TTH</td></tr>
<tr><td>32</td><td rowspan="2">BT, TT</td><td>All</td><td>BT, TT</td></tr>
<tr><td>16</td><td>BT, TT</td><td>All</td><td>BT</td><td>BTH</td></tr>
<tr><td>8</td><td rowspan="2">-</td><td colspan="3">BT, TTV</td><td>BTV</td><td>-</td></tr>
<tr><td>4</td><td colspan="3">BTV, TTV</td></tr>
</table>

As seen in Table 2.5, the presets control the depth of QT, BT, and TT partitions under the RA and AI. The slower and slow presets allow for up to three levels of BT or TT splits, achieving highly efficient compression at the expense of increased runtime. The medium preset reduces MTT depth to two levels for AI and only one level in RA, offering a balance between encoding speed and compression efficiency. In the fast and faster presets, partitioning complexity is minimized by limiting the encoder to only QT splits for RA, prioritizing encoding speed at the cost of reduced compression efficiency. The next section explores the impact of CU partitioning for each preset and analyzes CU size proportions across various resolutions.

Table 2.5 *Maximum depth values for each split mode and encoding preset in RA and AI*

Mode \ Preset	Faster		Fast		Medium		Slow		Slower	
	RA	**AI**	**RA**	**AI**	**RA**	**AI**	**RA**	**AI**	**RA**	**AI**
QT	4	4	4	4	4	4	4	4	4	4
MTT	0	0	0	1	1	2	2	3	3	3

To analyze the partitioning process in video frames across different presets and to visualize the distribution of block sizes and split modes within the frames, a custom

application named "*AnalyzeITV*" was developed. Figure 2.6 presents the application's interface, showcasing an example of its processing capabilities and the insights it generates. In the following sections, a detailed analysis of the partitioning process is conducted, highlighting the distribution patterns of block sizes and split modes within the presets.

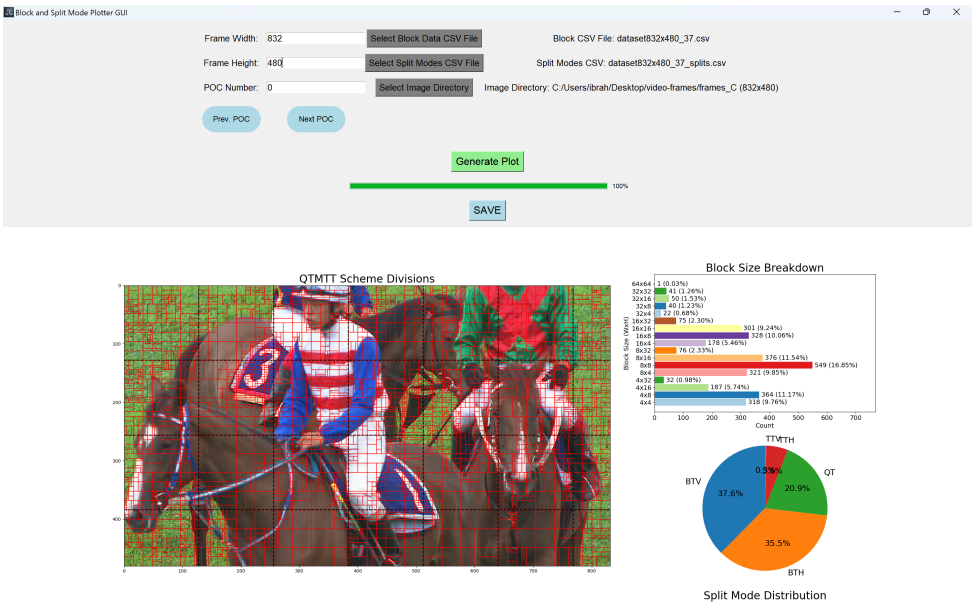
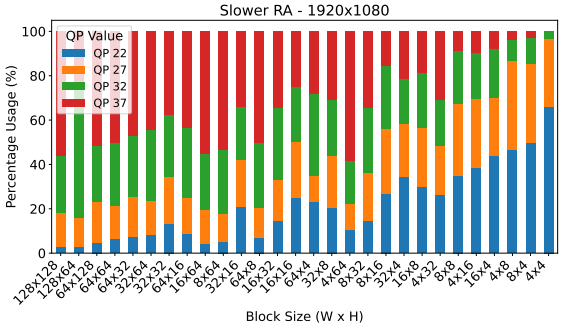


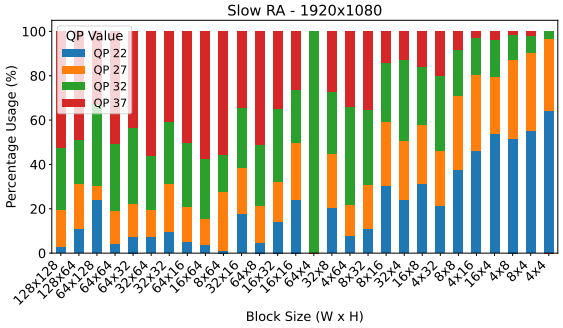
Figure 2.6 *AnalyzeITV application interface*

2.3.2 CU Sizes Breakdown Across Various Presets

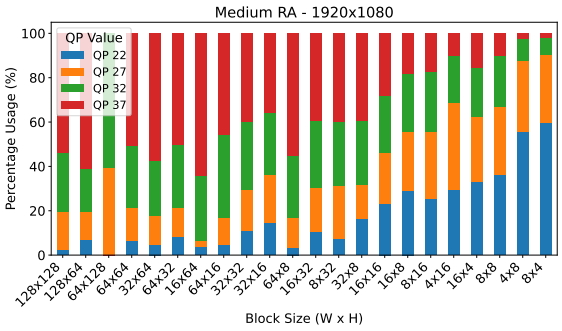
To examine the distribution of different CU sizes across presets, encoding was performed using five VVenC presets with four common QP values (22, 27, 32, and 37). This was done under both RA and AI configurations. For a precise comparison, the analysis focused on frame number 8 of the Cactus sequence from Class B, representing Full-HD resolution. Figure 2.7 illustrates the variation in the ratios of different block sizes for each QP in this frame. As shown in the figure, the QP value significantly impacts the distribution of block sizes. Lower QP values, such as 22, generally result in a higher proportion of smaller block sizes, as they preserve more detail and require finer granularity in encoding. Conversely, as the QP increases (to 27, 32, and especially 37), larger block sizes become more dominant. This is because higher QP values prioritize compression efficiency over detail retention, allowing larger blocks to represent coarser image regions, particularly in areas with less texture or motion. The distribution patterns observed in Figure 2.7 high-



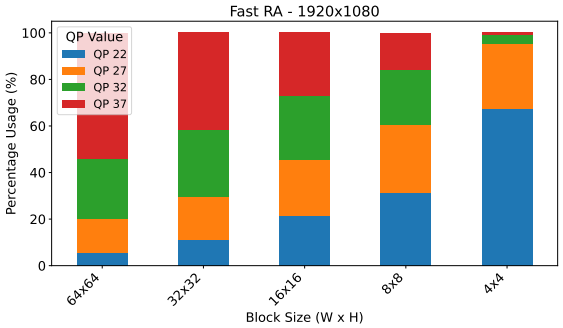
(a) *Slower*



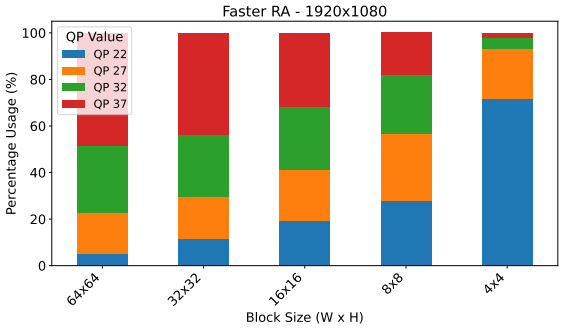
(b) *Slow*



(c) *Medium*



(d) *Fast*



(e) *Faster*

Figure 2.7 *CU size breakdown across the full-HD resolution*

light how QP adjustments influence the encoder's decisions regarding block size, balancing between compression and quality.

QP is crucial because it controls the balance between video quality and compression efficiency. Its consistent impact and flexibility across presets make it essential for effective encoding. Interestingly, although the VVenC presets (slower, slow, medium, fast, and faster) are designed to prioritize either encoding speed or quality, they show similar behavior when QP values are varied. Regardless of the preset, the general trend remains

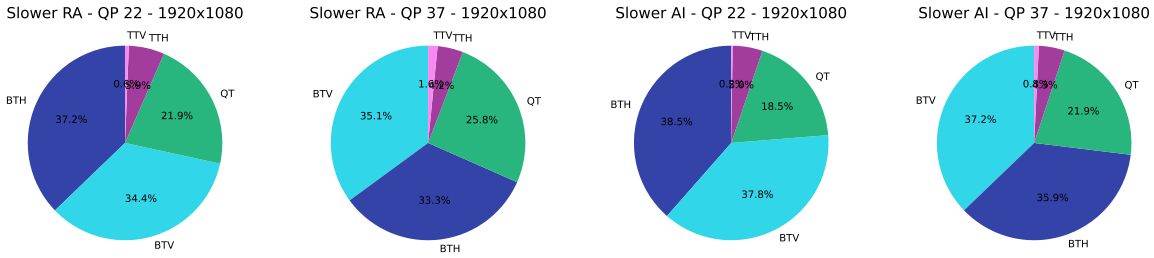
consistent: lower QP values lead to a higher frequency of smaller block sizes, while higher QP values result in larger block sizes. This suggests that, although each preset is optimized to favor speed or quality, the fundamental impact of QP on block size distribution remains uniform across all presets. However, the key difference between presets lies in the specific balance between quality and speed the slower presets take more time to refine block sizes and produce higher-quality results, whereas the faster presets process the frames more quickly by making less detailed adjustments. Despite these differences, the overall relationship between QP and block size distribution remains consistent across all presets.

2.3.3 Analysis of QTMTT Partitioning Across Different Encoding Presets

Building on the previous analysis, the block sizes in the VVenC encoder are determined by the QTMTT partitioning structure defined in the VVC standard. In VVenC, the depth and frequency of each split mode vary based on the selected preset and coding mode (RA or AI), as shown in Table 2.5. Figure 2.8 presents the distribution of split modes across presets at both lower QP (22) and higher QP (37) for RA and AI configurations. These results were extracted from the encoded frame number 8 of the Cactus sequence from Class B, which represents Full-HD resolution.

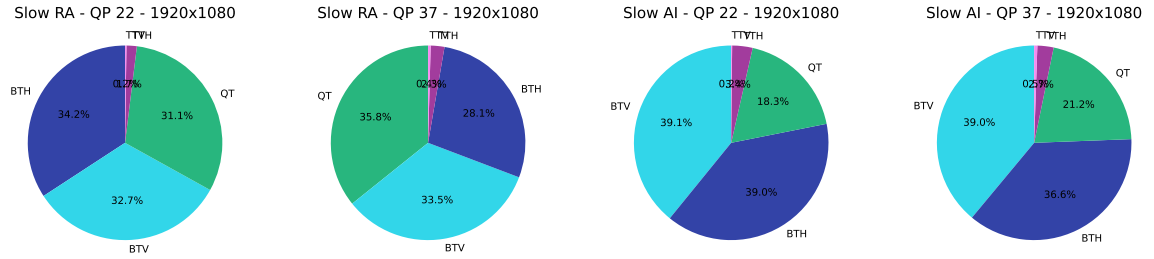
The slower preset shows that the most commonly used split mode is the BT in both horizontal and vertical directions, followed by QT, with TT being used the least. This behavior is typical for slower presets, which prioritize encoding quality and allow for deeper, more detailed partitioning. In AI mode, the BT split mode is more prevalent compared to RA mode, reflecting AI's need for finer partitioning to capture the details within individual frames due to the absence of temporal prediction. The QT split, restricted to the beginning of the CTU and not permitted after an MTT split, is primarily seen in the initial stages of splitting larger CU sizes, as it is used before more refined MTT partitioning begins.

The slow preset follows a similar trend, with BT being the dominant split mode in AI mode. In RA mode, the use of QT is more pronounced, as partitioning is less granular compared to AI, where more detailed BT splits are favored. In the medium preset, the RA configuration displays a higher usage of the QT split mode due to the reduced MTT depth, which limits more complex partitions. BT remains favored over TT, but with fewer



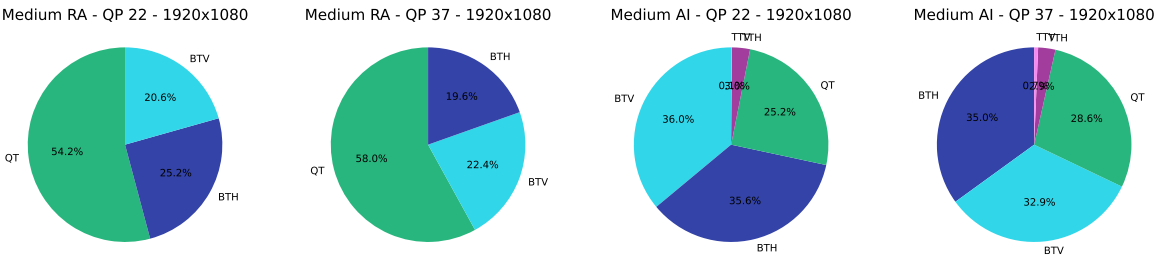
(a) *RA, QP=22* **(b)** *RA, QP=37* **(c)** *AI, QP=22* **(d)** *AI, QP=37*

(A) Slower



(e) *RA, QP=22* **(f)** *RA, QP=37* **(g)** *AI, QP=22* **(h)** *AI, QP=37*

(B) Slow



(i) *RA, QP=22* **(j)** *RA, QP=37* **(k)** *AI, QP=22* **(l)** *AI, QP=37*

(C) Medium

Figure 2.8 *Split modes breakdown in the full-HD resolution under RA and AI configurations with QP=22 and QP=37*

partitions, QT usage increases. When the QP value rises to 37, less partitioning occurs overall, leading to a greater reliance on QT to handle the larger block sizes. In AI mode, the medium preset exhibits a shift toward more MTT splits, as the MTT depth increases to 2, allowing for more BT splits to be employed compared to QT and TT.

In the fast and faster presets, the encoding process is streamlined to prioritize speed, allowing only the QT split mode. As a result, the distribution in Figure 2.8 shows 100%

QT usage for both QP 22 and QP 37, regardless of whether RA or AI is used. These presets sacrifice the flexibility of detailed partitioning in favor of faster encoding times, which simplifies the overall process but may lead to less precise encoding, particularly in complex scenes.

The results from Figure 2.8 illustrate how the VVenC encoder's presets adjust partitioning strategies to balance quality and speed. Slower presets focus on detailed partitioning using BT and TT splits, especially in AI mode, while faster presets reduce partitioning complexity by relying solely on QT splits, optimizing for encoding speed at the expense of finer detail retention. This behavior showcases the encoder's ability to adapt to various use cases, depending on the priority of quality or performance.

2.3.4 Impact of Split Mode Configurations on Encoding Performance

The impact of the different split modes on the partitioning process and encoding performance in the VVenC encoder was further evaluated by disabling specific modes and observing their effects on the BDBR and ETR. Tests were conducted using the default settings of the VVenC encoder with the slower preset under the RA configuration, applied to the CTC sequences. The slower preset was selected for its close alignment with VTM performance, while the RA configuration was chosen due to VVenC's optimization for this use case [9]. The results are shown in Table 2.6, which provides a comparison of BDBR and ETR when using only the QT mode, as well as combinations of QT with BT and QT with TT.

Table 2.6 *Impact of disabling QT, BT, and TT on BDBR and ETR*

Split Modes Configuration	BDBR (%)	ETR (%)
QT Only	81.57	86.40
QT + BT	4.36	42.53
QT + TT	13.01	65.73

When using only the QT mode, the BDBR increases significantly by 81.57%, indicating a significant loss in compression efficiency, as the lack of BT and TT modes limits the encoder's ability to adaptively partition blocks. However, this also results in a substantial ETR of 86.40%, meaning the encoding process becomes much faster. The trade-off here is clear: while the only QT configuration drastically reduces encoding time, the cost in terms

of increased bitrate and decreased quality is extremely high, making this configuration unsuitable for applications where quality is a priority. In the case of using both QT and BT, the BDBR is reduced to a much more manageable 4.36%, while the ETR remains at a substantial 42.53%. This suggests that the BT mode plays a crucial role in maintaining compression efficiency while still offering significant time savings. The relatively low increase in BDBR, combined with a meaningful reduction in encoding time, indicates that this configuration strikes a more favorable balance between speed and quality compared to using QT alone. Using QT and TT together results in a BDBR of 13.01% and an ETR of 65.73%. While this combination also achieves significant time savings, the increase in BDBR is noticeably higher than when using QT with BT. This highlights that the TT mode, while offering some flexibility in partitioning, does not provide the same level of efficiency as the BT mode in terms of balancing quality and encoding speed.

These findings demonstrate the critical role of split modes in balancing encoding time and compression efficiency. Disabling key split modes like QT or BT can significantly accelerate encoding but at the cost of increased bitrate and reduced quality. However, the insights from this analysis also point to the potential for optimizing the partitioning process in the VVenC encoder. By improving the decision-making process for selecting the appropriate split mode—rather than exhaustively testing all possible divisions—there is an opportunity to achieve significant time savings without sacrificing as much compression performance. This analysis serves as the foundation for the optimization strategies presented in this thesis. The primary objective is to minimize the time spent on partitioning decisions, particularly in the RA and AI configurations, by refining the partitioning process. The findings from this investigation have led to the development of two novel partitioning strategies designed to enhance the efficiency of the VVenC encoder while achieving a good balance between encoding time and video quality. Additionally, there are numerous other aspects of the VVenC encoder that could be further optimized, such as the intra-prediction tools, which will be explored in the subsequent section.

2.4 Achieving Real-Time Performance in VVenC

This section aims to assess VVenC's performance in real-time encoding scenarios, evaluating its speed and efficiency for time-sensitive applications. It explores how VVenC measures up against real-time requirements and examines key factors that enhance its performance, such as thread parallelism and tiles parallelism on optimized hardware.

2.4.1 Experimental Setup

In this test, VVenC version 1.7.0 was used to encode a selection of 100 videos featuring different scenes and different resolutions from a private dataset. Encoding was conducted under the random access configuration, where VVenC is particularly optimized [9]. To evaluate performance across quality levels, resolutions from 240p to 1080p were tested. Additionally, three hardware setups were used: a cluster node (CRN01), a Mac Mini (M1), and a Mac Studio (M2).

— **The cluster node (CRN01)** based on Dell PowerEdge R630, operating on x86 architecture, utilizes dual Intel Xeon E5-2650 v4 CPUs (24 cores in total), 384GB DDR4 ECC RAM, and SSD storage (2x 200GB for the system, 2x 800GB shared via GlusterFS).

— **The Mac Mini (M1)** equipped with an ARM-based Apple M1 chip with an 8-core CPU (4 performance cores and 4 efficiency cores), an 8-core GPU, and 16GB of unified memory.

— **The Mac Studio (M2)** also ARM-based, and is equipped with an M2 Ultra chip featuring a 24-core CPU (16 performance cores and 8 efficiency cores), a 60-core GPU, 64GB of unified memory.

2.4.2 Real-time Capabilities and Enhancement Through Thread Parallelism

The results of our encoding experiments with the VVenC encoder across different core configurations 4, 8, 16, and 24 cores—demonstrate varying levels of performance, especially when it comes to reach the target of real-time encoding. In this evaluation, thread parallelism is utilized to accelerate the encoding process by encoding frames in parallel. The objective is to determine how well VVenC performs in approaching real-time encoding and to observe the efficiency gains achieved by increasing the number of threads. By distributing frame encoding across multiple threads, VVenC aims to optimize processing time, allowing for faster encoding across different hardware setups.

Results of the cluster node (CRN01): The analysis of average Frames per Second (FPS) values across different core counts (4, 8, and 16) on the x86-based cluster node with the VVenC encoder provides insights into scalability and efficiency. Tests cover various presets and resolutions, targeting a framerate of 60 FPS. Generally, increasing core

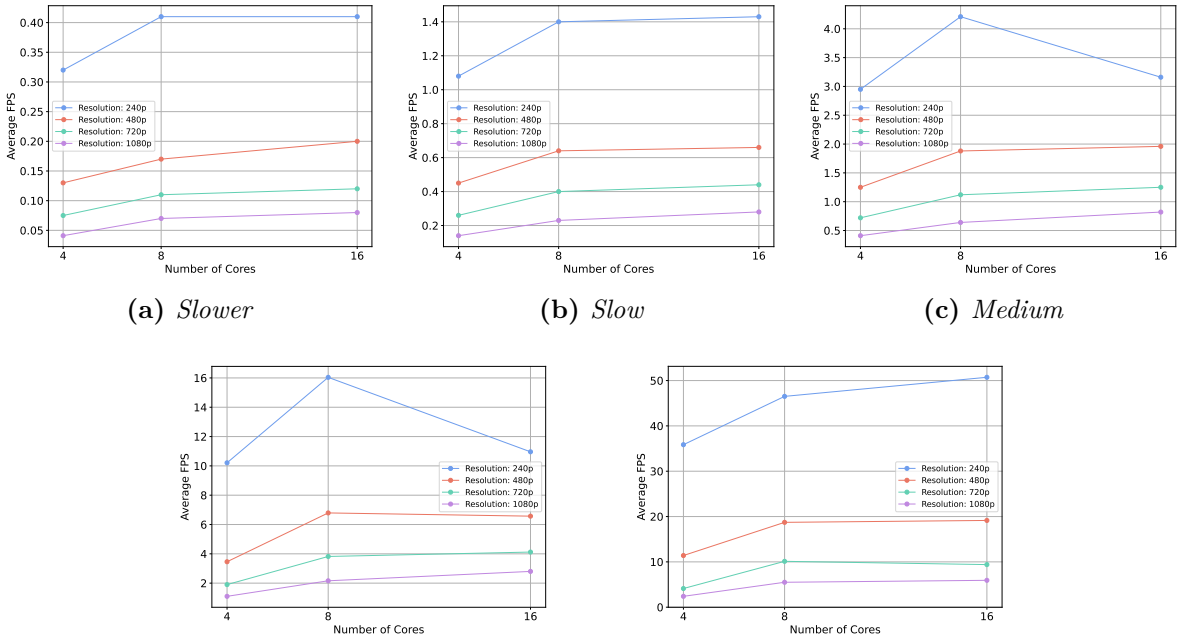
counts leads to higher FPS across all tested presets and resolutions, though the degree of improvement varies.

Table 2.7 *Average FPS for different presets and resolutions across core counts*

		Number of cores			Speedup Ratios	
Preset	**Resolution**	**4**	**8**	**16**	**8/4 Ratio**	**16/4 Ratio**
Slower	416×240	0.32	0.41	0.41	1.28	1.28
	832×480	0.13	0.17	0.2	1.31	1.54
	1280×720	0.075	0.11	0.12	1.47	1.6
	1920×1080	0.041	0.07	0.08	1.71	1.95
Slow	416×240	1.08	1.4	1.43	1.3	1.32
	832×480	0.45	0.64	0.66	1.42	1.47
	1280×720	0.26	0.4	0.44	1.54	1.69
	1920×1080	0.14	0.23	0.28	1.64	2.0
Medium	416×240	2.95	4.21	3.16	1.43	1.07
	832×480	1.25	1.88	1.96	1.5	1.57
	1280×720	0.72	1.12	1.25	1.56	1.74
	1920×1080	0.41	0.64	0.82	1.56	2.0
Fast	416×240	10.21	16.04	10.96	1.57	1.07
	832×480	3.46	6.79	6.57	1.96	1.9
	1280×720	1.9	3.82	4.12	2.01	2.17
	1920×1080	1.1	2.16	2.8	1.96	2.55
Faster	416×240	35.85	46.51	50.72	1.3	1.42
	832×480	11.4	18.72	19.14	1.64	1.68
	1280×720	4.12	10.12	9.4	2.46	2.28
	1920×1080	2.4	5.5	5.93	2.29	2.47

The results in Table 2.7 show that moving from 4 to 8 cores yields a noticeable increase in performance, with speedup ratios for the 8-core configuration over the 4-core configuration ranging between 1.28 and 2.46, depending on the resolution and preset. For instance, at 720p in the "Faster" preset, the FPS improves from 4.12 on 4 cores to 10.12 on 8 cores, resulting in a 2.46x speedup. This suggests that at higher resolutions and faster presets, the encoder benefits significantly from the addition of more cores due to the increased demand for computational resources. However, the benefits of scaling from 8 cores to 16 cores vary more noticeably by resolution and preset. For example, at 1080p on the "Fast" preset, moving from 8 to 16 cores provides a 1.29x speedup (from 2.16 to 2.8 FPS), indicating that the encoder's scalability diminishes slightly at higher core counts. Additionally, some resolutions, such as 240p on the "Faster" preset, show a minimal speedup (1.3x) when increasing from 4 to 8 cores, reflecting that lower resolutions may not fully utilize higher core counts due to their lighter processing load. Notably, the

"Faster" preset at 1080p yields a 2.47x speedup when increasing from 4 to 16 cores (from 2.4 to 5.93 FPS), which highlights that higher preset settings tend to benefit more from additional cores.



(a) *Slower* **(b)** *Slow* **(c)** *Medium*

(d) *Fast* **(e)** *Faster*

Figure 2.9 *Average FPS performance for each preset across different core counts and resolutions.*

Figure 2.9 shows the speedup of each VVenC preset with different core counts on the cluster node's x86 architecture. The results highlight VVenC's ability to make good use of multi-core setups, especially at higher resolutions and with more intensive presets. Performance improves as core count increases, but gains become smaller beyond 8 cores in some cases, especially at lower resolutions. This analysis suggests that while VVenC can effectively utilize multi-core processing, careful selection of core counts is essential depending on resolution and preset to maximize performance gains without unnecessary resource allocation. This study highlights potential areas for optimizing core usage, especially in lower-resolution encoding scenarios where the speedup is less pronounced.

Figure 2.10 compares the 8-core x86-based CRN01 and the 8-core ARM-based Mac Mini M1, highlighting the performance advantages of ARM, particularly in faster presets and lower resolutions. For the "Slower", the M1 achieves 0.149 FPS at 1080p, more than double the CRN01 value of 0.07 FPS, and 0.92 FPS at 240p, outperforming the CRN01 result of 0.41 FPS. With the "Slow", the M1 reaches 3.48 FPS at 240p compared to

1.4 FPS on the CRN01, indicating higher efficiency on ARM. In the "Medium", the M1 achieves 10.61 FPS at 240p, a 152% increase over the CRN01 result of 4.21 FPS. The gap increases in faster presets; with the "Fast" at 240p, the M1 records 43.46 FPS compared to 16.04 FPS on the CRN01. In the "Faster", the M1 scales up to 122.87 FPS at 240p, far surpassing the CRN01 result of 46.51 FPS. These results underscore the advantage of ARM in encoding efficiency, making the M1 highly suitable for high-throughput encoding tasks, particularly at lower resolutions and faster presets.

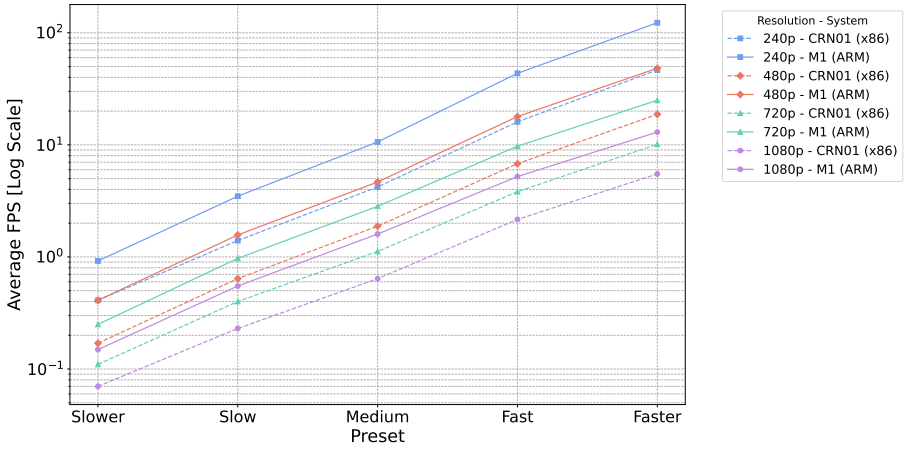
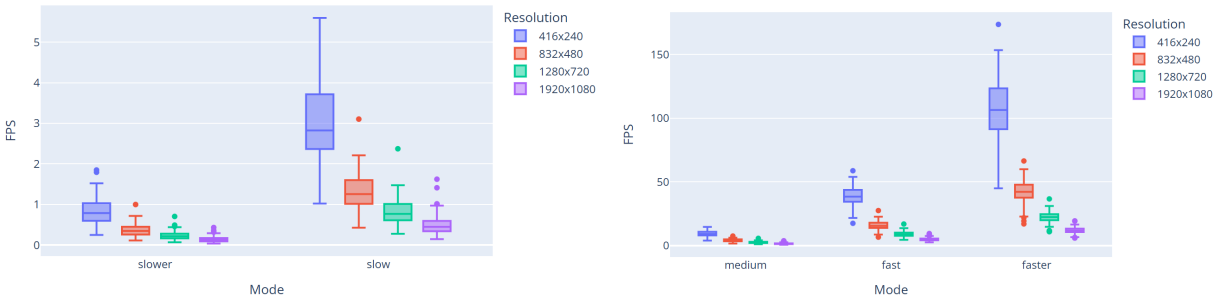


Figure 2.10 *Average FPS for ARM-based (M1) and x86-based (CRN01 cluster node) architectures across different presets and resolutions*

Results of the Mac Mini (M1) Vs the Mac Studio (M2) at 25 FPS: The 8-core setup of the Mac Mini (M1) illustrated in Figure 2.11, it is evident that this configuration struggles to achieve real-time speeds, particularly at higher resolutions and slower encoding presets. In "slower" and "slow" modes, the 8-core system consistently falls well below the 25 FPS mark across all resolutions, indicating that it is far from real-time encoding capabilities for high-quality video. Even in the "fast" and "faster" modes, the 8-core configuration only approaches real-time performance at lower resolutions like 416×240. For higher resolutions, especially at 1920×1080, the 8-core setup is significantly limited, with FPS often below 10, underscoring that this configuration is better suited for lower-resolution encoding tasks when speed is essential.

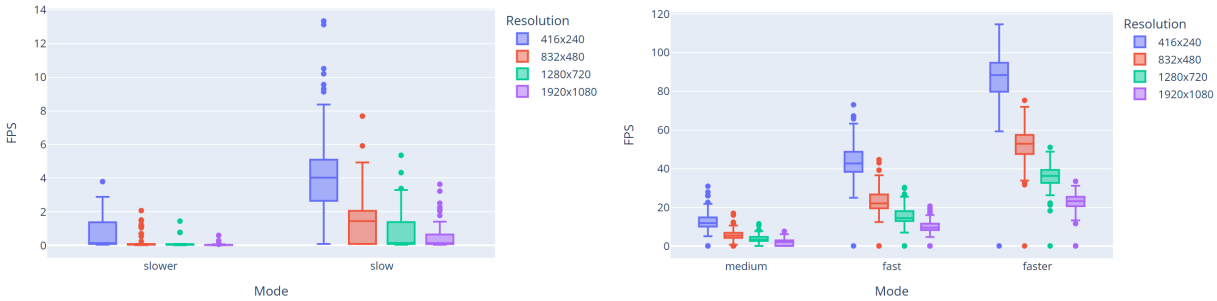
The 24-core configuration shown in Figure 2.12 delivers the highest performance, pushing the VVenC encoder closer to real-time encoding at multiple resolutions and modes. At lower resolutions such as 416×240, the "fast" and "faster" modes with 24 cores easily surpass the 25 FPS mark, offering ample headroom for real-time encoding and more



(a) *Slower and Slow* **(b)** *Medium, Fast, and Faster*

Figure 2.11 *Configuration: Mac Mini M1, 8-cores*

demanding use cases. This configuration also shows promising results for moderate resolutions (e.g., 1280×720) in faster encoding modes, achieving or exceeding the 25 FPS target. However, at the highest resolution of 1920×1080, even the 24-core setup struggles to consistently maintain real-time speeds in the "slower" and "slow" modes. This limitation suggests that although increasing core counts brings performance gains, certain high-resolution, high-quality encoding tasks with VVenC may face bottlenecks that are not fully alleviated by core scaling alone.



(a) *Slower and Slow* **(b)** *Medium, Fast, and Faster*

Figure 2.12 *Configuration: Mac Studio M2, 24-cores*

Overall, while increased core counts lead to clear improvements in FPS, achieving true real-time encoding remains challenging for high-quality and high-resolution settings with VVenC. The 8-core setup falls short of real-time capabilities in most cases, especially for higher resolutions. The 24-core configuration is closest to real-time, achieving or surpassing 25 FPS in lower resolutions and fast modes, yet it falls short for the most demanding settings. This suggests that while core scaling improves performance, additional optimizations, possibly in memory handling, the encoder's algorithms such as the partitioning block, are needed to achieve consistent real-time encoding across all resolutions

and quality settings.

Further insights into the evaluation of the VVenC encoder across a range of resolutions and architectures are available on our experimental results homepage [54], shown in Figure 2.13. This resource provides detailed performance metrics, including bitrate efficiency, FPS, quality, and encoding time across different video, resolutions, and hardware configurations. The interactive dashboard allows users to explore the data, comparing how VVenC presets perform under various conditions, enabling a deeper understanding of its adaptability and efficiency. The results can be accessed at: `https://vvenc-dash.fly.dev/`.

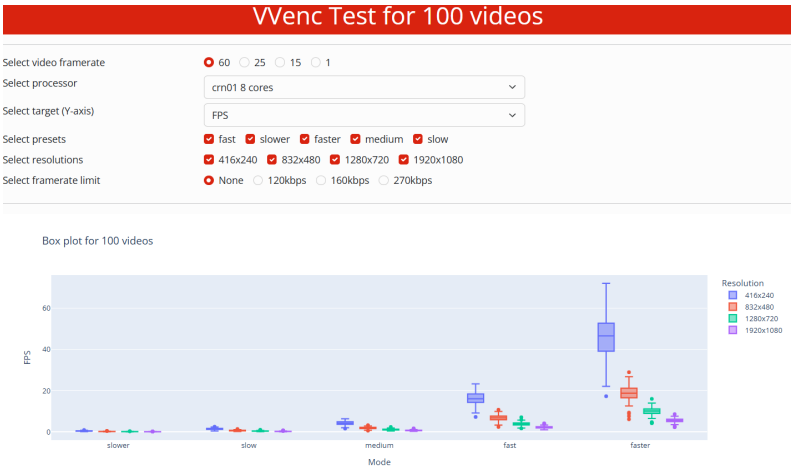


Figure 2.13 *VVenC performance evaluation results homepage*

2.4.3 Enhancements Through Tile Parallelism

As discussed in the previous section, increasing the number of cores does not scale effectively for encoding performance. To overcome this, tiles are introduced, allowing for independent encoding processes (see Chapter 1, section 1.4.1, page 16). By dividing the frame into tiles, VVenC can leverage thread parallelism more effectively, enabling each tile to be processed independently. Tests were conducted using tiles parallelism to assess its impact on encoding performance. Tiles parallelism is a technique that divides the video frame into independent regions, or "tiles", which can be processed simultaneously across multiple cores. This approach aims to enhance encoding efficiency by leveraging multi-core architectures more effectively, particularly for high-resolution videos where encoding time is a critical factor. Figure 2.14 illustrates the results of our tests evaluating performance in terms of FPS across different numbers of tiles and threads.

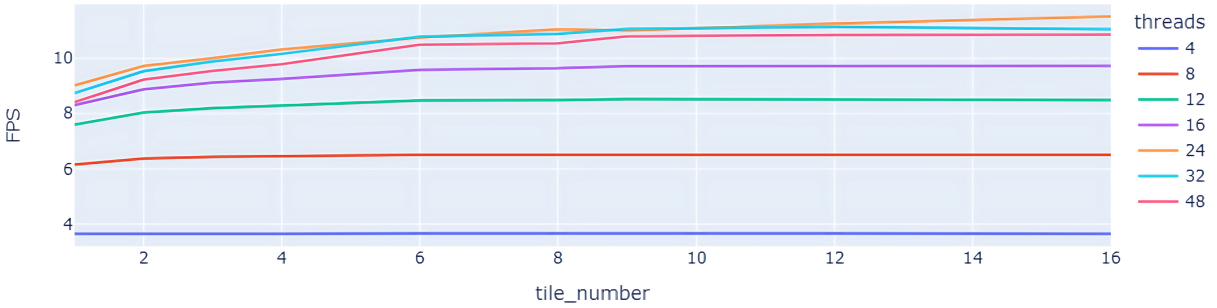


Figure 2.14 *FPS improvement based on the number of tiles and threads used for encoding*

The results showed that tiles parallelism can significantly improve FPS, especially at higher resolutions, where the workload can be distributed evenly across cores. However, slight variations in bitrate efficiency and quality were observed, due to tile boundaries that can introduce minor artifacts in some cases. Overall, tiles parallelism demonstrated a promising potential to bring the VVenC encoder closer to real-time performance, especially when used with hardware configurations featuring many cores. This experiment underscores the value of parallel processing techniques in optimizing encoding speed without a substantial sacrifice in quality.

2.5 Conclusion

This chapter presented a comprehensive analysis of the VVenC encoder, focusing on its optimized implementation of the VVC standard to achieve a balance between high visual quality and reduced encoding time. The evaluation of different presets revealed the substantial impact of partitioning schemes, block sizes, and split modes on overall encoder performance. The chapter also explored VVenC's ability to approach real-time encoding performance on different hardware architectures, including ARM and x86, across various presets and resolutions. These findings highlight key areas for improvement within the VVenC framework, particularly regarding the computational demands of the partitioning block. This analysis forms the basis for the following chapters, which will explore optimization strategies to reduce the complexity of this block in both AI and RA coding modes, ensuring the VVenC encoder remains effective for practical applications. The insights gained here will guide future research and development efforts to further refine video encoding processes for improved efficiency and quality.

Chapter 3

ML-Based Fast QTMTT Partitioning for VVenC Encoder in Intra Coding

3.1 Introduction

Machine Learning (ML) and Deep Learning (DL) have shown significant potential across various fields, particularly in optimizing video encoding processes due to their capability to manage complex, multidimensional problems. However, applying these techniques to real-time encoding remains challenging, especially when trying to balance speed, efficiency, and resource constraints. This chapter addresses this challenge by introducing a novel ML-based approach that integrates seamlessly into the Versatile Video Encoder (VVenC) encoder, offering a practical solution for real-time video applications.

This chapter focuses on reducing the computational complexity of the Versatile Video Coding (VVC) standard by optimizing the partitioning process within the VVenC encoder under the All Intra (AI) configuration. The primary objective is to minimize encoding time while maintaining a balance between coding efficiency and video quality. Specifically, we propose a fast Quadtree with Multi-Type Tree (QTMTT) partitioning strategy that employs a set of binary classifiers within Light Gradient Boosting Machine (LightGBM) models to predict the most likely split mode for each Coding Unit (CU). This approach bypasses the exhaustive Rate-Distortion Optimization (RDO) process, resulting in significant encoding speedup. The chapter begins with an overview of the partitioning process in VVC intra-coding and a review of existing methods for complexity reduction. We then detail our optimized CU partitioning method within the VVC framework, followed by an experimental analysis that demonstrates the effectiveness of our approach compared to existing solutions reported in the literature.

3.2 Related Works on CU Partitioning in Intra Coding

As mentioned earlier in Chapter 1, VVC, like most hybrid encoders, follows five major steps [7], as illustrated in Figure 3.1: partitioning, prediction, transform, quantization, and entropy coding. The partitioning step is responsible for dividing the frame into smaller blocks for encoding, and subsequently, the prediction step is performed, which may be intra or inter-frame. These are then fed into the transform and quantization processes, which are finally followed by entropy coding.

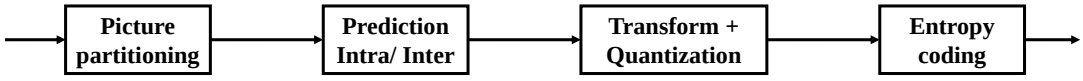


Figure 3.1 *The main steps of VVC encoder*

Several state-of-the-art approaches leverage heuristic methods, ML, and DL techniques to optimize the partitioning process.

3.2.1 Fast CU Partitioning Strategies for HEVC

In High Efficiency Video Coding (HEVC), the process of block partitioning is critical to achieving efficient video compression as aforementioned, particularly in the AI configuration. However, this process is computationally intensive, prompting researchers to explore various optimization strategies.

3.2.1.1 Heuristic Approaches for HEVC

Heuristic approaches have been widely adopted in early attempts to reduce the computational complexity of HEVC by accelerating the CU partitioning process. One common strategy involves utilizing Rate-Distortion (RD) cost metrics to guide faster partition decisions. For instance, Cho et al. [55] proposed a method that evaluates RD costs during CU encoding, featuring two key components: early splitting and early pruning tests. The early splitting test uses low-complexity RD cost estimates and a Bayes decision rule to determine whether a CU should be divided into sub-CUs, skipping the need for full RD cost calculations. Conversely, the early pruning test applies full RD costs to decide if a CU should remain unsplit, thus avoiding unnecessary partitioning. This method achieved a 50.2% reduction in encoding time with only a 0.6% increase in Bjøntegaard Delta Bit

Rate (BDBR). At a more aggressive level, it provided a 63.5% speedup with a 3.5% BDBR increase. Similarly, Zhang et al. [56] proposed a heuristic approach that uses gradients to estimate CU sizes quickly by analyzing average gradients in both the horizontal and vertical directions. Their method resulted in a 54% reduction in encoding time with a 0.7% increase in BDBR.

In another study, Min et al. [57] introduced a heuristic algorithm to optimize CU size in the AI configuration by leveraging a complexity score metric to predict the spatial complexity of blocks. This score is calculated using the l1 norm of luminance pixel differences. The global and local edge complexity scores are then compared against predefined thresholds to determine whether a block should be split or left unsplit. Their approach achieved a 52.3% reduction in computational complexity with a modest BDBR increase of 0.82%.

3.2.1.2 Machine Learning and Deep Learning Approaches for HEVC

ML techniques have been extensively explored to optimize the complexity of HEVC encoding, offering promising solutions for improving efficiency without compromising video quality.

One notable approach was presented by Amna et al. [48], who modeled the CU partitioning problem as a binary classification task using a fuzzy support vector machine (FSVM). By leveraging image features alongside Support Vector Machines (SVMs), they developed classification models that enhanced CU partitioning accuracy and optimized the RDO process. This method demonstrated both reduced complexity and improved rate-distortion performance.

Huang et al. [58] proposed a DL approach by developing a Convolutional Neural Network (CNN) model aimed at predicting CU depth decisions in the AI configuration. Their CNN model employed a two-level structure: one level used the current block and its borders as input, while the other focused solely on the current block. This dual-input strategy extracted multiple feature sets, which were then used to generate a partition map that consolidated split decisions for depth levels 0, 1, and 2. Additionally, a heuristic method based on the standard deviation was applied to skip Prediction Unit (PU) partitioning for 8×8 CUs. Implemented within the HEVC test Model (HM) 16.5 encoder, this method achieved a 66.7% reduction in computational complexity with a BDBR loss of only 1.71%.

Feng et al. [59] proposed another CNN-based approach that predicted a depth map to optimize Coding Tree Unit (CTU) partitioning. Rather than providing outputs for indi-

vidual depths, their method generated a matrix where each value represented the depth of an 8×8 block within 64×64 CTUs. This matrix, trained using a multi-scale L1 loss function, reflected local texture complexity, and was translated into split flags that predicted the partitioning structure. Tested under the HM 16.20 encoder in AI configuration, this approach reduced complexity by 65.6%, with a modest BDBR increase of 2.02%.

Xu et al. [60] explored a hierarchical approach to CU partitioning by predicting a partition map that represented the structure of entire CTUs across all three depth levels. For the AI configuration, a CNN with three normalization layers—each corresponding to a specific depth level—was used to make binary decisions about whether a CU should be split. In the Low Delay (LD)-P configuration, the method employed Long-Short Term Memory (LSTM) cells to capture CU depth correlations across frames. Their results demonstrated a 62% reduction in complexity for the AI configuration and 54.2% for the LD-P configuration, with a corresponding BDBR increase of 2.25% and 1.5%, respectively.

3.2.2 Fast CU Partitioning Strategies for VVC

VVC utilizes an advanced RDO process to make efficient decisions at various stages of encoding, such as partitioning block, intra/inter prediction, and transform selection. While this approach significantly enhances video quality and compression efficiency, it also increases the computational complexity of the encoder due to the exhaustive testing required to determine the optimal coding mode for each block.

A notable study by Tissier et al. [8] investigated strategies for reducing computational complexity across different modules of the VVC standard. Using the VVC Test Model (VTM) 3.0 under AI configuration, the authors encoded Common Test Conditions (CTC) sequences and recorded ground truth values for key parameters such as block partitioning, transform, and intra-prediction. In a subsequent test phase, the encoder directly applied these optimal choices, bypassing the RDO process. This study demonstrated that accurately predicting the optimal partitioning mode can reduce encoding time by up to 97%, followed by intra prediction, which offers a potential reduction of 65%, and Multiple Transform Selection (MTS), with a potential reduction of 55%. These findings highlight a significant opportunity to reduce computational complexity in VVC, particularly by focusing on optimizing the partitioning process. To address this challenge, several optimization techniques have been proposed, focusing on reducing the computational complexity of the encoding process through early termination algorithms and fast CU partitioning strategies. These approaches aim to skip unnecessary split modes, thereby accelerating

the encoding process with minimal impact on video quality.

3.2.2.1 Heuristic Approaches for VVC

Fu et al. [61] developed a fast CU partitioning algorithm that introduces two early skipping strategies. The first strategy targets the early termination of vertical splits, including both binary and ternary modes, while the second strategy focuses on bypassing horizontal ternary splits. Implemented in VTM 1.0, this algorithm yielded an average encoding time reduction of 45%, with a BDBR increase of approximately 1.02%. In a related effort, Cui et al. [62] proposed an early termination algorithm that leverages a directional gradient approach to determine the necessity of splitting a block either horizontally or vertically. This method is applicable to both binary and ternary split modes, offering a versatile solution for different scenarios. Integrated into the VTM 5.0 reference software, this algorithm achieved a significant 52% reduction in encoding time. However, this improvement came with a slight trade-off, resulting in a BDBR increase of 1.2%.

3.2.2.2 Machine Learning Approaches for VVC

The integration of ML into video coding has led to the development of several innovative algorithms designed to optimize the VVC encoding process.

In [63], Saldanha et al. introduced a configurable fast block partitioning method that leverages LightGBM models. These models directly predict CU splits, allowing the encoding process to bypass the RDO phase. Their approach, embedded in VTM 10.0, achieved a 54.20% reduction in encoding time, with a BDBR increase of 1.42%.

Parallel to these advancements, Chen et al. [64] employed SVM classifiers in developing a fast CU partitioning algorithm. By utilizing features such as entropy, texture contrast, and Haar-wavelet transformations, this method effectively selected division directions. Tested using the VTM 4.0, it resulted in a 51.23% reduction in encoding time, with a BDBR increase of 1.62%.

Expanding on these efforts, Quan et al. [65] presented a two-stage ML approach using Random Forest (RF) classifiers integrated into VTM 7.0. The first classifier directly identifies the best partition mode for both simple and complex CUs, while the second predicts partition termination for ambiguous cases. This method achieved a 57% reduction in encoding time, with a BDBR increase of 1.21%.

Additionally, Wu et al. [66] proposed a ML-based algorithm that accelerates VTM 10.0 by employing two SVM classifiers for the early termination of redundant partitions based on texture information. This approach resulted in a 63.16% reduction in encoding time, with a BDBR loss of 2.71%.

3.2.2.3 Deep Learning Approaches for VVC

A notable contribution in this area is the two-stage learning-based technique proposed by Tissier et al. [46], which has been implemented in VTM 10.2. This method employs a CNN to analyze the spatial features of an input block, generating a probability vector that predicts partitioning decisions along each 4×4 edge. Subsequently, a Decision Tree (DT) model utilizes these spatial features to predict the most probable splits for each block. By focusing the RDO process on the most likely configurations, this approach significantly reduces encoding time. This proposed technique demonstrated encoding time reductions of 47.4% and 70.4%, with corresponding BDBR increases of 0.79% and 2.49% for top-3 and top-2 configurations, respectively.

Another contribution by Li et al. [67], where they adopted a DL approach specifically designed to expedite the intra-mode encoding process in VVC by predicting CU partition modes. This method, employing a Multi-Stage Exit (MSE) CNN, processes CTUs to identify the most probable split modes based on CU size. Integrated into VTM 7.0, their approach yielded encoding time reductions ranging from 44.65% to 66.88%, with a corresponding BDBR increase between 1.322% and 3.188%.

Building upon the concept of efficient partitioning, Qiuwen et al. [68] proposed a fast CU partitioning decision method grounded in DenseNet architecture. This method, incorporated into VTM 10.0, involves segmenting CTUs into four 64×64 blocks, which are analyzed by a CNN to predict the likelihood of partitioning within each 4×4 block. The approach achieved a 53.98% reduction in encoding time, accompanied by a BDBR increase of 1.80%.

Similarly, Hoang et al. [69] introduced an Early-Terminated Hierarchical CNN model that effectively predicts the CU map, facilitating the early termination of the block partitioning process. Implemented in VTM 12.1, this strategy resulted in a 30.29% reduction in encoding time, with a BDBR increase of 1.39%.

In a related development, Zhang et al. [47] advanced the field with a Global Convolutional Network (GCN) module, which utilizes large kernel size convolutions to capture global CU information, thus optimizing partitioning predictions within the QTMTT struc-

ture. This method, integrated into VTM 10.0, demonstrated encoding time reductions between 51.06% and 61.15%, with BDBR increases ranging from 0.84% to 1.52%. Moreover, this technique was successfully applied to accelerate the VVenC encoder in its slower preset, further reducing encoding time while preserving average coding performance.

Table 3.1 *Comparison of leading state-of-the-art techniques for VVC optimization*

Approach	Reference Software	Technique	BDBR (%)	ETR (%)
Jing Cui [62]	VTM 5.0	Heuristic	1.23	51.01
Fu [61]	VTM 1.0	Heuristic	1.02	45.00
Tissier [46]	VTM 10.2	ML + DL	0.79	47.40
Saldanha [63]	VTM 10.0	ML (LightGBM)	1.42	54.20
Chen [64]	VTM 4.0	ML (SVM)	1.62	51.23
Quan [65]	VTM 7.0	ML (RF)	1.21	57.00
Guoqing [66]	VTM 10.0	ML (SVM)	2.71	63.16
Li [67]	VTM 7.0	DL	1.32	44.65
Qiuwen [68]	VTM 10.0	DL	1.80	53.98
Hoang [69]	VTM 12.1	DL	1.39	30.29
Zhang [47]	VTM 10.0	DL	1.52	61.15

A summary of these proposed approaches in the literature is presented in Table 3.1 for easier comparison. The results indicate that CNNs provide strong trade-offs between complexity reduction and performance. However, their hardware implementation can be challenging due to the high computational requirements and large number of parameters needed for high accuracy. Additionally, models such as RF and LightGBM, as demonstrated by Quan et al. [65] and Saldanha et al. [63], respectively, outperform other ML models in terms of complexity reduction when applied to VTM software.

3.3 Proposed Method for CU Partitioning in Intra Coding

In the VVC standard, the RDO process is used in the CU partitioning stage to improve coding efficiency. This process evaluates all possible split modes and selects the one with the lowest RD cost, as explained in Section 1.4.1 (page 16). While this process is essential for achieving optimal coding efficiency, it can be computationally expensive and time-consuming. To address this, our approach focuses on optimizing the CU partitioning process by selectively skipping certain evaluations within the QTMTT structure, which significantly reduces encoding time without compromising coding performance. As illus-

trated in Figure 3.2, our approach utilizes a set of binary classifiers to predict the most probable split mode for each CU. It integrates an optimized implementation of VVC, known as VVenC [9], with an efficient machine learning algorithm, LightGBM [10]. LightGBM is particularly well-suited for this application due to its relatively low computational complexity and ease of integration, especially when compared to other ML algorithms [10]. These characteristics make it an ideal choice for improving encoding speed while maintaining a balance between efficiency and performance.

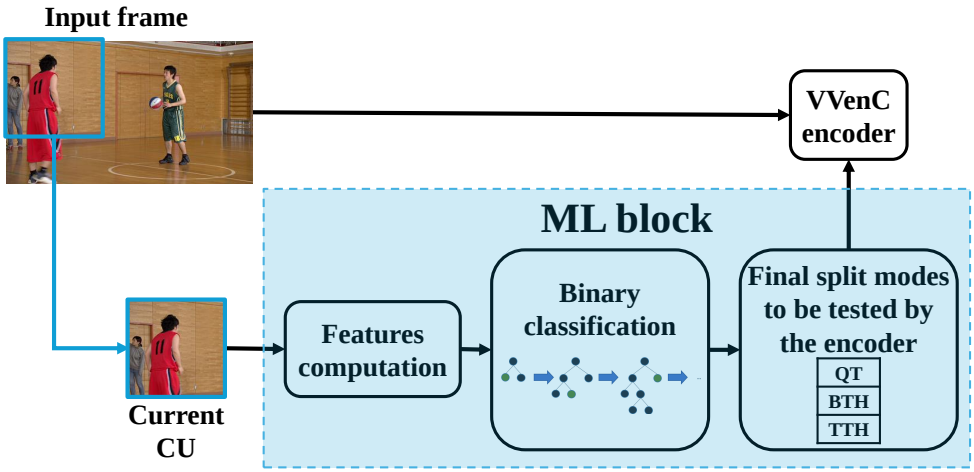


Figure 3.2 *ML-based approach for optimizing CU partitioning in the VVenC encoder in AI coding*

The main stages of our approach are illustrated in Figure 3.3 [70]. First, we created a set of video sequences from several image databases, including Div2k, 4K images, and Flickr2k. These sequences were encoded using a modified VVenC encoder, which extracted CU texture information and other features to generate a dataset of split modes corresponding to the six options in the QTMTT scheme: No Split (NS), Quadtree (QT), Binary Tree Horizontal (BTH), Binary Tree Vertical (BTV), Ternary Tree Horizontal (TTH), and Ternary Tree Vertical (TTV). During preprocessing, we balanced the dataset and used feature mutual information to select the most relevant features for training. We then trained LightGBM models to predict the most likely split mode for each CU by analyzing features from the pixel domain, which captured important information about the CU's texture complexity. Additionally, instead of using a single multi-class classifier, we divided the problem into five binary classifiers, each focused on a specific split mode. This approach enhances classification performance by allowing each classifier to specialize in a particular decision. Furthermore, these classifiers make decisions based on risk

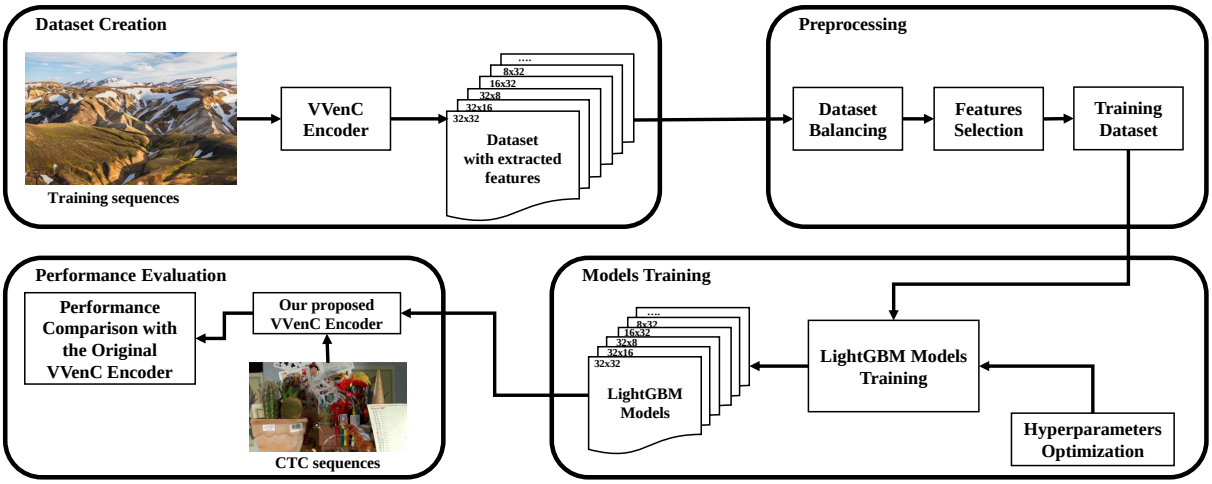


Figure 3.3 *Framework of the proposed approach for CU partitioning decision using LightGBM models integrated into the VVenC encoder*

intervals rather than conventional thresholds, which provide a good trade-off between quality and complexity reduction. To further optimize model performance, we used the Optuna tool [71] to fine-tune the hyperparameters before training the classifiers. The trained LightGBM models were then integrated into the VVenC encoder to predict the most likely split mode, allowing the encoder to bypass less probable modes. Finally, we evaluated the proposed encoder using the common test conditions [40] with 22 CTC video sequences at four different Quantization Parameter (QP) values: 22, 27, 32, and 37. The results are discussed in Section 3.4.

3.3.1 Fast CU Partition Decision Structure

The proposed solution aims to predict the optimal partitioning for each CU to achieve the best image quality while minimizing RD cost. To accomplish this, the number of split modes evaluated during the RDO process is reduced. Rather than assessing all possible split modes, the focus is placed on selecting the best option from six potential modes.

To solve this six-class classification problem, we adopted a strategy that uses five consecutive binary classifiers, as shown in Figure 3.4, rather than a single LightGBM multi-classifier. This approach offers more flexibility in the decision-making process and allows each classifier to be trained on specific features, improving classification performance.

The method explores eight specific CU sizes ($\boldsymbol{CS}$), where $\boldsymbol{CS} \in \{32\times32,\ 32\times16,\ 16\times32,\ 16\times16,\ 32\times8,\ 8\times32,\ 8\times16,\ 16\times8\}$. These sizes were selected because they are

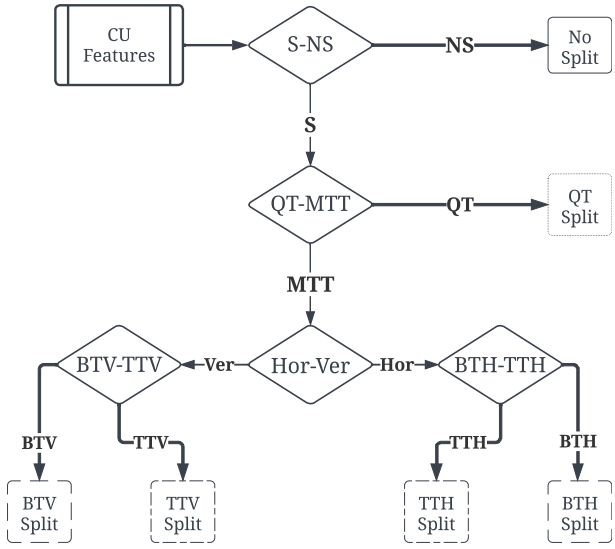


Figure 3.4 *The proposed algorithm for CU partition selection*

more commonly present in video frames compared to other sizes and allow for a wider range of split mode combinations as previously described in Chapter 2. By focusing on these frequently occurring sizes, it becomes possible to significantly reduce the number of split modes that need to be evaluated, thereby decreasing encoding time. For CU sizes outside these classes, the RDO process is applied directly within the encoder, as illustrated in Figure 3.5.

The five binary classifiers are used in a specific order, and each is named according to its function as follows:

- **S-NS:** This classifier determines whether a block should be divided into smaller sub-blocks or retained in its original size.
- **QT-MTT:** This classifier decides the type of tree structure used for dividing the CU, choosing between the QT structure and Multi-Type Tree (MTT) structures.
- **Hor-Ver:** This classifier establishes the direction of the split, determining whether it should be horizontal or vertical.
- **BTH-TTH:** This classifier evaluates whether a BTH or TTH split is the optimal choice for the given CU.
- **BTV-TTV:** This classifier assesses whether a BTV or TTV split is the best option for the CU.

The flowchart in Figure 3.5 illustrates the process of encoding a CU using our ML-based classification approach for partitioning. The process begins with encoding a CU

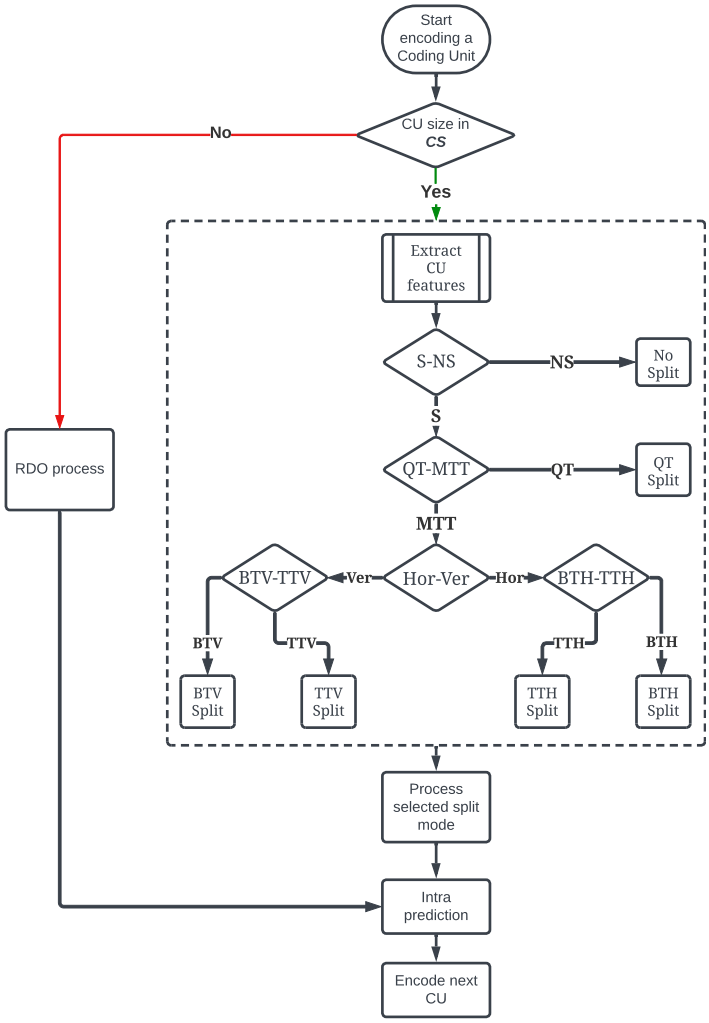


Figure 3.5 *Flowchart of the proposed approach*

and checking whether its size falls within the predefined CU size set (***CS***). If the CU size is not within these specified sizes, the process directly proceeds to the RDO process. However, if the CU size is part of the ***CS*** set, our solution calculates and extracts the CU features to feed the LightGBM classifiers. Then, the classification is made following the presented algorithm in Figure 3.4. Subsequently, the most probable split type is selected according to the prediction process detailed in subsection 3.3.4 to be processed by the RDO process. Otherwise, if the CU size is not in ***CS*** set, the encoder calls the RDO process to test all possible split types and select the one with the lowest RD cost. Finally, the CU partitioning process is terminated for the current CU and started for the next one. This approach combines ML-based classification with the traditional RDO process to speed up the CU partitioning process while maintaining good encoding performance.

3.3.2 Features Description and Computation

3.3.2.1 Features Description

To ensure robust feature extraction, several pixel-domain features of CUs and sub-CUs were selected. These features include the QP, with a focus on the four commonly used values (22, 27, 32, 37), as the coding modes are strongly influenced by the QP value. Additionally, key features such as horizontal and vertical gradients, block pixel mean, block variance, and texture complexity differences between sub-blocks were incorporated. The calculation of these features follows uses the pixel values denoted by I.

— **Quantization Parameter (QP):** $QP \in \{22, 27, 32, 37\}$

— **Vertical gradient, Δ_V:**

$$\Delta_V = \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} |I(i,j) - I(i,j+1)| \tag{3.1}$$

where W and H denote the width and height of the CU (or sub-CU), respectively, and $I(i,j)$ represents the pixel intensity at coordinates (i,j).

— **Horizontal gradient, Δ_H:**

$$\Delta_H = \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} |I(i,j) - I(i+1,j)| \tag{3.2}$$

— **CU Pixels Mean, μ_{CU}:**

$$\mu_{\text{CU}} = \frac{1}{W \cdot H} \sum_{i=1}^{W} \sum_{j=1}^{H} I(i,j) \tag{3.3}$$

where μ_{CU} is the mean intensity of the CU.

— **CU Pixels Variance, σ^2_{CU}:**

$$\sigma^2_{\text{CU}} = \frac{1}{W \cdot H} \sum_{i=1}^{W} \sum_{j=1}^{H} \left(I(i,j) - \mu_{\text{CU}}\right)^2 \tag{3.4}$$

— **Sub-CU's Texture Complexity, *SCTC*:**

$$SCTC_s = \frac{1}{k} \sum_{i=1}^{k} \left(\sigma_i^2 - \overline{\sigma_s^2}\right)^2 \tag{3.5}$$

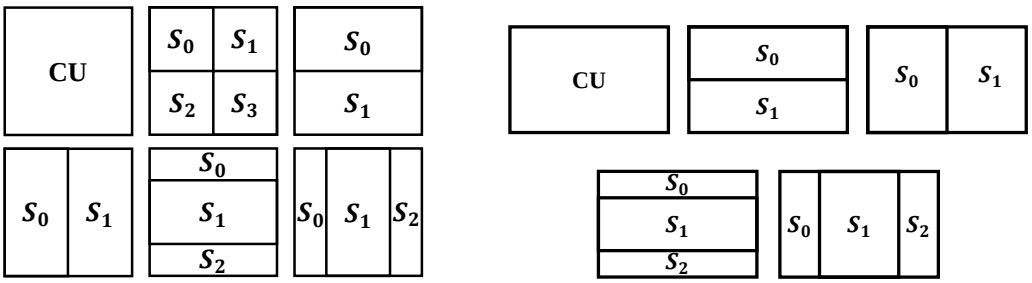
where σ_i^2 represents the variance of the i-th sub-CU, calculated using Equation (3.4). Meanwhile, $\overline{\sigma_s^2}$ refers to the average variance of the sub-CUs for a given split mode, denoted by s. The variable k indicates the number of sub-CU partitions, as detailed in Table 3.2:

Table 3.2 *Split modes with their corresponding sub-CU parts number*

Split mode (s)	QT	BTH	BTV	TTH	TTV
Number of sub-CUs parts (k)	4	2	2	3	3

3.3.2.2 Features Computation

Features computations are performed for the entire CU and each of its sub-CU components, designated as S_i. For a square-shaped CU, the features for the sub-CUs are computed as illustrated in Figure 3.6a. In the scenario of a rectangular CU, the feature calculations are conducted by assessing its respective sub-CUs, as depicted in Figure 3.6b.



(a) *Squared sub-CUs blocks* **(b)** *Rectangular sub-CUs blocks*

Figure 3.6 *CU blocks and sub-CU blocks for feature extraction*

3.3.3 Database and Models Training

3.3.3.1 Database Presentation

To enhance and improve the training process of the LightGBM models, a dataset was created for this purpose. As the study focuses on AI coding, no temporal relationship between frames is required. Therefore, five public picture databases were selected for the dataset, as described in [46] which are: 4K images, Div2k [72], JPEG-AI [73], HDR Google [74], and Flickr2k [75]. The images were then down-scaled to generate various resolutions, and finally, the pseudo-video sequences were created by combining the images

of the same resolution, as shown in Table 3.3 which presents the number of images per resolution.

Table 3.3 *Distribution of our dataset by image resolutions*

Resolution	**240p**	**480p**	**720p**	**1080p**	**4K**	**8k**	**Total**
Number of images	500	500	579	2557	654	418	5208

The modified VVenC encoder, configured with the "Slower" preset, was used to encode the pseudo-video sequences and extract CU features, with the corresponding split modes serving as labels.

3.3.3.2 LightGBM Models Training

LightGBM is an efficient gradient boosting framework designed for distributed and efficient learning, particularly useful in classification tasks. Developed by Microsoft, it stands out for its use of gradient-based one-sided sampling and exclusive feature bundling, which significantly reduce memory usage and increase the speed of computations [10]. LightGBM generally outperforms RF and DT models in binary classification tasks due to its superior prediction accuracy, faster training, and quicker prediction times [76]. It is particularly suited for handling large datasets and performs well with categorical features as it uses a histogram-based algorithm that bins continuous features into discrete bins, thus speeding up the training process. It supports both binary and multiclass classification, making it a versatile tool in the ML practitioner's toolkit. Its ability to handle sparse data and its lower memory usage compared to other boosting algorithms make it an attractive choice for many real-world classification problems [10].

LightGBM models have many hyper-parameters that can be adjusted to enhance their classification performance. These include the maximum number of leaves per tree, learning rate, tree depth, and the minimum required samples at a leaf node. Additionally, other hyper-parameters dictate the proportion of samples and features utilized in training each tree [10]. Tuning these hyper-parameters is critical for optimizing the performance of LightGBM models. Techniques such as grid search and Bayesian optimization are commonly employed to ascertain the optimal settings for these parameters. This study utilizes the Tree-structured Parzen Estimator (TPE) algorithm [77], implemented through the Optuna framework which is a versatile Python library that supports various optimization methods including random search and grid search [71]. Table 3.4 details the optimized hyperparameters values obtained.

Table 3.4 *Training parameters for our LightGBM models*

Parameter	Parameter Setting
Number of features	65
Number of estimators	30
Maximum Depth	15
Learning rate	0.05
Boosting type	GBDT

The LightGBM models were trained on a system equipped with an Intel Xeon(R) W-2145 CPU running at 3.70 GHz, an NVIDIA GeForce RTX 2080 Ti GPU, and 64 GB of RAM, using Ubuntu 20.04 LTS as the operating system. Each LightGBM classifier was trained independently for different CU sizes, specifically: $\boldsymbol{CS} \in \{32\times32,\ 32\times16,\ 16\times32,\ 16\times16,\ 32\times8,\ 8\times32,\ 8\times16,\ 16\times8\}$. After training, these models were integrated into the VVenC encoder following the flowchart illustrated in Figure 3.5, to predict the optimal split mode for CUs during the encoding process.

3.3.3.3 Performance Evaluation

To evaluate the effectiveness of our classifiers, we assessed their performance using standard ML metrics, including accuracy, precision, recall, and F1-score. These metrics are presented in detail in section 1.7.1 (page 33).

Table 3.5 *Accuracy and F1-score for each classifier*

Classifier	Metric	32×32	32×16	32×8	8×32	16×32	16×16	16×8	8×16
S-NS	Accuracy (%)	98.28	96.21	94.27	95.77	96.36	97.46	93.51	92.83
	F1-score (%)	97.29	96.12	93.95	94.85	96.14	97.22	93.18	92.41
QT-MTT	Accuracy (%)	86.65	-	-	-	-	82.24	-	-
	F1-score (%)	86.73	-	-	-	-	82.17	-	-
Hor-Ver	Accuracy (%)	87.60	90.21	84.17	83.23	93.16	87.35	77.51	79.83
	F1-score (%)	87.52	89.95	84.27	83.38	92.88	87.21	77.35	78.16
BTH-TTH	Accuracy (%)	84.55	87.21	-	85.64	93.16	87.76	-	74.36
	F1-score (%)	84.13	87.38	-	85.14	92.96	86.94	-	74.12
BTV-TTV	Accuracy (%)	85.18	85.67	81.62	-	72.73	82.68	79.56	-
	F1-score (%)	85.34	84.03	81.11	-	70.98	81.87	77.12	-

The results in Table 3.5 show the accuracy and F1-scores for each classifier across different CU sizes. The accuracy values are generally high, reflecting the robustness of the classifiers in predicting the CU split modes. For instance, the S-NS classifier achieves exceptionally high accuracy, with values reaching up to 98.28% for 32×32 blocks, indicating strong performance in handling larger block sizes. The F1-scores are similarly high,

with a peak of 97.29% for the same block size, which confirms both high precision and recall. When comparing the classifiers, the S-NS outperforms others, especially in larger CU sizes such as 32×32 and 16×16, where it consistently achieves accuracy above 95%. This suggests that S-NS is particularly effective in capturing the characteristics of larger CU blocks, which may have more distinguishable patterns, allowing for more confident predictions. In contrast, the QT-MTT classifier has comparatively lower accuracy and F1-scores, particularly for the 16×16 CU size, with values around 82%. The Hor-Ver classifier shows moderate accuracy across various CU sizes, maintaining values above 80% for most configurations but peaking at 93.16% for 16×32 CUs. This classifier appears better suited for CU dimensions where horizontal and vertical splits dominate, as indicated by its high accuracy for non-square partitions. Both the BTH-TTH and BTV-TTV classifiers display slightly lower accuracy in specific cases, such as smaller CU sizes, with BTH-TTH reaching 87.76% for 16×16 and BTV-TTV reaching 85.67% for 32×16. However, these classifiers still perform well overall, achieving over 85% accuracy in several cases.

Compared to the LightGBM classifiers proposed by Saldanha et al. [63], our models demonstrate a clear advantage, with many classifiers achieving higher accuracy across multiple CU sizes. Notably, several classifiers in our approach exceed 90% accuracy, with peak performances near 98%, suggesting a considerable improvement in classification reliability. This level of accuracy is especially important for reducing encoding time without compromising quality, as accurate split predictions allow the model to reduce the load of the exhaustive RDO checks, significantly speeding up the encoding process.

3.3.4 Prediction Process and Decision Making

Misclassification is a common problem in classification tasks, where classifiers may incorrectly assign instances to the wrong class [78]. This problem often arises from the use of default thresholds for classification decisions, which may not be optimal for every dataset or task, potentially leading to subpar performance. To enhance classification accuracy, we utilized risk intervals to refine decision-making. A Risk Interval is defined as a range of predicted probabilities where making a confident decision is challenging due to high uncertainty. In our approach, binary classifiers were trained with two potential outputs: class A and class B, corresponding to the pairs {(S, NS), (QT, MTT), (Hor, Ver), (BTH, TTH), (BTV, TTV)}. The classification decision is made based on the output probability of the binary classifier. If the probability exceeds a specified threshold, Δ_{Max}, the instance is assigned to class A. Conversely, if the probability falls below another threshold, Δ_{Min},

it is assigned to class B. If the probability lies within the risk interval, i.e., between Δ_{Min} and Δ_{Max}, further testing is performed to determine the appropriate class, as illustrated in Figure 3.7. Here, Δ_{Min} and Δ_{Max} represent the minimum and maximum boundaries for class assignments A and B, respectively.

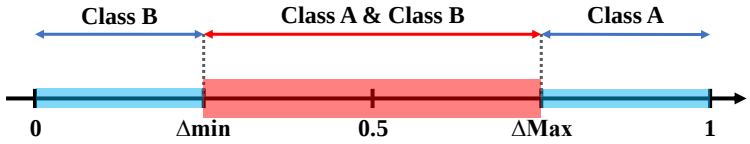
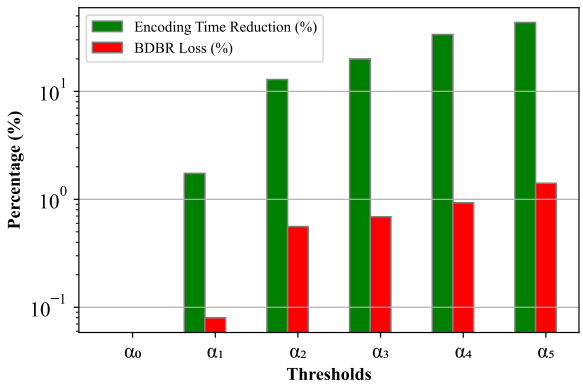
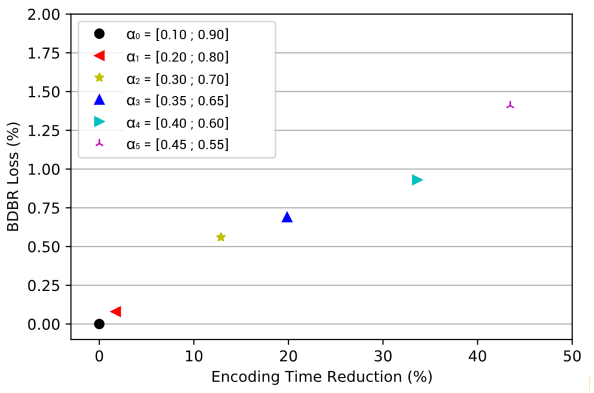


Figure 3.7 *Risk interval in ML*

The concept of risk intervals provides a more nuanced approach to evaluating classifier performance by considering the level of uncertainty associated with classification outcomes. Instead of relying on fixed thresholds, which may not be ideal for every scenario, risk intervals allow for a more flexible and context-sensitive decision-making process. This method acknowledges that the output probabilities of classifiers are not always definitive and may fall within a range of uncertainty. By incorporating risk intervals, the classification process becomes more adaptable to cases where probabilities are uncertain. Risk intervals help explore the trade-off between quality and complexity reduction, as shown in Figure 3.8.



(a) *Trade-offs in risk intervals across various threshold percentages*



(b) *Trade-offs in risk intervals showing BDBR loss and ETR across thresholds*

Figure 3.8 *Comparison of risk interval trade-offs in terms of ETR and BDBR loss for video sequences encoded with different classification thresholds*

A larger risk interval indicates a more cautious approach, leading to minimal quality degradation, but at the cost of achieving less complexity reduction. Further analysis determined that the risk interval boundaries, denoted as Δ_{min} and Δ_{Max}, need to be optimized

for each classifier. To establish the most effective boundaries, a video sequence was created by concatenating 10 images, each with a resolution of 832×480. This video sequence was then encoded using the VVenC encoder with the slower preset in AI configuration mode. The encoding process was repeated six times, testing six predefined risk intervals (α_0, α_1, α_2, α_3, α_4, α_5) during each run. The results were compared against the RD performance metrics of the VVenC encoder using the slower preset in AI coding mode, which served as the reference benchmark.

The predefined risk interval values are defined as follows:

— $\alpha_0 = [0.10 ; 0.90]$
— $\alpha_1 = [0.20 ; 0.80]$
— $\alpha_2 = [0.30 ; 0.70]$
— $\alpha_3 = [0.35 ; 0.65]$
— $\alpha_4 = [0.40 ; 0.60]$
— $\alpha_5 = [0.45 ; 0.55]$

The effectiveness of these risk intervals was evaluated using the BDBR, defined in Equation 1.15 (page 29), and the Encoding Time Reduction (ETR) calculated using Equation 1.17 (page 30). The trade-offs between time-saving and coding efficiency are illustrated in Figure 3.8a. This figure demonstrates that reducing the risk interval size, from α_0 to α_5, leads to greater time savings but also results in an increased BDBR loss. This illustrates how the size of the risk interval affects classifier performance.

This analysis revealed that the risk intervals α_2 and α_5, with their corresponding boundaries Δ_{min} and Δ_{Max}, offered the optimal trade-offs between encoding efficiency and time-saving.

— Larger Risk Interval : $\alpha_2 = [0.30 ; 0.70]$
— Smaller Risk Interval : $\alpha_5 = [0.45 ; 0.55]$

Table 3.6 *Risk intervals for each classifier*

Classifier	32×32	32×16	32×8	8×32	16×32	16×16	16×8	8×16
S-NS	α_5	α_5	α_5	α_5	α_5	α_5	α_5	α_5
QT-MTT	α_5	-	-	-	-	α_5	-	-
Hor-Ver	α_5	α_2	α_2	α_2	α_2	α_2	α_2	α_2
BTH-TTH	α_2	α_2	-	α_2	α_2	α_5	-	α_2
BTV-TTV	α_2	α_2	α_2	-	α_2	α_2	α_2	-

Table 3.6 presents the different sets of risk interval values for each classifier. Classifiers with lower accuracy tend to have higher rates of false positives and false negatives, result-

ing in increased uncertainty in their predictions. Consequently, these classifiers require larger risk intervals (α_2) to handle this uncertainty. In contrast, classifiers with higher accuracy can utilize smaller risk intervals (α_5) due to their reduced prediction uncertainty.

3.4 Experimental Results

3.4.1 Implementation Details and Evaluation Environment

To assess the effectiveness of our proposed approach, we evaluated it on a distinct set of video sequences, different from those used for training. This evaluation is crucial for understanding how well the model generalizes to new data, particularly in video coding where the characteristics of video content can vary significantly. For this purpose, we utilized the 22 CTC video sequences from the CTC test sequences [40]. These sequences were encoded using the VVenC v1.3.1 encoder [52], specifically the Full Feature encoder (***vvencFFapp***), in AI coding mode. The encoding process was carried out across the five predefined presets and for four QP values: 22, 27, 32, and 37.

For performance assessment, we used the BDBR metric [79] to quantify the bitrate efficiency improvement of our method compared to a reference. The BDBR calculation is based on Equation 1.15 (page 29), with full details provided in Section 1.6.1. Additionally, we analyzed encoding time reduction (ETR) to evaluate time savings, calculated using Equation 1.17 (page 30) and further explained in Section 1.6.2.

3.4.2 Performance Evaluation of the Proposed VVenC Encoder Presets: Slower, Slow, Medium, Fast and Faster

To evaluate our approach, we initially implemented classifiers to predict the split mode for the four CU sizes from the set $\boldsymbol{CS} \in \{32\times32, 32\times16, 16\times32, 16\times16\}$, as introduced in our previous work [78]. We subsequently expanded the set of CU sizes in section 3.4.2.1 to $\boldsymbol{CS} \in \{32\times32, 32\times16, 16\times32, 16\times16, 32\times8, 8\times32, 8\times16, 16\times8\}$, allowing for a more comprehensive evaluation by the classifiers [70]. This section presents insights using the slower preset as the baseline for comparison.

3.4.2.1 Performance Evaluation Applied on Four CU Sizes Using the Slower Preset

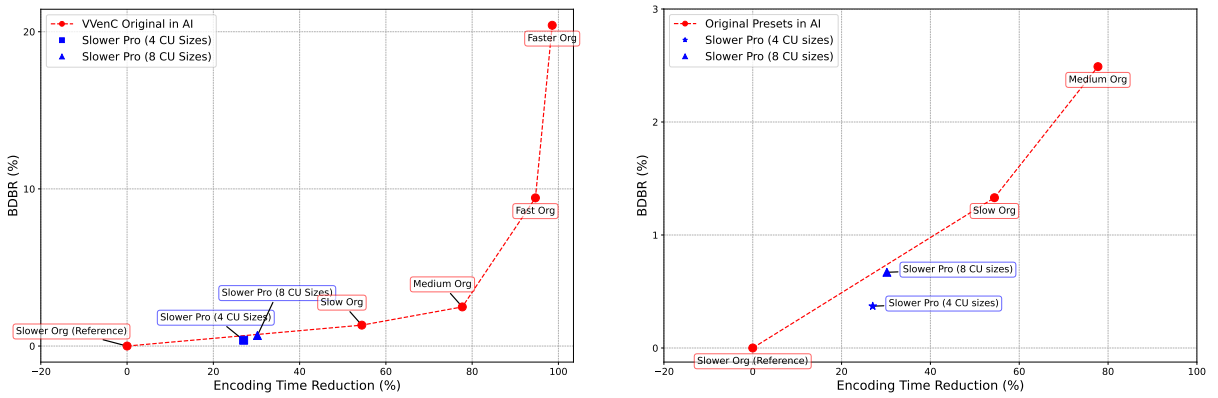
Table 3.7 provides a comparison of the coding performance between our proposed algorithm and the VVenC anchor encoder, using the "Slower" preset as the benchmark. In our approach, we applied a reduced set of CU sizes, restricting the CU size set ($\boldsymbol{CS}$) to four specific sizes as follows: $\boldsymbol{CS} \in \{32\times32, 32\times16, 16\times32, 16\times16\}$ [78].

Table 3.7 *Performance evaluation of the proposed solution using four binary classifiers*

Class	Video	PSNR (dB)	BDBR (%)	ΔETR (%)
A1	Campfire	-0.02	0.11	23.23
	Tango2	-0.01	0.07	22.73
	FoodMarket4	-0.02	0.71	29.29
	Average	**-0.02**	**0.30**	**25.08**
A2	DaylightRoad2	-0.02	0.12	25.05
	CatRobot1*	-0.03	0.08	22.27
	ParkRunning3	-0.03	0.48	25.77
	Average	**-0.03**	**0.23**	**24.36**
B	MarketPlace	-0.01	0.27	26.87
	RitualDance	-0.03	0.68	27.58
	BQTerrace	-0.02	0.39	29.88
	BasketBallDrive	-0.02	0.44	32.25
	Cactus	-0.01	0.15	25.49
	Average	**-0.02**	**0.39**	**28.41**
C	PartyScene	-0.04	0.04	25.72
	BQMall	-0.02	0.34	30.30
	BasketBallDrill	-0.01	0.21	24.40
	RaceHorsesC	-0.01	0.13	23.81
	Average	**-0.02**	**0.18**	**26.06**
D	BQSquare	-0.01	0.09	28.29
	BlowingBubbles	-0.01	0.08	25.69
	BasketBallPass*	-0.06	0.88	33.48
	RaceHorsesD	-0.01	0.17	25.55
	Average	**-0.02**	**0.31**	**28.25**
E	FourPeople	-0.02	0.30	29.68
	Johnny	-0.01	0.35	27.34
	KristenAndSara	-0.01	0.26	25.98
	Average	**-0.01**	**0.30**	**27.67**
F	ArenaOfValor	-0.11	1.25	31.06
	SlideEditing	-0.09	0.61	30.24
	BasketBallDrillText	-0.01	0.20	26.03
	SlideShow	-0.13	1.47	29.77
	Average	**-0.08**	**0.88**	**29.28**
Total Average		**-0.02**	**0.37**	**27.02**

The results in Table 3.7 demonstrate a consistent reduction in encoding time (averaging around 27%) across all video classes, while the average increase in BDBR remains minimal (around 0.37%), indicating that the proposed solution achieves a good speed-up with a negligible loss in coding efficiency. For example, Class A1 videos (*Campfire*, *Tango2*, and *FoodMarket4*) show an average BDBR increase of 0.30% with a corresponding 25.08% reduction in encoding time. Similarly, Class F videos, which are typically more complex, exhibit a higher BDBR increase (0.88%) but still achieve an impressive 29.28% reduction in encoding time. These results demonstrate the robustness of our optimization, providing substantial speed improvements without severely impacting video quality, as indicated by the small Peak Signal-to-Noise Ratio (PSNR) changes.

To visualize the performance gains, Figure 3.9a illustrates the relationship between BDBR increase and encoding time reduction for different encoder presets. It also compares our proposed method with the standard VVenC encoder presets, including "Slower", "Slow", "Medium", "Fast", and "Faster". The proposed slower using the four CU sizes (represented by a blue square in Figure 3.9a) strikes a well-balanced trade-off between encoding speed and bitrate efficiency. Compared to the reference "Slower" preset, our method achieves a significant encoding time reduction of around 27% with only a minor increase in BDBR.



(a) *Proposed slower (4 CU sizes) and (8 CU sizes) vs. VVenC original presets*

(b) *Zoom on the proposed slower (4 CU sizes) and (8 CU sizes)*

Figure 3.9 *Comparison of the proposed slower configurations with VVenC original presets, showing ΔETR(%) and BDBR(%) trade-offs*

When extending the CU size set to include eight sizes, an additional encoding time reduction of approximately 31% was observed, with a slight increase in bitrate, as shown in Figure 3.9b. This figure illustrates the trade-offs achieved using the two ***CS*** sets on

the slower preset. The following section 3.4.2.2 provides a detailed analysis of the results obtained with the extended ***CS*** set, including the outcomes for the other presets as well. The results in Table 3.7 and Figure 3.9b indicate that the proposed CU partitioning strategy not only enhances encoding speed but also maintains competitive rate-distortion performance.

3.4.2.2 Performance Evaluation of the Proposed VVenC Encoder Presets: Slower, Slow, Medium, Fast, and Faster

To evaluate our proposed approach, we encoded the 22 CTC sequences using the five presets and compared the results to those obtained with the Slower preset of the VVenC anchor encoder, which served as the reference. The experimental results presented in Table 3.8 show that the proposed Slower preset achieves an average encoding time

Table 3.8 *Trade-offs between ETR(%) and encoding efficiency BDBR(%): results of the proposed solution at The 5 presets compared to the slower original*

Preset	Video	Slower		Slow		Medium		Fast		Faster	
		BDBR (%)	ΔETR (%)	BDBR (%)	ΔETR (%)	BDBR (%)	ΔETR (%)	BDBR (%)	ΔETR (%)	BDBR (%)	ΔETR (%)
A1	Campfire	0.49	32.56	1.71	79.94	2.56	81.15	9.88	95.68	16.06	98.37
	Tango2	-0.40	30.76	0.76	77.61	1.07	68.94	8.30	96.15	21.56	97.58
	FoodMarket4	0.23	32.15	1.66	69.96	1.86	69.40	6.51	94.89	16.55	97.89
	Average	**0.11**	**31.82**	**1.38**	**75.84**	**1.83**	**73.16**	**8.23**	**95.57**	**18.06**	**97.95**
A2	DaylightRoad2	0.94	20.61	2.33	67.90	3.57	79.57	14.74	96.88	26.69	98.95
	CatRobot1	0.50	18.21	2.28	62.54	3.31	75.36	11.36	95.51	24.05	98.45
	ParkRunning3	0.61	10.88	1.72	56.81	2.17	74.53	8.10	94.75	13.49	98.48
	Average	**0.68**	**16.57**	**2.11**	**62.42**	**3.02**	**76.49**	**11.40**	**95.71**	**21.41**	**98.63**
B	MarketPlace	0.45	34.31	1.63	64.70	2.08	85.77	7.35	98.97	14.90	99.52
	RitualDance	0.94	38.07	2.19	65.75	3.11	87.83	8.62	98.88	17.59	99.28
	BQTerrace	1.32	33.92	2.95	68.56	4.36	89.39	11.04	98.44	22.63	99.42
	BasketBallDrive	0.49	33.25	1.99	67.92	3.39	87.74	12.08	99.62	27.52	99.58
	Cactus	0.73	30.62	1.77	66.43	2.93	88.22	9.90	98.79	19.85	99.50
	Average	**0.79**	**34.03**	**2.11**	**66.67**	**3.17**	**87.79**	**9.80**	**98.94**	**20.50**	**99.46**
C	PartyScene	0.41	33.88	1.38	68.25	2.53	89.60	8.05	97.29	15.41	99.56
	BQMall	0.91	27.94	2.20	66.40	3.75	88.56	11.51	97.70	22.95	99.70
	BasketBallDrill	1.36	23.20	3.65	63.24	5.63	87.39	12.00	98.28	30.61	99.61
	RaceHorsesC	0.51	26.56	1.30	65.91	2.41	88.55	7.96	97.76	15.83	99.56
	Average	**0.80**	**27.90**	**2.13**	**65.95**	**3.58**	**88.53**	**9.88**	**97.76**	**21.20**	**99.61**
D	BQSquare	0.37	30.30	1.45	67.37	2.69	88.98	8.80	97.88	16.02	99.59
	BlowingBubbles	0.47	37.37	1.11	68.34	2.26	87.85	7.29	96.96	15.66	99.51
	BasketBallPass	0.76	26.63	1.78	67.20	3.30	86.99	10.08	97.46	20.76	99.44
	RaceHorsesD	0.48	32.85	1.15	67.83	2.32	87.23	7.85	97.18	17.01	99.44
	Average	**0.52**	**31.79**	**1.37**	**67.69**	**2.64**	**87.76**	**8.51**	**97.37**	**17.36**	**99.50**
E	FourPeople	1.32	41.84	2.58	64.86	3.99	81.87	10.93	95.67	22.96	98.68
	Johnny	1.11	39.34	2.41	66.22	4.05	81.14	11.71	95.21	25.40	98.14
	KristenAndSara	0.95	36.29	2.17	61.48	3.50	80.10	10.82	95.05	24.21	98.21
	Average	**1.13**	**39.16**	**2.39**	**64.19**	**3.85**	**81.04**	**11.15**	**95.31**	**24.19**	**98.34**
Classes average		**0.67**	**30.21**	**1.91**	**67.12**	**3.01**	**82.46**	**9.83**	**96.78**	**20.45**	**98.91**

reduction of 30.21% with a minimal increase in BDBR loss of 0.67%. This level of BDBR loss is considered negligible, indicating that the observed speedup in encoding time is achieved without significantly compromising video quality. Among the proposed presets, the Slower preset demonstrated the maximum encoding time reduction of 41.84% for the *FourPeople* sequence, while the *ParkRunning3* sequence exhibited the minimum reduction at 10.88%. While the original "Slow" preset achieves a 54% time reduction with a 1.33% increase in bitrate, our approach enables this preset to reach a time-saving of up to 67.12%, with an average BDBR loss of only 1.91% compared to the reference. This demonstrates a significant optimization for runtime efficiency.

For the "Medium" preset, a time-saving of 77% is achieved with a bitrate increase of 2.49%. However, when our approach is applied, this preset shows a substantial encoding time reduction of approximately 82.46%, with a BDBR loss of 3.01%. This result offers a balanced trade-off between encoding time and compression efficiency.

The "Fast" and "Faster" presets deliver substantial encoding time reductions but at the cost of higher BDBR losses. Specifically, the "Fast" preset reduced encoding time by 96.78% with a BDBR loss of 9.83%, while the "Faster" preset achieved a 98.91% reduction in encoding time, accompanied by a BDBR increase of 20.45%.

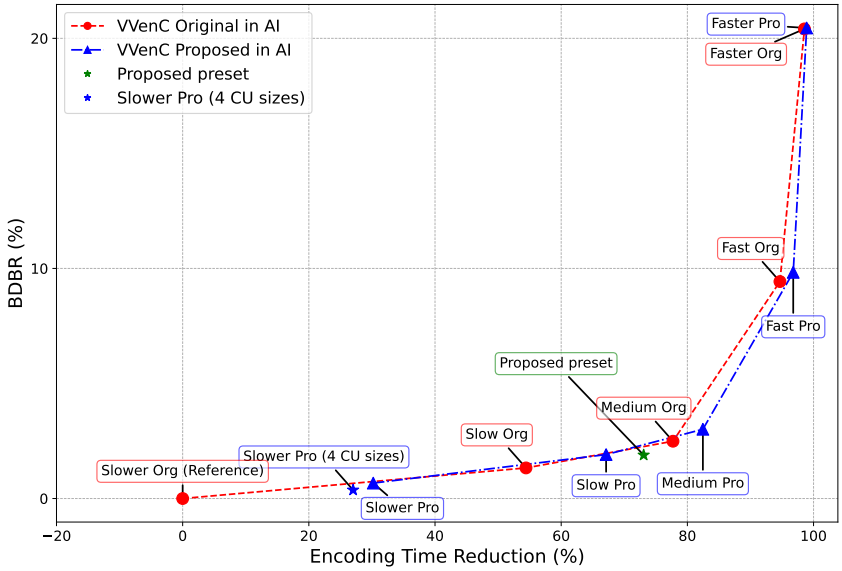


Figure 3.10 *Original vs. proposed VVenC encoders presets trade-offs in terms of ΔETR(%) and BDBR(%)*

These experimental results are illustrated in Figure 3.10, clearly demonstrating significant speedups for the Slower, Slow, and Medium presets. Additionally, a new preset,

marked by a green star in Figure 3.10, offers a notable reduction in encoding time with only a modest increase in BDBR. This proposed preset is based on the original Medium preset but incorporates adjusted parameters, such as increasing the MTT depth to align more closely with the Slower preset, enhancing overall performance. The results in Figure 3.10 illustrate a clear trade-off between encoding time and BDBR loss. Specifically, as encoding time decreases, video quality also diminishes. Thus, the choice of preset will depend on specific user requirements, particularly the preferred balance between faster encoding and maintaining video quality.

Figure 3.11 illustrates the speedups and shifts in each preset point from the original presets, using the VVenC anchor, to the new points achieved through the application of our ML-based approach. These speedups highlight the effectiveness of the proposed method in accelerating the presets, with only negligible increases in bitrate.

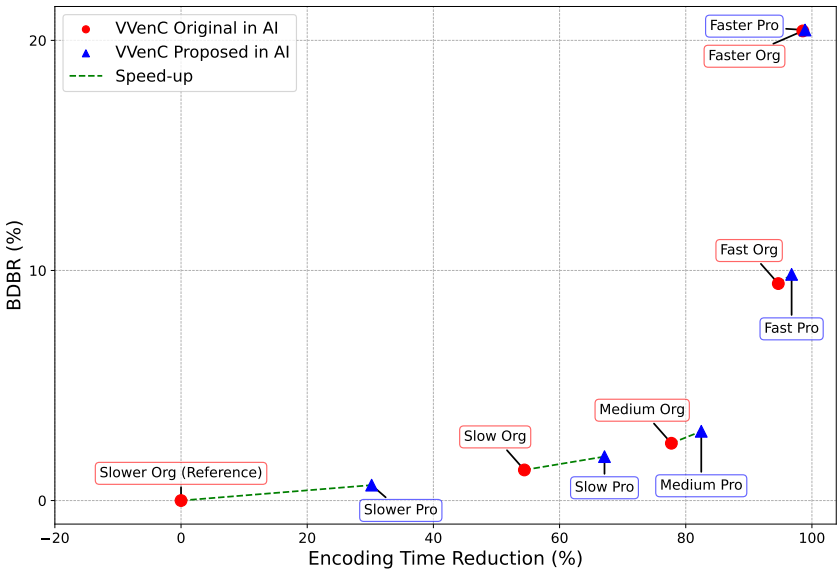


Figure 3.11 *Original vs. proposed VVenC encoders presets speed-ups*

To further evaluate our proposed solution, we compared each preset individually using both the original and proposed encoders. The results of this comparison are detailed in Table 3.9.

Consequently, the results indicate that the proposed approach achieved an average reduction in encoding time of over 27% across the various presets. At the same time, it maintained a good level of image quality, as evidenced by the BDBR measurements. The BDBR loss tends to decrease slightly from the Slower preset to the Medium preset, likely due to adjustments in configuration parameters related to the CU partitioning

Table 3.9 *Comparison of each preset individually using the original and the proposed encoders*

Preset	ETR (%)	BDBR (%)
Slower	30.21	0.67
Slow	28.12	0.58
Medium	23.67	0.52
Fast	21.98	0.38
Faster	25.63	0.05

process, such as the maximum depth of the QTMTT structure and the maximum CTU size. Additionally, the Fast and Faster presets are limited to QT with reduced MTT depth for intra-frames in Fast, making performance improvements in these presets more challenging. Overall, these results demonstrate that the proposed approach effectively balances the trade-off between encoding time and image quality. This balance is crucial for users who need to manage the resource-intensive and time-consuming nature of the encoding process. By adopting the proposed approach, users can potentially save time and resources while still achieving high-quality image output.

3.4.3 Comparison with Existing Methods in the Literature

In this subsection, we compare the proposed solution with other state-of-the-art methods in terms of ETR and BDBR loss. Table 3.10 summarizes the average trade-offs across different video sequence classes. The state-of-the-art methods were evaluated using the VTM reference software, while the proposed approach was integrated into VVenC 1.3.1 [52], using the Slower preset as a reference. Additionally, a new preset was proposed to further enhance the encoding speed with only a minor BDBR degradation. Both presets demonstrate significant improvements in the encoding process, achieving substantial time savings while preserving notable image quality.

It is important to note that a direct comparison between the proposed approach implemented in the VVenC encoder and state-of-the-art techniques using the VTM reference software may not fully reflect the significance of the results, as VVenC is already more optimized than VTM. A more relevant comparison would be between the proposed approach and other state-of-the-art methods also integrated into the VVenC encoder and optimized for similar use cases.

The results presented in Table 3.10 indicate that, in terms of encoding efficiency, the proposed approach using the Slower preset achieved the best performance measured by the BDBR metric. This was followed by the methods presented by *Saldanha* [63] and

Table 3.10 *Performance comparison of the proposed solution with the state-of-the-art approaches in AI coding*

Class	Saldanha [63]		Guoqing [66]		Zhang [47]		Slower Proposed		Proposed Preset	
	BDBR (%)	ΔETR (%)	BDBR (%)	ΔETR (%)	BDBR (%)	ΔETR (%)	BDBR (%)	ΔETR (%)	BDBR (%)	ΔETR (%)
A1	1.05	53.39	2.18	58.70	1.65	61.19	0.11	31.82	1.25	74.75
A2	1.11	52.63	2.25	63.11	1.87	63.04	0.68	16.57	2.01	74.31
B	1.40	58.26	2.87	66.93	1.52	64.43	0.79	34.03	2.04	77.89
C	1.65	51.99	2.92	62.27	1.39	58.61	0.80	27.90	2.17	72.15
D	1.24	49.92	1.98	60.01	0.95	56.35	0.52	31.79	1.47	69.87
E	2.07	58.44	4.08	66.78	1.97	62.74	1.13	39.16	2.38	69.48
Average	**1.42**	**54.20**	**2.71**	**63.16**	**1.52**	**61.15**	**0.67**	**30.21**	**1.89**	**73.07**

Zhang [47], with BDBR values of 1.42% and 1.52%, respectively. In contrast, *Guoqing* [66] experienced a higher BDBR degradation of 2.71%. Regarding ETR, the proposed preset significantly outperforms the other methods, achieving an average reduction of 73.07%. This is followed by the approach of *Guoqing* [66], *Zhang* [47], *Saldanha* [63], and finally, the proposed solution with the Slower preset. The Slower preset offers the most minimal BDBR degradation at just 0.67% while providing a substantial time-saving of 31.21%, demonstrating a favorable trade-off compared to other methods. The proposed preset further enhances time savings by up to 73.07% compared to existing works, with a modest increase in BDBR loss of 1.89%.

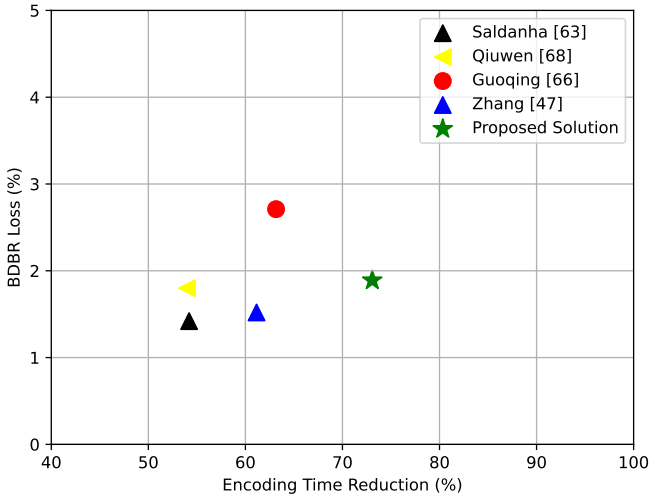


Figure 3.12 *Performance comparison of the proposed solution with the state-of-the-art approaches in AI coding*

Overall, the proposed preset effectively balances encoding time reduction with bitrate

increase. As shown in Figure 3.10, the proposed preset, marked by the green star, is positioned below the curves of both the original and proposed VVenC presets. This placement demonstrates its ability to save more time while incurring a reasonable BDBR loss of 1.89%, outperforming all other presets in terms of time savings. Figure 3.12 further illustrates the position of the proposed solution, also marked by the green star, showcasing its advantage in achieving substantial encoding time reductions with minimal bitrate increases. This positions the proposed approach favorably when compared to existing methods. These results underscore the promising nature of the proposed solution, though there remains potential for further improvements in encoding efficiency.

3.5 Conclusion

In this chapter, a novel approach was introduced using a set of LightGBM binary classifiers to enhance the CU partitioning process in VVC intra-coding. Integrated into the VVenC encoder, this method involves extracting CU features and training LightGBM classifiers to predict the most likely split mode within the QTMTT structure. To enhance classification accuracy, predefined risk intervals were applied to each classifier. Additionally, a new preset was proposed to optimize the balance between encoding quality and speed, achieving a 73.07% reduction in encoding time with only a 1.89% increase in BDBR. Experimental results confirmed that this approach significantly reduces encoding time while maintaining high encoding efficiency, outperforming several state-of-the-art methods.

In the next chapter, the work will focus on inter-coding under the Random Access (RA) configuration. Specifically, it will explore how LightGBM classifiers can improve CU partitioning efficiency by leveraging temporal and texture features to reduce the computational load of the conventional RDO process, enabling faster encoding without sacrificing coding efficiency.

Chapter 4

Low-Complexity QTMTT Partitioning Scheme for VVC Inter Coding

4.1 Introduction

This chapter extends the optimization efforts from Chapter 3, shifting the focus from intra-coding to inter-coding within the Random Access (RA) configuration of the Versatile Video Encoder (VVenC) encoder. As outlined in Chapter 1, Versatile Video Coding (VVC) employs sophisticated techniques to reduce bitstream size while preserving high video quality, with the enhanced Coding Tree Unit (CTU) partitioning scheme playing a critical role. This scheme, based on the Quadtree with Multi-Type Tree (QTMTT) structure [7], introduces additional partitioning options that enhance compression efficiency by allowing more flexible block splits. However, as discussed in Chapter 2, this added flexibility significantly increases computational complexity, creating a pressing need for optimization. To address this complexity in inter-coding, this chapter proposes an Machine Learning (ML)-based approach that leverages Light Gradient Boosting Machine (LightGBM) classifiers [10] to optimize the QTMTT partitioning process under the RA configuration [80]. The LightGBM classifiers are trained on features extracted from original frames, including coding information, texture, and temporal complexity features. This method is evaluated across multiple video sequences from the Common Test Conditions (CTC) dataset and benchmarked against the VVenC encoder version 1.7.0 [19], demonstrating substantial reductions in encoding time with minimal impact on the Bjøntegaard Delta Bit Rate (BDBR). The chapter begins with a review of recent advancements in complexity reduction for partitioning blocks in video encoders, with a focus on the High Efficiency Video Coding (HEVC) and VVC standards. Following this, the proposed ML-based method for reducing Coding Unit (CU) partitioning complexity in VVC is presented

in detail. Experimental results are then provided, comparing the effectiveness of this approach with existing methods.

4.2 Related Works on CU Partitioning in Inter Coding

In both HEVC and its successor, VVC, optimizing CU partitioning is essential for improving encoding efficiency and reducing processing time. This section delves into the CU partitioning process in VVC's inter-coding mode and discusses various complexity reduction strategies applied to tree partitioning in HEVC and VVC. These strategies are categorized into heuristic, ML, and Deep Learning (DL) approaches. By reviewing state-of-the-art techniques, we highlight their effectiveness and limitations, setting the stage for our research aimed at enhancing CU partitioning in the VVenC encoder under the RA configuration. As detailed in Chapter 2, VVenC provides presets that balance encoding speed and quality, including faster, fast, medium, slow, and slower options. Each preset adjusts the encoder's internal parameters to meet varying use case requirements. One of the critical aspects of these presets is the configuration of the partitioning block, which directly impacts its performance [9]. The structure of this block in VVenC is governed by Quadtree (QT) and Multi-Type Tree (MTT) partitions, with different QT, Binary-Tree (BT), and Ternary-Tree (TT) depth values as shown in Table 4.1.

Table 4.1 *Maximum depth values for each split mode and encoding preset*

Split Mode \ Preset	Faster	Fast	Medium	Slow	Slower
QT	4	4	4	4	4
BT	0	0	1	2	3
TT	0	0	1	2	3

In the slower and slow settings, the encoder maximizes partitioning depth with up to three levels of BT or TT splits, enabling the most detailed and efficient compression at the cost of increased runtime. Moving to the medium setting, the encoder permits up to two levels of BT or TT splits, balancing speed and compression efficiency. In the fast and faster settings, the encoder employs minimal partitioning complexity, using only QT to allow for rapid encoding with reduced compression efficiency [9]. Consequently, CU partitioning significantly increases the encoder's computational complexity, especially in

inter-coding mode, which involves multiple reference frames and Motion Vectors (MVs). To mitigate this complexity, several optimization methods are discussed in the following section.

4.2.1 Complexity Reduction Strategies for Tree Partitioning in HEVC

Most literature works for the HEVC standard were implemented using the reference software known as HEVC test Model (HM) [17].

4.2.1.1 Heuristic Approaches in HEVC

Huang et al. [81] proposed a comprehensive solution to optimize CU partitioning in HEVC by integrating several techniques. These include CU depth pre-selection, early termination for both CU and Prediction Unit (PU), and a fast decision tree for Transform Unit (TU) selection. The approach is based on a Rate-Distortion (RD)-complexity optimization framework that assesses the RD-complexity of both fast and conventional algorithms. This assessment involves comparing the complexity of the two algorithms and evaluating the RD loss of the fast algorithm relative to the traditional method. Temporal and spatio-temporal features are utilized to estimate the RD loss. A comparison between the RD-complexity cost and a predefined threshold determines whether the initial CU depth (depth 0) should be skipped. Additionally, early termination for CU depth, a fast TU decision tree, and early PU termination are implemented using the same RD-complexity principle but with different features. For CU depth early termination, features such as the number of bits generated by the motion vector difference and the mean absolute levels of quantized transform coefficients are considered. The fast TU decision tree and early PU termination use features like the mean absolute Hadamard transform difference and the maximum difference among sub-blocks within the CU. This combined approach, tested under HM 16.7 in the RA configuration, achieved a complexity reduction of 46% to 70%, with a BDBR increase ranging from 0.48% to 2.36%.

4.2.1.2 Machine Learning and Deep Learning Approaches in HEVC

Correa et al. [82] introduced a method for early termination of the Rate-Distortion Optimization (RDO) process in HEVC by utilizing a set of binary decision trees. These trees predict whether further partitioning is necessary for CUs of sizes 64×64, 32×32,

and 16×16. The decision trees leverage information gained from various features, such as RD costs for different splitting modes and the merge flag, to identify the most significant factors for decision-making. Key features include the chosen Prediction Unit (PU) splitting mode, the depth of neighboring CTUs, and the ratio between RD costs of different inter-splitting modes. For early termination of the PU structure, additional binary trees assess the likelihood of further partitioning for CUs ranging from 64×64 to 8×8. Additionally, two decision trees are employed to optimize the residual QT structure for 32×32 and 16×16 CUs. This ML-based approach achieved a 65% reduction in computational complexity with a BDBR increase of 1.36%.

In a subsequent extension of their work, Correa et al. [83] developed an online training scheme focused specifically on the early termination of the CU partitioning process. This approach utilizes three decision trees tailored to different CU sizes. Their analysis revealed that models trained on-the-fly using the first frame of a sequence achieved higher accuracy compared to offline-trained models. Consequently, the first frame is encoded using the traditional RDO process to collect training data, which is then pre-processed to address imbalances in split decisions. This method, sensitive to temporal relationships between frames, requires retraining during scene changes. When applied to the HM 16 encoder in the RA configuration, this approach resulted in a 34.4% reduction in complexity with a minimal BDBR increase of 0.2%.

Grellert et al. [84] employed a Support Vector Machine (SVM) to predict and skip unnecessary sub-partition evaluations in HEVC. Their method began with an analysis of 28 features, including coding flags, RD cost metrics, and MV direction, to determine their relevance to CU partitioning decisions. The analysis showed that the most effective features varied with CU size; for example, the number of encoded bits was crucial for 16×16 CUs but less so for 64×64 CUs. To address these differences, separate SVMs were initially trained for each CU size (64×64, 32×32, and 16×16). However, they found that F1 scores remained relatively consistent across different resolutions, allowing the use of a single SVM model for all sizes. The SVM models were trained to balance RD performance and complexity reduction, with a threshold applied to the SVM outputs to decide whether to skip a split. Depending on the threshold setting, this method achieved complexity reductions ranging from 34.9% to 52.4%, with BDBR increases between 0.13% and 1.11%, under the HM 16.8 in the RA configuration.

4.2.2 Complexity Reduction Strategies for Tree Partitioning in VVC

Most of the literature has focused on implementations using the widely adopted VVC Test Model (VTM) [18], with fewer studies exploring other implementations, such as VVenC [52].

4.2.2.1 Heuristic Approaches in VVC

Tang et al. [85] introduced a fast algorithm for CTU partitioning in both intra and inter-coding modes of VVC. For intra-coding, the algorithm utilizes a block-level Canny edge detector [86] to identify edge direction and strength, enabling selective skipping of certain partition modes. In inter-coding, the method employs a custom metric based on pixel value differences between the current frame and its reference frames to eliminate less likely partitions. By setting a threshold for these differences, the algorithm can terminate partitioning early, reducing unnecessary processing. This approach achieves a 31% reduction in encoding time with a BDBR increase of 1.34%.

In another study, Wei et al. [87] developed a method to reduce partitioning complexity in both intra and inter-prediction modes of VVC. The algorithm leverages historical data from previously analyzed partitions to identify the most efficient partitioning paths. It introduces direction-based and position-based skips for BT and TT partitions, as well as depth-based skips for MTT partitions. This method yields significant Encoding Time Reduction (ETR): 42.47% in All Intra (AI), 40.46% in RA, and 40.38% in Low Delay (LD)-P configurations, with BDBR increases of 0.45%, 1.08%, and 1.18%, respectively.

Shang et al. [88] proposed a technique that predicts the necessity of splitting a CU based on coding information from neighboring CUs. The method also uses temporally optimal coding modes generated during prediction to limit candidate modes, thereby accelerating the coding process. Additionally, it assesses the motion complexity of the current CU by analyzing neighboring prediction modes, enabling earlier skipping of unnecessary modes. Implemented in VTM 11.0, this approach achieves an average complexity reduction of 40.08%, accompanied by an increase of 1.56% in BDBR.

4.2.2.2 Machine Learning Approaches in VVC

Amestoy et al. [89] introduced a ML-based solution to address the increased encoding time associated with the Quadtree plus Binary-Tree (QTBT) structure in VVC. Utilizing

Random Forest (RF) classifiers, their method predicts the most likely partition modes for each coding block, allowing for a tunable balance between complexity reduction and coding efficiency. Implemented in the 'Joint Exploration Model (JEM) 7.0, the approach achieved encoding complexity reductions ranging from 30% to 70%, with a corresponding BDBR increase of 0.7% to 3.0%.

Kulupana et al. [90] proposed an approach that combines specialized Decision Trees (DTs) and RF classifiers to predict QTMTT and horizontal/vertical split decisions. Their method includes a tree selection algorithm that identifies the most accurate DTs from each forest based on an offline validation set. When implemented in the VTM software, this approach achieved ETRs of 32% to 41%, with a BDBR increase of 0.82% to 1.33%.

Li et al. [91] developed a temporal prediction model for optimal CU partitioning. This model uses data from previously encoded frames, along with motion vector difference information, to refine predictions. Although this method achieves a 23.19% ETR with a 0.97% BDBR increase, its reliance on sequential processing of frames can limit parallelism, posing challenges for simultaneous encoding operations.

Li et al. [92] also proposed an RF-based method focusing on the temporal information and texture complexity of CUs, applied across various sizes (64×64, 32×32, 16×16). Using the Gini coefficient [93] for decision-making at each node, this method achieved a 15.27% reduction in encoding time in RA mode, with a BDBR increase of 1.87%.

Xie et al. [94] developed an RF-based early termination algorithm that uses features extracted from both temporal co-located blocks and spatially adjacent blocks of the current CU. This model predicts whether to terminate the inter-prediction process early. When implemented in VVenC, this method resulted in an average ETR of 7.71% in RA mode, with a BDBR increase of 1.48%.

4.2.2.3 Deep Learning Approaches in VVC

In the realm of deep learning, Yeo et al. [95] presented a Convolutional Neural Network (CNN)-based fast split mode decision algorithm for VVC inter-prediction. Their multi-level tree CNN architecture classifies the split mode of each CU by analyzing factors such as the original and residual images, Picture Order Count (POC), and CU-level Quantization Parameter (QP) value. Implemented in VTM 11.0, this method achieved an average complexity reduction of 11.53%, with a BDBR increase of 1.01%, which may not be ideal for real-time applications due to its relatively low time savings.

Liu et al. [96] proposed a lightweight CNN-based method for predicting partitioning

grids at the CTU level, achieving ETRs ranging from 17% to 30% with a BDBR increase of 0.37% to 1.18%. In subsequent work, Liu et al. [97] introduced a U-Net-based CNN approach that predicts optimal partition paths using multi-scale motion vector fields and a partition pruning algorithm, further enhancing VVC encoding efficiency.

Tissier et al. [98] combined deep learning and ML techniques to reduce VVC encoder complexity in the RA configuration while minimizing BDBR loss. Their method utilizes a CNN to process the current CTU alongside two reference CTUs, identified based on motion vectors with the lowest RD cost. A multi-class DT classifier then predicts split probabilities using CNN-derived features. The encoder tests the top-N predicted splits, balancing complexity reduction with BDBR performance. This approach achieved approximately a 31.8% complexity reduction with a 1.11% BDBR increase, although the use of CNNs introduces significant computational overhead, which can hinder real-time processing.

Pan et al. [99] developed a fast encoder algorithm featuring a multi-branch CNN that performs binary classification at the CU level to determine "Partition" or "Non-partition," fully exploiting texture and motion characteristics to enhance implementation efficiency.

In summary, while heuristic, ML, and DL techniques have each contributed to optimizing the VVC partitioning block, they also present distinct challenges. Heuristic methods may lack generalizability, ML approaches, though adaptable, often require extensive computational resources and high-quality training data, and DL techniques, while effective, demand large datasets and significant computational power, complicating their integration and interpretation. Achieving a balance between ETR and BDBR increase remains crucial for effective video compression.

Table 4.2 summarizes techniques for optimizing CU partitioning in the VVC standard, including software versions, ETRs, BDBR increases, and publication years. While most studies utilize the VTM [18] encoder, Jingyi [92] and Kundan [94] employed the optimized VVenC [9] with the medium preset in the RA configuration. Our research extends these optimizations to the slow and slower presets, enhancing VVenC's suitability for real-time applications. Notably, the fast and faster presets in VVenC exclusively use QT, making additional optimizations unnecessary for these two settings.

While VTM is effective for benchmarking, it is not optimized for practical use. In contrast, VVenC offers a more efficient VVC implementation, as detailed in Chapter 3. It delivers comparable performance with reduced runtime [9], targeting real-time applications by optimizing various encoding tools. However, despite these improvements, VVenC

Table 4.2 *Overview of techniques for VVC inter prediction optimization*

Approach	Software Reference	Optimization Technique	ETR (%)	BDBR (%)	Year of Publication
Tang et al. [85]	VTM 4.0	Heuristic	31	1.34	2019
Amestoy et al. [89]	JEM 7.0	ML	30-70	0.7-3.0	2020
Yeo et al. [95]	VTM 11.0	DL	11.53	1.01	2021
Pan et al. [99]	VTM 10.0	DL	30.63	3.0	2021
Kulupana et al. [90]	VTM 6.0	ML	32-41	0.82-1.33	2021
Yue Li et al. [91]	VTM 7.0	ML	23.19	0.97	2022
Liu et al. [96]	VTM 10.0	DL	17-30	0.37-1.18	2022
Li Jingyi et al. [92]	VVenC 1.0	ML	15.27	1.87	2022
Wei et al. [87]	VTM 8.0	Heuristic	40.46	1.08	2022
Tissier et al. [98]	VTM 7.0	DL+ML	31.3-51	1.65-3.79	2022
Shang et al. [88]	VTM 11.0	Heuristic	40.08	1.56	2023
Xie et al. [94]	VVenC 1.0.0	ML	7.71	1.48	2023
Liu et al. [97]	VTM 13.0	DL	33.8-52.2	1.14-2.60	2023

still faces challenges due to the inherent computational complexity of VVC, especially in CU partitioning [8]. This complexity results in higher latency and increased computational demands [7], posing difficulties for real-time applications such as live streaming, video conferencing, and interactive gaming, where low latency and responsiveness are critical [100].

To address these challenges, our research focuses on further optimizing the VVenC encoder by improving its CU partitioning through a ML-based approach, which is discussed in detail in the following sections.

4.3 Proposed CU Partitioning Approach for Inter Coding

In the field of video compression, the CU partitioning process significantly impacts both compression efficiency and encoding time. A study by Tissier et al. [8] demonstrated that optimizing this process could reduce computational complexity by over 97%. Building on this, our approach introduces a novel method to enhance CU partitioning within the VVC framework, specifically targeting the VVenC encoder [80]. This method employs ML to accelerate CU partitioning by integrating features related to spatial and temporal complexities. These features include spatial metrics such as gradient values, pixel variance, and mean pixel values, which assess texture complexity and guide decisions on CU splitting. Temporal features, derived from motion vector variances between consecutive

frames, capture the motion dynamics essential for inter-frame coding. These features were used to train a set of LightGBM binary classifiers that predict the optimal split modes for each CU. These predictions are then utilized by the VVenC encoder to evaluate and select the best partition mode. As a result, our approach aims to optimize the encoding process by reducing its runtime while preserving compression efficiency. The architecture of the proposed approach is shown in Figure 4.1, illustrating the enhancement of the partitioning block in the VVenC encoder. Initially, the inter-frame processing begins by extracting the Motion Vector Field (MVF) for 4×4 pixel blocks within the frame, referencing the motion information from previous frames. This MVF is then used to derive the temporal features of the current CU. Additionally, texture features are extracted. Then, these extracted features are fed as input to the binary classifiers within the ML block. These classifiers predict the most probable split modes for the CU based on the extracted features. The predicted split modes are stored in a vector and subsequently evaluated by the encoder. The encoder tests these modes for the current CU, and upon processing the CU, it moves on to the next CUs. This process continues until the entire video is encoded into a bitstream file.

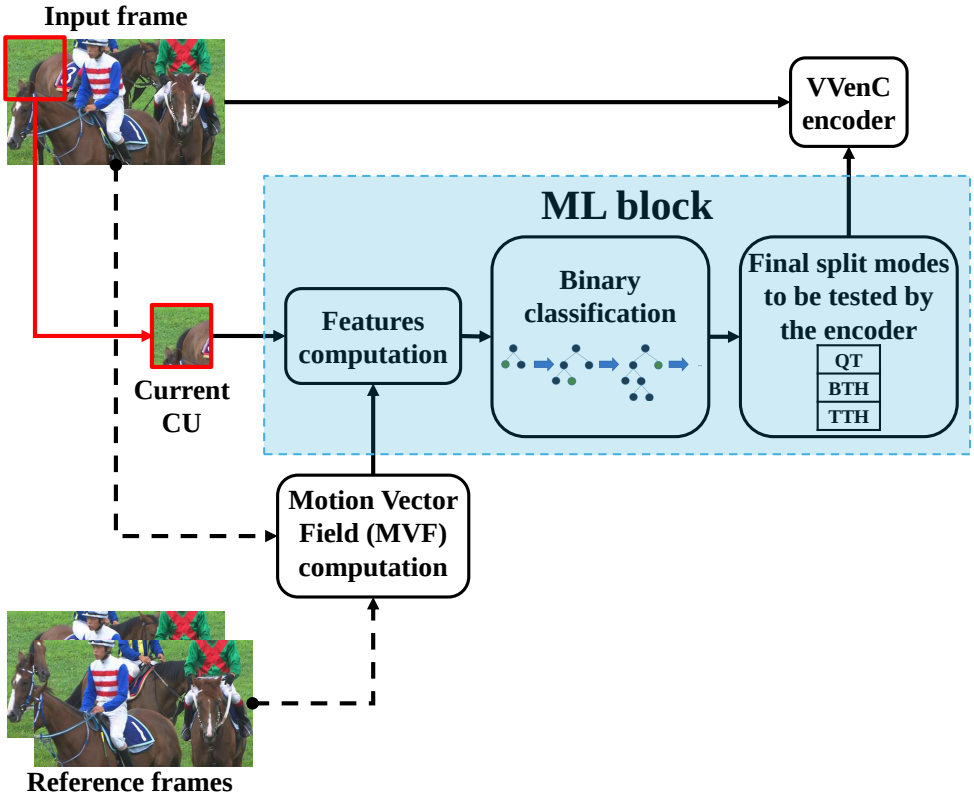


Figure 4.1 *ML-based approach for optimizing CU partitioning in the VVenC encoder*

4.3.1 Texture and Temporal Features Extraction

In video encoding, intra-frames correlate CU partition modes with texture complexity, while inter-frames introduce an additional layer of complexity because they depend on reference frames for prediction [7]. Our research enhances CU partitioning under RA configuration by integrating texture and temporal complexity, aiming to improve classification performance for optimal partitioning strategies. Furthermore, higher texture complexity generally indicates further splitting due to the close relation between the QTMTT structure and image texture. Whereas, image areas with more complex textures are usually further split. On the other hand, texture complexity can also reflect the direction selected for the partition modes [70]. When the complexity of texture in the horizontal direction is higher than in the vertical direction, CU will be more likely to be divided in the horizontal direction, and vice versa. Given these observations and their computational implications, a set of features was carefully selected to train binary classifiers. These features provide a comprehensive view of spatial and texture complexity, which is critically important for image quality. Additionally, temporal complexity describes the dynamics of a video and is essential for inter-coding mode. This method combines temporal and texture complexity features to improve prediction accuracy, which makes classifiers versatile and effective in handling a variety of video content.

4.3.1.1 Texture Complexity Features

Since coding modes can vary significantly across different QPs, our approach incorporates data from a range of commonly used QP values (22, 27, 32, and 37), as specified in modern video coding standards such as VVC [40]. This selection ensures that the dataset captures a wide variety of coding decisions, enabling the models to manage CUs effectively across various levels of quantization. To account for texture complexity, specific features are computed from pixel values. Initially introduced in our previous works and discussed in Chapter 3, these features are crucial for enabling the models to adapt to diverse content across varying QP conditions. The features listed below are explained in more detail, with their definitions provided in the following section 3.3.2.1 (page 74).

— **Quantization Parameter (QP)**
— **Vertical gradient (Δ_V)**
— **Horizontal gradient (Δ_H)**
— **CU Pixels Mean (μ_{CU})**

— **CU Pixels Variance** (σ^2_{CU})
— **Sub-CU's Texture Complexity** (***SCTC***)

4.3.1.2 Temporal Complexity Features

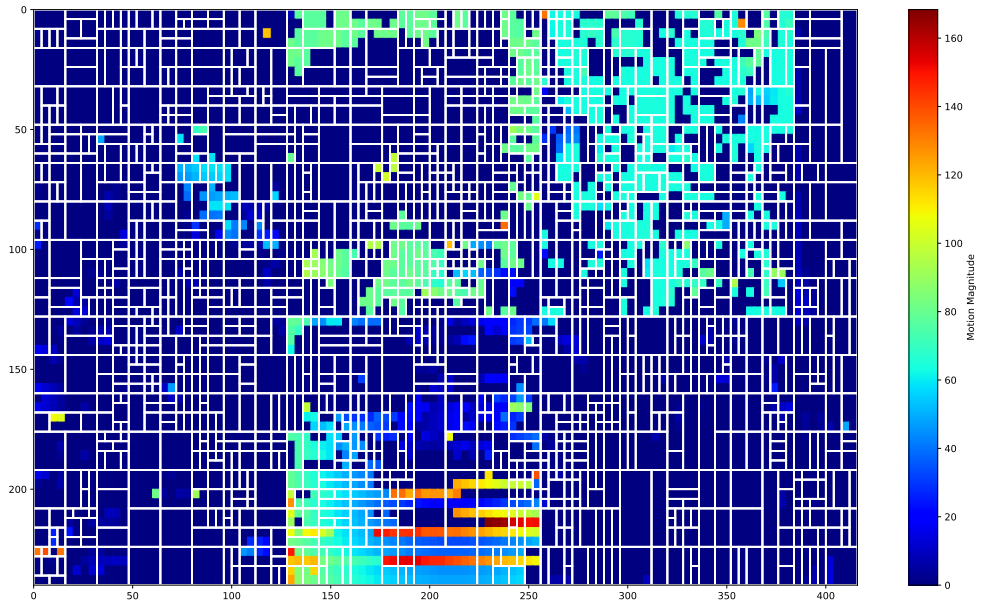
Temporal complexity was assessed by selecting features that quantify changes between consecutive frames. These features rely on MV information, frame relations, and frame differences, which are crucial for inter-coding. These features were calculated using the MVF created by the MVs associated with each 4×4 pixel block within the frame. These MVs are crucial in computing the variance within each CU and its corresponding sub-CUs. The process leverages the 4×4 MVs that define the movement patterns within each block or sub-block. This method provides a detailed understanding of motion consistency and variability across different sections of the video frame.

Figure 4.2a illustrates the MVs of each 4×4 block within frame number 16 of the *RaceHorses* video sequence, captured at a resolution of 416×240 pixels. Additionally, Figure 4.2b complements this by depicting the magnitude of each 4×4 block in the same frame, employing the QTMTT scheme for CU partitioning. As illustrated in Figures 4.2a and 4.2b, dynamic elements within the frame exhibit higher magnitude values and more varied MV directions, necessitating increased CU partitioning levels. Conversely, static elements display stable motion patterns, resulting in fewer divisions.

— **Temporal Layer (Tid)**: Indicates the temporal layer of a frame in a video sequence.
— **QT depth (QTD)**: Represents the QT depth for a block, related to motion or texture complexity.
— **MTT depth (MTTD)**: Similar to QTD, but for the MTT structure.
— **MVs Variance**: Each CU is subdivided into smaller blocks of 4×4 pixels. The variance of MVs both in horizontal (MV_x) and vertical (MV_y) directions within these blocks provides significant insights into the motion characteristics across the CU. Given a CU of any size with width W_{CU} and height H_{CU}, partitioned into N 4×4 blocks, where N is dependent on the CU dimensions and calculated as defined in Equation (4.1), the variance of the MVs is calculated as follows:

$$N = \frac{W_{CU} \times H_{CU}}{16} \tag{4.1}$$

(a) *Frame representation combined with its 4×4 MVs directions*



(b) *QTMTT-based frame partitioning with 4×4 MVs magnitudes*

Figure 4.2 *Visualization of MVs in 4×4 blocks: MVs directions indicated by red arrows in (a) and MVs magnitudes with CU partitioning structure in (b)*

$$\sigma^2_{MV_x} = \frac{1}{N}\sum_{i=0}^{N-1}(MV_x(i) - \mu_{MV_x})^2 \tag{4.2}$$

$$\sigma^2_{MV_y} = \frac{1}{N}\sum_{i=0}^{N-1}(MV_y(i) - \mu_{MV_y})^2 \tag{4.3}$$

where $MV_x(i)$ and $MV_y(i)$ are the horizontal and vertical components of the MV for the i-th 4×4 block within the CU, respectively. The mean values, μ_{MV_x} and μ_{MV_y}, are computed as the average of the MV components across all blocks in the CU as described in Equation (4.4) and Equation (4.5):

$$\mu_{MV_x} = \frac{1}{N}\sum_{i=0}^{N-1} MV_x(i) \tag{4.4}$$

$$\mu_{MV_y} = \frac{1}{N}\sum_{i=0}^{N-1} MV_y(i) \tag{4.5}$$

Feature computations are conducted for the entire CU as well as each of its sub-CU components, as discussed in detail in Chapter 3, Section 3.3.2.2 (page 75).

4.3.2 Classification Algorithm Structure

Previous research demonstrates that early identification of whether a CU requires splitting, as well as the determination of the split direction (horizontal or vertical for BT and TT types), can significantly lower computational load [63, 70, 78, 89]. These decisions lead to binary classification challenges: determining whether a split should occur and its direction.

Our approach employs LightGBM for classification, selected for its effectiveness in handling nonlinear classification scenarios [10]. Furthermore, instead of using a single multiclass classifier to categorize the five classes (QT, Binary Tree Horizontal (BTH), Binary Tree Vertical (BTV), Ternary Tree Horizontal (TTH), Ternary Tree Vertical (TTV)) or for each CU size, five separate binary classifiers were employed. Each binary classifier independently predicts whether the current CU should be split based on its features. This segmentation enhances the separation between classes and thereby improves classification performance. Consequently, five binary classifiers were introduced: C_{QT}, C_{BTH}, C_{BTV}, C_{TTH}, and C_{TTV}. Each classifier functions as an independent split examiner, where each classifier C_s determines whether to apply the split s or not. These classifiers play a critical role in the decision-making process by assessing the necessity of each potential split.

Algorithm 1 *CU Partitioning Optimization Algorithm Using LightGBM Binary Classifiers*

Require: Coding Unit (CU)
Ensure: Optimal split mode for the CU or termination of its encoding
Initialize binary classifiers for each split type: C_{QT}, C_{BTH}, C_{BTV}, C_{TTH}, C_{TTV}
$P_{QT} = P_{BTH} = P_{BTV} = P_{TTH} = P_{TTV} = 0$
if No partition mode is allowed **then**
    Terminate encoding for the current CU **return** Encoding termination signal
end if
Initialize a list to hold split probabilities: L_P
for Each CU in the CTU **do**
    $\mathbf{F}$ = Compute Feature(CU, W, H)
    if canSplit(QT) **then**
        $P_{QT} = C_{QT}(\mathbf{F})$
    end if
    if canSplit(BTH) **then**
        $P_{BTH} = C_{BTH}(\mathbf{F})$
    end if
    if canSplit(BTV) **then**
        $P_{BTV} = C_{BTV}(\mathbf{F})$
    end if
    if canSplit(TTH) **then**
        $P_{TTH} = C_{TTH}(\mathbf{F})$
    end if
    if canSplit(TTV) **then**
        $P_{TTV} = C_{TTV}(\mathbf{F})$
    end if
    L_P.append($[P_{QT}, P_{BTH}, P_{BTV}, P_{TTH}, P_{TTV}]$)
end for
Sort L_P in descending order based on probabilities
if $\max(P)_{P \in L_P} < TH_{NS}$ **then**
    Terminate encoding of the current CU **return** Encoding termination signal
else
    Select the top-3 split modes with the highest probabilities
    Test these split modes on the current CU
    Select and apply the optimal mode on this CU
end if
Proceed to encode the next CU

The proposed Algorithm 4.3.2 details our method for accelerating the CU partitioning in the RA coding mode. By leveraging binary classifiers trained to predict the likelihood of various CU split modes, including QT, horizontal and vertical binary splits (BTH and BTV), and asymmetric horizontal and vertical ternary splits (TTH and TTV), the algorithm efficiently prioritizes the most promising split configurations for each CU. In this framework, Algorithm 4.3.2 initializes the classifiers by integrating them into the encoder. Subsequently, the first CU embarks on its split mode determination by feature extraction as described in section 4.3.1. This includes assessing the size of each CU for

its compatibility with certain modes before initiating the classification process. If the size criteria are met, the CU is evaluated by each eligible classifier to assess the probability of optimal representation by one of the five possible split modes. These classifiers are built to detect which division mode best balances encoding efficiency with computational complexity, quantifying this through a probability score denoted as P_{xx}. Once each classifier has evaluated the CU, the resulting probabilities are aggregated into a list, denoted L_P, in descending order. The algorithm then prioritizes split modes based on these probability scores. If the CU size permits, the algorithm selects the top-N modes with the highest probability scores as candidates for encoding. Our experiments show that limiting the evaluation to only the top-1 or top-2 modes results in notable quality degradation, with BDBR increases of approximately 8% and 5%, respectively. This effect is more pronounced in slower encoding presets, further impacting overall quality. By contrast, selecting N=3 offers an optimized balance between bitrate increase and runtime reduction. However, if the CU size allows for fewer than three modes, all available modes are considered candidates. Subsequently, the encoder tests these candidates to identify the one that minimizes the RD cost. Conversely, suppose all predicted probabilities fall below a predefined threshold, denoted as TH_{NS} and set to 0.25. In that case, the encoder determines that no further splitting is necessary for the current CU and proceeds to encode it accordingly. This iterative process continues until all CUs within the CTU have been processed.

To address the potential misclassification issue and ensure robustness in the encoding process, additional checks on CU size and the number of modes selected as candidates have been implemented. This measure aims to guarantee that at least one of the selected modes represents the optimal split for a given CU. By including these checks, the algorithm introduces a dual selection mechanism, offering flexibility and resilience. This mechanism provides a secondary option, ensuring that even if the primary split mode is not accurately identified, a close alternative is available. Thus, the algorithm enhances its adaptability and robustness, ultimately contributing to more reliable encoding outcomes.

4.3.3 Training Methodology

Figure 4.3 depicts the workflow of the proposed method, which is comprised of three main stages: dataset creation, binary classifiers training, and the implementation and evaluation of the proposed solution.

Initially, a large dataset of training video sequences is used to extract features related

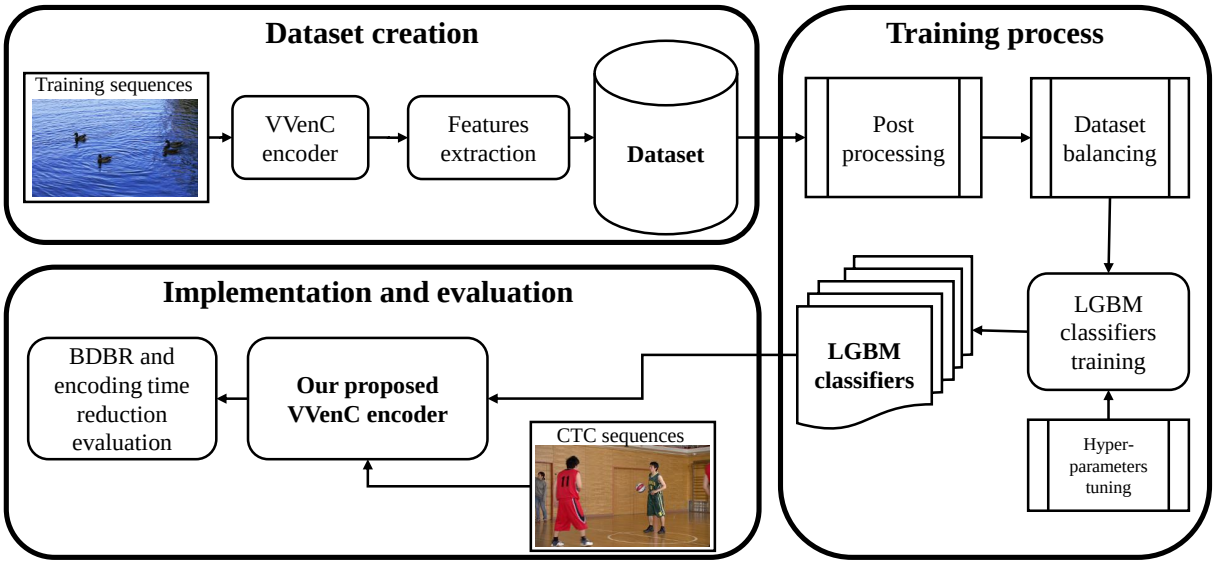


Figure 4.3 *Proposed method workflow: dataset creation, training, and performance evaluation in the VVenC encoder*

to texture and temporal complexity. These features, along with their corresponding CU split modes, form the basis of our dataset. Due to its superior coding efficiency, the slower preset is used for extracting features during the encoding process [19]. Subsequently, this dataset is used to train a set of binary classifiers. These classifiers are then integrated into the VVenC encoder to enhance its CU partitioning efficiency. Finally, to validate the effectiveness of the proposed method, we conducted evaluations using the slower, slow, and medium presets, with the slower serving as the benchmark for comparison. The evaluation results are detailed in section 4.4.

4.3.3.1 Dataset Creation

The dataset was generated by encoding a set of video sequences using the VVenC encoder under the RA configuration. During this process, CU features were extracted along with their corresponding split modes as labels, capturing all relevant textural and temporal characteristics of each CU. Each CU was labeled according to its split mode (e.g., QT, BTH, BTV, TTH, TTV, or No Split (NS)). Subsequently, separate datasets were created for training the five binary classifiers (C_{QT}, C_{BTH}, C_{BTV}, C_{TTH}, and C_{TTV}), with each dataset containing instances of the specific split mode under consideration, as well as instances with no split. This approach ensured balanced datasets with two classes: 'split' and 'no split', resulting in five combinations: (QT, NS), (BTH, NS), (BTV, NS),

(TTH, NS), and (TTV, NS).

Table 4.3 *Breakdown of CTU distribution across different resolution classes in the dataset*

Class	CTUs per frame	Encoded B-frames	CTUs per video	Number of videos	Total CTUs
A1	512	124	63488	4	253952
A2	480	124	59520	17	1011840
B	120	124	14880	9	133920
C	18	124	2232	9	20088
D	3	124	372	9	3348
E	50	124	6200	4	24800
Total	-	-	-	52	1447948

Table 4.3 details the final dataset, which consists of $1,447,948$ CTUs distributed across different resolution classes: A1 ($253,952$ CTUs), A2 ($1,018,840$ CTUs), B ($133,920$ CTUs), C ($20,088$ CTUs), D ($3,348$ CTUs), and E ($24,800$ CTUs). Higher resolution classes, like A1 and A2, contain more CTUs because these videos typically have fewer subdivisions into smaller CUs, requiring a larger sample size to achieve sufficient CUs for analysis. Conversely, lower resolution classes, such as C and D, have fewer CTUs but are more frequently subdivided into smaller CUs. It is important to note that the video sequences used to create this dataset are different from the CTC sequences used for evaluation. This difference helps the binary classifiers better generalize their predictions on new video data.

4.3.3.2 Binary Classifiers Training

As discussed in Chapter 3, Section 3.3.3.2 (page 76), LightGBM is an efficient gradient boosting framework developed by Microsoft, particularly useful for classification tasks [10]. Its effectiveness has been demonstrated in our previously proposed approach under AI. Table 4.4 presents the optimized hyperparameter values identified for this approach under the RA configuration.

The training process prioritizes balancing both the CU sizes and split modes to form a balanced dataset. This method enhances the effectiveness of training the LightGBM models and prevents classification issues related to data imbalance. The dataset was divided into three subsets: 70% for training, 15% for validation, and 15% for testing. These subsets are separate, with no overlap. The training set is used to train the model and help it learn from the data. The validation set is used to tune the model. The test set

Table 4.4 *Training parameters for LightGBM models*

Parameter Name	Parameter Setting
Learning rate	0.05
Maximum Depth	10
Number of leaves	32
Number of estimators	96
Boosting type	GBDT
Objective	binary

measures how well the model performs on new and unseen data. Following training and validation, the classifiers were integrated into the VVenC encoder 1.7.0, as outlined in Algorithm 4.3.2, to predict the split mode for CUs during the encoding process. Finally, the evaluation results of this solution are detailed in the following section 4.4.

4.4 Experimental Results

4.4.1 Implementation Details and Evaluation Metrics

The proposed algorithm is implemented in the slower, slow, and medium presets of the VVenC 1.7.0, and evaluated following the Joint Video Experts Team (JVET) common test conditions alongside the specified configurations indicated in Table 4.5. The evaluation is performed under RA configuration, with a Group of Pictures (GOP) size of 32, an I-frame period of 64, and QP values set to 22, 27, 32, and 37. This evaluation includes a variety of video sequences grouped into six different categories: A1 and A2 (4k), B (1080p), C (480p), D (240p), E (720p), and F (various resolutions).

Table 4.5 *Test and evaluation configurations of the proposed method*

Encoder	VVenC 1.7.0
Coding Configuration	Random Access (RA)
GOP size	32
Intra-Period	64
Reference Preset	Slower
Quantization Parameter (QP)	22, 27, 32, 37

To assess the performance of our approach, widely used metrics in the field of video coding were employed, including BDBR [39] and encoding time reduction (ETR), as

discussed in Section1.6.2. The BDBR calculation follows Equation 1.15 (page 29), while (ETR) is calculated using Equation 1.17 (page 30).

These two metrics are invaluable for comparing the effectiveness of new encoding algorithms or modifications to existing ones. Lower BDBR and encoding time values are desirable in scenarios with limited storage or bandwidth, such as streaming and mobile applications, as they indicate efficient resource use and suitability for real-time applications.

4.4.2 Classification Performance Evaluation

To assess the effectiveness of the binary classifiers, performance was assessed using standard ML metrics, including accuracy, precision, recall, and F1-score. These metrics are discussed in detail in Section 1.7.1 (page 33). As a result, the accuracy, precision, recall, and F1-Score of each classifier are summarized in Table 4.6.

Table 4.6 *Performance evaluation results for each binary classifier*

Classifier	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
C_{QT}	90.36	87.85	93.75	90.70
C_{BTH}	70.22	70.10	70.62	70.36
C_{BTV}	68.93	67.97	71.94	69.90
C_{TTH}	77.40	73.19	85.98	79.07
C_{TTV}	75.80	71.15	86.20	77.95

These classifiers were carefully trained using a combination of spatial and temporal features extracted from video sequences to determine if the CU should be split in one of five modes or left unsplit and encoded as it is. According to the assessment criteria, the QT classifier C_{QT} demonstrated outstanding performance, achieving an accuracy rate of 90.36% and an F1-Score of 90.70%. This is followed by the binary and ternary classifiers, with the highest accuracy reaching 77.40% (for C_{TTH}). Additionally, upon comparing the performance of these classifiers with those employed by Amestoy in [89], it is evident that the proposed classifiers showed superior accuracy on average.

Furthermore, these classifiers were evaluated using confusion matrices, as illustrated in Figure 4.4. The C_{QT} classifier outperformed the others, accurately predicting whether a CU should be split using the QT mode. In contrast, the BT and TT classifiers exhibited some confusion between their split modes, due to their close relationship. To reduce potential misclassifications, a strategy was implemented where the top-3 partition modes,

ranked by probability, are consistently evaluated. This approach ensures that the three most probable split modes are always considered, effectively minimizing losses in compression efficiency. Testing revealed that evaluating only the top-1 or top-2 modes led to quality degradation, with BDBR increases of approximately 8% and 5%, respectively, contributing to greater degradation in slower presets. To mitigate this, evaluating the top-3 split modes was chosen to reduce BDBR losses, thus maintaining better compression performance.

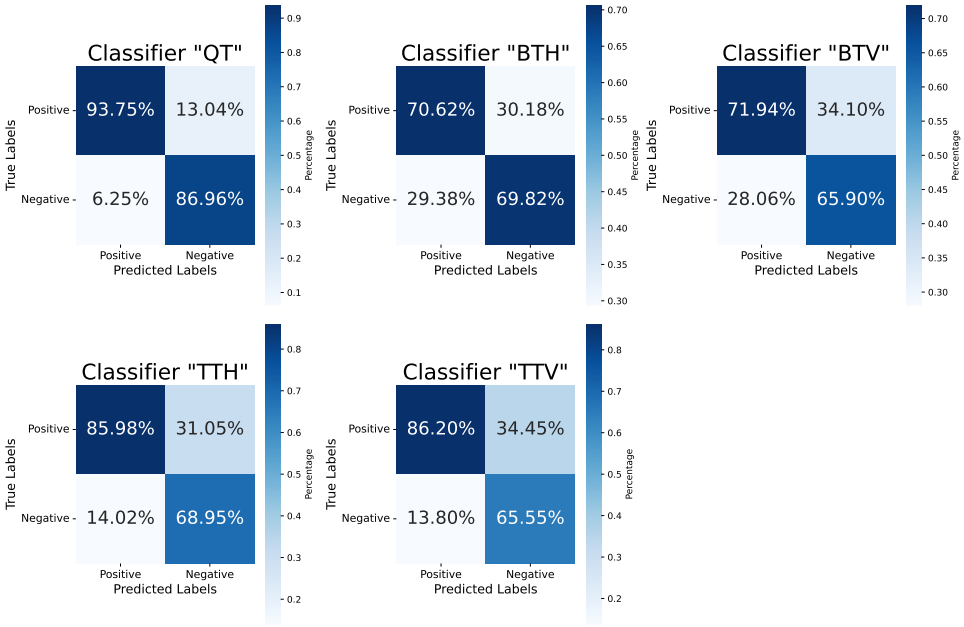


Figure 4.4 *Confusion matrices for each binary classifier*

4.4.3 Performance Evaluation of the Proposed VVenC Encoder Presets

This section analyzes various presets (slower, slow, medium) over the CTC video classes (A1, A2, B, C, D, E, F). Table 4.7 illustrates the trade-offs between encoding efficiency, measured by BDBR, and ETR, measured by ΔETR. Results are shown for the CTC sequences from class A1 to class F. The table provides the average results for each class individually, as well as for all sequences combined from class A1 to F. Each preset corresponds to different encoding speeds and efficiency levels, providing an in-depth analysis of the performance range of the VVenC encoder. The slower preset serves as the baseline for our comparison, as depicted in Figure 4.5. When comparing the slow

preset to this baseline, it demonstrated a 5.78% increase in BDBR, indicating reduced coding efficiency. However, this trade-off resulted in a significant reduction in encoding time, achieving up to a 74.38% decrease across all CTC classes. The primary goal of VVC development is to improve the quality of high-resolution videos [7]. Simplifying the encoding process for these high-resolution videos is crucial, as the time required for encoding grows with the resolution. classes A1 and A2 include 4K resolution sequences, while class B includes Full-HD sequences.

Table 4.7 *Trade-offs between ETR(%) and BDBR(%): performance evaluation of proposed VVenC presets vs. benchmark slower original preset*

	Video	Slower Proposed		Slow Original		Slow Proposed		Medium Original		Medium Proposed	
		BDBR (%)	ΔETR (%)	BDBR (%)	ΔETR (%)	BDBR (%)	ΔETR (%)	BDBR (%)	ΔETR (%)	BDBR (%)	ΔETR (%)
A1	Campfire	2.18	50.60	3.10	70.70	4.35	77.45	8.14	91.11	9.45	92.14
	Tango2	2.59	36.13	5.01	73.38	6.53	78.66	11.56	91.41	13.21	91.86
	FoodMarket4	1.51	33.54	4.80	72.19	5.83	77.02	9.24	89.62	9.96	90.13
	Class Average	**2.09**	**40.09**	**4.30**	**72.09**	**5.57**	**77.71**	**9.65**	**90.71**	**10.87**	**91.38**
A2	DaylightRoad2	3.58	39.43	8.71	73.40	11.29	78.87	16.79	91.88	19.49	92.30
	CatRobot1	3.17	39.98	9.13	73.32	11.58	79.38	16.38	91.83	18.74	92.26
	ParkRunning3	1.78	52.85	4.14	75.13	4.96	79.68	8.69	92.46	9.39	93.09
	Class Average	**2.84**	**44.09**	**7.33**	**73.95**	**9.28**	**79.31**	**13.95**	**92.06**	**15.87**	**92.55**
B	MarketPlace	2.36	43.37	5.63	74.01	6.86	79.68	10.50	91.87	11.72	92.53
	RitualDance	2.67	44.25	4.77	72.29	6.26	78.76	11.08	91.51	12.68	92.19
	BQTerrace	2.78	41.28	5.54	76.68	7.22	80.91	11.10	92.25	12.79	92.66
	BasketBallDrive	3.19	47.95	5.80	75.45	7.78	80.06	13.70	92.84	15.85	93.37
	Cactus	2.71	46.25	5.46	76.01	7.01	80.88	11.48	92.37	13.03	93.05
	Class Average	**2.74**	**44.62**	**5.44**	**74.89**	**7.03**	**80.06**	**11.57**	**92.17**	**13.21**	**92.76**
C	PartyScene	2.79	47.18	5.57	75.79	6.99	80.81	10.78	92.72	12.21	93.05
	BQMall	3.81	45.30	6.16	76.42	8.36	81.86	13.53	93.21	15.96	93.54
	BasketBallDrill	3.46	46.32	4.99	76.52	6.87	81.00	11.94	93.03	13.52	93.34
	RaceHorsesC	3.36	47.13	6.50	70.96	8.51	78.58	14.20	92.49	15.98	93.07
	Class Average	**3.36**	**46.48**	**5.81**	**74.92**	**7.68**	**80.56**	**12.61**	**92.86**	**14.42**	**93.25**
D	BQSquare	2.79	32.61	5.99	80.55	7.57	84.15	11.25	92.69	12.51	92.97
	BlowingBubbles	2.84	44.94	5.09	78.99	6.53	83.40	10.77	93.59	12.32	93.68
	BasketBallPass	3.52	47.98	4.91	77.54	6.78	82.39	11.92	93.81	13.73	93.96
	RaceHorsesD	3.57	44.46	5.28	73.25	7.33	78.96	12.74	92.46	14.69	92.89
	Class Average	**3.18**	**42.50**	**5.32**	**77.58**	**7.05**	**82.23**	**11.67**	**93.14**	**13.31**	**93.38**
E	FourPeople	2.90	42.04	4.36	77.72	6.22	81.59	9.50	92.06	11.41	92.25
	Johnny	2.22	31.98	4.56	77.91	6.18	81.05	9.81	90.73	11.49	90.76
	KristenAndSara	2.49	37.45	4.77	77.43	6.47	80.77	9.71	91.31	11.55	91.44
	Class Average	**2.54**	**37.16**	**4.56**	**77.69**	**6.29**	**81.14**	**9.67**	**91.37**	**11.48**	**91.48**
F	ArenaOfValor	3.59	52.21	6.10	76.49	8.46	80.52	13.82	92.80	16.12	93.60
	SlideEditing	3.41	39.92	2.81	53.90	5.29	65.65	11.64	82.67	13.65	84.66
	BasketBallDrillText	3.68	53.16	6.12	79.19	8.33	83.26	13.71	93.92	15.85	94.26
	SlideShow	3.63	44.81	15.68	68.62	18.97	74.73	29.35	89.42	31.79	90.14
	Class Average	**3.58**	**47.53**	**7.68**	**69.55**	**10.26**	**76.04**	**17.13**	**89.70**	**19.35**	**90.67**
	Total Average	**2.90**	**43.21**	**5.78**	**74.38**	**7.59**	**79.58**	**12.32**	**91.72**	**14.08**	**92.21**

Table 4.7 shows that for high-resolution classes A1, A2, and B, the slow preset ex-

hibited bitrate increases of 4.30%, 7.33%, and 5.44%, respectively. Despite the increase in bitrate, the slow preset offered an overall ETR of 73.64%, while the proposed slow preset further improved this with a 79.02% reduction compared to the slower reference. When directly comparing the slow and proposed slow presets, our method optimized the slow preset by providing a time-saving of 20% across all CTC classes, with only a modest 1.71% increase in BDBR. This indicates a more efficient trade-off between coding efficiency and encoding time. The medium preset offers a higher BDBR of 12.32% and a runtime reduction of 91.72% across all classes, highlighting its prioritization of speed over efficiency.

Table 4.7 clearly illustrates how class averages indicate the effectiveness of each preset. Slow presets balance speed and efficiency, suitable for applications requiring a compromise between these factors. In contrast, the medium presets prioritize speed, evident from their higher BDBR and faster encoding times. This analysis underscores the importance of selecting the appropriate preset based on application requirements, whether focusing on minimizing encoding time or maximizing compression efficiency. Table 4.7 also presents the BDBR and ETR of the proposed solution, reflecting performance across different scenarios. Compared to the original VVenC encoder, our method consistently achieves a better balance between reducing encoding time and maintaining a minimal increase in BDBR. For the slower preset, the proposed method consistently achieves significant improvements in ETR across all video classes. For instance, the average ETR improvement across class A1 videos is 40.09%, while class A2 and class B show enhancements of 44.09% and 44.62%, respectively. These improvements are crucial for applications requiring faster encoding times without compromising too much on quality. However, the BDBR values reveal a trade-off in bitrate efficiency. While the proposed method exhibits a slight increase in BDBR across all classes, the increases are relatively modest. For instance, the class A1 average BDBR is 2.09%, and for class A2, it is 2.84%. Class B, C, D, and E show BDBR increases of 2.74%, 3.36%, 3.18%, and 2.54%, respectively. The highest BDBR increase is observed in class F, with an average of 3.58%, suggesting a more significant impact on bitrate for certain types of content. These classes feature more complex motion and textures, demonstrating the preset's ability to handle such scenarios efficiently. Overall, the proposed slower preset achieves substantial time savings, with a maximum runtime reduction of 53.16% with the *BasketBallDrillText* sequence and a minimum reduction of 31.98% with the *Johnny* sequence, averaging 43.21% across all sequences. This preset only marginally increases the BDBR, with a maximum increase of 3.81% for the *BQMall*

sequence of class C, and a minimum increase of 1.51% for the *FoodMarket4* sequence, resulting in an overall average BDBR increase of just 2.9%. Thus, the proposed slower preset optimizes efficiency by significantly reducing encoding time while maintaining excellent compression performance.

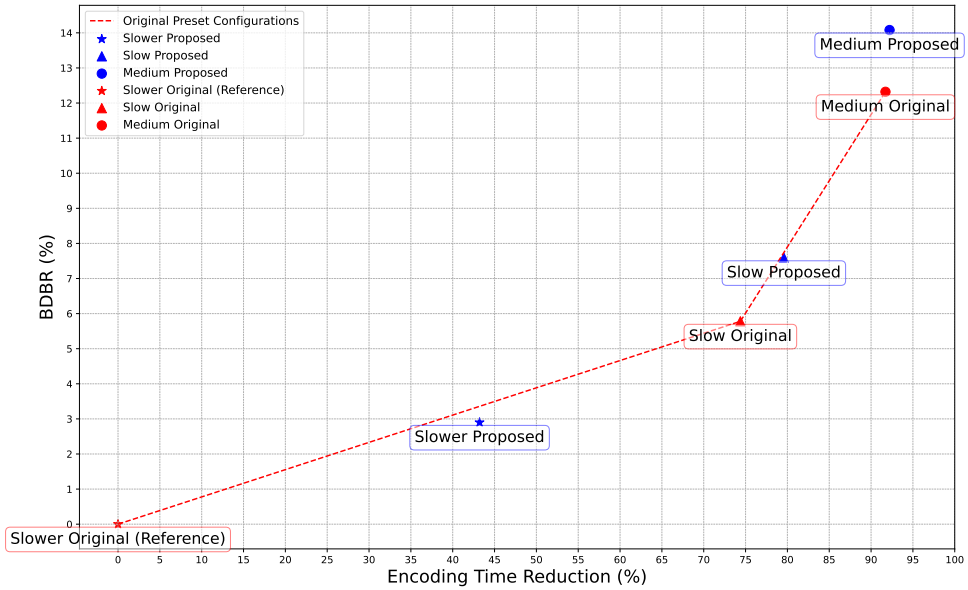
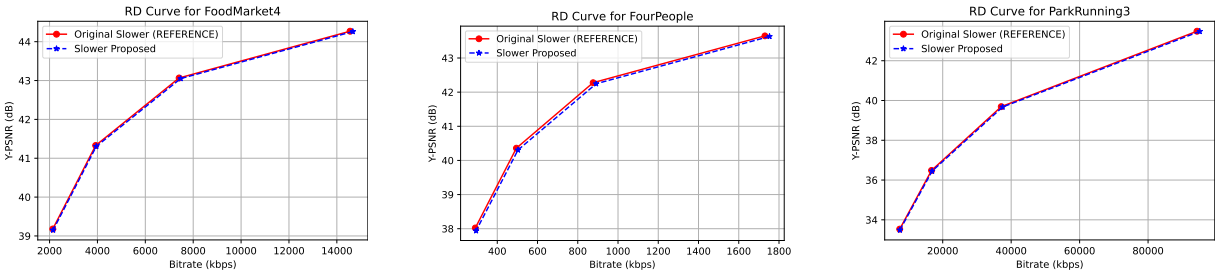


Figure 4.5 *Total average BDBR vs. ETR for original and proposed VVenC presets*

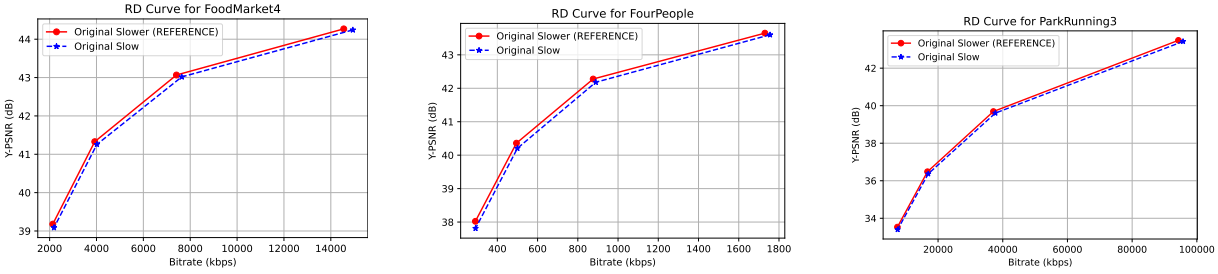
Figure 4.5 illustrates that the proposed slower preset enhances performance by decreasing reference encoding time without compromising compression efficiency. The modest increase in BDBR suggests minimal impact on quality, underscoring the effectiveness of this approach in delivering both efficient and rapid video encoding. The effectiveness of our method in handling complex motion patterns is especially noticeable in fast-moving scenes like *ArenaOfValor* and *SlideShow* of class F, showing a reduction in encoding time of over 52%. In Figure 4.5, the blue star representing the proposed slower preset is well positioned compared to the red symbols representing the original VVenC presets, indicating a better balance between ETR and BDBR. However, the proposed slow and medium presets are less efficient due to the MTT depths of each preset, making these presets less optimized compared to the slower ones.

Figure 4.6 presents objective video quality assessments using RD curves to validate the effectiveness of the proposed encoding method. The original slower serves as a reference and is compared against the proposed slower, as well as the VVenC encoder's slow and medium presets. This comparison demonstrates that the proposed method maintains high

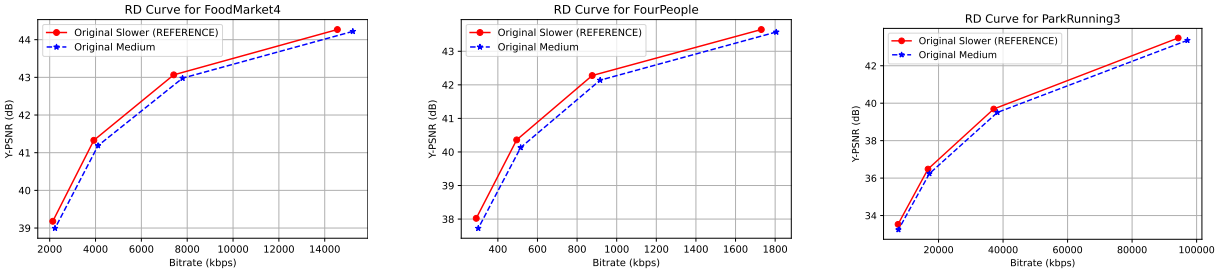
video quality while effectively reducing complexity. The analysis includes videos from high-resolution classes A1, A2, and B, specifically *FoodMarket4*, *FourPeople*, and *ParkRunning3* respectively. The RD curve for the proposed slower closely aligns with the original slower, indicating comparable compression efficiency. In contrast, the RD curve for the original slow shows a slight divergence, and the RD curve for the original medium exhibits a more significant deviation. These results suggest that the proposed slower achieves a more favorable balance between compression efficiency and encoding complexity, making it a more effective solution.



(a) RD curves of proposed slower



(b) RD curves of original slow



(c) RD curves of original medium

Figure 4.6 *RD performance comparison: proposed slower in (a), original slow in (b), and original medium in (c) for classes A1, A2, and B sequences*

4.4.4 Comparison with Existing Methods in the Literature

In this section, we compare our solution with some state-of-the-art techniques. Most methods proposed in the literature were implemented on the VTM encoder, as detailed in Table 4.2.

Table 4.8 *Comparative analysis of trade-offs between ETR(%) and BDBR(%): evaluating the performance of the proposed approach and some state-of-the-art methods*

Class	Yue Li [91]		Tissier [98]		Jingyi Li [92]		Proposed Slower	
	BDBR (%)	**ΔETR** (%)	**BDBR** (%)	**ΔETR** (%)	**BDBR** (%)	**ΔETR** (%)	**BDBR** (%)	**ΔETR** (%)
A1	1.05	28.08	1.81	51.10	1.59	16.61	2.09	40.09
A2	0.92	22.88	1.86	44.60	2.29	15.36	2.84	44.09
B	1.01	26.16	2.21	46.50	1.80	14.04	2.74	44.62
C	1.30	28.56	3.20	43.10	-	-	3.36	46.48
D	1.01	20.50	3.02	36.80	-	-	3.18	42.50
E	0.36	10.06	1.45	38.70	-	-	2.54	37.16
F	-	-	2.96	40.20	-	-	3.58	47.53
Total Average	**0.97**	**23.19**	**2.36**	**43.00**	**1.87**	**15.27**	**2.90**	**43.21**

Table 4.8 presents a comparative analysis of the trade-offs between ETR and BDBR across various video sequence resolutions and categories. Our method utilizes the "slower proposed" setting, which closely aligns with the VTM and employs all its performance tools [19]. This comparison benchmarks against approaches by Yue Li [91], Tissier [98], and Jingyi [92].

Our method achieves an average time-saving of 43.21%, outperforming Yue Li's (23.19%) and Jingyi's (15.27%), and slightly exceeding Tissier's (43%). When compared to Yue Li's approach, which employs a temporal prediction model, our method offers approximately double the time-saving at the cost of increased bitrate. Furthermore, our results are comparable to Tissier's, which uses CNN and VTM, demonstrating the effectiveness of our method. It is worth noting that achieving efficiency in VVenC is more challenging than in VTM, as VVenC is already optimized with various heuristics to reduce encoding time [19]. Moreover, our method consistently delivers the best trade-offs, particularly for more complex categories (C, D, and F), resulting in significant ETRs (44.62%, 42.50%, and 47.53%, respectively) with a slight increase in BDBR. Notably, for class F sequences across various resolutions, our method surpasses others by achieving greater time-saving while maintaining a moderate 3.58% increase in BDBR.

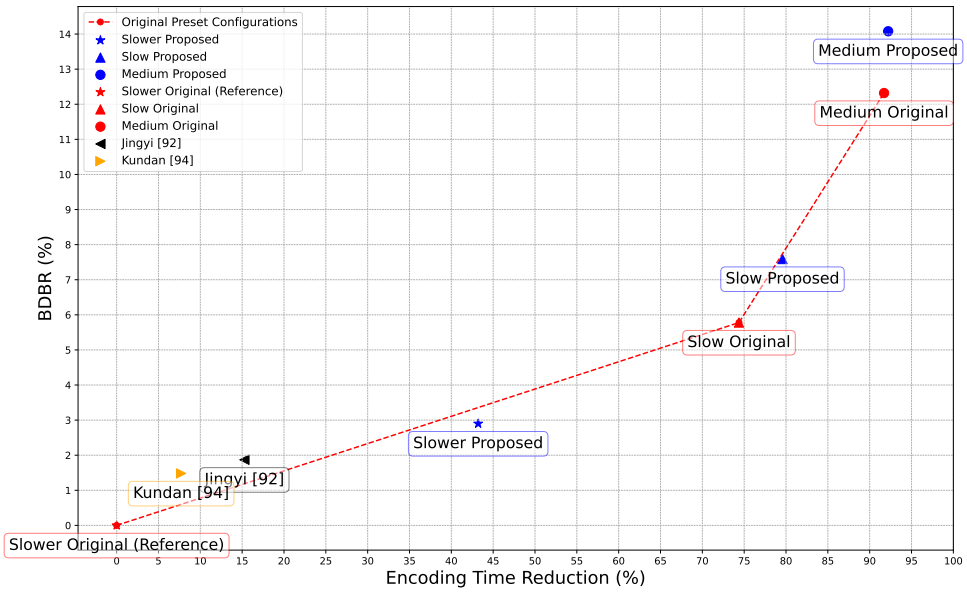


Figure 4.7 *Performance comparison: original and proposed presets compared to some state-of-the-art methods implemented on VVenC*

To ensure a fair comparison, our approach has been compared to notable studies using VVenC, specifically those by Jingyi Li [92] and Kundan [94]. These studies report reductions in encoding time of 15.27% and 7.71%, respectively, with bitrate increases of 1.87% and 1.48%. Compared to these results, our approach achieves a significantly greater ETR of 43.21% across all CTC classes, with a reasonable BDBR increase of 2.9%. For high-resolution classes, we achieved an ETR of 42.93% with a BDBR increase of 2.55%. Figure 4.7 illustrates that these state-of-the-art methods do not achieve more than a 16% reduction in encoding time.

In contrast, our approach offers significant time savings with only a slight increase in bitrate. Notably, there is a lack of studies focused on optimizing the VVenC encoder, with only the aforementioned two studies found. These results underscore the effectiveness of our approach in balancing computational complexity with compression efficiency, surpassing other techniques to achieve a superior trade-off essential for practical encoding scenarios.

4.5 Conclusion

This chapter presented a novel approach for optimizing VVenC under the RA configuration by reducing the complexity of CU partitioning. The method leverages coding and motion information across inter-frames to predict CU partition modes directly, bypassing the traditional RDO process in VVC. By employing LightGBM binary classifiers, split modes are accurately predicted, significantly accelerating the encoding process with minimal impact on the bitrate. Experimental results confirmed the effectiveness of this approach, achieving a runtime reduction of 43.21% with only a 2.9% increase in bitrate. This outcome demonstrates that the proposed method not only reduces computational demands for VVC encoding but also outperforms several state-of-the-art techniques in efficiency. These findings contribute valuable advancements to video coding, enhancing the feasibility of the VVC standard for real-time applications where speed and efficiency are essential.

Conclusion and Perspectives

The rapid growth in video content consumption, coupled with increasing demand for higher-quality and immersive media experiences, has necessitated the development of more efficient video compression techniques. However, advancements in compression efficiency have introduced significant computational complexities, particularly in the Versatile Video Coding (VVC) standard, which poses challenges for real-time video encoding and streaming applications. This heightened complexity, largely due to the sophisticated partitioning structure within VVC, limits its feasibility for widespread and real-time use. Various fast partitioning methods have been proposed to address this challenge and effectively reduce encoding demands. However, most efforts have focused on optimizing the VVC Test Model (VTM) encoder, which remains far from suitable for real-time applications due to its high computational requirements. In contrast, this thesis leverages the Versatile Video Encoder (VVenC) encoder, an optimized and practical implementation of VVC designed to bring real-time encoding feasibility closer. By targeting the VVenC encoder, this work addresses real-time constraints more effectively.

In this context, the research presented in this thesis has tackled critical challenges in video coding, with a specific focus on optimizing VVenC. Throughout this work, we have contributed to ongoing efforts to optimize video coding by proposing novel methods that reduce the computational complexity of VVenC while maintaining high coding efficiency. The first contribution, a statistical analysis and performance evaluation of VVenC, examined the encoder's operational characteristics across different coding modes and partitioning block sizes in both intra and inter-coding. This study also assessed VVenC's performance on different hardware platforms, including ARM and x86, to evaluate its potential for real-time encoding and the benefits of using multi-threading. This foundational study identified areas where targeted optimizations could yield significant reductions in computational load such as the partitioning block, guiding the subsequent optimization efforts.

The second contribution introduced a machine learning-based fast Quadtree with Multi-Type Tree (QTMTT) partitioning strategy specifically for intra-coding under the All Intra (AI) configuration. By employing Light Gradient Boosting Machine (LightGBM)

classifiers to predict optimal partitioning decisions. These classifiers are trained to predict the most probable partitioning mode, thereby reducing the need for the exhaustive Rate-Distortion Optimization (RDO) process typically required. By integrating these classifiers into the VVenC encoder, the proposed approach achieves significant reductions in encoding time, with up to 31% savings for the slower preset and up to 70% for the proposed medium preset. This is achieved while maintaining a balance between speed and coding efficiency, resulting in a slight increase in Bjøntegaard Delta Bit Rate (BDBR) of approximately 0.67% and 1.89% for the respective presets. The results of this work demonstrate significant time reductions in encoding processes with only a slight increase in bitrate.

The third contribution builds upon the optimization efforts for the VVenC encoder by focusing on reducing inter-coding complexity in the Random Access (RA) configuration. It introduces a low-complexity, learning-based QTMTT partitioning scheme that uses lightweight binary classifiers and inter-frame coding information to simplify the encoding process. This approach achieved significant reductions in runtime, with time savings of up to 43%, while maintaining an acceptable trade-off in coding efficiency, reflected by a BDBR increase of approximately 2.9%. These results demonstrate the effectiveness of the method in accelerating inter-coding processes with minimal impact on bitrate performance.

In conclusion, this thesis has established a robust foundation for optimizing the VVenC encoder, making significant contributions to the broader field of video compression. By developing and implementing machine learning-based approaches for Coding Unit (CU) partitioning in both intra and inter-coding scenarios, this work paves the way for further advancements in next-generation video coding standards. The integration of LightGBM classifiers to predict optimal partitioning decisions has demonstrated substantial reductions in encoding time while maintaining high compression efficiency, highlighting the potential of machine learning techniques in enhancing video encoding processes.

Despite these contributions, there remain several promising directions for future research to further enhance the efficiency and practicality of video encoders like VVenC, such as:

— **Real-Time Encoding Enhancements:** Future work could explore the integration of more advanced techniques, such as optimized Deep Learning (DL) models, to further improve the accuracy of partitioning decisions and reduce encoding time. Additionally, optimizing the encoder for specific hardware architectures, such as GPUs or FPGAs, could lead to even greater gains in real-time encoding performance.

— **Enhancing the Fast Presets of the VVenC:** Future work could improve the fast and faster presets by selectively enabling Multi-Type Tree (MTT) partitioning in high-confidence scenarios predicted by lightweight Machine Learning (ML) classifiers. This hybrid approach would leverage Quadtree (QT) for most cases to maintain encoding speed, while applying MTT only when significant coding efficiency gains are expected.

— **Adaptive Streaming and Multi-Rate Coding:** Extending the current research to address the challenges of adaptive streaming could be highly beneficial. Developing methods that optimize encoding efficiency across different bitrates and resolutions, particularly in multi-rate streaming scenarios, would help to meet the growing demands for adaptive video streaming in varying network conditions.

— **Energy Efficiency:** With the growing concern over energy consumption in data centers and mobile devices, future research could focus on developing energy-efficient video codecs. This could involve optimizing the VVenC to minimize energy usage without compromising on video quality, potentially through the use of low-power ML/DL models or by refining existing coding tools.

— **Emerging Video Formats:** As new video formats such as 360-degree video, Virtual Reality (VR), and Augmented Reality (AR) become more popular, the need for efficient encoding of these complex formats will grow. Future work could explore how the principles developed in this thesis can be adapted or extended to support the encoding of these emerging formats while maintaining low latency and high quality.

— **Hardware Implementation:** Another promising direction is the exploration of hardware acceleration for the proposed optimizations. Hardware implementations can provide the low-latency processing needed for real-time applications by offloading computationally intensive tasks from software to hardware.

— **End-to-End Video Coding Frameworks:** The development of end-to-end video coding frameworks is another critical area for future exploration. Current research often focuses on optimizing individual components of video coding, such as partitioning or motion estimation. However, there is growing interest in designing unified, end-to-end frameworks where all stages of video coding (from pre-processing to post-processing) are co-optimized using deep learning models.

These perspectives open up exciting opportunities for future research that could further advance the state-of-the-art in video coding technology. They hold promise for con-

tinuing to push the boundaries of video compression technology, ultimately contributing to more efficient and higher-quality video transmission and storage solutions.

FRENCH SUMMARY

5.1 Contexte Général

Ces dernières années, il y a eu un changement significatif dans la façon dont les gens consomment les médias, principalement en raison de l'essor des services de vidéo à la demande tels que Netflix et YouTube, des plateformes de streaming en direct comme Twitch et Kick, et de l'impact généralisé des réseaux sociaux tels que Facebook, Instagram et X (anciennement Twitter). En conséquence, le contenu vidéo domine désormais le trafic internet mondial. En 2022, Cisco a rapporté que le trafic vidéo constituait plus de 82% du trafic internet mondial, et ce chiffre a continué d'augmenter avec l'adoption croissante des formats haute résolution tels que la 4K et la 8K [1]. Selon un rapport de 2023 de Sandvine, le trafic vidéo a connu une augmentation de 24% en 2022, représentant 65% de tout le trafic internet au début de 2023 [2]. Cette tendance se reflète également dans l'utilisation des données mobiles, avec le rapport Ericsson Mobility estimant que la vidéo représentera 71% de tout le trafic de données mobiles en fin de 2023, une proportion qui devrait atteindre 80% d'ici 2028 [3]. L'ampleur mondiale de la vidéo en ligne est encore soulignée par un rapport de Statista, qui a révélé qu'à la fin de 2023, 92% des utilisateurs d'internet dans le monde avaient interagi avec du contenu vidéo en ligne [4], et que le nombre de spectateurs de vidéos en direct devrait atteindre 164,6 millions [5].

Cette explosion de la consommation de vidéos a été facilitée par des avancées significatives dans l'infrastructure réseau, notamment le déploiement généralisé des technologies 4G, 5G et fibre optique, qui ont considérablement augmenté la bande passante disponible. Cependant, le volume massif de données vidéo, combiné à la demande croissante de contenu de plus haute qualité et immersif, pose des défis importants en matière de bande passante, de stockage et de ressources computationnelles. Par exemple, une image vidéo 8K (Ultra High Definition (UHD)) contient 33 millions de pixels, nécessitant une bande passante bien plus importante que le contenu en Standard Definition (SD).

Pour répondre à ces défis, le développement de techniques de compression vidéo efficaces est devenu essentiel. Les codecs vidéo, responsables de la compression et de la décompression des fichiers vidéo, sont essentiels pour équilibrer la qualité de la vidéo et

la taille des fichiers. Au fil des ans, les normes de codage vidéo ont considérablement évolué, allant des premières versions comme le H.261 à des normes plus avancées telles que le High Efficiency Video Coding (HEVC) [6] et la plus récente Versatile Video Coding (VVC) [7]. Finalisée en juillet 2020, la norme VVC permet environ 40% d'économies de débit par rapport à l'HEVC tout en maintenant une qualité vidéo comparable, ce qui la rend particulièrement bien adaptée à la transmission de contenu en haute résolution [7].

Cependant, l'efficacité accrue de la compression de la VVC s'accompagne d'une augmentation significative de la complexité computationnelle. Le processus d'encodage de la VVC est estimé être de 6,6 à 25,8 fois plus long que celui de l'HEVC, selon les configurations d'encodage spécifiques utilisées [7]. Cette complexité accrue constitue un obstacle majeur à l'adoption généralisée de la norme VVC, en particulier dans les scénarios de streaming vidéo en temps réel où la faible latence est cruciale.

5.2 Motivations et Objectifs

Alors que la vidéo continue de jouer un rôle central dans les environnements grand public et professionnels, la nécessité de techniques de compression vidéo efficaces devient de plus en plus critique. En particulier, la norme VVC, bien qu'elle offre des améliorations significatives en termes d'efficacité de compression, introduit une complexité computationnelle importante qui pose des défis pour les applications en temps réel. Réduire cette complexité sans compromettre l'efficacité de la compression est essentiel pour permettre des expériences vidéo fluides, notamment dans les scénarios nécessitant une faible latence.

Cette thèse est consacrée à l'optimisation de l'un des aspects les plus gourmands en ressources computationnelles de la VVC : le processus de partitionnement. Selon [8], ce processus montre un potentiel élevé de réduction de la complexité, représentant 97% de la complexité totale de l'encodage. L'objectif principal est d'accélérer le processus d'encodage de la VVC en développant et en implémentant des algorithmes de partitionnement rapide basés sur des modèles d'apprentissage automatique. Ces algorithmes visent à réduire le temps d'encodage tout en préservant la haute efficacité de codage pour laquelle la VVC est reconnue. Le but ultime est de rendre l'encodage en temps réel réalisable, en minimisant les compromis entre la rapidité et la qualité vidéo.

Suite à la normalisation de VVC, l'équipe JVET a publié le logiciel VVC Test Model (VTM) comme une implémentation de référence. Bien que VTM soit très efficace pour le benchmarking et l'évaluation des performances des codecs, il souffre de temps d'encodage

significativement plus longs en raison de sa nature non optimisée. En revanche, cette recherche utilise l'encodeur appelé Versatile Video Encoder (VVenC) [9], développé par l'Institut Fraunhofer Heinrich Hertz (HHI). Le VVenC est une implémentation optimisée de la norme VVC, basée sur VTM, et offre des performances comparables tout en réduisant considérablement le temps d'encodage [9]. Ses optimisations, conçues pour des applications en temps réel et pratiques, font du VVenC un choix idéal pour notre recherche, offrant à la fois efficacité et performance.

En se concentrant sur les configurations All Intra (AI) et Random Access (RA) de l'encodeur VVenC, cette thèse propose des stratégies de partitionnement innovantes conçues pour améliorer l'efficacité de l'encodage. Ces optimisations ont été intégrées dans l'encodeur VVenC, et des tests approfondis ont démontré que les méthodes proposées surpassent les approches existantes. Les résultats ouvrent la voie à l'encodage en temps réel avec VVenC, offrant une solution pratique aux défis posés par la demande croissante de consommation vidéo moderne.

En abordant l'équilibre critique entre la qualité vidéo, la vitesse d'encodage et la complexité computationnelle, cette thèse contribue à l'avancement continu des technologies de compression vidéo, garantissant qu'elles répondent aux besoins évolutifs des environnements médiatiques actuels et futurs.

5.3 Contributions

Cette thèse propose trois contributions principales, chacune se concentrant sur des optimisations significatives de l'encodeur VVenC. L'analyse statistique sert de base, examinant la fréquence et la complexité des différents modes de codage et tailles de blocs de partitionnement dans les configurations de codage intra et inter au sein de l'encodeur VVenC. Cette analyse identifie les domaines clés où l'optimisation peut réduire considérablement la complexité computationnelle. Étant donné que l'encodeur VVenC est un codec vidéo relativement nouveau, les études sur la réduction de sa complexité, en particulier concernant les optimisations de partitionnement de blocs, sont limitées. En comblant cette lacune, notre recherche se concentre sur l'accélération du processus d'encodage dans VVenC grâce à des optimisations ciblées dans les configurations AI et RA. Nos contributions ont été intégrées dans le VVenC, et des tests approfondis montrent que nos méthodes surpassent les approches existantes, ouvrant la voie à l'encodage en temps réel avec VVenC. Les trois contributions principales sont détaillées ci-dessous :

I. Analyse Statistique et Évaluation des Performances de l'Encodeur VVenC La première contribution de cette thèse est une analyse statistique et une évaluation des performances de l'encodeur VVenC, une implémentation open-source et optimisée de la norme VVC [9]. Cette analyse commence par une introduction à l'encodeur VVenC, mettant en évidence ses caractéristiques uniques, ses innovations et la configuration de ses préréglages. L'étude utilise une méthodologie robuste, s'appuyant sur divers environnements de test et ensembles de données pour garantir un examen approfondi des performances de l'encodeur. Les principaux indicateurs de performance sont définis et utilisés pour comparer l'encodeur VVenC, offrant des perspectives sur son efficacité dans divers scénarios d'encodage, résolutions et débits binaires. Ce chapitre examine aussi les performances de l'encodeur VVenC sur des architectures matérielles incluant ARM et x86, afin de déterminer dans quelle mesure VVenC se rapproche du temps réel et d'identifier les gains obtenus en augmentant le nombre de threads. Les résultats de cette analyse fournissent une compréhension détaillée des caractéristiques opérationnelles du VVenC et identifient des domaines spécifiques où des optimisations supplémentaires peuvent entraîner des améliorations significatives en termes de vitesse d'encodage et de qualité. Ce travail fondamental prépare le terrain pour les efforts d'optimisation détaillés dans les chapitres suivants, garantissant que l'encodeur VVenC puisse répondre aux exigences des applications vidéo modernes.

II. Stratégie de Partitionnement Rapide QTMTT Basée sur l'Apprentissage Automatique pour le Codage Intra dans l'Encodeur VVenC La deuxième contribution clé de cette thèse est le développement d'une stratégie innovante basée sur l'apprentissage automatique pour optimiser le processus de partitionnement Quadtree with Multi-Type Tree (QTMTT) au sein de l'encodeur VVenC, spécifiquement pour les configurations de codage intra AI. La norme VVC apporte des améliorations significatives en termes d'efficacité de compression par rapport à son prédécesseur, HEVC, mais introduit également une complexité computationnelle considérable, notamment en raison de sa structure avancée de blocs de partitionnement. Pour réduire cette complexité, nous avons proposé un algorithme de partitionnement rapide des unités de codage (CU) utilisant des classificateurs Light Gradient Boosting Machine (LightGBM) [10]. Ces classificateurs sont entraînés à prédire le mode de partitionnement le plus probable, réduisant ainsi la nécessité du processus exhaustif de Rate-Distortion Optimization (RDO) habituellement requis. En intégrant ces modèles dans l'encodeur VVenC, notre approche réduit efficace-

ment le temps d'encodage tout en maintenant un équilibre entre gain de temps et efficacité de codage. Les résultats de ce travail démontrent des réductions significatives du temps d'encodage, variant de 30,21% dans le préréglage le plus lent (slower) à 82,46% dans le préréglage moyen (medium), avec seulement une légère augmentation du débit binaire. Cette contribution non seulement améliore les performances de l'encodeur VVenC, mais représente également une avancée substantielle vers la faisabilité de l'encodage vidéo en temps réel pour des contenus vidéo haute résolution et complexes.

III. Schéma de Partitionnement QTMTT à Faible Complexité Basé sur l'Apprentissage Automatique pour le Codage Inter dans l'Encodeur VVC La troisième contribution majeure de cette thèse concerne le développement d'un schéma de partitionnement QTMTT à faible complexité pour le codage inter dans l'encodeur VVenC, spécifiquement dans la configuration RA. La norme VVC intègre des avancées significatives en matière d'efficacité de compression vidéo, notamment grâce à sa structure complexe QTMTT. Bien que cette structure améliore grandement les performances de compression, elle introduit également des défis computationnels considérables, en particulier pour les applications en temps réel. Pour relever ces défis, nous proposons une stratégie de partitionnement optimisée qui exploite les techniques d'inter-prédiction, en incorporant à la fois des informations de codage et de mouvement des images inter-frames pour accélérer le processus d'encodage. Cette approche utilise une série de classificateurs légers LightGBM pour prédire le mode de découpage optimal pour chaque Coding Unit (CU), réduisant ainsi la nécessité des calculs exhaustifs. Implémentée dans l'encodeur VVenC, notre méthode a démontré une réduction significative du temps d'exécution, atteignant des gains de temps de 43,21% dans le préréglage le plus lent, avec seulement une augmentation minimale du débit binaire de 2,9%. Cette contribution améliore non seulement l'efficacité de l'encodeur VVenC, mais offre également une solution pratique pour réduire la complexité computationnelle dans les scénarios d'encodage vidéo à forte demande, tout en faisant progresser les capacités de la norme VVC.

5.4 Plan du Manuscrit

Cette section présente l'organisation du manuscrit de thèse. Il est structuré en une introduction générale, une conclusion générale, ainsi que quatre chapitres principaux. Chaque chapitre explore un aspect distinct de la compression vidéo, en mettant un accent

particulier sur l'optimisation de la norme VVC grâce à des approches innovantes basées sur l'apprentissage automatique (ML).

Le manuscrit commence par une introduction générale qui introduit la thèse en fournissant le contexte général, en discutant des motivations et en présentant les principales contributions de cette recherche.

Chapitre 1 fournit un aperçu complet du codage vidéo. Le chapitre commence par une introduction aux caractéristiques fondamentales de la vidéo, y compris des aspects tels que l'espace colorimétrique, les résolutions spatiales et temporelles, et la profondeur de bits. Ensuite, il retrace l'évolution des normes de codage vidéo, avec un accent particulier sur les normes HEVC et VVC. Ce chapitre propose également une analyse détaillée du processus de codage VVC, couvrant les principales étapes tels que les schémas de partitionnement, les modes de codage intra et inter-prédiction, le RDO et d'autres techniques pertinentes. Par ailleurs, une attention particulière est accordée au rôle de l'apprentissage automatique et de l'apprentissage profond dans la compression vidéo. Les avantages potentiels de ces approches sont mis en lumière, tout en examinant les défis liés aux exigences computationnelles et aux besoins importants en données.

Chapitre 2 se concentre sur l'analyse statistique et l'évaluation des performances de l'encodeur VVenC. Ce chapitre présente l'encodeur VVenC, en détaillant ses caractéristiques et configurations. Il propose également une analyse comparative approfondie des performances du VVenC dans divers scénarios d'encodage, résolutions et débits binaires, établissant une base pour comprendre où des optimisations supplémentaires peuvent être apportées. Ce chapitre examine aussi les performances de l'encodeur VVenC sur des architectures matérielles incluant ARM et x86, afin de déterminer dans quelle mesure VVenC se rapproche du temps réel et d'identifier les gains obtenus en augmentant le nombre de threads.

Chapitre 3 présente une stratégie basée sur l'apprentissage automatique pour le partitionnement rapide QTMTT au sein de l'encodeur VVenC, spécifiquement pour la configuration de codage intra (AI). S'appuyant sur les résultats du chapitre précédent, ce chapitre offre un aperçu complet des méthodes actuelles axées sur le partitionnement rapide pour VVC et HEVC sous la configuration AI. Il introduit une approche novatrice utilisant des classificateurs Light Gradient Boosting Machine (LightGBM) pour prédire les décisions de partitionnement. En intégrant cette méthode dans l'encodeur VVenC, des réductions significatives de la complexité computationnelle sont obtenues, comme le démontrent les résultats expérimentaux qui valident l'efficacité de cette approche.

Chapitre 4 étend les efforts d'optimisation à la configuration de codage inter de l'encodeur VVenC. Ce chapitre propose un aperçu complet des méthodes actuelles dans l'état de l'art axées sur le partitionnement rapide pour les normes VVC et HEVC sous la configuration RA. Il introduit un schéma de partitionnement QTMTT à faible complexité qui exploite les caractéristiques de texture et temporelles pour accélérer le processus d'encodage dans la configuration RA. La méthode proposée est évaluée de manière approfondie par rapport aux techniques de l'état de l'art, démontrant des améliorations substantielles en termes de vitesse d'encodage tout en maintenant une efficacité de codage élevée.

En conclusion, le manuscrit récapitule les principales contributions de cette thèse et explore les perspectives de recherche futures, visant à renforcer et à étendre l'impact de notre travail.

Bibliography

[1] Cisco, "Cisco annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper," 2020. Accessed: 2022-07-01. 1, 121

[2] Sandvine, "The Global Internet Phenomena Report January 2023," 2023. Accessed: 2023-01-09. 1, 121

[3] Ericsson, "Ericsson Mobility Report," 2023. Accessed: 2023-01-09. 1, 121

[4] Statista, "Leading video content types worldwide in 2021, by share of internet users who watch them," 2024. 1, 121

[5] Statista, "Live video viewership in the United States from 2018 to 2022," 2024. 1, 121

[6] 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. 1, 12, 13, 28, 32, 122

[7] 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. 1, 2, 12, 13, 15, 17, 18, 19, 22, 23, 32, 64, 90, 97, 99, 110, 122

[8] A. Tissier, A. Mercat, T. Amestoy, W. Hamidouche, J. Vanne, and D. Menard, "Complexity Reduction Opportunities in the Future VVC Intra Encoder," *IEEE 21st International Workshop on Multimedia Signal Processing, MMSP 2019*, 2019. 2, 66, 97, 122

[9] A. Wieckowski, J. Brandenburg, T. Hinz, C. Bartnik, V. George, G. Hege, C. Helmrich, A. Henkel, C. Lehmann, C. Stoffers, I. Zupancic, B. Bross, and D. Marpe, "VVenC: An Open And Optimized VVC Encoder Implementation," in *Proc. IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, 2021. 2, 3, 13, 40, 41, 54, 56, 70, 91, 96, 123, 124

[10] 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," in *Advances in*

Neural Information Processing Systems, vol. 30, Curran Associates, Inc., 2017. 4, 39, 70, 76, 90, 102, 106, 124

[11] J. Urban, "How Chroma Subsampling Works," 2017. Accessed: 2024-03-14. 8, 9

[12] G. J. Sullivan, P. N. Topiwala, and A. Luthra, "The H. 264/AVC advanced video coding standard: Overview and introduction to the fidelity range extensions," *Applications of Digital Image Processing XXVII*, vol. 5558, pp. 454–474, 2004. 12, 13, 32

[13] International Telecommunication Union, "ITU-T Recommendation H.261: Video codec for audiovisual services at px64 kbit/s." https://www.itu.int/rec/T-REC-H.261-199303-I/en, 1993. Accessed: 2024-07-09. 12

[14] ISO/IEC, "International Standard 11172-2: Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s – Part 2: Video," 1993. Standard. 12

[15] B. G. Haskell, A. Puri, and A. N. Netravali, *Digital video: an introduction to MPEG-2*. Springer Science & Business Media, 1996. 13

[16] F. H. Fitzek and M. Reisslein, "MPEG-4 and H. 263 video traces for network performance evaluation," *IEEE network*, vol. 15, no. 6, pp. 40–54, 2001. 13

[17] F. Bossen, X. Li, and K. Suehring, "AHG report: Test model software development (AHG3)," Tech. Rep. JVET-T0003, Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, July 2019. 15th JVET meeting, Gothenburg, Sweden. 13, 92

[18] Joint Video Experts Team (JVET), "VTM Software Repository, VVC Test Model (VTM)." https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM, 2021. Available online, Accessed: 2024-07-12. 13, 41, 94, 96

[19] Fraunhofer Heinrich Hertz Institute, "VVenC Encoder." https://github.com/fraunhoferhhi/vvenc/wiki, 2020. Accessed: 2024-07-12. 13, 41, 42, 43, 90, 105, 114

[20] Google, "VP8: An Open Video Codec for the Web." Developed by Google, 2010. Accessed: 2024-07-09. 13

[21] Google, "VP9: An Open Video Codec for the Web." Developed by Google, 2013. Accessed: 2024-07-09. 13

[22] WebM Project, "WebM: an open web media project," 2024. Accessed: 2024-08-16. 13

[23] Alliance for Open Media, "AV1: The AOMedia Video Codec," 2018. Available at https://aomedia.org/av1/. 13

[24] M. Kerdranvat, "The Video Codec Landscape in 2020," *ITU Journal: ICT Discoveries*, vol. 3, p. 11, 2020. 14

[25] 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. 19, 144

[26] 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. 19, 144

[27] M. Schäfer, B. Stallenberger, J. Pfaff, P. Helle, H. Schwarz, D. Marpe, and T. Wiegand, "Efficient fixed-point implementation of matrix-based intra prediction," in *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 3364–3368, IEEE, 2020. 19, 144

[28] J. Pfaff, A. Filippov, S. Liu, X. Zhao, J. Chen, S. De-Luxán-Hernández, T. Wiegand, V. Rufitskiy, A. K. Ramasubramonian, and G. Van der Auwera, "Intra Prediction and Mode Coding in VVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3834–3847, 2021. 19

[29] B. Bross, J. Chen, S. Liu, and Y.-K. Wang, "Versatile Video Coding Editorial Refinements on Draft 10," Technical Report JVET-T2001-v2, Joint Video Experts Team (JVET), September 2021. 22, 31

[30] 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. 22, 23

[31] T. Lu, F. Pu, P. Yin, S. McCarthy, W. Husak, T. Chen, E. Francois, C. Chevance, F. Hiron, J. Chen, *et al.*, "Luma mapping with chroma scaling in versatile video coding," in *2020 Data Compression Conference (DCC)*, pp. 193–202, IEEE, 2020. 22

[32] A. Norkin, G. Bjontegaard, A. Fuldseth, M. Narroschke, M. Ikeda, K. Andersson, M. Zhou, and G. Van der Auwera, "HEVC deblocking filter," *IEEE Transactions*

on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1746–1754, 2012. 22

[33] C.-M. Fu, E. Alshina, A. Alshin, Y.-W. Huang, C.-Y. Chen, C.-Y. Tsai, C.-W. Hsu, S.-M. Lei, J.-H. Park, and W.-J. Han, "Sample adaptive offset in the HEVC standard," *IEEE Transactions on Circuits and Systems for Video technology*, vol. 22, no. 12, pp. 1755–1764, 2012. 22

[34] L. Li, Y. Song, T. Ikenaga, and S. Goto, "A CABAC encoding core with dynamic pipeline for H. 264/AVC main profile," in *APCCAS 2006-2006 IEEE Asia Pacific Conference on Circuits and Systems*, pp. 760–763, IEEE, 2006. 23

[35] G. Sullivan, "Rate-Distortion Optimization for Video Compression," *IEEE Signal Processing Magazine*, vol. 84, 1998. 23, 34

[36] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952. 24

[37] M. Zhou, X. Wei, S. Wang, S. Kwong, C.-K. Fong, P. H. Wong, W. Y. Yuen, and W. Gao, "SSIM-based global optimization for CTU-level rate control in HEVC," *IEEE Transactions on Multimedia*, vol. 21, no. 8, pp. 1921–1933, 2019. 28

[38] A. Aaron and Z. Li, "Toward A Practical Perceptual Video Quality Metric," *Netflix TechBlog*, June 2016. 29

[39] ITU-T and ISO/IEC JTC 1, "Working practices using objective metrics for evaluation of video coding efficiency experiments," Technical Paper ITU-T HSTP-VID-WPOM and ISO/IEC DTR 23002-8, ITU-T and ISO/IEC JTC 1, 2020. [Online]. Available: https://www.itu.int/fr/publications/ITU-T/Pages/publications.aspx?parent=T-TUT-ASC-2020-HSTP1&media=electronic. 29, 107

[40] F. Bossen, J. Boyce, X. Suehring, X. Li, and V. Seregin, "JVET Common Test Conditions and Software Reference Configurations for SDR Video," tech. rep., Joint Video Experts Team (JVET), 2019. 31, 71, 81, 99

[41] 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, pp. 1–14, 2021. 35

[42] J. Park, B. Kim, J. Lee, and B. Jeon, "Machine learning-based early skip decision for intra subpartition prediction in VVC," *IEEE Access*, vol. 10, pp. 111052–111065, 2022. 35

[43] Y. Wang, X. Fan, C. Jia, D. Zhao, and W. Gao, "Neural network based inter prediction for HEVC," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, IEEE, 2018. 35

[44] W. Hamidouche, P. Philippe, C.-E. Mohamed, A. Kammoun, D. Menard, and O. Déforges, "Hardware-friendly dst-vii/dct-viii approximations for the versatile video coding standard," in *2019 Picture Coding Symposium (PCS)*, pp. 1–5, IEEE, 2019. 36

[45] S. Samir, M. Saumard, T. Damak, M. Jridi, and M. A. B. Ayed, "Random Forest Based Fast MTS Algorithm for VVC Encoder," in *LAIS-ICSOC*, 2024. 36

[46] 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*, pp. 1–1, 2023. 36, 68, 69, 75

[47] S. Zhang, S. Feng, J. Chen, C. Zhou, and F. Yang, "A GCN-based fast CU partition method of intra-mode VVC," *Journal of Visual Communication and Image Representation*, vol. 88, p. 103621, 2022. 36, 68, 69, 88

[48] M. Amna, W. Imen, B. Soulef, and S. Fatma Ezahra, "Machine Learning-Based approaches to reduce HEVC intra coding unit partition decision complexity," *Multimedia Tools and Applications*, vol. 81, no. 2, pp. 2777–2802, 2022. 37, 65

[49] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, "DVC: An End-To-End Deep Video Compression Framework," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10998–11007, 2019. 37

[50] Z. Hu, G. Lu, and D. Xu, "FVC: A New Framework towards Deep Video Compression in Feature Space," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1502–1511, 2021. 37

[51] A. Wieckowski, C. Stoffers, B. Bross, and D. Marpe, "VVenC: an open optimized VVC encoder in versatile application scenarios," in *Applications of Digital Image Processing XLIV* (A. G. Tescher and T. Ebrahimi, eds.), vol. 11842, p. 118420H, International Society for Optics and Photonics, SPIE, 2021. 41

[52] F. H. H. Institute, "VVenC, the Fraunhofer Versatile Video Encoder." `https://github.com/fraunhoferhhi/vvenc`, 2024. Accessed: 2024-07-12. 41, 42, 43, 44, 81, 87, 94

[53] I. Taabane, D. Menard, A. Mansouri, and A. Ahaitouf, "VVenC: Open Source VVC Encoder Documentation." https://vvenc.netlify.app/, 2024. Accessed: 2024-10-03. 44

[54] S. B. D. Mdalsi, I. Taabane, and D. Menard, "VVenC: VVenC-Dashboard." https://vvenc-dash.fly.dev/, 2024. Accessed: 2024-10-03. 61

[55] S. Cho and M. Kim, "Fast CU splitting and pruning for suboptimal CU partitioning in HEVC intra coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 9, pp. 1555–1564, 2013. 64

[56] T. Zhang, M.-T. Sun, D. Zhao, and W. Gao, "Fast intra-mode and CU size decision for HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 8, pp. 1714–1726, 2016. 65

[57] B. Min and R. C. Cheung, "A fast CU size decision algorithm for the HEVC intra encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 5, pp. 892–896, 2014. 65

[58] Y. Huang, L. Song, and E. Izquierdo, "CNN accelerated intra video coding, where is the upper bound?," in *2019 Picture Coding Symposium (PCS)*, pp. 1–5, IEEE, 2019. 65

[59] A. Feng, C. Gao, L. Li, D. Liu, and F. Wu, "CNN-based depth map prediction for fast block partitioning in hevc intra coding," in *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, IEEE, 2021. 65

[60] M. Xu, T. Li, Z. Wang, X. Deng, R. Yang, and Z. Guan, "Reducing complexity of HEVC: A deep learning approach," *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 5044–5059, 2018. 66

[61] T. Fu, H. Zhang, F. Mu, and H. Chen, "Fast CU partitioning algorithm for H.266/VVC intra-frame coding," *Proceedings - IEEE International Conference on Multimedia and Expo*, vol. 2019-July, pp. 55–60, 2019. 67, 69

[62] J. Cui, T. Zhang, C. Gu, X. Zhang, and S. Ma, "Gradient-based early termination of CU partition in VVC intra coding," *Data Compression Conference Proceedings*, vol. 2020-March, no. Dcc, pp. 103–112, 2020. 67, 69

[63] M. Saldanha, G. Sanchez, C. Marcon, and L. Agostini, "Configurable Fast Block Partitioning for VVC Intra Coding Using Light Gradient Boosting Machine," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 6, pp. 3947–3960, 2022. 67, 69, 78, 87, 88, 102

[64] F. Chen, Y. Ren, Z. Peng, G. Jiang, and X. Cui, "A fast CU size decision algorithm for VVC intra prediction based on support vector machine," *Multimedia Tools and Applications*, vol. 79, no. 37-38, pp. 27923–27939, 2020. 67, 69

[65] Q. He, W. Wu, L. Luo, C. Zhu, and H. Guo, "Random Forest Based Fast CU Partition for VVC Intra Coding," *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, BMSB*, vol. 2021-Augus, pp. 31–34, 2021. 67, 69

[66] G. Wu, Y. Huang, C. Zhu, L. Song, and W. Zhang, "SVM based fast CU partitioning algorithm for VVC intra coding," *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 2021-May, no. Mic, 2021. 68, 69, 88

[67] T. Li, M. Xu, R. Tang, Y. Chen, and Q. Xing, "DeepQTMT: A Deep Learning Approach for Fast QTMT-Based CU Partition of Intra-Mode VVC," *IEEE Transactions on Image Processing*, vol. 30, pp. 5377–5390, 2021. 68, 69

[68] Q. Zhang, R. Guo, B. Jiang, and R. Su, "Fast CU Decision-Making Algorithm Based on DenseNet Network for VVC," *IEEE Access*, vol. 9, pp. 119289–119297, 2021. 68, 69

[69] X. Hoangvan, S. Nguyenquang, M. Dinhbao, M. Dongoc, and D. T. Duong, "Fast QTMT for H.266/VVC Intra Prediction using Early-Terminated Hierarchical CNN model," *International Conference on Advanced Technologies for Communications*, vol. 2021-Octob, pp. 195–200, 2021. 68, 69

[70] I. Taabane, D. Menard, A. Mansouri, and A. Ahaitouf, "Machine learning based fast QTMTT partitioning strategy for VVenC encoder in intra coding," *Electronics*, vol. 12, no. 6, p. 1338, 2023. 70, 81, 99, 102, 149

[71] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019. 71, 76

[72] 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 (CVPR) Workshops*, pp. 1122–1131, 2017. 75

[73] JPEG AI, "JPEG AI Dataset," 2022. 75

[74] Google Research, "HDR+ Dataset by Google Research," 2021. 75

[75] Flickr Community, "Flickr2K Dataset," 2020. 75

[76] "Random Forest vs Lightgbm." https://mljar.com/machine-learning/lightgbm-vs-random-forest/. Accessed: 2022-07-30. 76

[77] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011. 76

[78] I. Taabane, D. Menard, A. Mansouri, and A. Ahaitouf, "A Fast CU Partition Algorithm for VVenC Encoder in Intra Configuration," in *2022 IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 1–6, 2022. 78, 81, 82, 102, 149

[79] G. Bjontegaard, "Calculation of Average PSNR Differences between RD curves," Technical Report VCEG-M33, ITU-T SC16/Q6, Austin, Texas, USA, Apr. 2001. 13th VCEG Meeting. Available: https://www.itu.int/wftp3/av-arch/video-site/0104_Aus/VCEG-M33.doc. 81

[80] I. Taabane, D. Menard, A. Mansouri, S. Sahraoui, and A. Ahaitouf, "Low Complexity Learning-Based QTMTT Partitioning Scheme for Inter Coding in VVC Encoder," *IEEE Access*, vol. 12, pp. 141088–141103, 2024. 90, 97, 149

[81] B. Huang, Z. Chen, Q. Cai, M. Zheng, and D. O. Wu, "Rate-distortion-complexity optimized coding mode decision for HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 3, pp. 795–809, 2019. 92

[82] G. Correa, P. A. Assuncao, L. V. Agostini, and L. A. da Silva Cruz, "Fast HEVC encoding decisions using data mining," *IEEE transactions on circuits and systems for video technology*, vol. 25, no. 4, pp. 660–673, 2014. 92

[83] G. Correa, P. Dall'Oglio, D. Palomino, and L. Agostini, "Fast block size decision for HEVC encoders with on-the-fly trained classifiers," in *2020 28th European Signal Processing Conference (EUSIPCO)*, pp. 540–544, IEEE, 2021. 93

[84] M. Grellert, B. Zatt, S. Bampi, and L. A. da Silva Cruz, "Fast coding unit partition decision for HEVC using support vector machines," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 6, pp. 1741–1753, 2018. 93

[85] N. Tang, J. Cao, F. Liang, J. Wang, H. Liu, X. Wang, and X. Du, "Fast CTU partition decision algorithm for VVC intra and inter coding," in *2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 361–364, IEEE, 2019. 94, 97

[86] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, pp. 679–698, 1986. 94

[87] W. Kuang, X. Li, X. Zhao, and S. Liu, "Unified Fast Partitioning Algorithm for Intra and Inter Predictions in Versatile Video Coding," in *2022 Picture Coding Symposium (PCS)*, pp. 271–275, IEEE, 2022. 94, 97

[88] X. Shang, G. Li, X. Zhao, and Y. Zuo, "Low complexity inter coding scheme for Versatile Video Coding (VVC)," *Journal of Visual Communication and Image Representation*, vol. 90, p. 103683, 2023. 94, 97

[89] T. Amestoy, A. Mercat, W. Hamidouche, D. Menard, and C. Bergeron, "Tunable VVC frame partitioning based on lightweight machine learning," *IEEE Transactions on Image Processing*, vol. 29, pp. 1313–1328, 2019. 94, 97, 102, 108

[90] G. Kulupana, S. Blasi, *et al.*, "Fast Versatile Video Coding using Specialised Decision Trees," in *2021 Picture Coding Symposium (PCS)*, pp. 1–5, IEEE, 2021. 95, 97

[91] Y. Li, F. Luo, and Y. Zhu, "Temporal Prediction Model-Based Fast Inter CU Partition for Versatile Video Coding," *Sensors*, vol. 22, no. 20, p. 7741, 2022. 95, 97, 114

[92] J. Li, S. Zhang, and F. Yang, "Random Forest Accelerated CU Partition for Inter Prediction in H. 266/VVC," in *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 01–06, IEEE, 2022. 95, 96, 97, 114, 115

[93] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001. 95

[94] K. Xie, J. Zhou, S. Zhang, and F. Yang, "Fast Inter Prediction Mode Decision Method Based On Random Forest For H. 266/VVC," in *2022 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, pp. 1–5, IEEE, 2022. 95, 96, 97, 115

[95] W.-H. Yeo, B.-G. Kim, *et al.*, "CNN-based fast split mode decision algorithm for versatile video coding (VVC) inter prediction," *Journal of Multimedia Information System*, vol. 8, no. 3, pp. 147–158, 2021. 95, 97

[96] Y. Liu, M. Abdoli, T. Guionnet, C. Guillemot, and A. Roumy, "Light-weight CNN-based VVC inter partitioning acceleration," in *2022 IEEE 14th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*, pp. 1–5, IEEE, 2022. 95, 97

[97] Y. Liu, M. Riviere, T. Guionnet, A. Roumy, and C. Guillemot, "CNN-based Prediction of Partition Path for VVC Fast Inter Partitioning Using Motion Fields," *arXiv preprint arXiv:2310.13838*, 2023. 96, 97

[98] A. Tissier, W. Hamidouche, J. Vanne, and D. Menard, "Machine learning based efficient Qt-Mtt partitioning for VVC inter coding," in *2022 IEEE International Conference on Image Processing (ICIP)*, pp. 1401–1405, IEEE, 2022. 96, 97, 114

[99] Z. Pan, P. Zhang, B. Peng, N. Ling, and J. Lei, "A CNN-based fast inter coding method for VVC," *IEEE Signal Processing Letters*, vol. 28, pp. 1260–1264, 2021. 96, 97

[100] THEOplayer, "The Importance of Low Latency in Video Streaming." https://www.theoplayer.com/blog/the-importance-of-low-latency-in-video-streaming, 2024. Accessed: 2024-08-25. 97

Appendix A

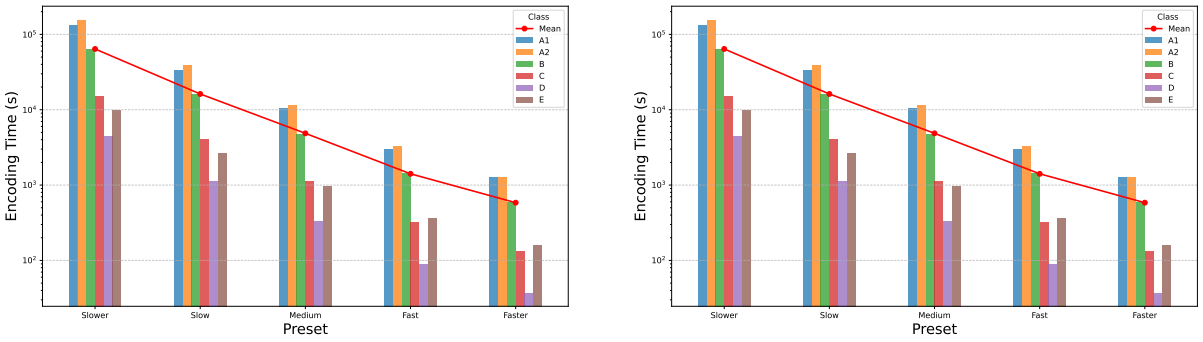
Annex: VVenC Performance Analysis Across Different Resolutions

This study presents experiments conducted across all resolution classes from the 6 Common Test Conditions (CTC) sets (A1, A2, B, C, D, and E). The experiments were performed using four Quantization Parameter (QP) values across the five Versatile Video Encoder (VVenC) presets, under both Random Access (RA) and All Intra (AI) configurations.

A.1 Evaluation of Encoding Time

Figure A.1 illustrates the distribution of encoding time for each VVenC preset across different resolution classes. The slower preset, which emphasizes compression efficiency, takes significantly more time to encode higher resolutions such as Ultra High Definition (UHD) (Classes A1, A2), followed by Full-HD (Class B), 480p (Class C), 720p (Class E), and finally 240p (Class D). The increase in encoding time for higher resolutions is expected, as larger frames require more complex processing. This pattern holds across all presets and for both RA and AI configurations. Higher resolutions involve a larger number of pixels and more intricate block partitioning, especially in RA mode where motion estimation demands significant computational resources.

As the encoding moves from the slower to the faster preset, the time required for encoding drops noticeably across all classes. This reduction results from optimizations that prioritize speed by simplifying internal encoder processes. A major factor in this speed-up is the adjustment in Multi-Type Tree (MTT) partitioning. In the slower preset, the encoder allows deeper block partitioning using Quadtree (QT), Binary-Tree (BT), and Ternary-Tree (TT), enabling highly flexible block sizes that adapt to local content com-



(a) *VVenC RA encoding time* **(b)** *VVenC AI encoding time*

Figure A.1 *VVenC original presets encoding time under RA and AI configurations using CTC classes*

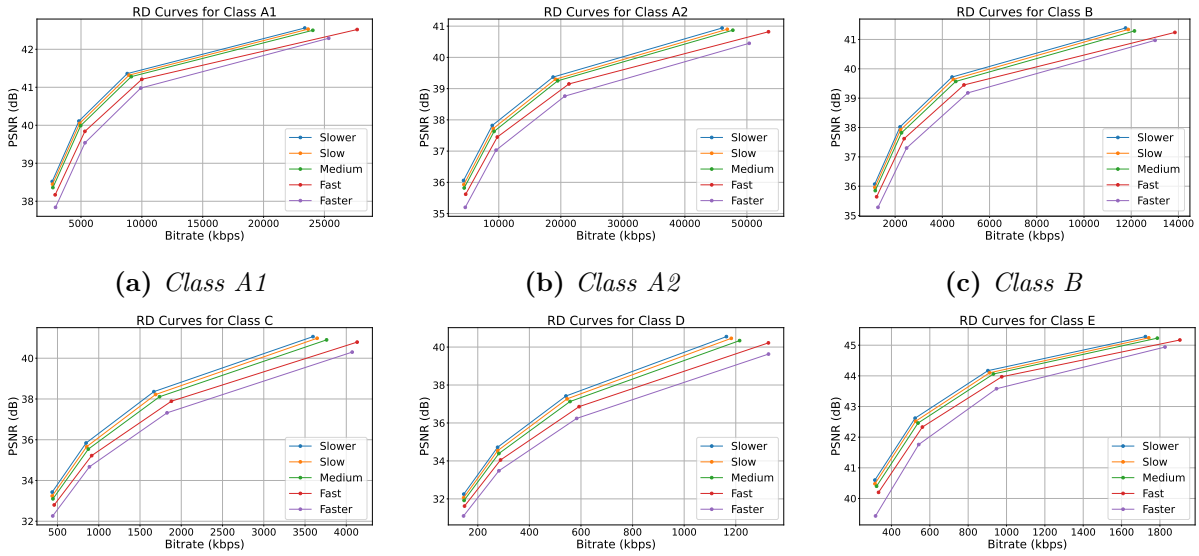
plexity. This flexibility improves compression but increases computational load, leading to longer encoding times.

By contrast, the faster preset reduces the maximum partitioning depth and restricts block partitioning options, often allowing only QT partitions, similar to the High Efficiency Video Coding (HEVC) standard. This simpler partitioning strategy decreases the number of potential block configurations the encoder must evaluate, significantly reducing computational complexity and, consequently, encoding time. These optimizations primarily target time-saving in higher resolutions, where the encoding workload is greatest. Overall, the shift from slower to faster presets demonstrates a deliberate trade-off between encoding time and compression efficiency. Limiting the partitioning flexibility and streamlining decision processes in the faster presets reduces encoding time substantially, particularly for high-resolution content, while slightly sacrificing compression performance.

A.2 Rate-Distortion Analysis

The Rate-Distortion (RD) curves illustrated in Figure A.2 provide a clear depiction of the compression efficiency of the five VVenC presets across the six CTC classes (A1, A2, B, C, D, and E) under the RA configuration. As expected, the slower preset consistently achieves the best compression efficiency, reflected in the steepest RD curves, which offer the highest Peak Signal-to-Noise Ratio (PSNR) for a given bitrate (see section 1.6.1, page 28). This superior performance is due to the exhaustive search and deeper block partitioning allowed by the encoder, resulting in better adaptation to content complexity

and finer optimization of bitrate.



(a) *Class A1* **(b)** *Class A2* **(c)** *Class B*

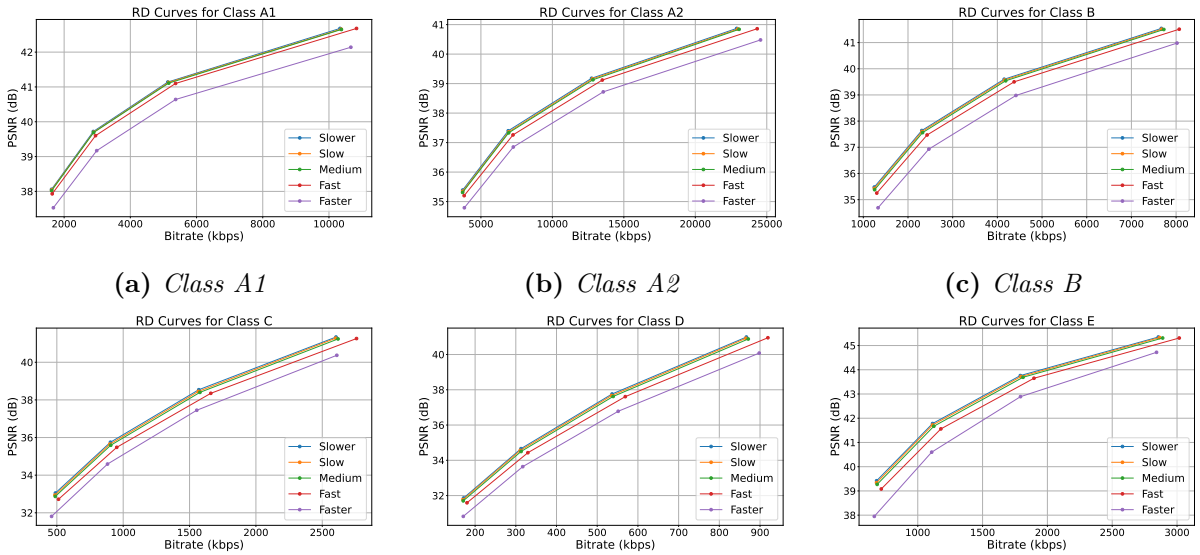
(d) *Class C* **(e)** *Class D* **(f)** *Class E*

Figure A.2 *VVenC original presets Rate-Distortion (RD) curves under RA configuration*

Moving toward the faster presets, a noticeable shift occurs in the RD curves, with a reduction in PSNR at equivalent bitrates. The trade-off between encoding time and compression efficiency becomes evident, as the faster preset employs more constrained partitioning and less complex encoding techniques, leading to a moderate decrease in coding efficiency. The curves for the medium preset lie between the slower and faster extremes, balancing time savings with reasonable compression performance. Across all classes, this trend holds consistent, but the impact varies based on resolution. Higher resolution classes, such as A1 (UHD) and B (Full-HD), exhibit a more pronounced decline in compression efficiency with faster presets, as these resolutions require more advanced block partitioning and motion estimation for optimal coding efficiency. Lower resolution classes like D (240p) show smaller variations in RD curves, as the reduced complexity of the content diminishes the need for the deep partitioning employed by the slower presets.

In the AI mode, where all frames are encoded independently without temporal prediction, the slower preset once again provides the best compression efficiency, yielding the highest PSNR for a given bitrate as depicted in Figure A.3. This is due to the more complex intra-prediction modes and deeper block partitioning allowed by the preset, enabling finer adaptation to the spatial details of each frame.

As the presets transition from slower to faster, the RD curves show a gradual reduction



(a) *Class A1* **(b)** *Class A2* **(c)** *Class B*

(d) *Class C* **(e)** *Class D* **(f)** *Class E*

Figure A.3 *VVenC original presets Rate-Distortion (RD) curves under AI configuration*

in compression efficiency, with the faster preset producing lower PSNR at similar bitrates. This efficiency loss is attributed to the simplified partitioning strategy and fewer intra-prediction options in the faster presets, leading to less optimal block representation. In AI mode, where the focus is on spatial prediction within each frame, these restrictions particularly impact high-resolution classes like A1 (UHD) and B (Full-HD), where the need for detailed intra-prediction is greatest. For lower resolution classes, such as D (240p), the differences in RD curves are less pronounced, as the reduced spatial complexity requires less sophisticated intra-coding techniques.

In summary, the slower and slow presets prioritize compression efficiency, with the slow preset maintaining many of the same internal parameters as the slower one, particularly in AI mode where the MTT depth remains unchanged. In contrast, the fast and faster presets make drastic reductions in block partitioning flexibility, Coding Tree Unit (CTU) size, and other encoding parameters to speed up the encoding process. However, this results in a noticeable decrease in compression efficiency, especially for high-resolution video content. The medium preset sits between these extremes, offering a compromise between speed and efficiency.

Appendix B

Annex: Assessing Intra Prediction Tools in VVenC Encoder Presets

B.1 Introduction

Versatile Video Coding (VVC) introduces several intra-prediction tools that play a key role in efficient video compression, particularly when handling various types of content, such as natural scenes, screen content, and synthetic images. The most notable tools in VVC include Intra Sub-Partitions (ISP), Matrix-Based Intra Prediction (MIP), and Multiple Reference Line (MRL). The Versatile Video Encoder (VVenC) encoder incorporates these tools from the VVC Test Model (VTM) encoder, offering flexibility through different presets like slower, slow, and medium. This section explores how these tools impact compression efficiency and encoding time across these presets. Each preset configuration file specifies whether a tool is enabled, disabled, or optimized for speed. Table B.1 shows the parameter settings for these tools in each preset. The parameter values show a gradual reduction in the use of intra prediction tools as presets progress from slower to faster. Slower presets employ more complex tools (like ISP, MIP, and MRL) to achieve precise intra-frame predictions. Conversely, faster presets disable most prediction tools to accelerate processing, suitable for scenarios where encoding speed is critical.

Table B.1 *Intra prediction tools values in each preset configuration file*

Parameter	Slower	Slow	Medium	Fast	Faster
ISP	1	3	3	0	0
MIP	1	1	1	0	0
MRL	1	1	1	0	0

B.1.1 Overview of Intra Prediction Tools in VVC

Intra Sub-Partitioning (ISP) allows the encoder to split a block into smaller sub-blocks, either vertically or horizontally, to improve the prediction of regions with strong directional structures [25]. Figure B.1 illustrates these partitions in both vertical and horizontal orientations. ISP is particularly effective for encoding areas with sharp edges, lines, or text, where traditional intra-prediction methods may struggle to capture finer details. By dividing the block into sub-partitions, the encoder can make more accurate predictions, resulting in enhanced coding efficiency.

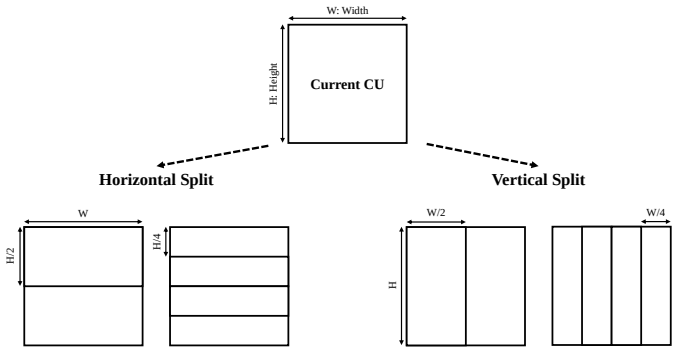


Figure B.1 *Intra Sub-Partitions (ISP)*

Matrix-based Intra Prediction (MIP) is a novel tool that uses matrix-based operations to predict pixel values within a block [27]. It is particularly useful for screen content and other synthetic imagery, where pixel patterns are highly regular and directional prediction modes may be insufficient. Instead of relying on simple directional relationships, MIP applies a matrix transformation to capture more complex correlations between neighboring pixels.

Multiple Reference Lines (MRL) enhances intra-prediction by allowing the encoder to choose from several reference lines, as shown in Figure B.2. This added flexibility improves prediction accuracy, especially in complex or noisy regions where a single adjacent reference line may not provide enough information [26]. MRL is particularly useful in natural scenes with varying textures or non-uniform structures, enabling more precise predictions and better overall compression efficiency in challenging areas.

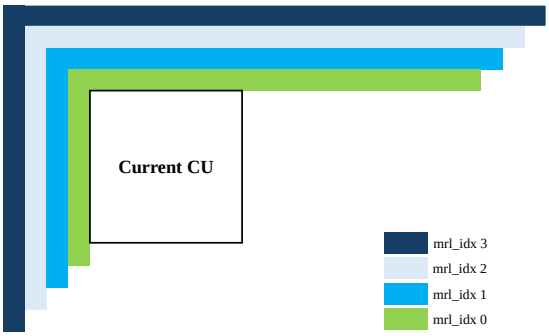


Figure B.2 *Multiple Reference Line (MRL)*

B.2 Impact of ISP, MIP, and MRL Configurations

The impact of disabling each tool was evaluated using two key metrics: Bjøntegaard Delta Bit Rate (BDBR) and Encoding Time Reduction (ETR) (see section 1.6, page 27). BDBR quantifies changes in bitrate efficiency, where a higher BDBR indicates that more bits are required to maintain the same quality, signaling a reduction in compression efficiency. ETR measures the percentage of time saved when a tool is disabled, with higher ETR values reflecting greater reductions in encoding time. These metrics were analyzed across three encoding presets: slower, optimized for maximum compression efficiency; slow, which balances encoding speed and efficiency; and medium, which prioritizes faster encoding but sacrifices some compression efficiency. Table B.2 summarizes the results for all presets when intra-prediction tools are individually and jointly disabled. In the table, "E" indicates the tool is enabled, and "D" indicates it is disabled.

Table B.2 *Impact of intra prediction tools on BDBR and ETR percentages across VVenC presets*

Configuration	ISP	MIP	MRL	Slower		Slow		Medium	
				BDBR(%)	ETR(%)	BDBR(%)	ETR(%)	BDBR(%)	ETR(%)
C1	D	E	E	0.55	20.58	0.55	25.27	0	0.23
C2	E	D	E	0.44	14.86	0.53	10.76	0.37	5.41
C3	E	E	D	0.28	1.12	0.29	0.84	0.36	2.41
C4	D	D	E	1.06	35.71	1.17	36.93	1	24.75
C5	D	E	D	0.86	21.72	0.88	26.74	1.02	21.53
C6	E	D	D	0.71	15.84	0.83	11.69	0.76	7.72
C7	D	D	D	1.37	36.38	1.54	38.79	1.46	23.22

In configuration *C1*, disabling ISP has the most pronounced impact on both compression efficiency and encoding time. In the slower preset, removing ISP raises BDBR by

0.55%, reflecting a clear degradation in compression efficiency, while ETR reaches 20.58%, showing that ISP is highly computationally intensive. The slow preset exhibits a similar trend, where disabling ISP results in a 0.55% increase in BDBR and a notable ETR of 25.27%, indicating that the computational cost of ISP becomes more apparent as speed gains are prioritized over compression. In the medium preset, however, the absence of ISP has negligible effects on BDBR, and the ETR impact is minimal at only 0.23%. This demonstrates that the role of ISP diminishes as encoding speed becomes the main priority, with fewer resources dedicated to intricate spatial prediction.

In configuration *C2*, MIP serves as a versatile tool that enhances prediction accuracy, particularly for structured or synthetic content, though it does not exhibit as large an impact as ISP. When MIP is disabled in the slower preset, there is a 0.44% increase in BDBR and a 14.86% reduction in encoding time, indicating that while MIP contributes to compression, it imposes a lower computational load compared to ISP. In the slow preset, the effects of disabling MIP become slightly more pronounced, with BDBR increasing by 0.53% and ETR by 10.76%. This shows that as speed becomes a priority, MIP retains a moderate balance between efficiency and complexity. In the medium preset, MIP has a smaller effect, with a BDBR increase of 0.37% and an ETR of 5.41%. This highlights that while MIP provides some compression benefits even in faster presets, its role is more prominent when handling structured content.

On the other hand, configuration *C3* shows that MRL has a comparatively minor influence on both bitrate efficiency and encoding complexity. In the slower preset, disabling MRL results in only a 0.28% increase in BDBR and a minimal ETR of 1.12%, indicating that MRL has limited computational demand. This trend persists in the slow preset, with a similar 0.29% increase in BDBR and an ETR of 0.84%, underscoring that MRL adds modest improvements in prediction accuracy at a low computational cost. In the medium preset, MRL yields a 0.36% increase in BDBR and an ETR of 2.41%, reinforcing its low-impact role in enhancing compression with minimal time savings.

B.3 Impact of Disabling Multiple Intra Tools Configurations

Table B.2 shows also the results of disabling multiple tools and illustrates how these tools complement each other: Disabling ISP and MIP in configuration *C4*, results in the largest BDBR increases and ETR values across all presets, particularly in the slower

preset, where compression efficiency is critical. For instance, disabling both tools leads to a 1.06% increase in BDBR and a 35.71% ETR.

Configuration *C5*, where ISP and MRL are disabled, shows a similar trend but with slightly lower BDBR and ETR values than *C4*. In the slower preset, disabling these tools increases BDBR by 0.86% and reduces ETR by 21.72%, suggesting that MRL has less impact on compression efficiency than MIP when combined with ISP. In the medium preset, BDBR rises by 1.02%, with a corresponding ETR of 21.53%, reinforcing that ISP remains a primary contributor to compression efficiency, even when used without MRL.

Configuration *C6*, which disables MIP and MRL, results in a balanced trade-off between BDBR and ETR across presets. In the slower preset, BDBR increases by 0.71%, while ETR is reduced by 15.84%, indicating that ISP alone can still maintain a reasonable level of compression efficiency with moderate computational demand. In the medium preset, BDBR increases by 0.76%, and ETR is reduced by 7.72%, reflecting the relatively minor role of MIP and MRL in speed-focused scenarios.

In configuration *C7*, disabling all three tools (ISP, MIP, and MRL) leads to the highest overall bitrate penalties (BDBR up to 1.54% in the slow preset) and the largest encoding time reductions (ETR of 38.79% in slow). This demonstrates that while these tools collectively contribute to high compression efficiency, they also add considerable computational complexity.

This section has examined the roles and impacts of three key intra-prediction tools—ISP, MIP, and MRL—in the VVC standard, particularly within the VVenC. Among these tools, ISP is the most critical for achieving high compression efficiency, but it also introduces the greatest computational overhead, particularly in the slower and slow presets. MIP plays a balanced role, offering moderate improvements in compression while keeping computational costs lower than ISP. MRL, on the other hand, is a lightweight tool that provides improvements in prediction accuracy with minimal impact on encoding speed. Each tool's importance varies depending on the encoding preset. In slower presets, where compression efficiency is crucial, these tools are indispensable for maintaining low bitrates. In faster presets (e.g., medium), their importance decreases, as encoding speed becomes the primary concern. This trade-off between computational complexity and compression efficiency is a special feature of the VVenC encoder, allowing for flexibility across a wide range of video applications.

B.4 Conclusion

Intra-prediction tools like ISP, MIP, and MRL are essential to VVC, affecting compression efficiency and complexity in VVenC. Each tool serves a unique purpose across encoding presets: ISP boosts compression but increases computational demand, MIP provides a balanced approach for structured content, and MRL enhances prediction accuracy with low time costs. As speed becomes a priority in faster presets, the roles of these intra-prediction tools decrease, demonstrating the adaptability of VVenC to various application needs. This study highlights how the presets within VVenC enable efficient video compression.

Appendix C

Personal Publications

C.1 Scientific Journals

— **[J1]** [70] **Taabane, I.**, Menard, D., Mansouri, A., and Ahaitouf, A. (2023). "Machine learning based fast QTMTT partitioning strategy for VVenC encoder in intra coding," *Electronics*, 12(6), 1338.

— **[J2]** [80] **Taabane, I.**, Menard, D., Mansouri, A., and Ahaitouf, A. (2024). "Low Complexity Learning-Based QTMTT Partitioning Scheme for Inter Coding in VVC Encoder," *IEEE Access*.

C.2 International Conferences

Conference Papers:

— **[C1]** [78] **Taabane, I.**, Menard, D., Mansouri, A., and Ahaitouf, A. (2022, November). "A Fast CU Partition Algorithm for VVenC Encoder in Intra Configuration." In *2022 IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 1-6. IEEE.

Oral Presentations:

— **[C2]** **Taabane, I.**, Menard, D., Mansouri, A., and Ahaitouf, A. (2022, October). "Machine Learning Based Fast CU Partitioning Algorithm For VVC Intra Configuration." In Euro-Mediterranean Conference on Materials, Components, and Systems (EMCM-DS).

— **[C3]** **Taabane, I.**, Menard, D., Mansouri, A., and Ahaitouf, A. (2023, December). "Fast CU Partitioning Approach for VVC Inter Coding." In the International Workshop of Multimodal Artificial Intelligence, Machine Learning and Applications.

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

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Titre : Optimisation de la Complexité Basée sur l'Apprentissage Automatique Léger pour l'Encodage VVC en Temps Réel

Mot clés : Compression Vidéo, Optimisation de la Complexité, Partitionnement (QTMTT), Apprentissage Automatique (ML), Versatile Video Coding (VVC), Versatile Video Encoder (VVenC)

Résumé : La croissance rapide des services de vidéo à la demande, des plateformes de streaming en direct et des réseaux sociaux a fait du contenu vidéo la forme dominante du trafic internet mondial, représentant plus de 82% des données. Cette explosion de la consommation de vidéos exige le développement de techniques de compression vidéo plus avancées. La norme VVC, introduite en juillet 2020, permet une réduction de 50% du débit binaire par rapport à son prédécesseur HEVC, mais augmente considérablement la complexité de l'encodage, jusqu'à 25,8 fois. Cette thèse se concentre sur la réduction de la complexité de l'encodage VVC tout en préservant la qualité vidéo. Elle présente une analyse statistique de l'encodeur VVenC pour identifier des opportunités d'optimisation, suivie du développement d'une technique basée sur l'apprentissage automatique utilisant des classificateurs LightGBM pour accélérer le partitionnement QTMTT dans le codage intra. La recherche étend également ces optimisations aux configurations de codage inter. Ces méthodes proposées réduisent significativement le temps d'encodage avec une augmentation minimale du débit binaire, améliorant ainsi la faisabilité de l'encodage VVC en temps réel pour des vidéos haute résolution.

Title: Lightweight Machine Learning-Based Complexity Optimization for Real-Time VVC Encoding

Keywords: Video Compression, Complexity Optimization, Quadtree with Multi-Type Tree (QTMTT) Partitioning, Machine Learning (ML), Versatile Video Coding (VVC), Versatile Video Encoder (VVenC)

Abstract: The rapid growth of video-on-demand services, live-streaming platforms, and social media has led to video content dominating global internet traffic, accounting for over 82% of data. This surge in video consumption necessitates the development of more advanced video compression techniques. The VVC standard, introduced in July 2020, achieves a 50% reduction in bitrate compared to its predecessor High Efficiency Video Coding (HEVC) but significantly increases encoding complexity, up to 25.8 times. This thesis focuses on reducing VVC encoding complexity while preserving video quality. It presents a statistical analysis of the VVenC to identify areas for optimization, followed by the development of a machine learning-based technique utilizing Light Gradient Boosting Machine (LightGBM) classifiers to accelerate QTMTT partitioning in intra-coding. The research also extends these optimizations to inter-coding configurations. These proposed methods significantly reduce encoding time with minimal increases in bitrate, enhancing the feasibility of real-time VVC encoding for high-resolution video content.