# Optimization and Acceleration of HEVC and VVC Video Compression Standards Using Artificial Intelligence on Parallel Processing Platforms.

Soulef Bouaafia

**RÉPUBLIQUE TUNISIENNE**

**MINISTÈRE DE L'ENSEIGNEMENT SUPÉERIEUR**

**ET DE LA RECHERCHE SCIENTIFIQUE**

**UNIVERSITÉ DE MONASTIR**

**FACULTÉ DES SCIENCES DE MONASTIR**

**Ecole Doctorale : Matériaux, Dispositifs et Microsystèmes (EDMD)**

# THÈSE

Présentée pour l'obtention du diplôme de

**DOCTORAT DE L'UNIVERSITÉ DE MONASTIR**

Faculté des Sciences de Monastir

Spécialité : ÉLECTRONIQUE ET MICROÉLECTRONIQUE

Présentée par:

## Soulef BOUAAFIA

**Sujet:**

---

**Optimisation et Accélération des Normes de Compression Vidéo HEVC et VVC utilisant l'Intelligence Artificielle sur des Plateformes de Traitement Parallèle.**

**Optimization and Acceleration of HEVC and VVC Video Compression Standards Using Artificial Intelligence on Parallel Processing Platforms.**

---

Directeur de thèse : **Fatma Ezahra SAYADI**

Soutenue le 05/11/2021 devant la Commission d'Examen composée de :

| | | |
|---|---|---|
| M. Nejib HASSEN | Professeur, ISIMM | Président |
| M. Ali DOUIK | Professeur, ENISO | Rapporteur |
| M. Khaled BEN KHALIFA | Maître de conférences, ISSATSO | Rapporteur |
| M. Chokri SOUANI | Professeur, ISSATSO | Examinateur |
| Mme. Fatma Ezahra SAYADI | Maître de conférences, ENISO | Directeur de Thèse |

**Laboratoire d'Électronique et Microélectronique (Code: LR99ES30)**

# Acknowledgments

***First, I would like to witness my gratitude to Almighty God, who gave me the courage and strength to pursue this thesis and opened the gates of knowledge for me.***

*I have started my final internship project for my master's degree in Electronics and Microelectronics Lab Research E$\mu$E. It was March 2017. I never thought this training would be the reason for choosing me to pursue a long and fruitful Ph.D. journey full of fluctuations (The 1$^{st}$ registration was on January 23, 2019). What's for sure is that I will never forget this opportunity which was wonderful, overwhelming, and full of lessons and responsibilities. Thankfully, my Ph.D. was completed successfully in which we delivered many journals and international conferences. This wouldn't have happened without the help and support of countless people over the past three years.*

*First of all, I am deeply grateful to my supervisors* ***Mrs. Fatma Ezahra SAYADI*** *and* ***Mrs. Randa KHEMIRI*** *who gave me this opportunity and believed in me from the very beginning. Knowing that this Ph.D. experience wasn't always easy, it was always a pleasure, with a lot of excitement, to be up to the challenge and to beat paper deadlines. Working with you improved me a lot as a student, as a researcher, and as a person. It wasn't straightforward for me to understand how to think like a researcher. You have taught me, both consciously and unconsciously, how good work is done. I appreciate all your contributions of time and ideas to make my Ph.D. experience productive. You have always listened to my ideas and discussions with you frequently led to progress. Your ability to approach research problems and your high scientific standards set an example. I admire your ability to balance research interests and personal pursuits. I am thankful for the excellent example you have provided me as a successful and ambitious researcher.*

*I would first of all like to thank the members of the jury for their presence, for their careful reading of my thesis as well as for the remarks they will address to me during*

*this defense in order to improve my work. I thank Professors* ***Ali DOUIK*** *and* ***Khaled BEN KHALIFA*** *for finding the time to read my thesis and for their valuable feedback. I would like to thank Professor* ***Chokri SOUANI*** *for being examinator of this jury. And finally, I would like to thank Professor* ***Nejib HASSEN*** *for the honor he gave me when he agreed to chair this jury.*

*I would like to thank all EμE laboratory members, especially the Lab head* ***Pr. Mohsen MACHHOUT*** *and* ***Pr. Mohamed ATRI*** *for their advice and their human qualities of listening and understanding.*

*This work would never be completed without the support of my family. Words cannot describe my gratitude for my parents, my brothers, and my sisters. Without them, I could never have reached this current level of success. To all of you, thank you for your continuous encouragements and devotion.*

*I would like to give special thanks to my friend* ***Dr. Seifeddine MESSAOUD*** *who have always encouraged me.*

**Soulef BOUAAFIA**

“ “ “ “ “ “ ‘

# Abstract

Video contents visualization has been revolutionized over the last decade with the advent of video-on-demand services, web-TV, video-sharing sites, live streaming service for individuals, and broadcast platforms offered by social networks. This led to an explosion of internet traffic. According to a recent Cisco study, video-driven internet traffic will quadruple between 2017 and 2022 and will represent 82% of overall internet traffic. The appearance of new video content, such as 360° video, Virtual Reality (VR), High Frame Rate (HFR) and the advent of very high spatial resolution 8K or even 16K leads to a significant increase in the amount of data to be transmitted. Consequently, efficient compression is essential to store or transmit this huge amount of data. Despite the considerable performance achieved by the video coding standards, the existing compression techniques showed their limitations and it is becoming increasingly difficult to meet the growing demands of data. Therefore, the adoption of new approaches such as machine learning based methods has great potential to address this challenge and can provide very promising results. The objective of this thesis is to introduce advanced techniques to significantly reduce the complexity of the High Efficiency Video Coding (HEVC) and the Versatile Video Coding (VVC) standards, while preserving the bitrate gain and ensuring a better video quality for users. These techniques, based on machine learning provide better performance in classification, in prediction and in efficient compression vs classical algorithms.

# Résumé

La visualisation de contenus vidéo a été révolutionnée au cours de la dernière décennie avec l'apparition des services de vidéo à la demande, de web-TV, de sites de partage de vidéos, de service de diffusion en direct pour les particuliers, et des plateformes de diffusion offertes par les reseaux sociaux. Ceci a conduit à une explosion du trafic internet. Selon une étude récente de Cisco, le trafic internet lié à la vidéo quadruplera entre 2017 et 2022 et représentera 82% du trafic internet global. L'apparition de nouveaux contenus vidéo, tels que la vidéo 360°, la Réalité Virtuelle (VR), le High Frame Rate (HFR) et l'avènement de très grandes résolutions spatiale 8K voire 16K conduit à une augmentation significative de la quantité de données à transmettre. Par conséquent, une compression efficace est essentielle pour stocker ou transmettre cette énorme quantité de données. Malgré les performances considérables obtenues par les normes de codage vidéo, les techniques de compression existantes ont montré leurs limites et il devient de plus en plus difficile de répondre aux demandes croissantes de données. Par conséquent, l'adoption de nouvelles approches telles que les méthodes d'apprentissage automatique représente un grand potentiel pour relever ce défi et peut fournir des résultats très prometteurs. L'objectif de cette thèse est d'introduire des techniques avancées pour réduire significativement la complexité des normes de codage vidéo High Efficiency Video Coding (HEVC) et Versatile Video Coding (VVC) tout en préservant le gain en débit et assurant une meilleure qualité de vidéo aux utilisateurs. Ces techniques basées sur l'apprentissage automatique offrent de meilleures performances en classification, en prédiction et en efficacité de compression par rapport aux algorithmes classiques.

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

**5G** : Fifth Generation 53

**AI** : Artificial Intelligence 16, 50, 74

**ALF** : Adaptive Loop Filter 14, 53

**AR** : Augmented Reality xviii

**ASIC** : Application Specific Integrated Circuit 74

**AVC** : Advenced Video Coding xix

**AXI** : Advanced eXtensible Interface 75

**BD-BR** : Bjontegaard Delta bitrate 41, 42, 63, 66

**BD-PSNR** : Bjontegaard Delta Peak Signal-to Noise Ratio 41

**CABAC** : Context-Adaptive Binary Arithmetic Coding 9

**CLBs** : Configurable Logic Blocks 77

**CNN** : Convolutional Neural Network xx, 19, 32, 53, 95

**CPU** : Central Processing Unit 80

**CTC** : Common Test Conditions 63

**CTU** : Coding Tree Unit 6, 56, 82

**CU** : Coding Unit xx, 6, 57, 95

**DBF** : De-Blocking Filter 9, 14, 53

**DCT** : Discrete Cosine Transform 9

**DL** : Deep Learning 18, 32

**DMA** : Direct Memory Access 80, 87

**DNNs** : Deep Neural Networks 19

**DST** : Discrete Sine Transform 9

**FFs** : Flip-Flops 77

**FPGAs** : Field Programmable Gate Arrays 73, 74

**GoP** : Group of Picture 31

**GPU** : Graphics Processing Unit 36

**HD** : High Definition xviii, 50

**HDR** : High Dynamic Range 2, 95

**HEVC** : High Efficiency Video Coding v, vi, xix, 2, 95

**HFR** : High Frame Rate v, vi, xviii, 2

**HLS** : High Level Synthesis 74, 96

**IOBs** : Input/Output Blocks 77

**IoT** : Internet of Things 50, 51

**IP** : Intellectual Property xxi, 74, 76

**ISO/IEC** : International Organization for Standardization and the International Electrotechnical Commission 3

**ITUT** : International Telecommunication Union, Telecommunication 3

**JCT-VC** : Joint Collaboration Team on Video Coding 3, 4, 34

**JVET** : Joint Video Exploration Team xix, 10

**JVT** : Joint Video Team 3

**LDP** : Low Delay P 41, 42, 46

**LSTM** : Long-and Short-Term Memory xx, 95

**LUTs** : Look-Up-Tables 77

**M-IoT** : Multimedia-Internet of Things xx, 50, 51, 96

**MB** : Macro-Block 6

**ML** : Machine Learning 16

**MOS** : Mean Opinion Score 52

**MPEG** : Moving Picture Experts Group 3

**MSE** : Mean Square Error 63

**PL** : Programmable Logic 74, 87, 96

**PS** : Processing System 74, 87, 96

**PSNR** : Peak Signal-to-Noise Ratio 63

**PU** : Prediction Units 6

**QoE** : Quality of Experience xx, 50, 52, 66, 95

**QoS** : Quality of Service 52

**QP** : Quantization Parameter 13, 41, 57, 61

**QTMT** : QuadTree plus Multi-type Tree 10, 97

**RA** : Random Access xiv, 63

**RDO** : Rate-Distortion Optimization 27, 46, 55, 82, 95

**ReLU** : Rectified Linear Unit 22, 32

**RTL** : Register-Transfer-Level 80

**SAO** : Sample Adaptive Offset 9, 14, 53

**SGD** : Stochastic Gradient Descenth 35, 39

**SoCs** : System-on-Chips 74

**SVM** : Support Vector Machines xx, 95

**TU** : Transform Units 6

**UHD** : Ultra-High Definition xviii, 10, 50, 95

**VCEG** : Visual Coding Experts Group 3

**VR** : Virtual Reality v, vi, xviii

**VVC** : Versatile Video Coding v, vi, xix, 2, 49, 55, 95

**WSE-DCNN** : Wide-activated Squeeze-and-Excitation Deep Convolutional Neural Network xx, 51, 55, 96

# General Introduction

With the development of multimedia computing, communication and display technologies, many video applications have emerged, such as TV broadcasting, video-on-demand, video conference, mobile video, video surveillance, 3D videos and Augmented Reality (AR), which can provide immersive telepresence and realistic visual perception experience. These video applications have been widely employed for multiple roles in human daily life, such as manufacturing, communication, national security, military, education, medicine, and entertainment. Nowadays, video data has been the majority data traffic over the internet and its volume grows explosively each year. The latest Cisco Visual Networking Index reports that Internet Protocol video traffic accounted for 75% of all Internet traffic in 2017, and they expect it to rise up to 82% by the year 2022 [CCH18]. On that occasion, million minutes of video contents will be delivered through the network in every second. To further enhance the immersive and realistic visual experiences, more high-end video applications emerge, such as High and Ultra-High Definition content (HD, UHD), Virtual Reality (VR), High Frame Rate (HFR) and 360° video and the advent of very high spatial resolution 8K or even 16K, which require larger data volume to represent higher fidelity and more details. Meanwhile, the number of video clients and cameras in use grows rapidly as the video demands keep boost in recent years, as HDTV, surveillance cameras, laptop and smart phones. The total amount of global video data doubles every two years, which is the bottleneck for data processing, storage and transmission.

Video coding is one of the basic technologies in video applications that allows video data to be structured and compressed more efficiently for computation, transmission and storage. It has been developed over three decades with four generations and the coding efficiency doubles every ten years. But there is a big gap as compared with the rapid growth of global video data doubling every two years. Achieving much higher compression efficiency and narrowing the gap in an effective way become urgent missions for video coding. Machine learning is a field of study that can learn from data, discover hidden patterns and make data-driven decisions. Due to its superior performance in learning from data, many emerging works have applied machine learning algorithms to video coding to further promote the coding performances, which becomes one of the most promising directions in both academic and industrial communities.

In this context, the advent of the video coding standard, High Efficiency Video Coding (HEVC), standardized in January 2013 [SOHW12] has made it possible to broadcast the UHD content on the communication network. The HEVC provides nearly 50% bitrate gain in comparison to the H.264/AVC standard for the same quality. However, HEVC is still not efficient enough to endure the burden of video transmission and storage for various large popular applications based on 8K and 360° videos. For this reason, Versatile Video Coding (VVC) appears the most recent video coding standard developed jointly by JVET, as known H.266 [BCO+21]. It is based on the same hybrid video coding block, as its predecessors from MPEG-2 to HEVC. VVC is designed to be both efficient and versatile to address today's media needs. This includes approximately $30\% - 50\%$ bitrate reduction over HEVC [WHB+20], as well as versatility by efficient coding of a wide range of video content and applications.

The main objective of this thesis is to significantly improve the coding efficiency of HEVC and VVC standards based on fast machine learning algorithms. This manuscript is structured in four chapters:

**Chapter I : Video Coding and Artificial Intelligence Backgrounds**

The first chapter introduces the most emerging video technologies nowadays. For this purpose, the hybrid aspects of the video coding standards are discussed first and then some essential modules to build a codec with this structure are detailed. In order to emphasize on the similarity of general structure between different video coding standards, some sections also provide equivalent historical and descriptive information from HEVC along with VVC. Meanwhile, we summarize the most challenges in video coding standards. After that, we introduce the recent advancements in machine learning and deep learning models and their categories. Finally, some related research based on video coding techniques are reviewed.

**Chapter II : Machine Learning Approach-based Fast CU Partition for Reducing HEVC Complexity**

The second chapter proposes a fast Coding Unit (CU) partition based on machine learning approaches to reduce the HEVC complexity of inter-mode. An online Support Vector Machines (SVM)-based fast CU partition method is proposed for reducing the encoding complexity of HEVC. Afterwards, to predict the CU partition of HEVC, a Deep Convolutional Neural Network (CNN) is proposed, which reduces the HEVC complexity at inter-mode. Unfortunately, these two machine learning algorithms do not explore the correlation of the CU partition across neighboring frames. A Long-and Short-Term Memory (LSTM) model was developed to learn the temporal dependency of the inter-mode CU partition. Therefore, a deep learning approach is proposed to predict the CU partition at inter-mode, which combines the CNN and LSTM structures. Finally, the obtained results are discussed in order to evaluate the performance of the proposed algorithms.

**Chapter III : Deep Learning based Video Quality Enhancement for the New Versatile Video Coding**

The third chapter proposes a deep learning algorithm-based VVC standard to enhance visual video quality while improving the user's Quality of Experience (QoE). The proposed Wide-activated Squeeze-and-Excitation Deep Convolutional Neural Network (WSE-DCNN) model is integrated into VVC standard to replace in-loop filtering in order to alleviate the coding artifacts, such as ringing, blocking, and blurring. The proposed VVC filtering technique is used in the Multimedia-Internet of Things (M-IoT) scenario-based smart city context to help the centralized cloud meet the user's required video quality. Finally, all simulation results obtained are interpreted and compared to the related existing methods.

**Chapter IV : Deep CNN Co-Design for HEVC CU Partition Prediction on FPGA-SoC**

The last chapter proposes a deep CNN based hardware-software design for HEVC CU prediction on FPGA-SoC. Our proposed work aims to accelerate the CNNs due to their computationally intensive. Hence, we create a hardware Intellectual Property (IP) core for each CNNs layer using the Vivado HLS tool. Then, we have designed a hardware-software architecture by importing the hardware IP cores based on the PYNQ-Z1 board. Finally, the achieved results are discussed and a comparative study is made.

Finally, the last part of this thesis will be reserved for a general conclusion that summarizes the results found and lists the different perspectives.

# Chapter I

# Video Coding and Artificial Intelligence Backgrounds

**Summary**

## I.1 Introduction

In this chapter, a brief review of the video compression structure is presented. For this purpose, the hybrid video coding standards scheme is first discussed and then some essential modules to build a codec with this structure are detailed. In order to emphasize on the similarity of general structure between different video coding standards, some sections also provide equivalent historical information from High Efficiency Video Coding (HEVC) along with Versatile Video Coding (VVC). Meanwhile, we summarize the most challenges in video coding standards. After that, we introduce the recent advancements in machine learning and deep learning models and their categories. Finally, this chapter provides a detailed literature review on different advanced video coding approaches that have been proposed.

The remainder of this chapter is organized as follows. Section I.2 presents the video compression history. The HEVC standard structure is described in Section I.3. Section I.4 introduces the VVC coding tools. Then, Section I.5 provides the video coding challenges. Afterwards, a detailed overview of artificial intelligence technique is exposed in Section I.6. The related research of video coding approaches is presented in Section I.7. Finally, Section I.8 concludes this chapter.

## I.2 Video Compression History

Every second in year 2021, more than a million minutes of video content will cross the network. It would take a person more than 5 million years to watch all videos of one month [CCH18]. This forecast is convincing for video coding experts to think of more efficient compression tools and technologies. These technologies are expected to address various emerging video formats, namely High Dynamic Range (HDR), High Frame Rate (HFR), high resolution videos (e.g. 4K, 8K and beyond), immersive 360° videos, screen content and more. Video coding standards aim at bringing format compatibility between devices. This enables the playback of any video file conforming the syntax of a given standard, with any device supporting it. From the industrial point of view, such unity facilitates the interaction between all components of the broadcast chain, including consumer electronics manufactures, broadcasters, content providers etc. This convenience in interaction, if achieved, can significantly accelerate the progress of the broadcast industry as a whole.

Historically, two major video coding standardization organizations have coexisted: Moving Picture Experts Group (MPEG), which belongs to the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC), and the Visual Coding Experts Group (VCEG), which belongs to the International Telecommunication Union, Telecommunication Standardization Sector (ITUT). The history of the different H.26x and MPEG-x families established by ITUT and ISO/IEC is shown in Figure I.1.

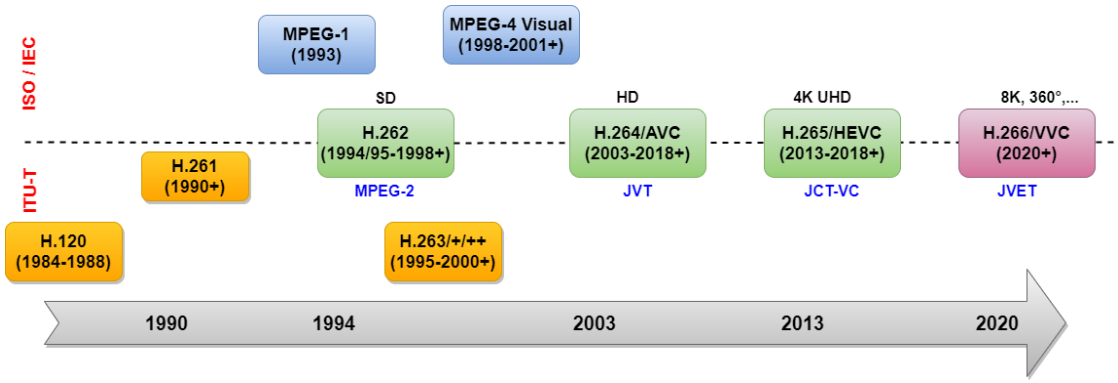



Figure. I.1: History of Video Coding Standardization

Most successful standardization acts of the MPEG were accomplished after its collaboration with the ITUT, in the late 90's. This joint collaboration, initially called Joint Video Team (JVT), then Joint Collaboration Team on Video Coding (JCT-VC), resulted in some of the most successful video coding standards in the family of "H.26x", notably H.264/Advanced Video Coding (AVC), developed in May 2003 [CKL06], and H.265/High Efficiency Video Coding (HEVC), finalized in January 2013 [SOHW12]. In October 2015, another collaboration between MPEG and VCEG formed the Joint Video Exploration Team (JVET) [MAS18] that was tasked with assessing the available compression technologies and exploring the requirements for a next-generation video compression standard. Hence, the new video coding standard called H.266/Versatile Video Coding (VVC) was standardized in July 2020 [BCO+21]. After the history presentation of the different video coding standards, the two latest HEVC and VVC will be detailed in next sections, since they will be used in this thesis.

## I.3 HEVC Standard

High Efficiency Video Coding (HEVC) is the sophisticated video coding standard, also known as H.265, standardized in 2013 by the JCT-VC [Ric13]. HEVC saves approximately 50% of bitrate for the same subjective video quality, with respect to its predecessor H.264/AVC standard. Thus, the HEVC codec is expected to ease the burden on global networks where High Definition (HD) and Ultra High Definition (UHD) video content is becoming more and more popular. HEVC is based on the basic hybrid structure as employed by previous standards since H.261. However, the standard contains series of incremental improvements [Ric13] over H.264/AVC in order to achieve better compression efficiency. The block diagram of a hybrid video coding layer conforming with the HEVC standard is illustrated in Figure I.2.

In accordance, a typical video encoder compliant with the HEVC standard would start by dividing each frame into block-shaped regions, with the exact block partitioning being conveyed to the decoder. The first picture of the video sequence is coded using only intra picture prediction, i.e., the prediction of the blocks in the picture is only

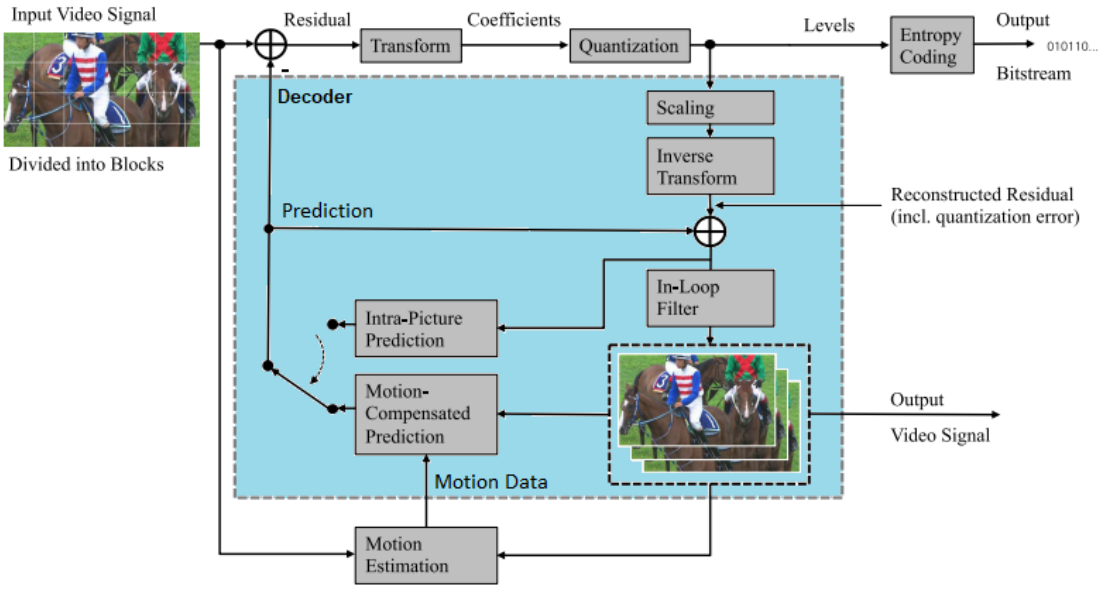



Figure. I.2: Block Diagram of the Hybrid Video Coding Layer for HEVC

based on the blocks from that same picture. For the remaining pictures of the video sequence, inter picture prediction is used. This type uses prediction information from other previously encoded pictures.

From Figure I.2, the result from the prediction is subtracted from the original block and the residual information is then transformed by a linear spatial transform. The transform coefficients are then scaled, quantized, compressed and transmitted in the receiver, together with the prediction information. The encoder also integrates the processing loop of the decoder in order to generate the same pictures as the output of the decoder. These pictures are then stored in a decoded picture buffer, and will be used for the prediction of the subsequent pictures. In the following, the general features of the hybrid video coding scheme used in HEVC will be described with more details.

### I.3.1 Sampled Representation of Pictures

Video sequence is typically captured using the RGB color space, which is not a particularity efficient representation for video coding. On the contrary, HEVC uses a more video coding friendly color space, the YCbCr, which divides the color space in 3 components: Y, known as luma, representing brightness; Cb and Cr, also known as chroma, which represent how much color deviates from gray towards blue and red, respectively.

As the human visual system is more sensitive to brightness, the typically used sampling scheme follows the 4:2:0 structure, meaning that four luma components are sampled for every chroma component. HEVC also supports each sample pixel value with 8 or 10 bits precision, with 8 bits being the most commonly used for HEVC standard [B+13] and 10 bits used for VVC standard [FJK+20].

### I.3.2 Block Partitioning in the HEVC Standard

In the former video coding standard H.264/AVC, Variable Macro-Block (MB) sizes ranging from 4×4 to 16×16 are supported [CKL06]. Whereas larger block sizes, reached at 64×64, are used in HEVC standard to facilitate the high definition video compression. Additionally, more flexible partitioning of video frames is supported to improve the

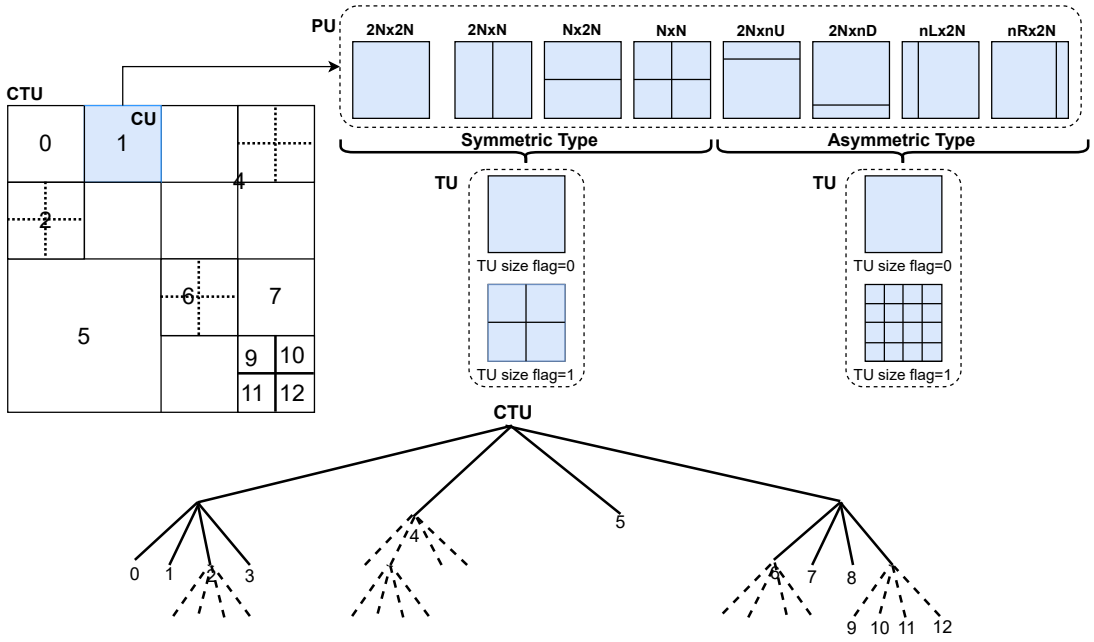



Figure. I.3: HEVC Quadtree Partitioning Structure, including CU, PU and TU (solid line for CU, dashed line for TU)

coding efficiency, where available block size varies from 4×4 up to 64×64, including symmetric partitioning, such as 2N×2N, 2N×N, N×2N and N×N, and also asymmetric motion partitioning (AMP) for instance 2N×nU, 2N×nD, nL×2N and nR×2N. In particular N×N is only allowed for minimum coding unit (CU) and AMP is not applied to CUs smaller than 16×16. Figure I.3 introduces the partitioning and quadtree structure

used in the HEVC standard, in which the analogous structure, called Coding Tree Unit (CTU), is splitted into CUs using a quad-tree partitioning structure, and a CU can be further sub-divided into Prediction Units (PU) for inter-frame or intra-frame prediction and its transformation is performed using one or more Transform Units (TU).

### I.3.3 Intra Prediction

In intra picture prediction, the information of adjacent CTU from the same picture is used for spatial prediction, as shown in Figure I.4. There are a total of 35 intra picture prediction modes available in HEVC, corresponding to 33 different directional modes, a DC and a planar mode. For directional mode encoding, the spatially neighboring decoded blocks are used as reference for the prediction, using the selected angle to cover the current PU. This mode is the most used for regions with strong directional edges. Directional mode prediction is consistent across all block sizes and prediction directions. DC mode encoding simply uses a single value matching the mean value of boundary samples for the prediction. Finally, the planar mode assumes an amplitude surface with a horizontal and a vertical slope derived from the boundaries. This mode is supported for all block sizes in HEVC.

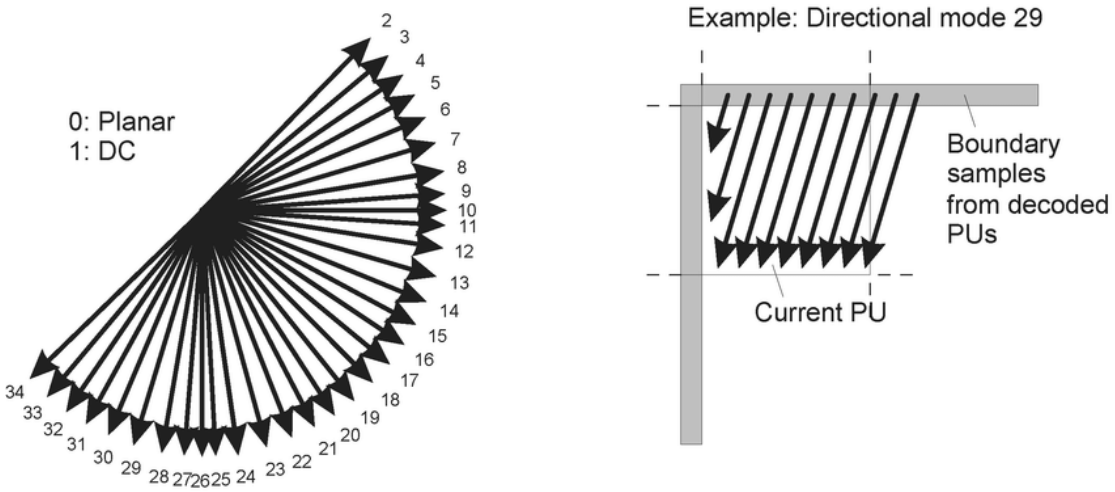



Figure. I.4: Intra Prediction Modes in the HEVC Standard

### I.3.4 Inter Prediction

In order to exploit the redundancies in the temporal adjacent images, inter-picture prediction based on previously coded pictures is an essential technique to obtain high compression rates. It consists of the application of the following two techniques: motion compensation and motion estimation. By using these techniques, pictures are predicted from previously encoded frames (uni-directional) or from previous and future frames (bi-directional), as shown in Figure I.5. The use of the bidirectional prediction is more complex, since it requires the video frames to be coded and stored out of order, so that future frames may be available.

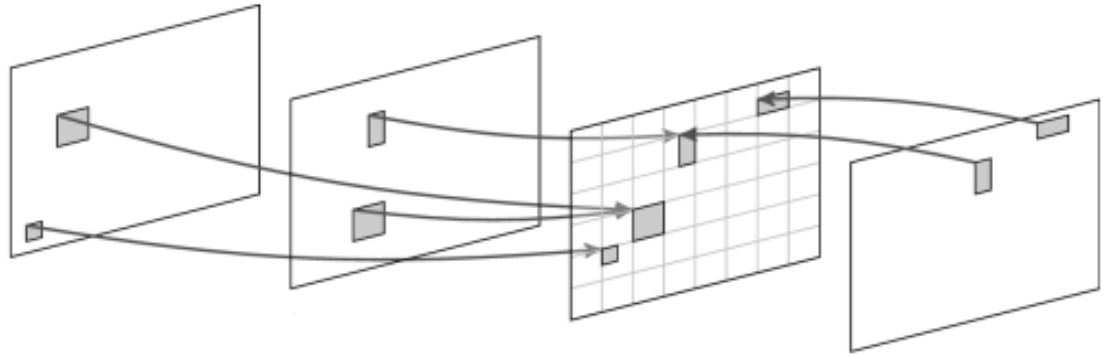


Figure. I.5: Example of Uni and Bi-directional Inter Prediction

Before the application of motion compensation technique, the encoder has to find a block similar to the one it is encoding on a previous/future encoded frame, referred to as a reference frame. Such searching procedure is known as motion estimation, resulting in the identification of a motion vector, which points to the position of the best prediction block in the reference frame. However, since the identified block will most likely not be an exact match of the encoding block, the resulting difference (residue) has to be encoded and transmitted to the decoding end, so that it can be read by the decoder. These residuals, originated from the difference between the predicted block and the actual block, are known as prediction errors.

The actual position of the prediction in the neighboring frames may be out of the sampling grid (where the intensity is unknown), so the intensities of the positions in between the integer pixels must be interpolated and the resolution of the motion vector

increased accordingly. For the interpolation in fractional luma sample positions, an 8-tap filter is used, while a 4-tap filter is used for chroma samples.

### I.3.5 Transform and Quantization

After the motion estimation, all the prediction error residuals are transformed into a set of coefficients for efficient transmission and storage. In the HEVC standard, as indicated in Figure I.3, TUs of size 4×4, 8×8, 16×16 and 32×32 are supported. The 2D transforms based on Discrete Cosine Transform (DCT) are designed for them and special efforts are particularly spent on selecting the value of the transfrom matrix for retaining the property of easy-to-implementation [SOHW12]. In addition, when transforming for 4×4 block size in intra-frame prediction mode, another integer transformation based on Discrete Sine Transform (DST) is available for use. The resulting transform coefficients are then quantized, before being sent to the construction of the coded bitstream. Quantization is a compression technique which converts a range of values into a single quantum value. The maximum Quantization Parameter of HEVC standard is set to 51.

### I.3.6 Entropy Coding

In the HEVC standard a bitstream is produced using motion parameters, prediction modes, quadtree partitioning information, quantized transform coefficients and some other control data through entropy coding. Only one entropy coding method, Context-Adaptive Binary Arithmetic Coding (CABAC), is specified in the standard. Although there is no change made on the core algorithm of CABAC, it is optimized on the aspects of context modeling, adaptive coefficient scanning, coefficient coding, sign data hiding and so on to improve its throughput.

### I.3.7 In-Loop Filters

Before writing the samples in the decoded picture buffer, they are processed first by a deblocking filter (DBF) and then by a sample adaptive offset filter (SAO). Block based coding schemes tend to produce blocking artifacts due to the fact that inner blocks are coded with more accuracy than outer blocks. To mitigate such artifacts, the decoded

samples are filtered by a DBF. After the deblocking has been processed, the samples are processed by SAO, a filter designed to allow for better reconstruction of the original signal amplitudes, reducing banding and ringing artifacts. SAO is performed on a per CTU basis and may or may not be applied, depending on the filtering type selected.

## I.4 VVC Standard

Versatile Video Coding (VVC) [BCO+21] [HBA+21] is the new generation video coding developed in July 2020, by the Joint Video Experts Team (JVET), as a successor of the HEVC [SOHW12]. As the next standard for sophisticated video coding technology, VVC allows up to $30\% - 50\%$ for bitrate savings while maintaining the same quality as HEVC. VVC has been designed to achieve improved compression capacity over previous standards such as HEVC, and at the same time to be highly versatile for effective use in a broadened range of applications. Some key application areas for the use of VVC particularly include UHD video (e.g. 4K or 8K resolution), video with a high dynamic range, and video for immersive media applications such as 360° omnidirectional video, in addition to the applications that have commonly been addressed by prior video coding standards.

Similar to its predecessor HEVC, VVC uses a block-based hybrid coding architecture with some coding tools that may be included or removed. The VVC architecture includes the inter-picture, intra-picture prediction, and transform coding with entropy coding.

### I.4.1 Block Partitioning in the VVC Standard

In VVC, each picture is split into non-overlapping squares called CTUs. The largest CTU size allowed in VVC is $128 \times 128$ pixels, larger than the maximum size allowed in HEVC, $64 \times 64$. Large blocks improve the efficiency of coding flat areas such as backgrounds, especially for high-resolution videos such as HD and 4K. In order to efficiently represent highly detailed areas such as textures and edges, VVC employs a flexible partitioning scheme that can CTUs partition sized of $128 \times 128$ down CUs as small as $4 \times 4$ pixels.

Figure I.6 shows one CTU divided into multiple CUs with a QuadTree plus Multi-

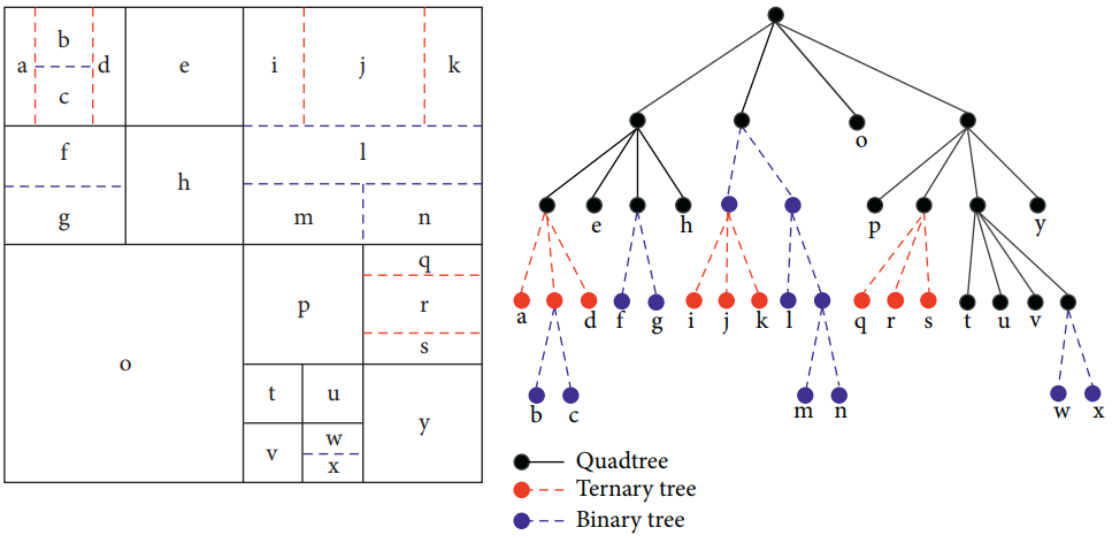



Figure. I.6: Example of Quadtree with Nested Multi-Type Tree Coding Block Structure for VVC

type Tree QTMT coding block structure, where the solid block edges represent quadtree partitioning and the dotted edges represent multitype tree partitioning with either binary or ternary splits. The first is the quadtree split that is also available in HEVC, which can recursively split CTU into squared CUs down to $4 \times 4$ pixels, smaller than the $8 \times 8$ minimum CU size in HEVC. The second part consists of binary-tree and ternary-tree splits that partition a block into two and three rectangles respectively. Both binary and ternary tree splits can operate in either horizontal or vertical directions, be recursively applied and mixed together in a nested multi-type tree.

The block partitioning in VVC is highly flexible and provides about 8 percent bitrate reduction over HEVC. However, this flexibility comes at a computational cost, especially on the encoder side, where many more permutations need to be evaluated to select the optimal partition.

### I.4.2 Intra Prediction

The number of directional intra modes in VVC is extended from 33, as used in HEVC, to 65. The new directional modes, not used in HEVC, are depicted by red dotted arrows, as mentioned in Figure I.7, whereas the planar and DC modes are unchanged for both video encoder. These denser directional intra prediction modes apply for all block sizes and

for both luma and chroma intra predictions. In HEVC, 33 angular prediction directions are defined from 45° to 135° in a clockwise direction. In VVC, the angular precision is basically doubled to produce 65 angles within that same range, and another 28 "wide-angle" prediction modes beyond this angular range can be used for non-square blocks, as illustrated in Figure I.8.

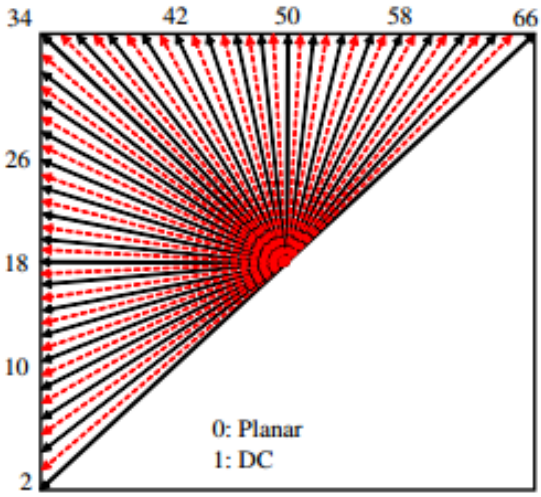



Figure. I.7: Intra Directional Modes in VVC

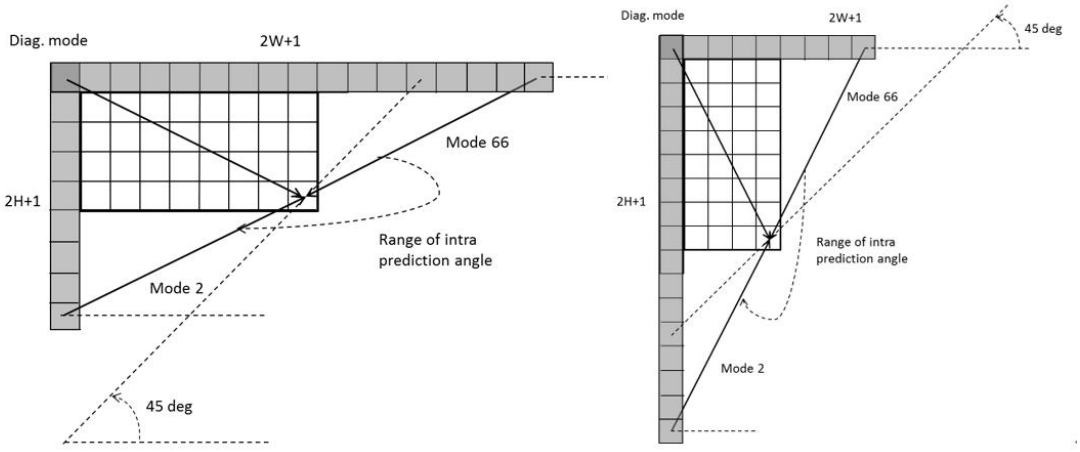



Figure. I.8: Wide-Angular Intra-Picture Prediction

### I.4.3 Inter Prediction

The basic concepts of uni-directional and bi-directional motion compensation from one or two reference pictures are mostly unchanged. However, there are some new tools

that have not been used in the last video coding standard. For each inter-predicted CU, motion parameters consisting of motion vectors, reference picture indices and reference picture list usage index, and additional information needed for the new coding feature of VVC to be used for inter-predicted sample generation. The motion parameter can be signalled in an explicit or implicit manner. When a CU is coded with skip mode, the CU is associated with one PU and has no significant residual coefficients, no coded motion vector delta or reference picture index. A merge mode is specified whereby the motion parameters for the current CU are obtained from neighbouring CUs, including spatial and temporal candidates, and additional schedules introduced in VVC. The merge mode can be applied to any inter-predicted CU, not only for skip mode. The alternative to merge mode is the explicit transmission of motion parameters, where motion vector, corresponding reference picture index for each reference picture list and reference picture list usage flag and other needed information are signalled explicitly per each CU.

Beyond the inter coding features in HEVC, VVC includes a number of new and refined inter prediction coding tools listed as follows; Extended merge prediction, 1/16th luma sample MV storage and $8 \times 8$ motion field compression, Bi-prediction with CU-level weight (BCW), and Bi-directional optical flow (BDOF), etc.

### I.4.4 Transform and Quantization

The size of transform block is increased from $4 \times 4$ to $64 \times 64$ in the VVC standard compared to the HEVC standard. In addition to the DCT-II used in HEVC, a multiple transformation selection (MTS) scheme is also used for residual coding of intra and inter coding blocks. The newly introduced transformation matrices are DST-VII and DCT-VIII. The change of the Quantization stage is the increase in the maximum Quantization Parameter (QP) from 51 to 63.

### I.4.5 Entropy Coding

In VVC, the CABAC technique is improved in comparison to the HEVC design. The three main modifications are: modified context modeling for transform coefficients, multi-hypothesis probability estimation with context-dependent updating speed and

adaptive initialization for context models (e.g. initial probability states of context models for inter coded slices can be initialized by copying states from previously coded pictures).

Table I.1: Coding Tools of VVC vs HEVC

| | HEVC | VVC |
|---|---|---|
| **Block partitioning** | ◆ Coding Tree Unit (CTU) quadtree (QT) structure<br>◆ From 64×64 to 8×8 Coding Unit (CU) size | ◆ CTU quadtree structure with nested multi-type tree (QT+MTT)<br>◆ From 128×128 to 4×4 CU size<br>◆ Chroma separate tree (CST)<br>◆ Local dual tree<br>◆ Virtual pipeline data units (VPDUs) |
| **Intra prediction** | ◆ DC, planar<br>◆ 33 directional prediction modes<br>◆ Linear interpolation | ◆ DC, planar<br>◆ 65 directional prediction modes<br>◆ Wide-angle prediction modes<br>◆ 4-tap interpolation filters (IFs) using 2 sets of filters<br>◆ Position-dependent prediction combination (PDCP)<br>◆ Multiple reference lines (MRL)<br>◆ Matrix-based intra-picture prediction (MIP)<br>◆ Cross-component linear model (CCLM)<br>◆ Intra sub-partitions (ISP) |
| **Inter prediction** | ◆ Merge mode<br>◆ Advanced motion vector prediction (MVP)<br>◆ 8-tap IFs for luma<br>◆ 4-tap IFs for chroma | ◆ Extended merge mode and MVP with<br>▪ History-based MVP (HMVP)<br>▪ Pair-wise average MVP candidate<br>▪ Subblock-based temporal MVP (SBTMVP)<br>▪ Merge with motion vector difference (MMVD)<br>▪ Symmetric motion vector difference (SMVD)<br>◆ Adaptive motion vector resolution (AMVR)<br>◆ 8-tap or 6-tap IFs for luma<br>◆ 4-tap IFs for chroma<br>◆ Geometric partitioning mode (GPM)<br>◆ Bi-prediction with CU-level weight (BCW)<br>◆ Combined Intra/Inter prediction (CIIP)<br>◆ Decoder-side motion vector refinement (DMVR)<br>◆ Bi-directional optical flow (BDOF)<br>◆ Affine motion<br>◆ Prediction refinement with optical flow (PROF) |
| **Forward/inverse transform and quantization** | ◆ Square transform (up to 32×32)<br>◆ Discrete cosine transform and discrete sine transform<br>▪ DCT-II and DST-VII<br>◆ Sign data hiding (SDH) | ◆ Square and rectangular transform (up to 64×64)<br>◆ Multiple transform selection (MTS)<br>▪ DCT-II, DST-VII, and DCT-VIII<br>◆ Non-separable secondary transform (LFNST)<br>◆ Subblock transform (SBT)<br>◆ Adaptive chroma QP offset<br>◆ SDH<br>◆ Dependent quantization (DQ)<br>◆ Joint coding of chroma residuals (JCCR) |
| **Entropy coding** | ◆ Context-adaptive binary arithmetic coding (CABAC)<br>◆ Coefficient group<br>◆ Reverse diagonal, horizontal and vertical coefficient scan | ◆ CABAC with multi-hypothesis probability estimates<br>◆ Additional coefficient group size<br>◆ Reverse diagonal coefficient scan only<br>◆ Improved probability model sections for absolute transform coefficient levels |
| **Loop filtering** | ◆ Deblocking filter (DF)<br>◆ Sample adaptive offset (SAO) | ◆ Luma mapping with chroma scaling (LMCS)<br>◆ Deblocking boundary handling modifications<br>◆ Deblocking long filter<br>◆ Luma-adaptive deblocking<br>◆ Sample adaptive offset (SAO)<br>◆ Adaptive loop filter (ALF)<br>◆ Cross-Component ALF (CC-ALF) |
| **Parallelization** | ◆ Slices<br>◆ Tiles<br>◆ WPP with CTU row delay of two CTUs | ◆ Slices<br>◆ Tiles<br>◆ Subpictures<br>◆ WPP with CTU row delay of one CTU |

### I.4.6 In-Loop Filters

In VVC, a remapping operation and three in-loop filters can be applied sequentially to the reconstructed picture to modify its representation domain and alleviate different types of artifacts. First, a new sample-based process called LMCS (Luma Mapping with Chroma Scaling) is performed. Then, a DBF is used to reduce blocking artifacts.

SAO is then applied to the deblocked picture to attenuate ringing and banding artifacts. Finally, an Adaptive Loop Filter (ALF) reduces other potential distortion introduced by the quantization and transform processes. The deblocking filter design is based on the one in HEVC but is extended with longer deblocking filters and a luma-adaptive filtering mode designed specifically for HDR video. While SAO is the same as in HEVC, and the deblocking is very similar, LMCS and ALF are new compared with previous standards. The design of ALF in VVC consists of two operations: ALF with block-based filter adaption for both luma and chroma samples and a cross-component ALF (CC-ALF) for chroma samples.

Table I.1 summarizes the main coding tools of HEVC and VVC. VVC has adopted many new coding tools in each coding stage [MMS+21].

## I.5 Video Coding Challenges

During the last decade, multimedia services and video applications have significantly increased due to the huge progress in digital technologies. The emerging video applications and image representation offer an immersive and more natural viewing experience. However, these new services require both higher quality and resolution (4K, 8K) to satisfy the quality of service required by the end users. To meet the increasing demands for video content at better qualities and higher resolutions, video compression technology is being researched and developed, due to its higher performance. However, this unmatched performance is achieved by increasing the encoder computational complexity mainly due to its block partition structure. Indeed, the complexity reduction has always been a popular challenge in the video coding field. For example, Figure I.9 shows that the greatest complexity lies in the selection of the optimal prediction mode, especially in the inter-mode [CMMC19].

In this context, many researchers aim to reduce the complexity for each standard module through fast methods. The efficiency and versatility of recent Machine Learning based approaches paved the way for researching more extensive ways to integrate Machine Learning solutions into video coding schemes. In fact, they are able to achieve

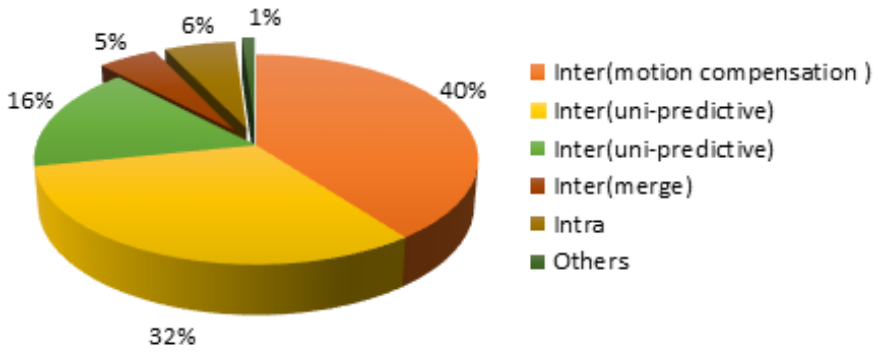



Figure. I.9: Example of HEVC Time Profile

higher compression efficiency than older video compression technologies. In the next section, we will review the new technologies, such as Artificial Intelligence, Machine Learning and Deep Learning.

## I.6 Artificial Intelligence: New Advancements and Innovations

Artificial Intelligence (AI) is a branch of computer science that deals with simulation of human intelligence by machines processes and computational rationality. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving. AI is a computer system able to perform tasks that normally require human intelligence, such as visual perception, speech recognition, decision making, and translation between languages. Machine learning and deep learning are subsets of AI, which are described in the following sections.

### I.6.1 Machine Learning

The learning activity is essential for the human beings in order to understand and recognize various parameters such as a voice, a person, an object, and others. One generally distinguishes the learning which consists of memorizing information [AM05] [SS94], and the learning by generalization [Wit74] [OW83] in which we usually build a model from learning examples to recognize new examples and scenarios. For the machines, it is easy

to handle a large amount of data but difficult to build a good model which is able to effectively recognize new objects in a new test. Machine learning (ML) is an attempt to understand and reproduce this learning facility in an artificial system. It therefore seems appropriate to use techniques from this field to discover and model knowledge and reduce the semantic gap [MMC13]. ML is at the crossroads of various fields such as artificial intelligence, statistics, cognitive science, probability theory, optimization, signal and information, and so on [B+01] [RN16] [Ful08]. It is therefore very difficult to give taxonomy of machine learning categories. Then, we briefly present in this section the four main types of machine learning techniques [DB17]: Supervised Learning [KZP07], Unsupervised Learning [HTF09], Semi-supervised Learning [CSZ09], and Reinforcement Learning [KLM96].

Supervised learning systems make use of labeled datasets [KZP07]. This training set of input-output pairs is used to find a deterministic function that maps any input to an output, predicting future input-output observations while minimizing errors as much as possible. While Unsupervised learning systems use unlabeled datasets to train the system [HTF09]. The objective of unsupervised learning is to derive structure from unlabeled data by investigating the similarity between pairs of objects, and is usually associated with density estimation or data clustering. Reinforcement learning systems do not experience a fixed dataset, but a feedback loop between the system and its experiences [KLM96]. A dynamic environment is considered in which state-action-reward triples are observed as the data. The objective of reinforcement learning is mapping situations to actions with the goal of maximizing rewards. Other existing learning systems that are a combination of two categories, such as semi-supervised learning that uses both labeled and unlabeled data [CSZ09].

Here, we limit our focus to supervised learning algorithms. There are a wide variety of tasks exist that could be solved with machine learning. However, two popular machine learning tasks are regression analysis and classification. Commonly used algorithms for classification technique [KZP07] include k-Nearest Neighbor, Support Vector Machine, Naïve Bayes, and Decision Trees, etc...In the following, we focus on describing the Support Vector Machine (SVM) considered as the most useful algorithm, due to its

ability to solve classification task problems. Indeed, SVM is a class of learning algorithm, initially used for discrimination that is, predicting a binary qualitative variable which is then generalized forecast a quantitative variable. In case of discriminating a dichotomous variable, they are based on the search for the optimal margin hyperplane. The hyperplane, where possible, classifies or separates the data correctly while being as far as possible from all observations, based on the training set. The principle is therefore to find a classifier or a discrimination function whose generalization capacity (forecast quality) is acceptable for the specific application. Therefore, the purpose of SVM is the reduction of discrimination problem to the linear problem of finding an optimal hyperplane [HCP+18].

In Figure I.10, the principle of the SVM algorithm has been shown. Finding the optimal hyperplane to differentiate classes is the major functionality of SVM techniques. The Figure I.10 (a) presents two classes consisting of circles and stars which need to be separated. SVM is a frontier which best segregates the two classes (hyperplane). Now the important question is how can one identify the right hyperplane?. The response is in the Figure I.10 (b) which maximizes the distance between the nearest data point (either class) and hyperplane will help us to decide the right hyperplane. This distance is defined as margin. So, the margin of the hyperplane $C$ is the highest as compared to $A$ and $B$. Hence, the hyperplane $C$ is the optimal hyperplane that can classify data set.

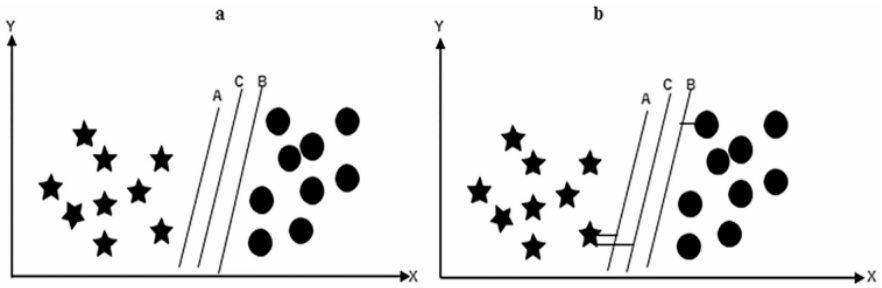



Figure. I.10: Example of SVM Classifier

### I.6.2 Deep Learning

The past decade has witnessed the emerging and booming of Deep Learning (DL), a class of techniques that are increasingly adopted in the hope of approaching the ultimate goal of artificial intelligence [WBAK20]. DL belongs to machine learning technology, and has the distinction of its computational models, known as deep artificial neural networks or deep networks for short, which are composed of multiple (usually more than three) processing layers, each layer is further composed of multiple simple but non-linear basic computational units. One benefit of such deep networks is believed to be the capacity for processing data with multiple levels of abstraction, and converting data into different kinds of representations. Note that these representations are not manually designed; instead, the deep network including the processing layers is learned from massive data using a general machine learning procedure. DL eliminates the necessity of handcrafted representations, and thus is regarded useful especially for processing natively unstructured data, such as acoustic and visual signal, since the processing of such data is considered a long-standing problem in the field of artificial intelligence.

Specifically for processing image/video, DL using Convolutional Neural Network (CNN) has revolutionized the paradigm in computer vision and image processing [BKSA20a]. CNN is one of the most commonly used supervised deep learning models, which is described in this chapter. This network structure was first proposed by Fukushima in 1988 [Fuk88]. In the 1990s, LeCun et al. [LBBH98] applied a gradient-based learning algorithm to CNNs and obtained successful results for the handwritten digit classification problem. After that, researchers further improved CNNs and reported state-of-the-art results in many recognition tasks. CNNs have several advantages over Deep Neural Networks (DNNs), including being more like the human visual processing system, being highly optimized in the structure for processing 2D and 3D images, and being effective at learning and extracting abstractions of 2D features.

Figure I.11 introduces the overall architecture of CNNs consisting of two main parts: Feature extraction and classification. In the feature extraction layers, each layer of the network receives the output from its immediate previous layer as its input and passes its output as the input to the next layer. In the classification part, the feature maps of

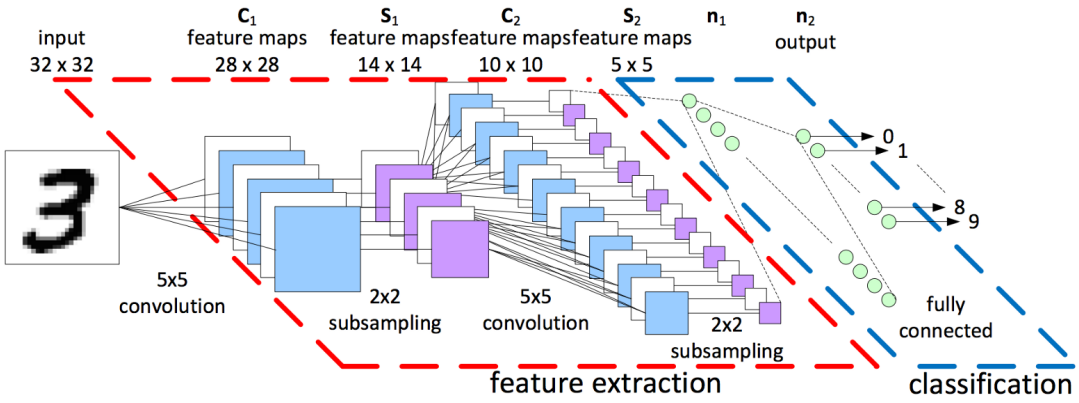



Figure. I.11: Overall Convolutional Neural Network Architecture

the last layer of the CNN are used as the input to a fully connected layer which is called classification layer using an activation function to calculate the score of the respective class. However, the CNN architecture consists of a combination of three types of layers: Convolutional, pooling, and fully connected layers. The mathematical details on the different layers of CNN are discussed below.

### I.6.2.1 Convoluional Layer

Convoluional Layer is the core building block of CNN network, in which its parameters consist of a set of learnable filters also known as kernels. The main task of the convolutional layer is to detect features found within local regions of the input image that are common throughout the dataset and mapping their appearance to a feature map. A feature map is obtained for each filter in the layer by repeated application of the filter across sub-regions of the complete image, i.e., convolving the filter with the input image, adding a bias term, and then applying an activation function. Therefore, four important hyperparameters in the convolutional layer are used, such as Filter Size, Number of Filters, Stride, and Zero Padding. The following equation shows the convolution operation.

$$x_j^l = f\left(\sum_{i \in M_j} x_i^{l-1} \times \ k_{ij}^l + b_j^l\right) \tag{I.1}$$

where $x_j^l$ is the output of the current layer, $x_i^{l-1}$ is the previous layer output, $k_{ij}^l$ is the kernel for the present layer, and $b_j^l$ are the biases for the current layer. $M_j$ represents a

selection of input maps.

#### I.6.2.2 Pooling Layer

In CNN, the sequence of convolution layer and activation function layer is followed by an optional pooling or down-sampling (also sub-sampling) layer to reduce the spatial size of the input and thus reducing the number of parameters in the network. A pooling layer takes each feature map output from the convolutional layer and down-samples it, i.e., pooling layer summarizes a region of neurons in the convolution layer. Two types of operations are mostly performed in this layer: Average pooling or max-pooling. In the case of the average pooling approach, the function usually sums up over N×N patches of the feature maps from the previous layer and selects the average value. On the other hand, in the case of max-pooling, the highest value is selected from the N×N patches of the feature maps. The poling operation can be defined in equation I.2, where down(·) represents a sub-sampling function.

$$x_j^l = down\left(x_i^{l-1}\right) \tag{I.2}$$

#### I.6.2.3 Fully Connected Layer

At the end, the stack of convolutional and pooling layers act as feature extraction stage while as the classification stage is composed of one or more fully connected layers followed by an activation function layer. The process of convolutional and pooling continues until enough features are detected. Next step is to make a decision based on these detected features. In case of classification problem, the task uses the detected features in the spatial domain to obtain probabilities that these features represent each class, that is, obtain the class score. This is done by adding one or more fully connected layers at the end. In fully connected layer, each neuron from previous layer is connected to every neuron in the next layer and every value contributes in predicting how strongly a value matches a particular class.

Additionally, a fully connected layer is connected to all features, and it is prone to overfitting. Overfitting refers to the problem when a model is trained and it works so

well on training data that it negatively impacts the performance of the model on new data.In order to overcome the problem of overfitting, a dropout layer can be introduced in the model in which some neurons along with their connections are randomly dropped from the network during training. Only the reduced network is trained on the data in that stage. The removed nodes are then reinserted into the network with their original weights. Dropout notably reduces overfitting and improves the generalization of the model.

#### I.6.2.4 Activation Functions

The output of each convolutional layer is fed to an activation function layer. The activation function layer consists of an activation function that takes the feature map produced by the convolutional layer and generates the activation map as its output. The activation function is used to transform the activation level of a neuron into an output signal. There are many activation functions and some of the commonly used activation functions are as follows: Rectified Linear Unit (ReLU) has gained some importance in recent years and currently is the most popular activation function for deep neural networks. Neural networks with ReLU train much faster than other activation functions. ReLU simply computes the activation by thresholding the input at zero. In other words, a rectified linear unit has output 0 if the input is less than 0, and raw output otherwise. It is denoted as.

$$f(x) = max(0, x) \tag{I.3}$$

The sigmoid function is mathematically represented in the equation I.4. It squashes the input into the range [0, 1].

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{I.4}$$

In addition, the hyperbolic tangent function (tanh) is similar to sigmoid function but its output lies in the range [-1, 1]. The advantage of tanh over sigmoid is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero. Moreover, softmax function is often used in the output layer of a neural network for classification. It is a more generalized logistic activation function which is

used for multiclass classification, which is defined as.

$$\sigma\left(x_j\right) = \frac{e^{x_j}}{\sum_{k=1}^{n} e^{x_k}} \tag{I.5}$$

#### I.6.2.5 Backpropagation Algorithm

For the training phase, the common algorithm used is the backpropagation [RZ85]. The training procedure requires us to be able to calculate the derivative of the cost function with respect to all the parameters of the neural network (the weights and biases of all the neurons in the input, hidden, and visible layers). The backpropagation algorithm is a clever procedure that exploits the layered structure of neural networks to more efficiently compute gradients [MBW+19]. This algorithm consists of a forward pass from the bottom layer to the top layer where one calculates the weighted inputs and activations of all the neurons. One then backpropagates the error starting with the top layer down to the input layer and uses these errors to calculate the desired gradients. This description makes clear the incredible utility and computational efficiency of the backpropagation algorithm. We can calculate all the derivatives using a single "forward" and "backward" pass of the neural network, equations are summarized in A. This computational efficiency is crucial since we must calculate the gradient with respect to all parameters of the neural net at each step of gradient descent.

## I.7 Related Research

The existing video coding complexity reduction works can be generally classified into two categories: heuristic and learning based approaches. This section reviews the complexity reduction approaches in these two categories.

### I.7.1 Heuristic Methods

In heuristic methods, several fast decision algorithms have been introduced [CMMC19, WLM+16, XLWM13, CK13, FSK+20, PK19]. To reduce the HEVC computational complexity, authors in [CMMC19] introduced a look-ahead stage-based fast partitioning and

mode decision algorithm. Wang et al. in [WLM+16] proposed a threshold-based splitting decision scheme with respect to the RD cost of each CU. It reduces the number of available intra candidates, adaptive reference frame selection and early termination of coding unit splitting. In [XLWM13], authors proposed a fast algorithm to split CU based on pyramid motion divergence at inter prediction. In addition, a fast early CU-splitting and pruning method with low complexity and full RD cost was developed by Cho et al. in [CK13]. In a similar way, authors in [FSK+20] proposed a fast QTMT partition algorithm based on variance and gradient to reduce the computational complexity brought in by the novel MT partitions in VVC. Reference [PK19] proposes a context-based ternary trees (TT) decision (C-TTD) method to significantly reduce TT computational complexity in VVC intra-coding. These methods are based on the statistics on the RD cost properties, temporal and spatial correlation, which limit their applicability and may be difficult to handle the situations with various contents, complex coding structures.

### I.7.2 Learning Methods

The past few years have exhibited great success in applying machine learning tools to enhance the video coding. In this vein, great efforts have been carried out to integrate machine learning tools in order to predict the CU partition to reduce HEVC complexity [CAAdSC14, ZKW+15, ZZP+17, ZZK+17, THM+21, LXT+21]. The search for the optimal partitioning has also been considered as a classification problem. For example, Corrêa et al. [CAAdSC14] proposed data mining techniques based on three early termination schemes to simplify the decision on the optimal CTU structures. To reduce the encoding complexity, Zhang et al. [ZKW+15] propose a CU early termination algorithm. In this work, the authors designed a CU depth decision process in HEVC and model it as a three-level of hierarchical classification decision. In this regard, an SVM-based fast HEVC encoding algorithm was proposed by Zhu et al. [ZZP+17] to predict both the CU partition and PU mode. The CU early termination is modeled as hierarchical binary classifications, whereas the PU selection is decided as a multi-class classification. In order to reduce the HEVC encoding complexity, Zhu et al. [ZZK+17] proposed a CU decision method based on fuzzy SVM that achieve a good trade-off be-

tween computational complexity reduction and RD performance. For VVC standard, authors [THM+21] developed a machine learning based efficient QT-MTT partitioning scheme for VVC intra encoders. Similarly, reference [LXT+21] proposes a deep learning approach to predict the QTMT-based CU partition, for drastically accelerating the encoding process of intra-mode VVC. A fast CU partition decision algorithm based on the improved Directed Acyclic Graph Support Vector Machine model to reduce the complexity of CU partition [ZWH+21]. In addition, other components of HEVC and VVC, such as in-loop filtering, are simplified to reduce the encoding complexity.

Overall, the main target of video coding is to minimize the bitrate while maintaining the visual quality. There are three key requirements on the video coding [OS11], including high compression ratio, low complexity, and high visual quality. In this context, this thesis contribution proposes to integrate the advancement techniques in the video coding standards, in order to achieve a compression efficiency considerably higher than the old video compression technologies.

## I.8 Conclusion

This chapter introduces the basic building blocks of a video compression system and a detailed description of the HEVC and VVC standards, which are the current state-of-the-art video coding standards. The coding tools comparison of HEVC versus VVC is provided. Then, the chapter also presents a video coding standards challenges. Moreover, the recent advancements technologies, such as artificial intelligence, machine learning and deep learning and their tools are surveyed. Finally, the related researches on HEVC and VVC complexity reduction are reviewed.

In the next chapter, in order to overcome the HEVC complexity, we propose to integrate machine learning solutions in HEVC standard to predict the CU partition at inter-mode. This chapter will provide more details about the three proposed machine learning algorithms instead of traditional rate-distortion optimization search in HEVC standard.

# Chapter II

# Machine Learning Approach-based Fast CU Partition for Reducing HEVC Complexity

**Summary**

## II.1 Introduction

The past decade has witnessed great success of machine learning technology in many disciplines, especially in computer vision and image processing. However, machine learning-based video coding remains in its infancy. With this objective in mind, this chapter proposes a fast CU partition algorithm based on machine learning to reduce both HEVC complexity and RD performance. We first propose an online Support Vector Machines (SVM) based fast CU partition, which reduces the HEVC complexity at inter-mode. Then, this chapter provides a Deep Convolutional Neural Network (CNN) structure to predict the CU partition at HEVC inter-mode. Since, the latter does not explore the correlation of the CU partition across neighboring frames, we have developed a Long-and Short-Term Memory (LSTM) structure to learn the temporal dependency of the inter-mode CU partition. Finally, a deep learning approach is proposed to predict the CU partition at inter-mode, which combines the CNN and LSTM structures.

The remainder of this chapter is organized as follows. Section II.2 introduces the HEVC CU partition structure. Then, Section II.3 presents the proposed CU partition-based machine learning. The framework performance evaluation and simulation results are analyzed in Section II.4. Finally, Section II.5 concludes this chapter.

## II.2 CU Partition Structure

In the HEVC standard [LLYY15], the CTU supports quadtree CU partitions with four levels of CU depth from 0 to 3, which corresponds to CU size ranging from $64 \times 64$ to $8 \times 8$, as shown in Figure II.1. In the CTU with $64 \times 64$ size, the split flag is set to 0, the Rate-Distortion (RD) cost $RD1$ is calculated. Then the sub-CUs of $32 \times 32$ size are obtained when the split flag changes to 1. The first one, $CU_1 0$ has an RD cost equal to $RD2$. The next depth is reached when the CU is partitioned into four CUs of size $16 \times 16$. The first CU ($CU_2 0$) of size $16 \times 16$ has an RD cost equal to $RD3$. When its

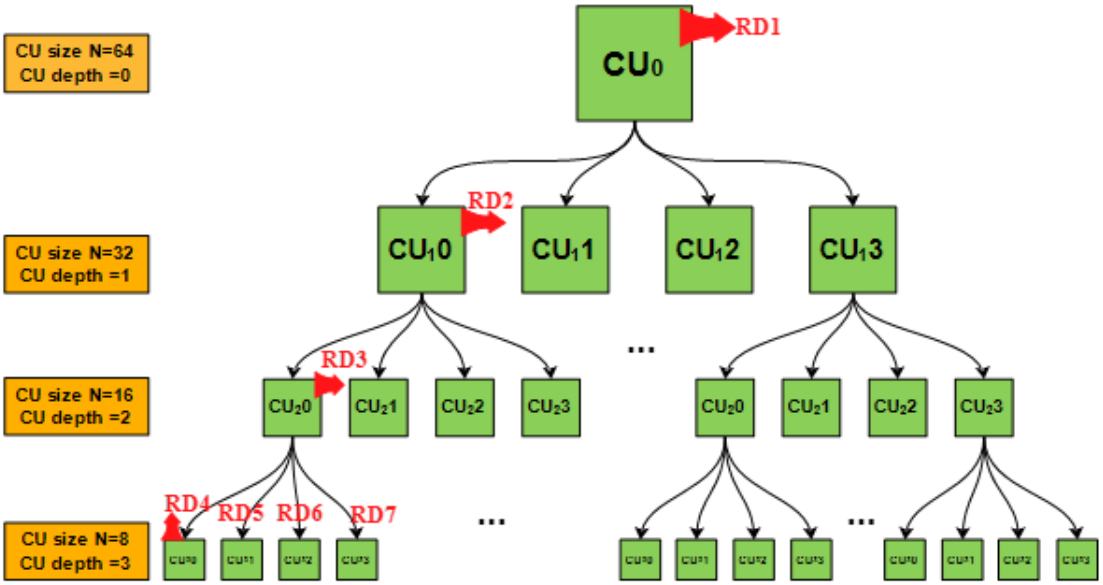



Figure. II.1: CU Partition Structure in HEVC

split flag is 1, the last depth (depth=3) is reached and it is therefore partitioned into four sub-CU of size $8 \times 8$. The RD cost for each sub-CU will be noted $RD4$, $RD5$, $RD6$ and $RD7$, respectively. The first decision will be taken from the bottom to the top by determining if the first CU of size $16 \times 16$ is checked or not. We need a comparison of the sum of the four RD cost of the sub-CU $8 \times 8$ with the $RD3$ of the CU $16 \times 16$ to make a decision. If the $RD3$ is greater than the sum of $RD4$, $RD5$, $RD6$ and $RD7$, the partitioning decision of $CU_20$ will be taken, otherwise $CU_20$ will not be split. The same for the other CUs, the decision is always based on the equation II.1. Generally, the Rate-Distortion Optimization (RDO) is a method to decide the optimal mode in video coding. After the full RDO search, the CU partition with the minimum RD cost is selected. It is worth noting that the RDO search is extremely time consuming, mainly attributed to the recursive checking process. In a $64 \times 64$ CTU, 85 possible CUs are checked, including 1, 4, $4^2$ and $4^3$ CUs with sizes of $64 \times 64$, $32 \times 32$, $16 \times 16$ and $8 \times 8$.

$$RD_{cost}\_CU > \sum_{k=0}^{3} RD_{cost}\_sub\_CU\,(k) \tag{II.1}$$

The CU partition can be considered as a combination of binary classifiers $\{F_l\}_1^3$ at three levels of decisions $l \in 1, 2, 3$ on whether to split a parent CU into sub-CUs. Accord-

ing to the CTU, we assume that the CUs are denoted as $CU$, $CU_i$, $CU_{i,j}$ corresponding to depth 0,1,2,3, where $i, j \in 0, 1, 2, 3$ are the index of sub-CUs. In each CU depth, we need to determine whether to split the current CU or not. The overall CU partition in a CTU is extremely complex, due to the large number of possible pattern combinations. For example, for a $64 \times 64$ CU, if $F_1(CU)= 1$, it will be split into four $32 \times 32$ CUs, i.e., $\{CU_i\}_{i=0}^{3}$. Since for each $CU_i$ there exist $1 + 2^4 = 17$ splitting patterns in $\{CU_i, j\}_{j=0}^{3}$, the total number of splitting patterns for CU is $1 + 17^4 = 83522$. There are too many types of CU partitions and it is hard to be solved by a single multi-class classification in one step. However, due to the large number of pattern combinations, the prediction is adopted at each decision level to yield $\tilde{F}_1(CU)$, $\{\tilde{F}_2(CU_i)\}_{i=0}^{3}$, and $\{\tilde{F}_3(CU_{i,j})\}_{i,j=0}^{3}$, which denotes the predicted $F_1(CU)$, $\{F_2(CU_i)\}_{i=0}^{3}$, and $\{F_3(CU_{i,j})\}_{i,j=0}^{3}$, respectively.

## II.3 Proposed CU Partition based on Machine Learning

### II.3.1 CU Partition based on SVM

In machine learning theory, SVM is a supervised learning tool that performs classification analysis [CV95]. In particular, the video coding mode decision process can be considered as a classification problem. A hyperplane technique is used in SVM to separate data from one space at one dimension to another at a larger dimension. SVM can transform data in a larger dimensional space by nonlinear transformation, if the data points are clearly not linearly separable in the input space. To separate the two classes of data points, SVM maps the sample data into a hyperspace. In addition, the main goal of SVM is to solve linear and nonlinear problems in order to find an optimal hyperplane. SVM classifier creates a hyperplane in order to maximize the margin between hyperplanes and support vectors [HCP+18].

The CU split decision can be modeled as a binary classification problem, with classes split and non-split [BKSA20b]. Here, we propose an online SVM as a machine learning technique, since it is robust and popular in solving the binary classification problem with significant computational advantages. The main idea is to find a hyperplane that can separate the training samples of different classes while maximizing the margin be-

tween these classes in order to determine the CU splitting level. According to equation II.2, the ideal weight vector $w$ is a linear combination of support vectors. Therefore, the support vectors are the training points that minimize the misclassification. Given training set with $N$ samples, $(\{x_i, y_i\}_{i=1}^{N})$, $x_i \in R^n$, while $y_i \in (\{-1, 1\})$, the hyperplane parameterized by the normal vector $w$ that maximizes margins can be found by solving the optimization problem.

$$\min_{w} \frac{\gamma}{2} \| w \|^2 + \frac{1}{n} \sum_{i=0}^{n} max(0, 1 - y(w \cdot x)) \tag{II.2}$$

where $\gamma \geq 0$ is the smoothing parameter and is defined by: $\gamma$=$1/nC$, where C is the parameter which need to be tuned during SVM training.

Mathematically, SVMs handle such situations by using a kernel function which maps the data to a different space where a linear hyperplane can be used to separate classes. In this work, Gaussian Radial Basis Function (RBF) is applied as the kernel function, which is defined as:

$$K(x_i, x_j) = \exp(-\frac{\| x_i - x_j \|^2}{2\sigma^2}) \tag{II.3}$$

Our approach therefore consists in determining when a $2N \times 2N$ block has to be "Split" or "Not-Split". The input feature vector is denoted by $x_i$ and $y_i$ is the output label indicating CU splitting or not. The following equation is the discriminant function:

$$f(x) = w^T \phi(x) + b \tag{II.4}$$

where the normal vector is denoted by $w$. The function $\phi(x)$ maps feature vector $x$, and $b$ is the bias.

The current CU splitting decision should be determined in each CU depth. Therefore, an SVM classifier is used in each CU depth to get the best combination of CU, PU and TU via evaluating the RD cost of all possible modes. Figure II.2 demonstrates the process of the proposed CU decision method. To speed up the coding process, the SVM classifier is online learned to early terminate the CU partition process, as the SVM run-time classification can take a much time. In the proposed scheme, three models are

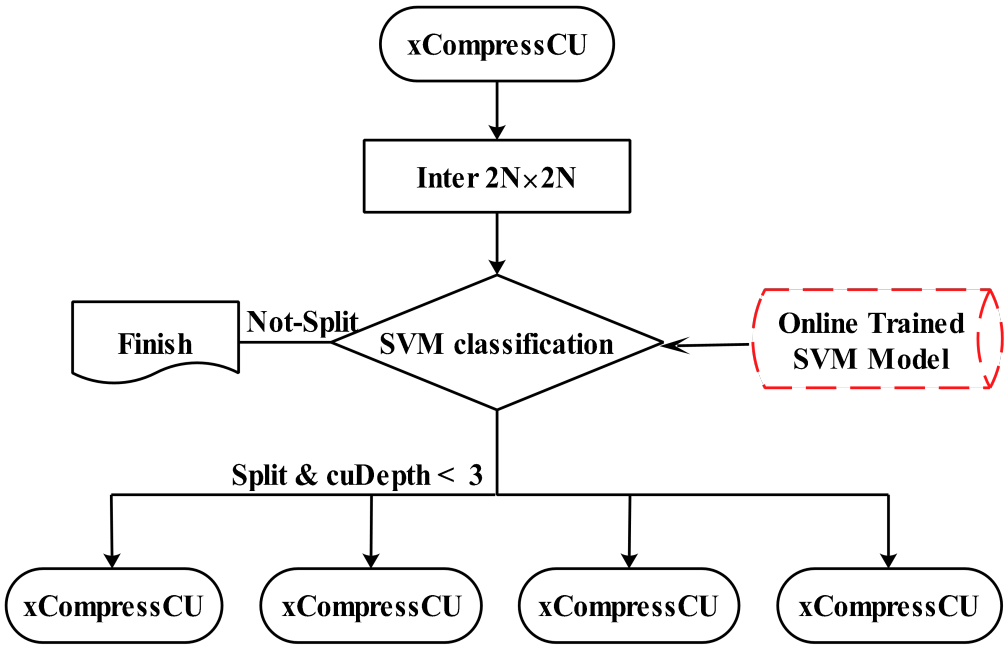



Figure. II.2: Flowchart of the Proposed Algorithm

trained online on the features selected for different CU sizes $64 \times 64$, $32 \times 32$, $16 \times 16$. The input features are summarized into two aspects; spatial and temporal adjacent encoded CUs, including the above-left, above, above-right, left and temporal co-located CUs corresponding to the current CU.

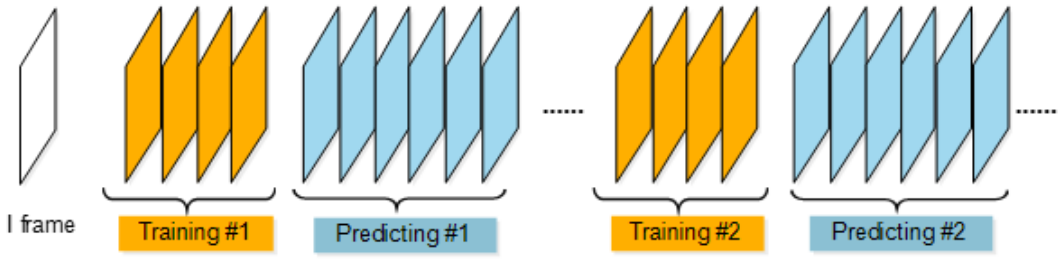



Figure. II.3: Online Training Mode

The first frames of Group of Picture (GoP) are encoded by HEVC test model when a video sequence is the encoding input. The selected features and the ground truths are extracted in the same time. After the online training and the classification process are achieved, the CU and the PU partitions of following inter frames will be predicted directly. Figure II.3 illustrates an example of an online training mode, where training frames are in yellow color and the predicting frames are in blue color. The advantage of

online training is that the properties of the video sequence of the training and testing are quite close which is better to improve the accuracy of the prediction.

### II.3.2 CU Partition based on Deep CNN

CNN is the most widely used DL model for video processing applications. According to the mechanism of the CU partition at inter-mode HEVC, a deep CNN structure is shown in Figure II.4. The residual CTU is fed into CNN architecture. Here, the residue is obtained by pre-coding the frame in HEVC. Our proposed architecture is composed of preprocessing layer, convolution layers, concatenated vector and fully connected layers. The preprocessing layer are residual CUs of CU, $CU_i$ or $CU_{i,j}$, corresponding to the three levels. Therefore, the residual block is subtracted by the mean intensity values to reduce the variation of the input CTU samples. Specifically, at the first level of CU partition, the mean value of CU is removed in accordance with the output of $\tilde{F}_1(CU)$. At the second level, four CUs $\{CU_i\}_{i=0}^{3}$ are subtracted by their corresponding mean values, matching the $2 \times 2$ output of $(\{\tilde{F}_2(CU_i)\}_{i=0}^{3})$. At the third level, $\{CU_{i,j}\}_{i,j=0}^{3}$ remove the mean values in each CU for the $4 \times 4$ output $(\{\tilde{F}_3(CU_{i,j})\}_{i,j=0}^{3})$.

After preprocessing layer, the three convolutional layers are used to extract features from data at all levels. The convolution layer is a mathematical operation that takes two inputs such as CU partition and filters. In each layer, the convolution kernels of all three levels have the same size. In our work, at the first convolutional layer, 16 kernels are used to extract the low features maps for the CU partition. At the second and third layers, feature maps are sequentially convoluted twice with $2 \times 2$ kernels (24 filters for the second layer and 32 filters for the third layer) to generate features at a higher level.The strides of all the above convolutions are equal to the widths of the corresponding kernels for non-overlap convolution.

The below design of the convolutional layer is in accordance with all possible non-overlap CUs at different sizes for CTU partition. At the end of the convolution, through the concatenation layer, the final feature maps are concatenated together and then flatten into a vector. In the following fully connected layers, features generated from the whole CTU are all considered to predict the CU partition at each single level.

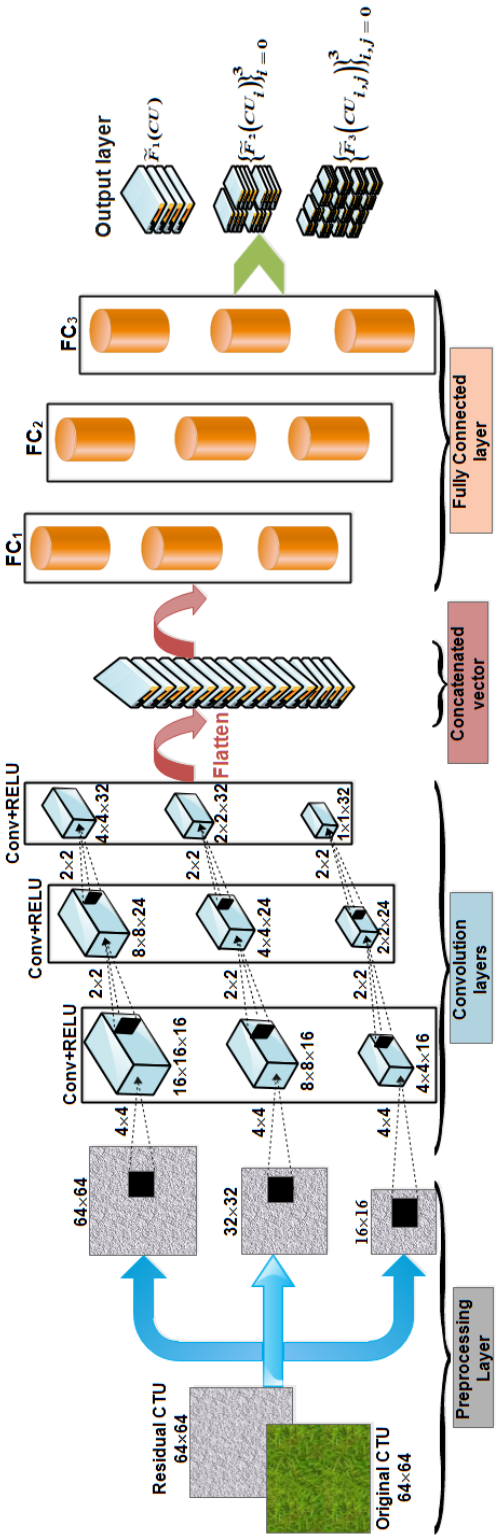



Figure. II.4: Deep CNN Architecture

Finally, the concatenated vector flows through three fully-connected layers as illustrated in Figure II.4, including two hidden layers and one output layer. The two hidden

fully connected layers successively generate feature vectors denoted by $(FC_l)_{l=1}^{3}$. The outputs of deep CNN are 1, 4, and 16 elements, such as the predicted binary labels $(\tilde{F}_1(CU))$ in $1 \times 1$, $(\{\tilde{F}_2(CU_i)\}_{i=0}^{3})$ in $2 \times 2$ and $(\{\tilde{F}_3(CU_{i,j})\}_{i,j=0}^{3})$ in $4 \times 4$ at three levels, respectively. In Deep CNN structure, the early termination may result in the calculation of the fully connected layers at levels 2 and 3 being skipped, thus saving computation time. Specifically, if CU is decided not to be split at level 1, the calculation of $(\{\tilde{F}_2(CU_i)\}_{i=0}^{3})$ is terminated early at level 2. If $\{CU_i\}_{i=0}^{3}$ are all not split, the $(\{\tilde{F}_3(CU_{i,j})\}_{i,j=0}^{3})$ at level 3 do not need to be computed for the early termination. The ReLU function is used to activate all convolutional layers and hidden fully connected layers, since this function has better convergence speed [GBB11]. Moreover, since all the labels for splitting or non-splitting are binary, all the output layers in three levels are activated with the sigmoid function.

### II.3.3 Training Phase

This section presents the training process for the proposed Deep CNN as shown in Figure II.5. We train our model in a supervised learning manner, in which the Deep CNN has been learned based on labeled data. In this context, we create the database for training the proposed model, which satisfy highly performances (high accuracy and low loss).

Afterwards, we establish a large-scale database for CU partition of the inter-mode HEVC (CPIH), in order to increase the prediction accuracy. However, to construct our CPIH database, we selected 114 raw video sequences with various resolutions from $352 \times 240$ to $2560 \times 1600$ [XDLW14, OSS+12, ML94, B+13]. These sequences are gathered into three sub-sets: 86 sequences for training, 10 sequences for validation, and 18 sequences for test. Table II.1 summarizes the chosen videos and the number of frames $(41,349)$ in our CPIH database.

First, we encoded the original database (114 video sequences) by original HEVC encoder common test condition at different Quantization Parameters (QP=22, 27, 32, 37) using Low Delay P configuration (using $encoder - lowdelay - P - main.cfg$ ) to obtain the residue and the ground truth CU depth. The ground truth CU depth files

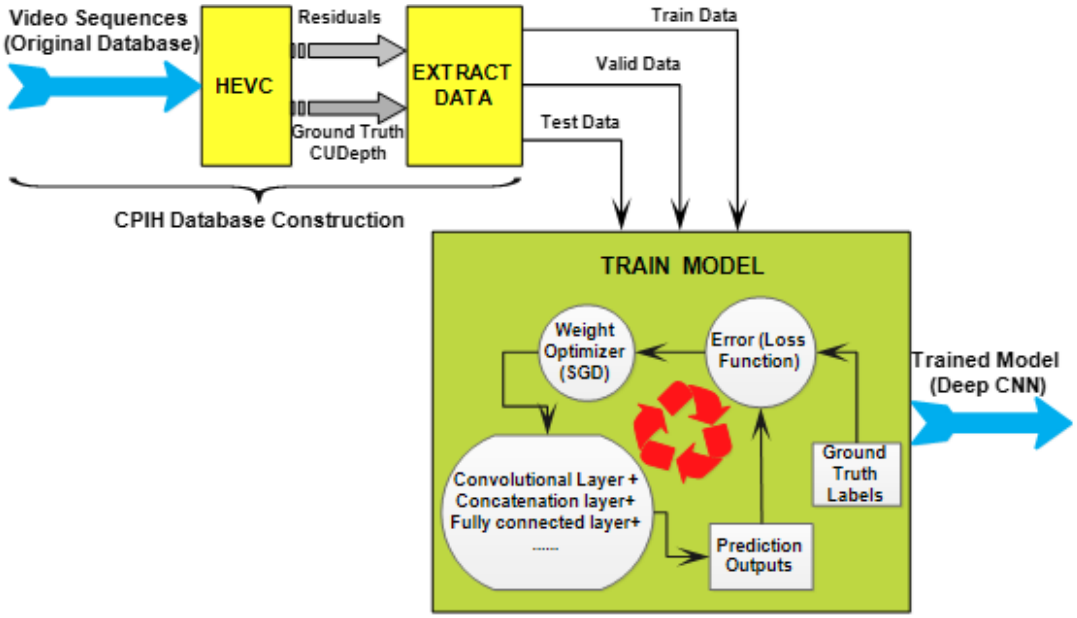



Figure. II.5: Training Process

contain the division probability of the entire sequences. Second, in order to construct a training sample, the training, validation, and test data are generated by implementing the 'EXTRACT DATA' program. The training data is used to train the model, while the validation data is used to determine when to stop the learning process. For the test data, 18 sequences of Classes $A - E$ from the JCT-VC are used to evaluate the performance of the proposed Deep CNN [B+13].

As shown in Figure II.5, the train model process summarizes the manner on how to train the model based on the CPIH database construction. The Stochastic Gradient Descent algorithm with momentum (SGD) is used as a powerful optimization algorithm to update the network weights at each iteration and minimize gradient error between the ground truth labels and the prediction outputs. This process will continue until the loss function reaches a minimum value. Furthermore, the Deep CNN model is trained at four QPs by using different sizes of CU, which varies from $16 \times 16$ to $64 \times 64$.

For learning our Deep CNN model, we assume that the cross entropy is applied as a loss function, which is defined in the equations II.5 and II.6:

$$L = \frac{1}{N} \sum_{n}^{N} L_n \tag{II.5}$$

Table II.1: Sequences in CPIH Database

| Resolutions | Train Data | | Valid Data | | Test Data | |
|---|---|---|---|---|---|---|
| | N. of video | N. of frame | N. of video | N. of frame | N. of video | N. of frame |
| 352x240(SIF) | 4 | 677 | - | - | - | - |
| 352x288(CIF) | 23 | 6,530 | 2 | 550 | - | - |
| 704x576(4CIF) | 4 | 2,280 | 1 | 600 | - | - |
| 720x486(NTSC) | 6 | 1,800 | 1 | 300 | - | - |
| 416X240(240p) | - | - | - | - | 4 | 1,900 |
| 832x480(480p) | - | - | - | - | 4 | 1,900 |
| 1280x720(720p) | 5 | 1,327 | 2 | 1,100 | 3 | 1,800 |
| 1920x1080(1080p) | 28 | 8,417 | 2 | 540 | 5 | 2,080 |
| 2048x1080(2k) | 16 | 8,048 | 2 | 1,200 | - | - |
| 2560x1600(WQXGA) | - | - | - | - | 2 | 300 |
| Total | **86** | **29,079** | **10** | **4,290** | **18** | **7,980** |

where $N$ is the number of training samples and $Ln$ represents the sum of the cross entropy:

$$L_n = Y(F_1^n(CU), \tilde{F}_1^n(CU)) + \sum_{i\in\{0,1,2,3\}} Y(F_2^n(CU_i), \tilde{F}_2^n(CU_i)) + \sum_{i,j\in\{0,1,2,3\}} Y(F_3^n(CU_{i,j}), \tilde{F}_3^n(CU_{i,j})) \quad \text{(II.6)}$$

where $Y$ denotes the cross entropy between the ground truth labels and the predicted labels. The labels predicted by our Deep CNN are represented by $(\{\tilde{F}_1(CU)\})$, $(\{\tilde{F}_2(CU_i)\}_{i=0}^3)$, and $(\{\tilde{F}_3(CU_{i,j})\}_{i,j=0}^3)\}_{n=0}^N$.

We use the Tensorflow-GPU deep learning framework to train our proposed Deep CNN on an NVIDIA GeForce GTX 480 GPU that can dramatically improve speed during training compared to the CPU. We adopt a batch mode learning method with a batch size of 64 where the momentum of the stochastic gradient descent algorithm optimization is set to 0.9. To train our Deep CNN, the base learning rate is set to decay exponentially to 0.01, changing every $1,000$ iterations. The total number of iterations

was $2,000,000$. Finally, the trained model can be used to predict the CU partition at HEVC inter-mode.

### II.3.4 CU Partition based on Deep CNN-LSTM

Considering the correlation of the HEVC CU partition of adjacent frames, we propose an LSTM network to learn the inter-frame temporal correlation for each video sequences [BKS+20]. The proposed framework that combines CNN-LSTM is presented in Figure II.6. The deep CNN is composed of three convolutional layers (Conv1, Conv2 and Conv3), a concatenated vector and a fully connected layers, as detailed in section II.3.2. As presented previously, the deep CNN parameters are learned based on the ground-truth and the residual CTU, then their extracted features $(FC_{1-l})_{l=1}^{3}$ are the input of the proposed LSTM network at frame $t$. These features $(FC_{1-l})_{l=1}^{3}$ are extracted at the first fully connected layer of the deep CNN. As shown in Figure II.6, the LSTM

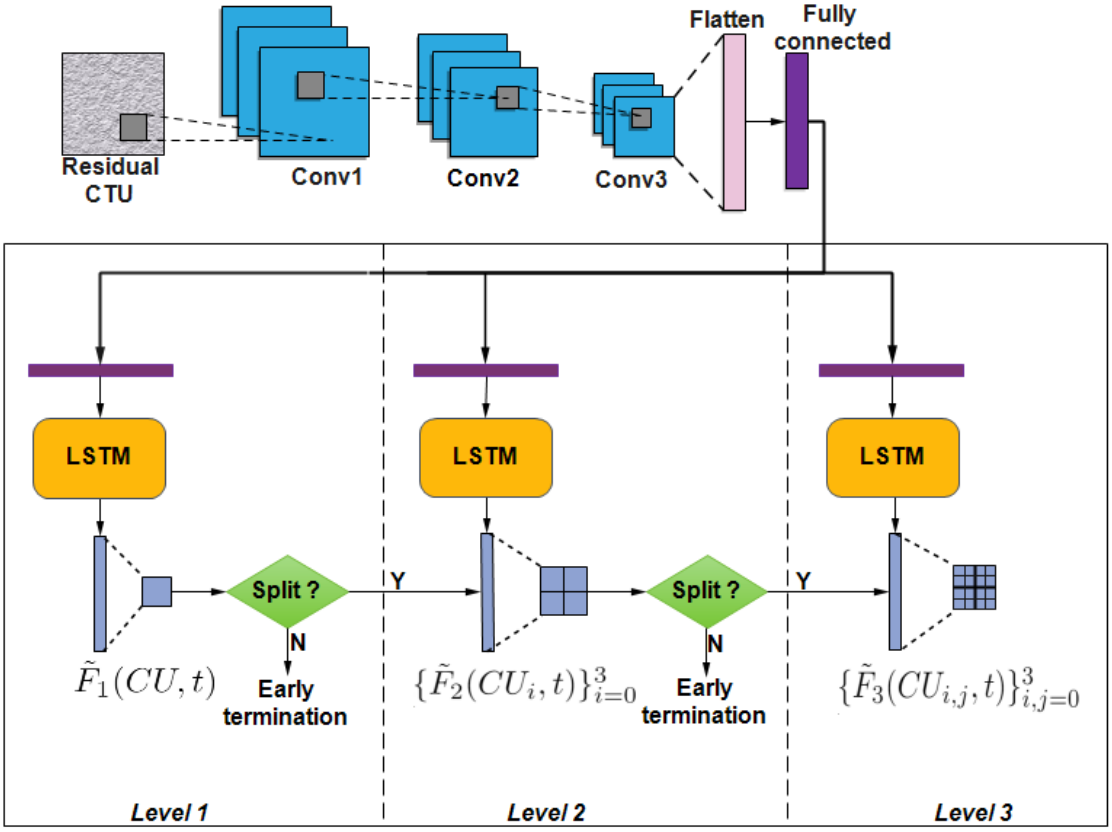



Figure. II.6: Proposed Framework

architecture is composed of three LSTM cells corresponding to three levels splitting of

each CU. Specifically, $\tilde{F}_1(CU,t)$ at level 1 indicating whether the $64\times64$ size CU will be split into $32\times32$ size sub-CUs or not. At level 2, $\{\tilde{F}_2(CU_i,t)\}_{i=0}^{3}$ and $\{\tilde{F}_3(CU_{i,j},t)\}_{i,j=0}^{3}$ designate respectively the CU partition labels from $32\times32$ to $16\times16$ and from $16\times16$ to $8\times8$.

At each level, two fully connected layers that contains a hidden layer and an output layer are followed the LSTM cells. In addition, the output features of the LSTM cells are denoted by $(FC'_l)_{l=1}^{3}$ at frame $t$. If the CU of the current level is predicted to be split, the LSTM classifier of the next level is activated to make decisions on the four subsequent CUs at the next level. Otherwise, the prediction on partitioning the current CTU is terminated, in order to save computational time. Finally, the CU splitting results of three levels are combined to represent the CTU partition in the form of 21-dimensional vector, which is composed of $\tilde{F}_1(CU,t)$, $\{\tilde{F}_2(CU_i,t)\}_{i=0}^{3}$ and $\{\tilde{F}_3(CU_{i,j},t)\}_{i,j=0}^{3}$, as shown in Figure II.6. The ReLU and the sigmoid activation functions are used to activate the hidden and the output layers, respectively [GBB11].

When predicting CTU partition, the long short-term dependency of CTU partition across frames can be taken into consideration in the LSTM network. As seen in Figure II.6, the temporal dependency is modeled by the LSTM cells, which are processed along with the encoded frames. Here, we take the LSTM cell of level $l$ at frame $t$ as an example to discuss the internal mechanism of the proposed LSTM. In fact, the LSTM cell consists of three gates, as shown in Figure II.7; the input gate $i_l(t)$, the forget gate $f_l(t)$, and the output gate $o_l(t)$.

At level $l$, $FC_{1-l}(t)$ represents the deep CNN input features at frame $t$, and $FC'_l(t-1)$ is the output features of the LSTM model of frame $t-1$. These three gates are presented in the following equations:

$$\begin{aligned} i_l(t) &= \sigma(W_i\cdot[FC_{1-l}(t),FC'_l(t-1)]+b_i)\\ o_l(t) &= \sigma(W_o\cdot[FC_{1-l}(t),FC'_l(t-1)]+b_o)\\ f_l(t) &= \sigma(W_f\cdot[FC_{1-l}(t),FC'_l(t-1)]+b_f) \end{aligned} \tag{II.7}$$

where the sigmoid function is denoted by $\sigma(\cdot)$. The weights are defined by $\{w_i,w_o,w_f\}$

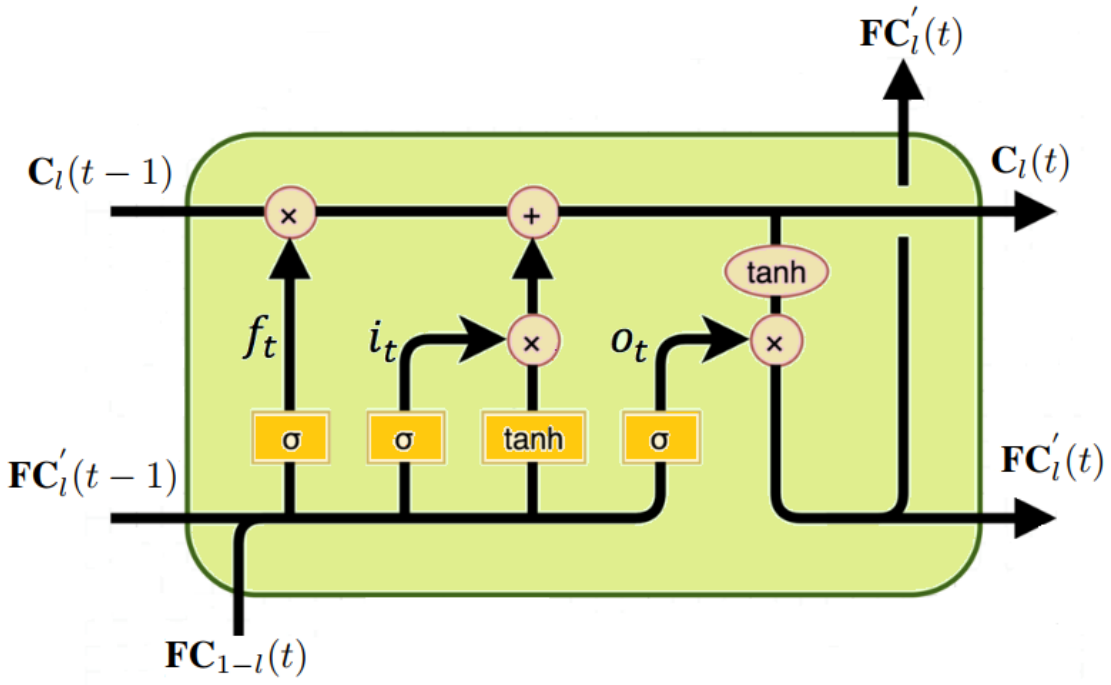



Figure. II.7: LSTM Cell

and the biases are denoted by $\{b_i, b_o, b_f\}$ for the three gates. At frame $t$, the state $c_l(t)$ of the LSTM cell can be updated by:

$$c_l(t) = i_l(t) \odot tanh(W_c \odot [FC_{1-l}(t), FC_l'(t-1)] + b_c) + f_l(t) \odot c_l(t-1) \tag{II.8}$$

where $\odot$ signifies the element-wise multiplication. The output of the LSTM cell $FC_l'(t)$ can be determined as follows:

$$FC_l'(t) = o_l(t) \odot c_l(t) \tag{II.9}$$

### II.3.5 Training Phase

In the training phase, the LSTM model was trained from the training set of the inter database given in Table II.1, which minimizes the loss function between the ground truth and the prediction of CTU partition. Figure II.8 shows the learning process. The SGD algorithm is considered to be the powerful optimization algorithm to learn the network structure through their feed-forward and backward sub-process. Where,

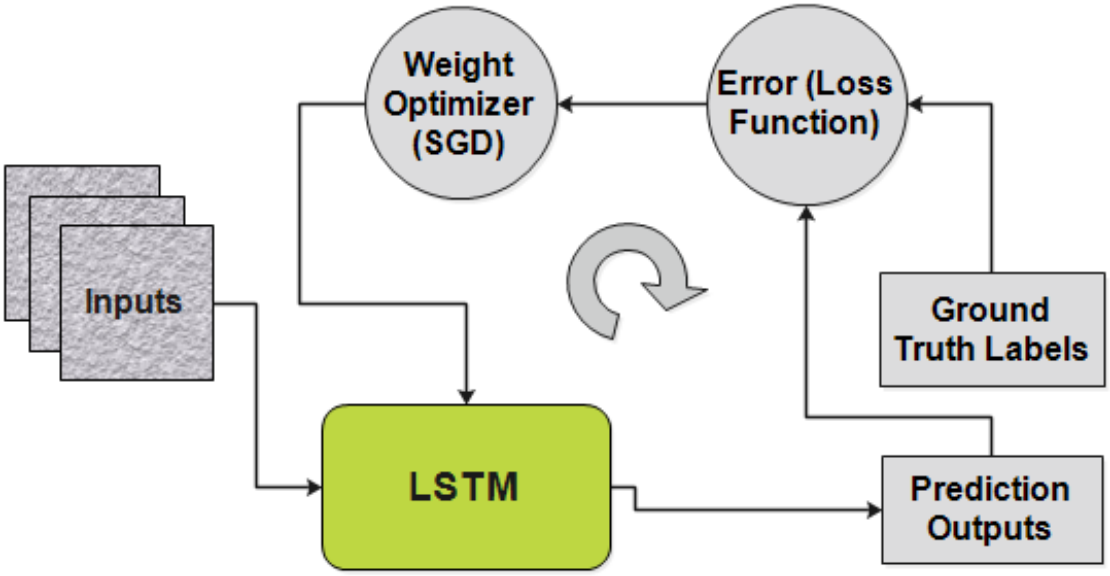



Figure. II.8: Learning Process

the cross entropy is selected to be the cost function of the gradient error calculator designated by $Y(\cdot\,,\cdot\,)$ in II.10. At frame $t$, the loss function $Ln(t)$ for the $n-th$ sample CU is written as follows:

$$L_n(t) = Y(F_1^n(CU,t), \tilde{F}_1^n(CU,t)) + \sum_{i\in\{0,1,2,3\}} Y(F_2^n(CU_i,t), \tilde{F}_2^n(CU_i,t)) + \sum_{i,j\in\{0,1,2,3\}} Y(F_3^n(CU_{i,j},t), \tilde{F}_3^n(CU_{i,j},t)) \tag{II.10}$$

However, over $N$ training samples alongside the $T-frames$, the LSTM network can be trained by optimizing the cost function, as defined in equation II.11.

$$L = \frac{1}{NT}\sum_{n}^{N}\sum_{t}^{T} L_n(t) \tag{II.11}$$

The training parameters were defined as follows; batch size, learning rate, and LSTM length ($T$) were set to 64, 0.001, and 20, respectively. Finally, the trained model was saved to be used after (in the framework), which aims to predict the inter-coding CU partition. For the test, the LSTM model works in stages, that is, when the prediction of the CU partition at frame $t-1$ is complete, the state and the output of the frame $t$ are computed. To further enhance the $RD$ performance and reduce the inter-coding

computational complexity, the bi-threshold decision scheme was adopted at three levels. Note that the upper and the lower thresholds at level $l$ are represented by $(\gamma_l)_{l=1}^{3}$ and $(\bar{\gamma}_l)_{l=1}^{3}$. At three levels, the LSTM network provides the predicted CU partition probability $P_l(CU)$. Consequently, the CU decides to be split only when $P_l(CU) > \gamma_l$. If $P_l(CU) < \bar{\gamma}_l$, the CU is not split. The interval $[\bar{\gamma}_l, \gamma_l]$ represents the uncertain zone, in which the possible splitting patterns of the current CU need to be traversed by HEVC for the *RDO* search. In this way, the HEVC complexity is reduced considerably by skipping the most redundant verification of the *RD* cost.

## II.4 Experimental Results

This section introduces the evaluation performance of the proposed HEVC method. Specifically, we first present the experimental settings. Then, the performance metrics are provided. Finally, the evaluation performance of the proposed machine learning approaches are discussed and compared to other related algorithms.

### II.4.1 Experimental Settings

In this section, we present the obtained results to validate the coding efficiency of the proposed deep learning framework. Our experiments were performed in the HM16.5 reference test model [Mod] using the Low Delay P (LDP) configuration. The QP values tested were 22, 27, 32 and 37 for encoding process. All simulations were tested on 18 JCT-VC videos from class A (2650×1600) to class E (1280×720) [B+13], as illustrated in Table II.2. The frames number used for each video sequences is 100. All implementations were executed on windows 10 OS platform with Intel ®core TM i7-3770 @ 3.4 GHz CPU and 16 GB RAM. To accelerate the speed of the network model training phase, we also used the NVIDIA GeForce GTX 480 GPU, but it was not used in the HEVC complexity reduction test. In the experiments, the Tensorflow-GPU deep learning framework was used.

Table II.2: Test Sequences

| Class | Resolutions | Sequences |
|---|---|---|
| A | 2560 × 1600 | PeopleOnStreet, Traffic |
| B | 1920 × 1080 | Kimono, ParkScene, Cactus, BQTerrace, BasketballDrive |
| C | 832 × 480 | BasketballDrill, BQMall, PartyScene, RaceHorses |
| D | 416 × 240 | BasketballPass, BQSquare, BlowingBubbles, RaceHorses |
| E | 1280 × 720 | FourPeople, Johnny, KristenAndSara |

### II.4.2 Performance Metrics

The RD performance analysis is performed based on the Bjontegaard Delta bitrate (BD-BR) and the Bjontegaard Delta Peak Signal-to Noise Ratio (BD-PSNR) [Bjo01]. The BD-BR represents the average bitrate savings that calculated between two RD curves for the same video quality, where negative BD-BR values indicate actual bitrate savings and positive values indicate how much the bitrate is increased. BD-PSNR is the overall PSNR difference of RD curves with the same bitrate in decibel. Not forgetting that the coding time is modeled as the critical metric for the validation performance of the HEVC at inter-mode, as shown in the following equation:

$$\triangle\ T = \frac{T_{Proposed} - T_{Original}}{T_{Original}} \times 100\ (\%) \tag{II.12}$$

where $T_{Proposed}$ and $T_{Original}$ are the coding time of the proposed approach and the original HEVC algorithm, respectively.

### II.4.3 Performance Evaluation with Online SVM and Deep CNN

Table II.3 gives a comparison of our two proposed methods, Deep CNN and Online SVM, in terms of complexity reduction and RD performance using LDP configuration.

The experimental results show that our Deep CNN model achieves a significant complexity reduction of around 53.99% with 1.80% BD-BR compared to the online SVM at LDP configuration. On the other hand, our online SVM demonstrates significant coding losses in BD-BR of 3.55% and an average decrease of 52.28% in time reduction.

Table II.3: Performances Comparison between Deep CNN and Online SVM

| Sequences | Online SVM | | | Deep CNN | | |
|---|---|---|---|---|---|---|
| | BD-BR (%) | BD-PSNR (dB) | $\triangle T$ (%) | BD-BR (%) | BD-PSNR (dB) | $\triangle T$ (%) |
| PeopleOnStreet | 3.65 | -0.154 | -56.56 | 1.20 | -0.051 | -50.67 |
| Traffic | 3.94 | -0.106 | -58.07 | 1.49 | -0.041 | -57.90 |
| Kimono | 1.12 | -0.036 | -44.18 | 1.38 | -0.044 | -43.26 |
| ParkScene | 1.67 | -0.048 | -52.60 | 1.43 | -0.041 | -64.14 |
| Cactus | 1.68 | -0.034 | -41.38 | 2.44 | -0.047 | -52.57 |
| BQTerrace | 1.67 | -0.029 | -41.45 | 2.22 | -0.034 | -58.43 |
| BasketballDrive | 1.72 | -0.039 | -51.17 | 2.28 | -0.051 | -51.30 |
| BasketballDrill | 2.72 | -0.098 | -55.87 | 1.43 | -0.052 | -53.54 |
| BQMall | 5.11 | -0.192 | -55.96 | 2.24 | -0.085 | -52.25 |
| PartyScene | 4.38 | -0.169 | -52.93 | 1.48 | -0.057 | -51.54 |
| RaceHorses | 4.67 | -0.173 | -51.25 | 1.41 | -0.053 | -42.22 |
| BasketballPass | 4.94 | -0.217 | -55.49 | 1.85 | -0.083 | -52.42 |
| BQSquare | 6.63 | -0.227 | -55.92 | 2.09 | -0.073 | -52.79 |
| BlowingBubbles | 4.56 | -0.157 | -51.87 | 1.71 | -0.061 | -46.55 |
| RaceHorses | 7.48 | -0.315 | -50.81 | 1.32 | -0.058 | -38.01 |
| FourPeople | 1.78 | -0.054 | -51.90 | 1.06 | -0.029 | -67.54 |
| Johnny | 3.60 | -0.076 | -60.12 | 3.99 | -0.083 | -69.66 |
| KristenAndSara | 2.51 | -0.070 | -53.57 | 1.31 | -0.082 | -67.20 |
| Overall | **3.55** | **-0.121** | **-52.28** | **1.80** | **-0.057** | **-53.99** |

As it can be seen, our proposed Deep CNN obtains significantly best results in terms of execution time for class $E$ sequences, this is caused by the low motion activities displayed in these sequences, which leads to larger partitions. For the same reason, it is possible to observe a slightly higher encoding time for high-resolution sequences compared to lower resolution ones.

From the overall performance evaluation, we can find that the proposed method Deep CNN outperforms the online SVM in terms of both complexity reduction and RD

performance of inter-mode HEVC, as seen in Table II.3. This implies that the proposed Deep CNN is robust in reducing complexity of inter-mode HEVC when compared to the online SVM. This is due to the fact that the CNN works well with visual images recognition while SVM is used widely in classification problems. Additionally, it is difficult to parallelize SVM but the CNN architecture inherently supports parallelization.

### II.4.4 Performance Evaluation with Deep CNN and CNN-LSTM

Table II.4: Performances Comparison between Deep CNN and CNN-LSTM

| Sequences | Deep CNN | | | Proposed CNN-LSTM | | |
|---|---|---|---|---|---|---|
| | BD-BR (%) | BD-PSNR (dB) | $\triangle T$ (%) | BD-BR (%) | BD-PSNR (dB) | $\triangle T$ (%) |
| PeopleOnStreet | 1.20 | -0.051 | -50.67 | 1.70 | -0.017 | -48.88 |
| Traffic | 1.49 | -0.041 | -57.90 | 1.53 | -0.059 | -66.38 |
| Kimono | 1.38 | -0.044 | -43.26 | 1.65 | -0.052 | -47.77 |
| ParkScene | 1.43 | -0.041 | -64.14 | 2.79 | -0.081 | -70.82 |
| Cactus | 2.44 | -0.047 | -52.57 | 1.73 | -0.033 | -53.85 |
| BQTerrace | 2.22 | -0.034 | -58.43 | 1.75 | -0.030 | -65.62 |
| BasketballDrive | 2.28 | -0.051 | -51.30 | 2.02 | -0.045 | -52.77 |
| BasketballDrill | 1.43 | -0.052 | -53.54 | 1.67 | -0.061 | -48.23 |
| BQMall | 2.24 | -0.085 | -52.25 | 1.38 | -0.090 | -48.07 |
| PartyScene | 1.48 | -0.057 | -51.54 | 0.96 | -0.038 | -59.30 |
| RaceHorses | 1.41 | -0.053 | -42.22 | 1.47 | -0.055 | -54.32 |
| BasketballPass | 1.85 | -0.083 | -52.42 | 1.26 | -0.056 | -56.67 |
| BQSquare | 2.09 | -0.073 | -52.79 | 1.27 | -0.046 | -60.79 |
| BlowingBubbles | 1.71 | -0.061 | -46.55 | 0.97 | -0.034 | -50.14 |
| RaceHorses | 1.32 | -0.058 | -38.01 | 1.60 | -0.018 | -50.33 |
| FourPeople | 1.06 | -0.029 | -67.54 | 2.71 | -0.071 | -72.42 |
| Johnny | 3.99 | -0.083 | -69.66 | 2.46 | -0.083 | -73.59 |
| KristenAndSara | 1.31 | -0.082 | -67.20 | 2.93 | -0.094 | -74.91 |
| Overall | **1.80** | **-0.057** | **-53.99** | **1.78** | **-0.053** | **-58.60** |

For further performance evaluation of the proposed scheme, Table II.4 shows the coding performance between the proposed CNN-LSTM framework and the deep CNN [BKSA20a]. The proposed scheme CNN-LSTM is better than the deep CNN in terms of computational-complexity and RD performance. Specifically, the execution time of this method is 58.60% on average, exceeding the 53.99% obtained with deep CNN only.

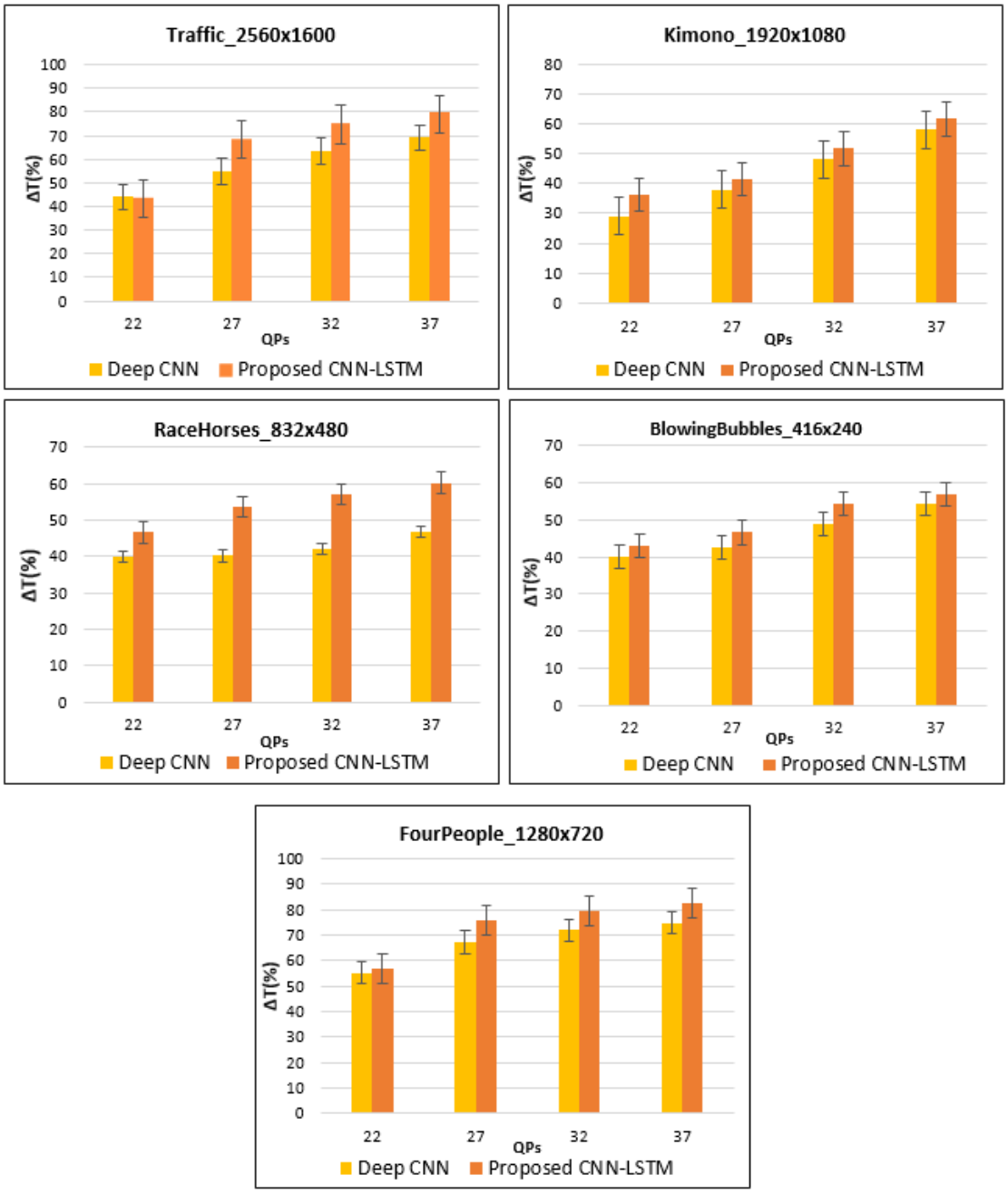



Figure. II.9: Encoding Time of the Proposed CNN-LSTM and Deep CNN

On the other hand, the proposed approach can reduce the BD-PSNR performance by -0.053dB, which is better than -0.057dB achieved by the proposed Deep CNN. Furthermore, our proposed approach has an average BD-BR performance of 1.78%, better than that of [BKSA20a]; 1.80%. In our experiments, we note that the proposed deep learning CNN-LSTM achieves high HEVC complexity reduction at inter-coding, because it is capable to predict all the CU splitting of an entire CTU at the same time. The proposed algorithm also performs well in terms of BD-PSNR performance, due to the high accuracy of the predicted CU partition. Consequently, the learning scheme based on CNN-LSTM achieves a good compromise between RD performance and coding complexity in order to predict inter-mode CU partition of HEVC. This is mainly due to the LSTM ability to resolve the temporal correlation through adjacent frames.

For more evaluation, the reducing complexity of CNN-LSTM versus deep CNN under all video sequences $(A - E)$ at LDP configuration is proved in Figure II.9. As shown in this Figure, the proposed approach allows higher encoding time when the $QP$ value increases from 22 to 37. Overall, the proposed deep learning approach outperforms best in terms of time saving than the deep CNN. Consequently, the proposed scheme is better for reducing the HEVC complexity of inter-coding and for finding an optimal $CU$ partition, compared to traditional RDO research.

### II.4.5 Comparative Study

To evaluate the encoding performance of the proposed learning approach, our experimental results are compared to other state of the art methods. Table II.5 summarizes the proposed scheme performances compared to the works-based learning process. From this table, the proposed scheme outperforms others schemes in terms of complexity-RD performance. In [LZZ+19], Li et al. proposed a CU early termination-based reinforcement learning to reduce the computational-complexity for HEVC. In [TTAA19], Tahir et al. developed a fast method for reducing HEVC encoding time based on random forest classifier. Xu et al. [XLW+18] proposed a fast CU partition algorithm for HEVC, including inter and intra prediction based on deep learning approach for reducing HEVC complexity. Specifically, the proposed approach achieves a maximum execution time

Table II.5: Comparative Study

| Sequence | [XLW+18] | | | [LZZ+19] | | | [TTAA19] | | | Proposed CNN-LSTM | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BD-BR (%) | BD-PSNR (dB) | $\Delta T$ (%) | BD-BR (%) | BD-PSNR (dB) | $\Delta T$ (%) | BD-BR (%) | BD-PSNR (dB) | $\Delta T$ (%) | BD-BR (%) | BD-PSNR (dB) | $\Delta T$ (%) |
| PeopleOnStreet | 1.05 | -0.045 | -47.50 | 5.45 | -0.250 | 29.84 | 1.85 | -0.085 | 27.22 | 1.70 | -0.017 | -48.88 |
| Traffic | 1.99 | -0.052 | -60.60 | – | – | – | 3.20 | -0.102 | 59.66 | 1.53 | -0.059 | -66.38 |
| Kimono | 1.49 | -0.048 | -56.03 | 0.35 | -0.010 | 34.74 | 2.43 | -0.079 | 50.49 | 1.65 | -0.052 | -47.77 |
| ParkScene | 1.47 | -0.042 | -58.72 | 2.84 | -0.090 | 46.42 | – | – | – | 2.79 | -0.081 | -70.82 |
| Cactus | 2.07 | -0.043 | -56.87 | 2.79 | -0.065 | 43.22 | 2.46 | -0.060 | 53.06 | 1.73 | -0.033 | -53.85 |
| BQTerrace | 1.09 | -0.017 | -60.01 | 2.15 | -0.038 | 38.70 | 1.74 | -0.032 | 51.89 | 1.75 | -0.030 | -65.62 |
| BasketballDrive | 2.26 | -0.052 | -55.84 | 2.06 | -0.046 | 39.45 | 1.93 | -0.046 | 49.16 | 2.02 | -0.045 | -52.77 |
| BasketballDrill | 1.95 | -0.072 | -55.19 | 3.90 | -0.148 | 32.12 | 2.24 | -0.089 | 46.55 | 1.67 | -0.061 | -48.23 |
| BQMall | 1.91 | -0.071 | -50.74 | 5.56 | -0.227 | 37.11 | 1.86 | -0.071 | 43.32 | 1.38 | -0.090 | -48.07 |
| PartyScene | 1.01 | -0.039 | -46.83 | 4.74 | -0.210 | 31.58 | 2.58 | -0.108 | 30.33 | 0.96 | -0.038 | -59.30 |
| RaceHorses | 0.87 | -0.032 | -46.22 | 2.10 | -0.180 | 26.03 | 1.25 | -0.048 | 27.08 | 1.47 | -0.055 | -54.32 |
| BasketballPass | 1.45 | -0.066 | -49.04 | – | – | – | 3.12 | -0.147 | 38.24 | 1.26 | -0.056 | -56.67 |
| BQSquare | 0.77 | -0.028 | -46.91 | 3.38 | -0.145 | 35.72 | 3.02 | -0.117 | 34.30 | 1.27 | -0.046 | -60.79 |
| BlowingBubbles | 1.29 | -0.044 | -45.62 | 3.41 | -0.136 | 24.73 | 3.99 | -0.154 | 39.87 | 0.97 | -0.034 | -50.14 |
| RaceHorses | 1.11 | -0.047 | -41.86 | – | – | – | – | – | – | 1.60 | -0.018 | -50.33 |
| FourPeople | 1.83 | -0.052 | -64.37 | 1.66 | -0.058 | 65.28 | 1.83 | -0.063 | 78.98 | 2.71 | -0.071 | -72.42 |
| Johnny | 1.69 | -0.038 | -66.49 | 0.90 | -0.020 | 64.05 | 5.45 | -0.139 | 79.31 | 2.46 | -0.083 | -73.59 |
| KristenAndSara | 1.55 | -0.045 | -67.23 | 1.58 | -0.050 | 64.67 | 3.30 | -0.108 | 77.58 | 2.93 | -0.094 | -74.91 |
| Overall | **1.49** | **-0.046** | **-54.2** | **2.56** | **-0.099** | **43.33** | **2.97** | **-0.107** | **54.57** | **1.78** | **-0.053** | **-58.60** |

of 75% and 58.60% on average and gives an increase in BD-BR of 1.78% with a little reduction in BD-PSNR of -0.053 dB.

In fact, the proposed approach achieves a higher computational-complexity reduction for video sequences with low motion activities and homogeneous regions, where the blocks CU partition is larger and the percentage of splitting cases is lower, such as "KristenAndSara" video sequence. Similarly, the existing methods prove a high encoding time for class E video sequences. For example, [LZZ+19] achieves 64% encoding complexity and 1.58% BD-BR increase for sequence "KristenAndSara", as shown in Table II.5. For the same sequence, [TTAA19] gives 77% time saving with an increase in the BD-BR of 3.30% on average of four QPs. In addition, the work proposed in [XLW+18] achieves 67.23% encoding time with 1.55% BD-BR on average. With regard to the ultra-high definition sequences like "PeopleOnStreet", the computational complexity reduction of our proposed approach is slightly lower, since these sequences have high motion and camera movement, which are encoded in a small CU partition. Hence, the proposed scheme performs better in terms of both RD performance and complexity reduction of HEVC as compared to the previous works. Overall, all approaches are better adapted to low-motion video content.

On the other hand, on average, 43% time saving is reduced by [LZZ+19] with an increase in BD-BR of 2.56% and a decrease in BD-PSNR of -0.099dB. The proposed method presented in [TTAA19] allows 54.57% encoding time while the BD-BR increases by 2.97% and the BD-PSNR degradation reaches -0.107dB. Regarding the work presented in [XLW+18], the proposed method surpasses ours in terms of BD-BR and BD-PSNR, while our proposed approach allows a significant coding time saving of 58.60% compared to this work. When comparing our work to the state of the art schemes [LZZ+19], [TTAA19], and [XLW+18] describing earlier, we can conclude that the proposed CNN-LSTM-based learning method proves the best coding efficiency of HEVC at inter-mode in order to predict the CU partition.

## II.5 Conclusion

In this chapter, we proposed a fast CU partition based on machine learning approaches to reduce the HEVC complexity of inter-mode. An online SVM-based fast CU partition method was proposed to reduce the encoding complexity of HEVC. Then, to predict the CU partition of HEVC, a Deep CNN was proposed, which reduces the HEVC complexity at inter-mode. Unfortunately, these two machine learning algorithms do not explore the correlation of the CU partition across neighboring frames. Therefore, a deep learning approach was proposed to predict the CU partition at inter-mode, which combines the CNN and the LSTM structures. Simulations results show and prove the efficiency of the proposed framework in saving a significant encoding complexity, compared to other previous approaches-based machine learning tools.

In the next chapter, we provide a deep learning technique to improve the visual quality of reconstructed video for the VVC standard in order to meet the demands required by the users.

# Chapter III

# Deep Learning based Video Quality Enhancement for the New Versatile Video Coding

**Summary**

## III.1 Introduction

Multimedia-IoT (M-IoT) is an emerging type of Internet of Things (IoT) relaying multimedia data (image, video, audio and speech, etc...) [NQA+20]. The rapid growth of M-IoT devices enables the creation of a massive volume of multimedia data with different characteristics and requirements [MBB+20]. With the development of Artificial Intelligence (AI), AI-based M-IoT systems have been recently designed and deployed for various video-based services for contemporary daily life, like video surveillance with HD and UHD and mobile multimedia streaming. These new services need higher video quality in order to meet the Quality of Experience (QoE) required by users [ARM18].

This chapter proposes a deep CNN-based in-loop filtering approach, denoted as the Wide-activated Squeeze-and-Excitation Deep Convolutional Neural Network (WSE-DCNN). The proposed approach provides new powerful in-loop filtering without exploiting traditional ones for the VVC standard. Indeed, the main goal is to effectively remove compression artifacts and enhance the compressed video quality. The proposed method improves the QoE of end-users.

The remainder of this chapter is organized as follows. Section III.2 presents the background. Section III.3 introduces the proposed M-IoT scenario. Then, the proposed deep CNN-based in-loop filtering in VVC standard is defined in section III.4. Next, we evaluate the proposed method in section III.5. Finally, section III.6 refers to the conclusion of this chapter.

## III.2 Background

The growing multimedia portfolio, including Big Data processing, Cloud Computing and the IoT [MBB+20], has a direct impact on our lifestyle. M-IoT is considered as a major network technology enabling the interconnection and interaction between humans, health-centers, industries, and objects like cameras, transport, and sensors [CJH16]. In addition, M-IoT systems combine the networking technologies for computer vision, image processing, and connectivity. Yet, they can be used in driving assistance, surveil-

lance such as crime and fire detection, and remote sensing such as high-speed object-tracking [NQA+20]. Real-world multimedia applications including, smart industry 4.0 and agriculture 4.0, smart traffic monitoring, smart cities, smart homes, smart health, and smart environment with intelligent surveillance systems [NQA+20], are illustrated in Figure III.1. However, several issues, such as interoperability, security, data size, reliability, storage and computational capacity need to be well resolved to process multimedia data [ZKH+19].

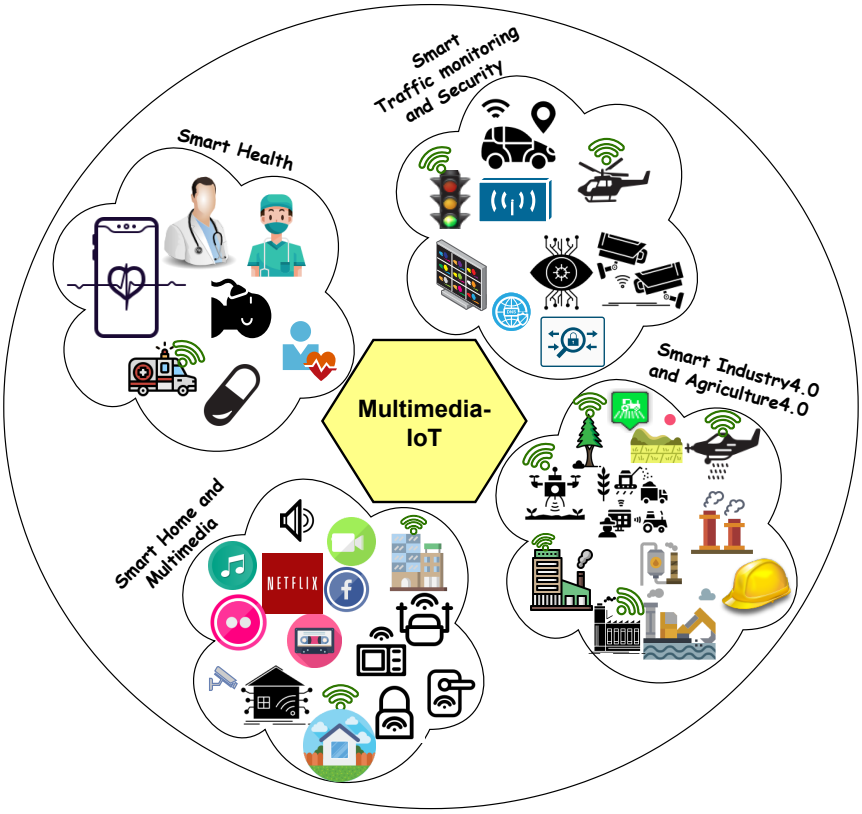



Figure. III.1: M-IoT Use Cases

Compared to traditional IoT, M-IoT has powerful functionality such as fast and reliable data delivery. Therefore, it imposes high Quality of Service (QoS) requirements and demands efficient network architecture. In this context, QoE represents the perspective of the end-user's QoS. QoE can be depicted as objective or subjective. The users Objective QoE is difficult to measure and varies considerably according to the requirements. However, service providers are interested by the subjective QoE to assess the network Mean Opinion Score (MOS) [ARM18].

As emerging technologies have rapidly evolved, multimedia services and video applications have grown tremendously. Higher image resolution (4K, 8K), especially for video games and monitoring tasks, are needed to satisfy the QoS specifications of end-users. In traditional multimedia encoding methods, data are compressed only in one time and decoded in every playing time. M-IoT devices are more concerned with uploading the data in uplink transmission. The latter poses challenges on computationally powered constrained M-IoT devices. Therefore, VVC which is a powerful multimedia encoding/decoding technique, has been widely adopted. VVC [BCL19] is the new generation video coding developed in July 2020, by JVET, as a successor of the High Efficiency Video Coding (HEVC) [Wie15], more details are given in chapter I. As the next standard for sophisticated video coding technology, VVC allows up to $30\% - 50\%$ for bitrate savings while maintaining the same quality as HEVC [BKS21]. Although VVC aims to keep the compressed video in high quality with additional encoding features, it still suffering from compression artifacts and can lead to a decrease in the QoE. Hence, the QoE of VVC compressed video needs to be improved.

On the other hand, VVC adopts the block coding and the quantization structure, many different forms of distortion still exist, such as blocking artifacts, blurring and ringing artifacts, as illustrated in chapter I. The blocking artifacts affect the visual quality. While these distortions are permanent and cannot be removed entirely, special filters can be used to reduce them. Hence, loop filters play an important role in reducing artifacts problems and in improving video and image qualities.

Unlike HEVC, in-loop filtering techniques (ie. De-Blocking Filter (DBF), Sample Adaptive Offset (SAO), and Adaptive Loop Filter (ALF)) are applied in the VVC standard. These filters remove the video compression artifacts and enhance the visual quality of the reconstructed video. Indeed, the DBF purpose is designed with the use of discontinuity-based smoothing filters to minimize artifacts along block boundaries [NBF+12]. In order to reduce ringing artifacts through compensation, SAO is used as a filter added after DBF, which applies shifts to samples based on the encoder lookup table and analyzes signal amplitudes using a histogram [FAA+12]. ALF is the latest loop filtering considered as a new feature in VVC. ALF reduces distortions between reconstructed

and original images [TCY+13]. Although these conventional in-loop filtering can relieve specific artifacts, it is difficult to overcome the complex distortion introduced by video compression. To meet this challenge, powerful deep learning approaches have been proposed for video quality enhancement [JLLL19, CCWL20]. Among these techniques, CNN is the most robust and efficient processing method for recognizing and analyzing images and videos[KSH12, MC19, BAH18]. These approaches using CNN-based in-loop filtering and post-processing are proposed to reduce visual artifacts and to achieve high performance. Indeed, regarding the challenges of 5G and M-IoT technologies, such as low latency cost, high speed rate, and high video and image resolution quality, the VVC original loop filtering became insufficient to meet resolution requirement of M-IoT-based applications. To address these critical issues, QoE must be considered and improved in order to ensure QoS for end-users [NQA+20].

## III.3 Proposed M-IoT Scenario-based Architecture for Multimedia Data

Without loss of generality, we propose an M-IoT scenario in the context of smart city, as illustrated in Figure III.2. It consists of a set of M-IoT devices, like cameras and multimedia devices, that are capable to acquire multimedia contents from the real and physical word. After that, the sensed multimedia data is sent to the centralized cloud computing for processing, via the network layer, using different transmission technologies such as LP-WAN [MBA+20]. M-IoT devices are more concerned with uploading data in uplink transmission, which poses challenges for constrained computational M-IoT devices. Several metrics can be considered, in this step, like the delay, jitter, and packet loss rate. Our interest is shifting to central computing, such as M-IoT data compression and encoding/decoding.

After M-IoT data acquisition step, data is compressed once and decoded whenever played. Traditionally, video encoding/compression is achieved by utilizing spatial and temporal redundancies. In this context, video quality is considered as the potential challenge in the VVC standard, that must be improved in this phase, especially when the

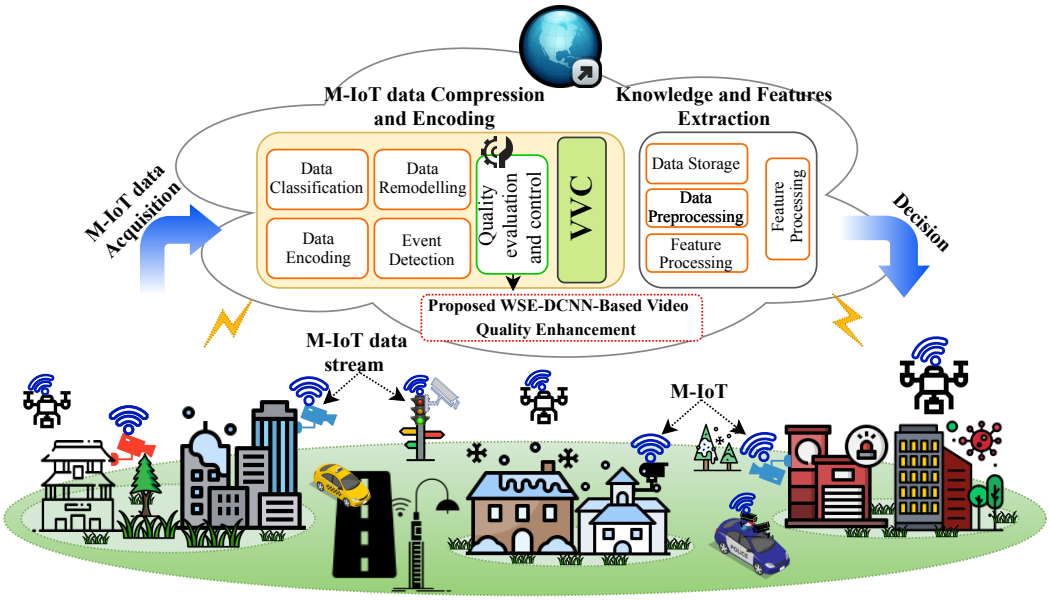



Figure. III.2: M-IoT Scenario-based Centralized Video Quality Enhancement

huge collected multimedia data are structured/unstructured, with high velocity, and with different resolutions. Therefore, the QoE depends on the video quality performances, is denoted as the metric that should be maximized. Based on the modeling, given in [PV17], the QoE is modeled considering the bitrate (BR) as formulated in (III.1).

$$QoE_{BR} = a \times log(BR) + b \tag{III.1}$$

where $a$ and $b$ denote coefficients determined during the experiment. However, this parameter, just like the PSNR metric, will also be used for the proposed WSE-DCNN-based in-loop filtering to evaluate video quality.

## III.4 Proposed Method

This section introduces the proposed method and describes the WSE-DCNN architecture and how this is integrated into VVC standard to replace the traditional loop filtering technique in order to improve the video quality.

### III.4.1 Proposed WSE-DCNN-based In-Loop Filtering For VVC

In our study, the proposed WSE-DCNN model replaces the original VVC loop filtering module (including, DBF, SAO, and ALF), as shown in Figure III.3. The principal

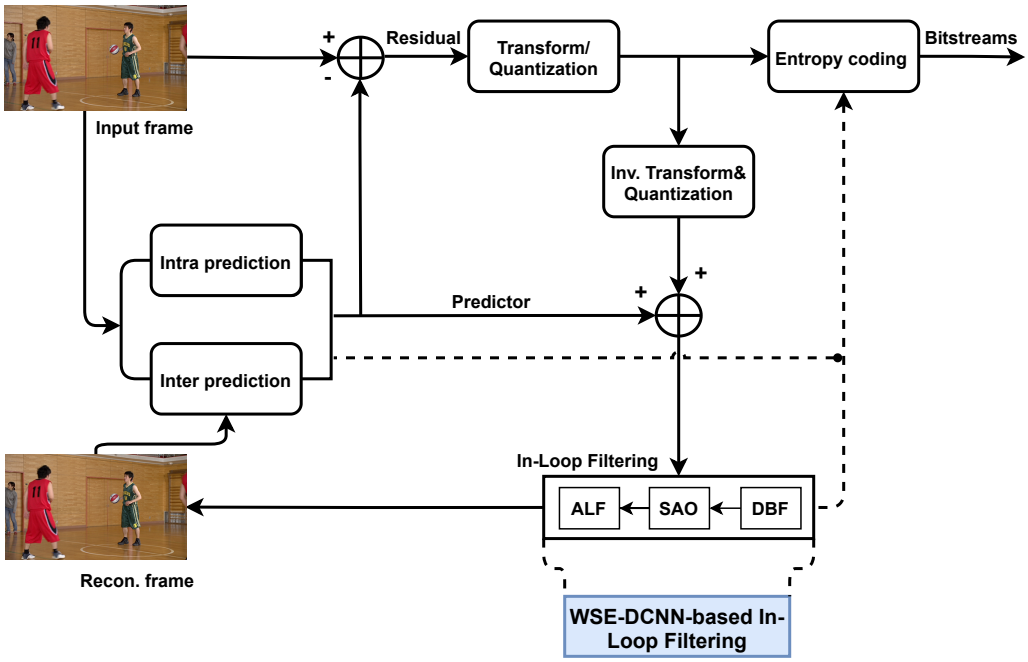



Figure. III.3: Proposed VVC Standard Framework

goal of this strategy is to improve the visual quality of the reconstructed frame while maintaining coding gains. The RDO technique is used to determine whether to apply to each CU the proposed WSE-DCNN in-loop filter. Equation III.2 is given for the RDO metric.

$$J = D + \lambda R, \tag{III.2}$$

where the distortion between the original and the reconstructed frame is denoted by $D$, the coding bits needed represents by $R$ and the Lagrange multiplier controlling the trade-off between $D$ and $R$ is $\lambda$.

The CTU level on/off control is adopted to avoid a reduction in $RDO$ performance. The frame level filtering would be shut off to prevent over-signal, if the enhancement quality is not worth to cost the signaled bits. Specifically, the control flags at the CTU-level and frame-level are designed as follows. For each CTU, if the RD performance of the filtered CTU achieves better quality, the corresponding CTU control flag is enabled,

otherwise the flag is disabled. After all the CTUs in one frame are determined, the frame-level RD cost before and after filtering are calculated using equation III.2 indicated by $J1$ and $J2$, respectively. If $J1 > J2$, the frame-level flag will be enabled. Hence the corresponding frame-level flag can be encoded in the slice header and CTU-level control flags can be signaled into each corresponding CTU syntax. Otherwise, the frame-level flag is disabled and CTU-level flags will not be encoded for transmission anymore.

### III.4.2 WSE-DCNN Architecture

The concept of the proposed architecture is illustrated in Figure III.4. The proposed architecture, is shared by luma ($Y$) and two chroma ($U$ and $V$), where the three components will be filtered simultaneously. The proposed WSE-DCNN model consists of six inputs, three denoting the $YUV$ reconstructed and three including the QP and the CU for luminance and chrominance. Meanwhile, these inputs are first normalized to provide better convergence in the learning phase, and then fed to a WSE-DCNN-based in-loop filtering. Hence, the three ($Y/U/V$) reconstruction are normalized to $[0, 1]$ based on the highest bit depth value. This implies that the normalized values ($P'(x, y)$) are achieved by formula (III.3).

$$P''(x, y) = \frac{P'(x, y)}{1 << B - 1}, x = 1, ..., W, y = 1, ..., H \tag{III.3}$$

where the bit depth is denoted by $B$, $P''(x, y)$ is the normalized value in normalized $Y/U/V$ at (x,y), $W$ and $H$ are the width and the height of the reconstructed frame respectively.

Various QPs contribute to a variety of reconstructed video quality. This makes it easier to use a single set of parameters to fit reconstructions with different qualities. $QP$ should be normalized to $QPmap$ following the formula (III.4).

$$QPmap(x, y) = \frac{QP}{63}, x = 1, ..., W, y = 1, ..., H \tag{III.4}$$

Regarding the other inputs, there are $CU$ partition of luma ($Y$) and chroma ($UV$) components. Since the blocking artifacts are mainly caused by CU block partition. The

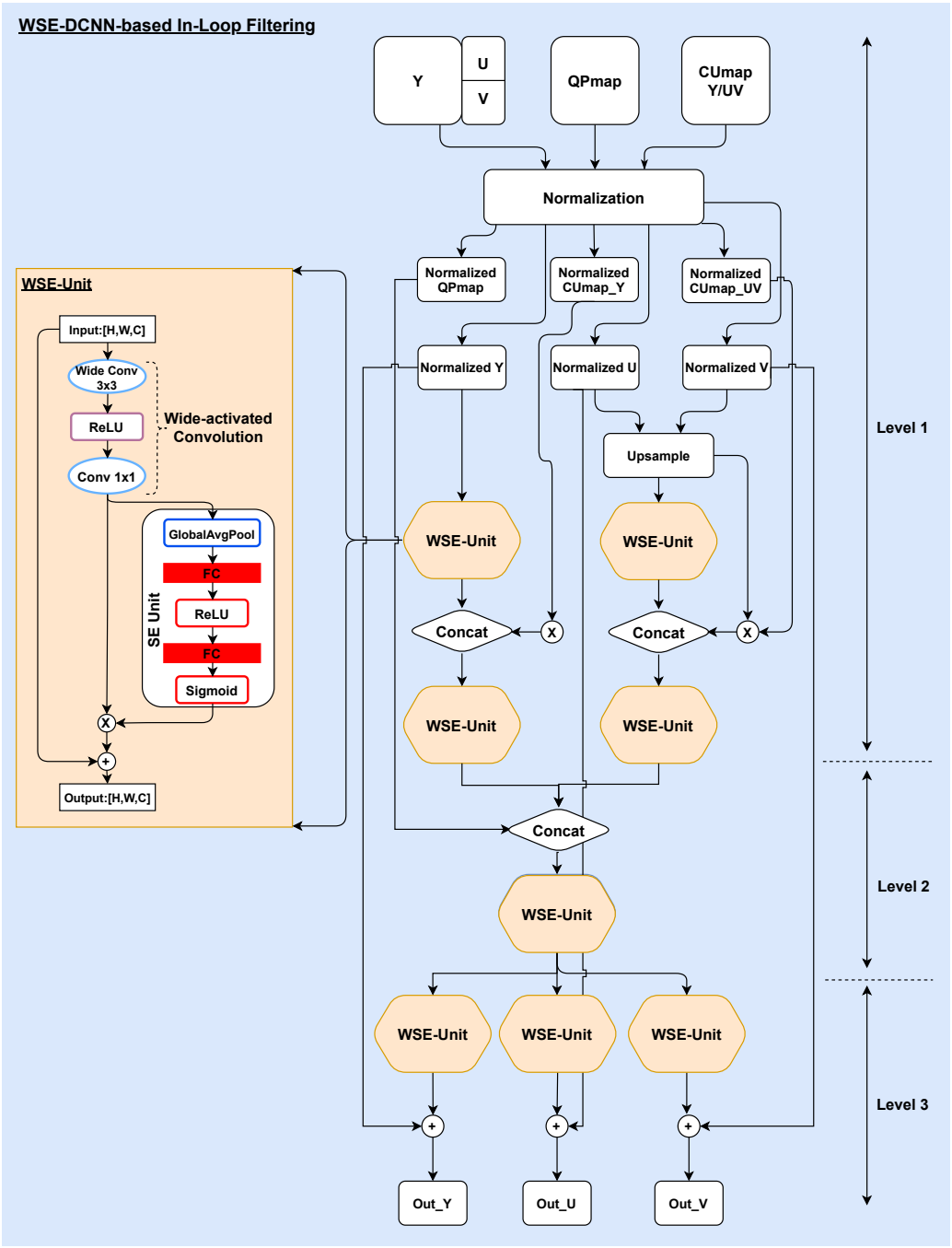



Figure. III.4: WSE-DCNN Architecture

*CU* block partition is converted into Coding Unit maps (*CUmaps*) and normalized and then, it is considered as an input to the network. For example, for each *CU* in each frame, the boundary position is filled with two and the other positions are filled with one. However, two *CUmaps* can be obtained, one as $Y - CUmap$ for luma and the other

denoted by $UV - CUmap$ for chroma.

As shown in Figure III.4, the processing of WSE-DCNN-based in-loop filtering has three levels. At the first level, the three components of $YUV$ are processed through WSE-blocks, and each one of them is fused with its corresponding $CUmap$. In addition, $CUmap$ would be multiplied by its own corresponding channel before being concatenated to feature maps. In the second level, the feature maps of different channels are connected together and then processed by several WSE-blocks. At this level, the $QPmap$ is also concatenated. Finally, in the last level, the three channels are processed separately again to generate the final residual image. Then the original input will be implemented as a residual CNN.

The $WSE$ module is considered to be the basic unit of the WSE-DCNN-based in-loop filtering proposed in the VVC standard, as shown in Figure III.4. Specifically, this basic unit is composed of the Wide-activated convolution [YFY+18] and the Squeeze-and-Excitation ($SE$) [HSS18] operation. The Wide-activated convolution performs very well in super-resolution and noise reduction tasks. It consists of a wide $3 \times 3$ convolution followed by ReLU [NH10] activation function and a narrow $1 \times 1$ convolution. Next comes, the $SE$ operation which is the most technical operation used to weight each convolutional layer. It can use the complex relationship between different channels and generates a weighting factor for each channel.

The WSE unit consists of the following phases as depicted in Algorithm 1, given a feature map X with shape $H \times W \times C$, where $C$ means channel amounts:

- A wide $3 \times 3$ convolution followed by ReLU and a convolution layer with kernel size is $1 \times 1$. Given $Y_1$ is the channel defined in Algorithm 1 and $Y_2$ is the output of the second convolution layer.

- Each channel obtains a value according to the squeeze operation using Global Average Pooling (GAP) $Y_3(k)$ as shown in Algorithm 1.

- The excitation operation is described by two fully connected layers followed by ReLU and sigmoid ($\sigma$) activation functions, respectively. As shown in Algorithm 1, $Y_4$ is the first fully connected layer followed by ReLU, which is refined by a certain

**Algorithm 1** WSE-Unit

**Input:** $X \in \{H, W, C\}$
**Output:** $Y \in \{H, W, C\}$

**for** *number of Epochs* **do**
  **while** *True* **do**
    **Wide-activated Conv-Function(X):**
      $Y_1 = ReLU(W_1 X + b_1)$
      $Y_2 = W_2 Y_1 + b_2$
      **return** $(Y_2)$
      **Squeeze-and-Excit-Function**($Y_2$)**:**
        **while** *True* **do**
          ***Call-Squeeze-Operation***($Y_2$)***:***
          $Y_3(k) = \frac{1}{HXW} \sum_{i=1}^{H} \sum_{j=1}^{W} Y_2(i, j, k)$.
          **return**($Y_3$)
          ***Call-Excitation-Operation***($Y_3$)***:***
          $Y_4 = ReLU(W_4 Y_3 + b_4)$.
          $Y_5 = \sigma(W_5 Y_4 + b_5)$.
          **return**($Y_5$)
        **END while**
      **return**($Y_5$)
      **WSE Function**($Y_2, Y_5$)**:**
      $Y_6(i, j, k) = Y_2(i, j, k) \times Y_5(k)$,
      $\forall i \in \{1, ..., H\}, \forall j \in \{1, ..., W\}$,
      $\forall k \in \{1, ..., C\}$.
      **return**($Y_6$)
  **END while**
**END for**

ratio $r$. Then, the second fully connected layer followed by the sigmoid activation function which is denoted by $Y_5$, and it gives each channel a smoothing gating ratio in the range of [0,1].

- According to WSE function, each $Y_2$ channel is multiplied by the gating ratio $r$, as defined Algorithm 1.

- Finally, when the number of input equals to the output channels $C$, a skip connection will be added directly from input to output to learn the residue. Otherwise, there is no skipped connection.

## III.5 Experimental Results

In this section, we will present the performance of the proposed loop filtering scheme based on WSE-DCNN in the VVC standard through experimental results. We evaluated the performances, in terms of RD performance, and QoE of the proposed scheme. Specifically, we first present the description of the dataset collection in the experiments. Then, model training, testing, and evaluation are provided. Finally, objective evaluations and subjective visualizations are presented. Furthermore, the performance of the proposed method is illustrated and compared with other CNN based in-loop filter algorithms.

### III.5.1 Dataset Collection

In this work, we exploited the public large video dataset BVI-DVC [MZB20], especially developed for training the deep video compression methods. According to the work cited in [MZB20], all selected sequences are progressive-scanned at a spatial resolution of 3840×2160, with frame rates ranging from 24 fps to 120 fps, a bit depth of 10 bit, and in YCbCr 4:2:0 format. All of them are truncated to 64 frames without scene cuts, using the segmentation method described in [MWZ+15]. To further increase data diversity and provide data augmentation, 200 video clips were spatially down-sampled to 1920×1080, 960×540 and 480×270 using a Lanczos third order filter. The BVI-DVC dataset includes 800 video sequences with different contents at four different resolutions. Table III.1 summarizes the key features of BVI-DVC video training dataset used in this study. Figure III.5 shows the sample frames of twenty example sequences.

In this context, we selected 80% of video sequences for the training model and 20% for validation from the BVI-DVC video training dataset. These sequences are compressed by VVC reference software (VTM-4.0) [VTM20] with QP values (22, 27, 32, 37) under RA configuration. For each QP, the reconstruction video images, including luma and chroma components, and their corresponding ground truth are divided into 64×64 patches, which were selected in a random order.

Table III.1: Key Features of BVI-DVC Video Training Database [MZB20]

| Features | BVI-DVC [MZB20] |
|---|---|
| Image or Videos | Video |
| Sequences Number | 800 |
| Images Number in each video | 64 |
| Max Resolution | 2160p |
| Min Resolution | 270p |
| Bit Depth | 10 |
| Various textures | Yes |

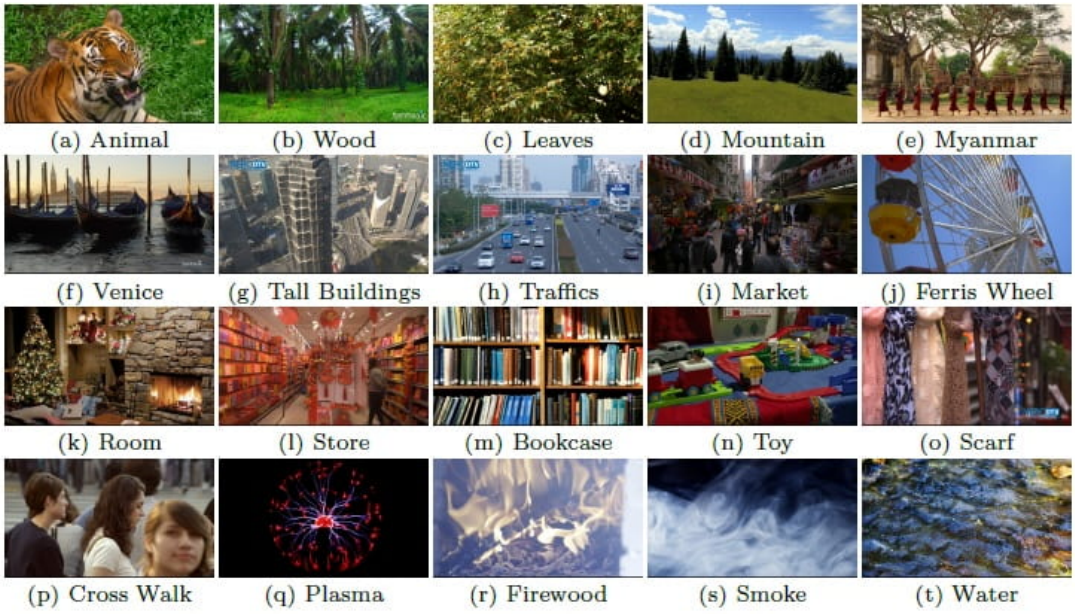



Figure. III.5: Sample Frames of Sequences from the BVI-DVC Database

### III.5.2 Deep Model Training and Testing Evaluation

The proposed deep learning model is trained offline in a supervised learning manner. During training phase, the Tensorflow-GPU [AAB+16] is used as a deep learning framework to train the proposed model. The training parameters used in our experiments are summarized as follows: the batch size is set to 128, the training epochs to 200, the learning rate is set to 0.001 and weight decay of 0.1 for every 50 epochs. The Adam [Da14] optimizer is used to train our deep model. The training platform uses windows

10 OS with Intel®core TM i7-3770 @3.4 GHz CPU and 16 GB RAM and an NVIDIA GeForce RTX 2070 GPU.

The Mean Square Error (MSE) is applied as a loss function between the ground truth image and the reconstructed image [MBA+20]. The following equation III.5 defines the MSE loss function.

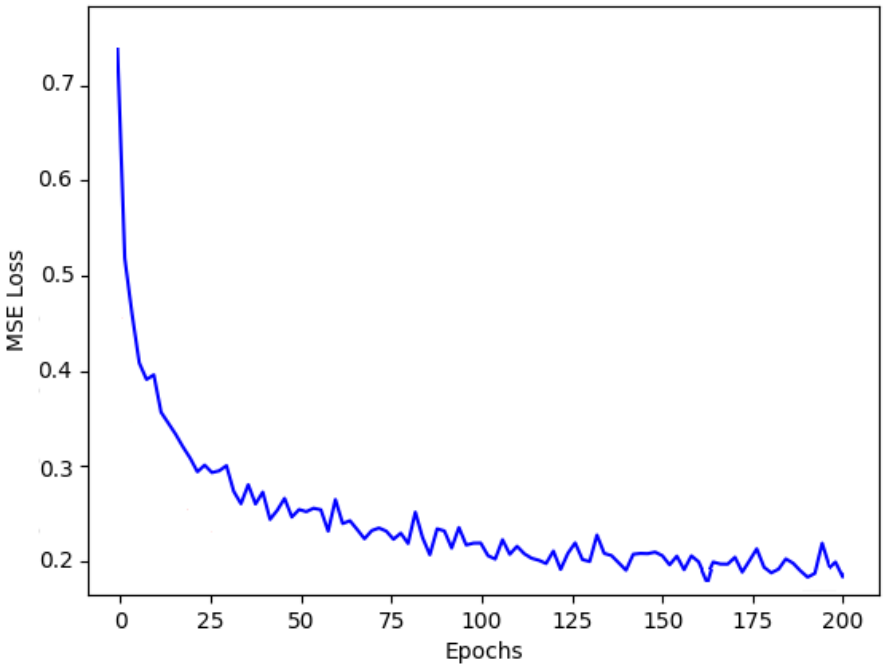
$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} ||F(Y_i, \theta) - X_i||_2^2 \tag{III.5}$$

Let $X_i$ is the ground truth of the proposed model, where $i \in \{1, ..., N\}$. The output of the WSE-DCNN model is denoted by F$(\cdot)$, where $Y_i$ represents the compressed images, $i \in \{1, ..., N\}$ and $\theta$ is the parameter set of the proposed framework. The loss function has indeed converged on a minimum value, it means that our model is well trained. To prove the efficiency of the proposed WSE-DCNN network, Figure III.6 shows the MSE loss and the validation Peak Signal-to-Noise Ratio (PSNR) during training process. The PSNR is defined by the equation III.6 [OSS+12]. As we can see, the MSE loss function performs well in terms of convergence performance.

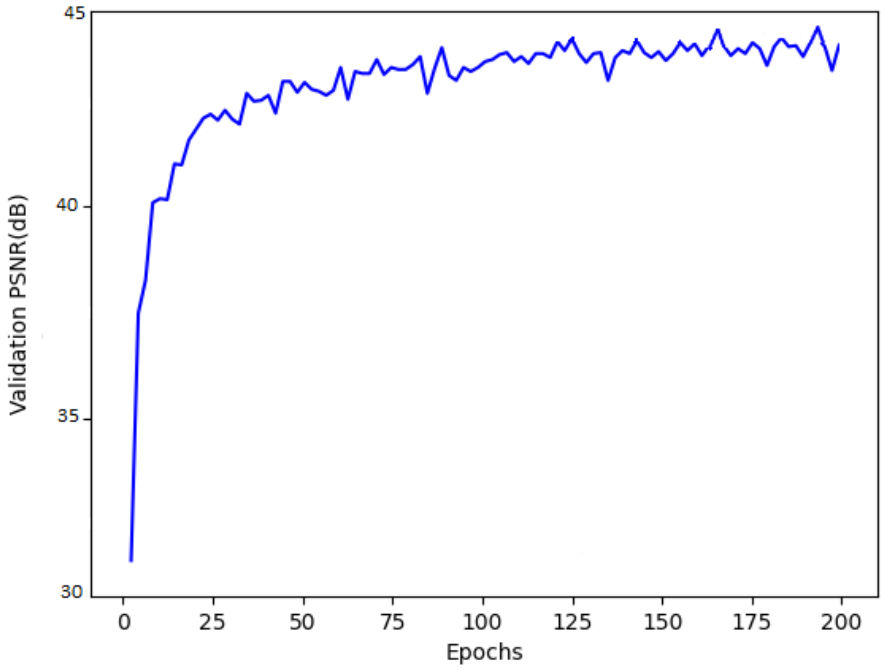
$$PSNR = 10 \times log\frac{(2^B - 1)^2}{MSE} \tag{III.6}$$

Where B is the number of bits per sample of the video sequence and the MSE is defined in the equation III.6.

In the testing phase, our proposed WSE-DCNN model is integrated into VVC standard to replace the traditional in-loop filtering method. All simulations are tested under the VVC JVET Common Test Conditions (CTC) [BBL+19] using Random Access (RA) configuration at $QP$ values (22, 27, 32, 37). The (VTM-4.0) with traditional filters enabled is used in our experiments. From VVC CTC, seventeen test sequences were used for performance evaluation, including class A1 (3840×2160), class A2 (3840×2160), class B (1920×1080), class C (832×480), and class D (416×240). To evaluate the coding performance of the proposed model, Bjontegaard-Delta bitrate (BD-BR) [Bjo01] is applied as an assessment metric.




(a) MSE Loss




(b) Validation PSNR

Figure. III.6: Training MSE Loss and Validation PSNR

### III.5.3 WSE-DCNN Evaluation

The RD performance results of the proposed model compared to the original VVC standard are shown in Table III.2. Columns $Y$, $U$, and $V$ in the table show the BD-BR

Table III.2: Performance Evaluation of the Proposed model under RA Configuration

| Class | Sequences | $Y(\%)$ | $U(\%)$ | $V(\%)$ | $T_{enc}(\%)$ | $T_{dec}(\%)$ |
|---|---|---|---|---|---|---|
| Class A1 | Tango2 | -2.89 | -10.02 | -11.35 | 147 | 939 |
| 4K | Campfire | -1.22 | -2.75 | -10.28 | 119 | 1537 |
| Class A2 | CatRobot1 | -1.89 | -10.76 | -8.03 | 151 | 975 |
| 4K | DaylightRoad2 | -1.47 | -12.36 | -2.55 | 149 | 875 |
| Class B | Kimono2 | -0.51 | -8.13 | -20.63 | 82 | 1524 |
| 1080p | ParkScene | -4.18 | -9.25 | -12.94 | 125 | 1947 |
| | Cactus | -2.36 | -12.27 | -9.70 | 117 | 1154 |
| | BasketballDrive | -2.53 | -4.83 | -7.82 | 107 | 1981 |
| | BQTerrace | 0.11 | -2.88 | 0.63 | 116 | 1619 |
| Class C | BasketballDrill | -3.84 | -7.01 | -9.97 | 137 | 1312 |
| WVGA | BQMall | -3.89 | -11.48 | -10.92 | 118 | 1137 |
| | PartyScene | -4.65 | -9.69 | -9.63 | 115 | 1112 |
| | RaceHorses | -1.35 | -10.70 | -13.66 | 94 | 1079 |
| Class D | BasketballPass | -3.40 | -8.21 | -7.79 | 125 | 4628 |
| WQVGA | BQSquare | -5.27 | -4.39 | -11.44 | 137 | 2001 |
| | BlowingBubbles | -4.15 | -8.52 | -5.19 | 115 | 2045 |
| | RaceHorses | -5.08 | -18.04 | -19.74 | 114 | 2164 |
| **Overall** | | **-2.85** | **-8.89** | **-10.05** | **122** | **1648.76** |

of $Y$, $U$, and $V$ components, respectively. Ratios of the encoding and decoding time are denoted by $T_{enc}$ and $T_{dec}$ of the proposed model compared to the original one. The encoding time is defined by the equation III.7.

$$T = \frac{T_{Pro}}{T_{Orig}} \times 100\% \tag{III.7}$$

where the encoding or decoding time of the proposed method and original VVC are defined by $T_{Pro}$ and $T_{Orig}$, respectively.

As illustrated in Table III.2, the proposed scheme achieves better mean coding gains when it is integrated into VVC. 2.85% BD-BR savings for luma $Y$ component under RA configuration is obtained, while achieving 8.89% and 10.05% BD-BR reduction for both chroma $U$ and $V$ components. The proposed scheme offers significant RD-compression performance mainly in $U$ and $V$ chrominance for all test sequences. It is also clear that the coding performance differs for several sequences. This means that the proposed model is impacted by video sequence information. Moreover, the proposed model performs well in terms of coding gains for high motion or rich texture video sequences, as well as, Tango2, DaylightRoad2, Kimono2, and RaceHorses, etc. In summary, the proposed technique outperforms better the VVC with traditional in-loop filtering algorithm in terms of RD performance.

Regarding complexity reduction, the time differences between the proposed VVC algorithm and the original VVC standard for encoding and decoding, are summarized in Table III.2. NVIDIA GeForce RTX 2070 GPU is used to measure the encoding and decoding time of the proposed filtering technique. From this table, the mean encoding time obtained is about 122% and the mean decoding time is about 1648.76% compared to the original VVC algorithm under RA configuration. Therefore, we note that the proposed model achieves a significant reduction in coding complexity compared to the original VVC algorithm.

To demonstrate the effectiveness of our proposed filtering model integrated into the VVC standard, PSNR is also used as a quality measure [OSS+12], which is calculated by the following equation III.8:

$$PSNR_{YUV} = \frac{6 \times PSNR_Y + PSNR_U + PSNR_V}{8} \tag{III.8}$$

The BQSquare video sequence encoded with $QP$ equals to 37 under RA configuration is deployed in order to show the subjective visual quality and to further verify the effec-

(a) (b)

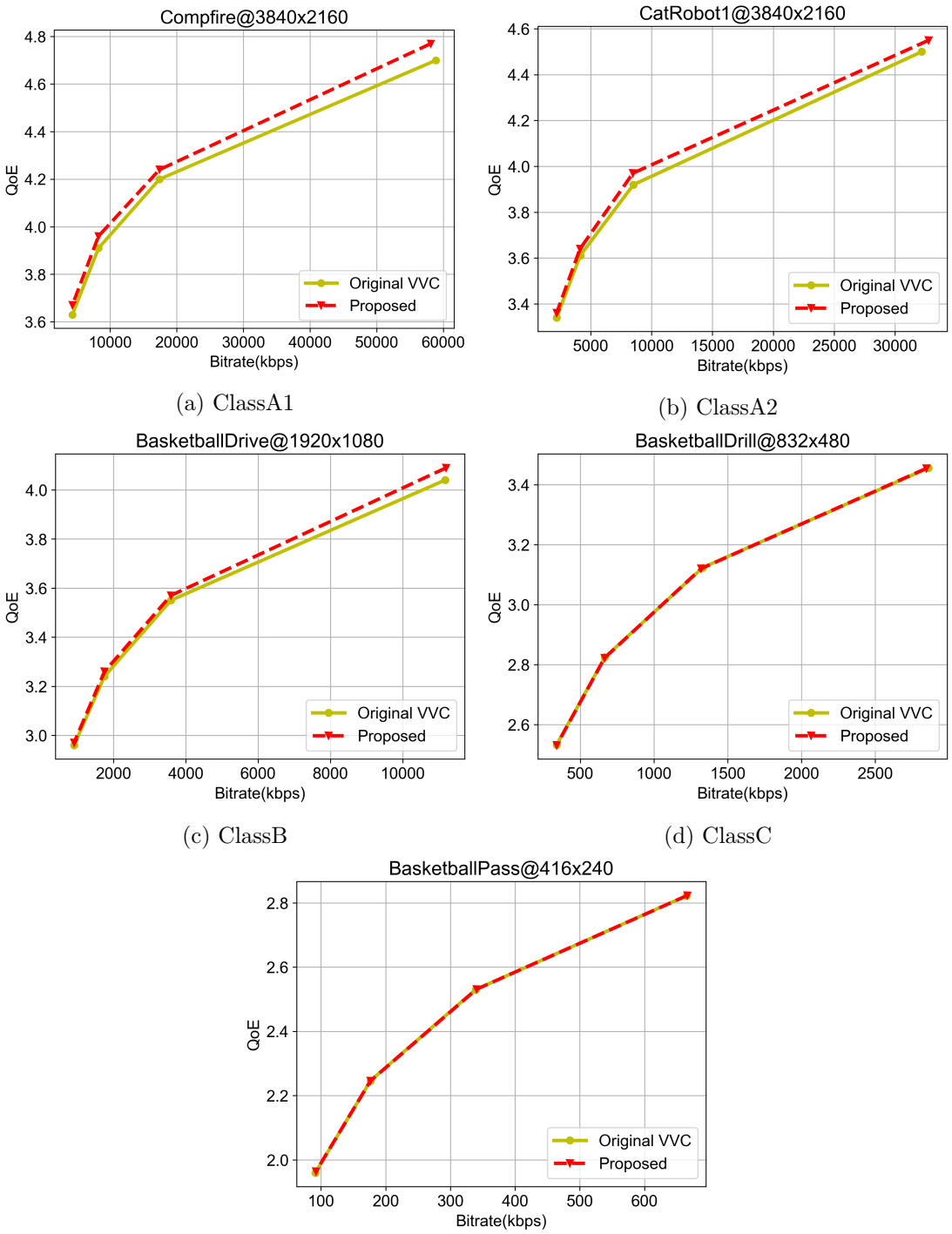
(c) (d)

Figure. III.7: Ablation Study. Subjective Visual Quality Comparison (the 12th frame of BQSquare with $QP$=37: (a) Original; (b) VVC without in-loop filtering ($PSNR$=31.17dB); (c) VVC ($PSNR$=31.37dB); (d) VVC-based proposed model ($PSNR$=31.68dB)

tiveness of the proposed model. Figure III.7 shows the comparison of video subjective quality. It is clear that frame details are blurry being compressed by the original VVC standard, but become clearer after being filtered by the proposed model. In fact, the proposed model effectively eliminates blocking artifacts as well as ringing artifacts and blurring, which improves visual quality, compared to the VVC standard with/without traditional in-loop filtering. Therefore, we compare the QoE variation with respect of bitrate of the proposed technique with the original VVC for class $A1$ to class $D$ using RA configuration at four QPs, as shown in Figure III.8. It is remarkable that the suggested technique meets the QoE requirements of the end users, especially in high resolution video sequences, such as in class $A1$, class $A2$, and class $B$.

### III.5.4 Comparative Study

We also compared the proposed approach with other filtering models based on CNN network. Table III.3 shows the comparison of encoding performance with other approaches




(a) ClassA1

(b) ClassA2

(c) ClassB

(d) ClassC

(e) ClassD

Figure. III.8: Comparison of QoE Variation with Respect to bitrate

[CCWL20], [KK19], [WWG+19], and [HLS20] in terms of reducing RD-complexity under RA using VVC CTC. In [CCWL20], the authors proposed an in-loop filter algorithm-based Dense Residual convolutional neural Network (DRN) to improve the reconstructed

video quality. This network is integrated after *DF*, before *SAO*, and *ALF* into VVC VTM-4.0 test model. This model is trained using the DIV2K dataset [DIV18]. More-

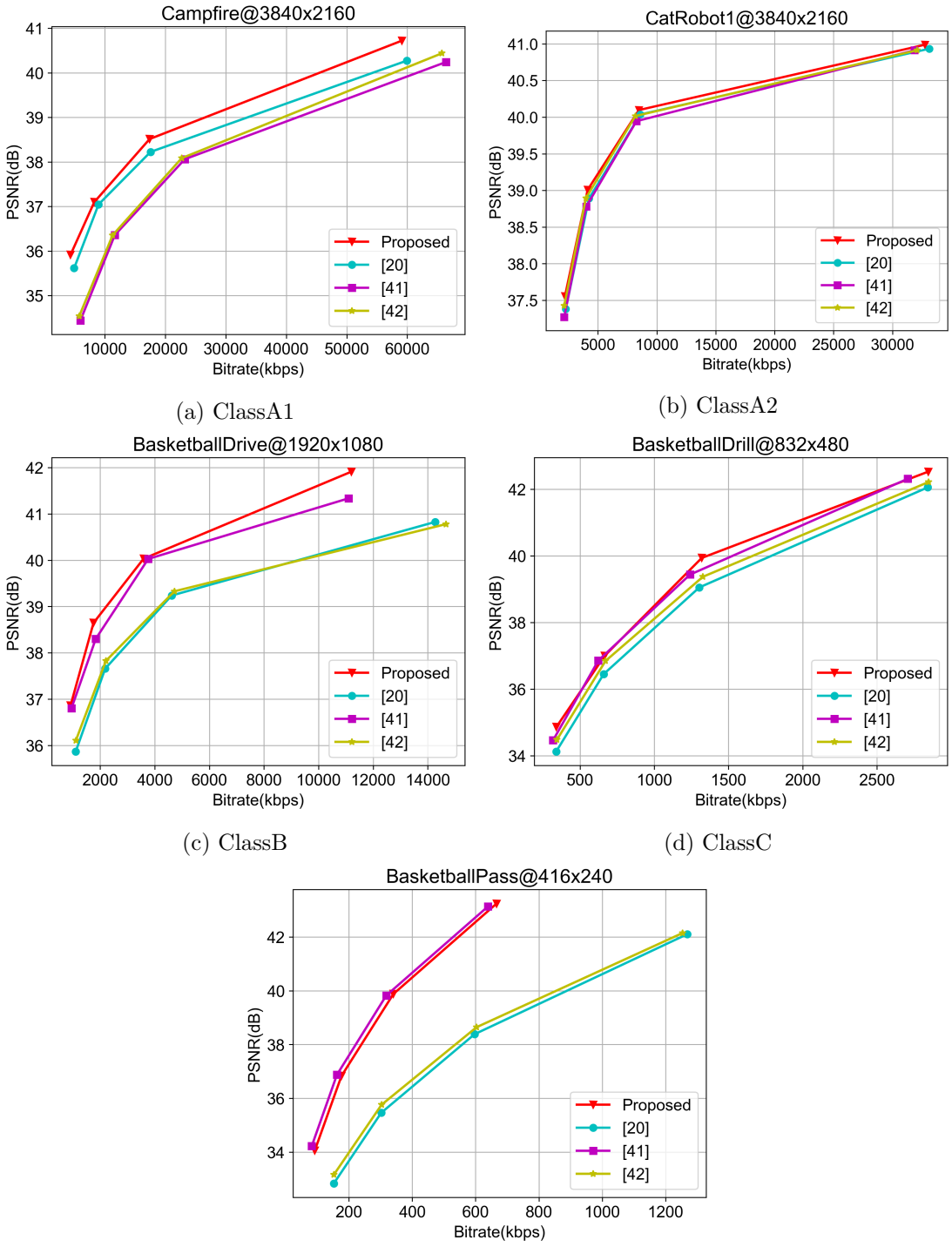
Table III.3: Coding Performance Comparison with other Approaches

| Class | schemes | $Y(\%)$ | $U(\%)$ | $V(\%)$ | $T_{enc}(\%)$ | $T_{dec}(\%)$ |
|---|---|---|---|---|---|---|
| Class A1 | [CCWL20] | -1.27 | -3.38 | -5.10 | 106 | 6967 |
| 4K | [KK19] | 0.87 | 0.12 | 0.22 | 149 | 81711 |
| | [WWG+19] | 0.18 | 0.63 | -2.95 | 138 | 123657 |
| | [HLS20] | -1.10 | -0.30 | 0.31 | – | – |
| | Ours | **-2.05** | **-6.38** | **-10.81** | **133** | **1238** |
| Class A2 | [CCWL20] | -2.21 | -5.74 | -2.88 | 106 | 6435 |
| 4K | [KK19] | -1.12 | -0.52 | -2.11 | 142 | 81263 |
| | [WWG+19] | -0.98 | 7.16 | -3.34 | 138 | 121571 |
| | [HLS20] | -1.94 | 0.89 | -2.24 | – | – |
| | Ours | **-1.68** | **-11.56** | **-5.29** | **150** | **925** |
| Class B | [CCWL20] | -1.13 | -4.73 | -4.55 | 106 | 7011 |
| 1080p | [KK19] | -0.83 | -0.47 | -1.20 | 143 | 69595 |
| | [WWG+19] | 0.64 | -4.16 | -3.67 | 149 | 154962 |
| | [HLS20] | -2.51 | -1.28 | -0.89 | – | – |
| | Ours | **-1.89** | **-7.47** | **-10.09** | **109** | **1645** |
| Class C | [CCWL20] | -1.39 | -3.63 | -4.36 | 106 | 8110 |
| WVGA | [KK19] | -1.76 | -3.64 | -6.80 | 122 | 46645 |
| | [WWG+19] | -1.17 | -4.38 | -1.61 | 129 | 122434 |
| | [HLS20] | -4.03 | -4.17 | -5.62 | – | – |
| | Ours | **-3.43** | **-9.72** | **-11.05** | **116** | **1160** |
| Class D | [CCWL20] | -1.39 | -1.96 | -3.08 | 105 | 4217 |
| WQVGA | [KK19] | -2.95 | -3.27 | -7.35 | 123 | 32155 |
| | [WWG+19] | -3.13 | -6.26 | -5.15 | 122 | 104265 |
| | [HLS20] | -5.33 | -4.11 | -5.83 | – | – |
| | Ours | **-4.47** | **-9.79** | **-11.04** | **122** | **2709** |

over, a CNN-based in-loop filter algorithm is proposed for both intra and inter pictures placed before $ALF$ with $DBF$ and $SAO$ disabled [KK19]. This method is implemented into VVC VTM-3.0 standard [KK19]. Yet, in [WWG+19], the authors proposed a CNN based loop filter placed all traditional filters in VVC which trained based on the DIV2K [DIV18] dataset and implemented in VTM-5.0. In addition, Huang et al. [HLS20] proposed a novel multi-gradient convolutional neural network based in-loop filter for VVC to replace the original DBF and SAO filters. This network trained based on the DIV2K dataset [DIV18] and implemented in VTM-3.0.

As shown in Table III.3, the proposed WSE-DCNN framework integrated into VVC standard achieves a best $RD$ performances for both luminance and two chrominance for all test sequences from class A1 to class D, as compared to previous proposed approaches. These results imply that the proposed model performs well in terms of both objective and subjective visual qualities. Regarding the computational complexity, the proposed method outperforms other approaches in terms of encoding time for class $A2$. Compared with related works in [CCWL20], [KK19], and [WWG+19], the methods proposed exceed our proposed scheme in terms of complexity reduction. In conclusion, the suggested method leads to efficient performance in almost all test sequences in terms of $RD$ performance, demonstrating the efficiency and universality of the WSE-DCNN solution compared to other methods. Whereas the computational complexity of VVC standard is still limited.

For further evaluation, we have provided $RD$ performance curves of the suggested model-based in-loop filtering versus the other three methods under RA configuration with four $QPs$ for class A1 to class D. Figure III.9 shows the comparison in terms of $RD$ performance (PSNR based on bitrate). By comparing the associated approaches, we can conclude that the proposed filtering model considerably improves the $RD$ performance of the VVC standard. Our proposed in-loop filtering model works well especially in high resolution video sequences, as well as in class A1, class A2, and class B.




(a) ClassA1

(b) ClassA2

(c) ClassB

(d) ClassC

(e) ClassD

Figure. III.9: RD-performance Curves of the Proposed Model Compared to other three Approaches

## III.6 Conclusion

This chapter proposed a deep learning algorithm-based VVC standard to enhance visual video quality while improving the user's QoE. The proposed WSE-DCNN framework is implemented into VVC standard to replace in-loop filtering in order to alleviate the coding artifacts, such as ringing, blocking, and blurring. The proposed VVC filtering technique is used in the M-IoT scenario-based smart city context to contribute to the centralized cloud that attempts to meet the required user's video quality. Compared to the traditional VVC-based filters, the proposed framework achieves the best compression performance in terms of objective and subjective quality.

Deep learning algorithms, such as CNNs, have significant higher accuracies than traditional algorithms, but they require huge amounts of computational resources and memory access due to the large number of parameters in the layers operation, which represents a computational challenge. Therefore, many hardware accelerators, such as FPGAs have been used to improve CNN performance, which is the purpose of the next chapter.

# Chapter IV

# Deep CNN Co-Design for HEVC CU Partition Prediction on FPGA-SoC

**Summary**

## IV.1 Introduction

Convolutional Neural Networks are widely used, due to their excellent performance, in many computer vision applications, such as facial recognition, image classification tasks, speech recognition programs, video gaming, etc. However, CNNs require a large number of memory resources and they are also computationally intensive. Field Programmable Gate Arrays (FPGAs), especially the new technology FPGA-SoC, are considered as the most promising platforms for accelerating CNNs, due to their high performance capabilities, energy efficiency, and reconfigurable property.

This chapter provides a hardware-software architecture based on an accelerated CNN model for a video compression application. We first accelerate the CNN layers to build an Intellectual Property (IP) cores using Vivado High Level Synthesis (HLS). Then, we create a hardware-software architecture based on a CNN's IP cores designed and integrated in the Programmable Logic zone (PL) which is connected to the Xilinx Processing System (PS) that manage all processing tasks on the FPGA-SoC board.

The remainder of this chapter is organized as follows. Section IV.2 provides the background in which the preliminary study are included. Section IV.3 discusses the proposed CNN accelerator on FPGA-SoC. Section IV.4 describes the experimental results and the discussions. Section IV.5 concludes this chapter.

## IV.2 Background

Recent AI methods, such as deep learning, have enjoyed considerable success in various machine learning tasks because of their powerful learning ability [KLP20]. They have been broadly applied in many signal processing areas including computer vision, image processing, data mining. However, computationally intensive deep learning algorithms had to be run on embedded devices [HALK21], such as FPGAs. Especially, the new technology FPGA-SoC are considered as the most promising platforms for accelerating AI methods, due to their real-time performance, high energy efficiency and flexible designs.

### IV.2.1 FPGA-SoC

Nowadays, the limitations of Application Specific Integrated Circuit (ASIC) System-on-Chips (SoCs) make them incompatible with a large number of applications in terms of time-to-market, flexibility and upgrade capability. There is a clear need for a more flexible solution, and this is what motivates the System-Programmable-Chip, a specific flavour of SoC implemented on a programmable, reconfigurable device. However, the FPGAs is the popular solution used, which are inherently flexible devices that can be configured to implement any arbitrary system, including embedded processors if needed. FPGAs can also be reconfigured as often as desired, thus offering a more fundamentally flexible platform than ASICs for implementing SoCs.

Based on the Xilinx Ultra-Scale architecture, Zynq provides an even more ideal platform for implementing flexible SoCs: Xilinx markets the device as an 'All-Programmable SoC' (APSoC). The Zynq architecture consists of two main parts: a PS formed around a dual-core ARM Cortex-A9 processor, and a PL, which is equivalent to that of an FPGAs, as illustrated in Figure IV.1. It also has built-in memory, a variety of peripher-

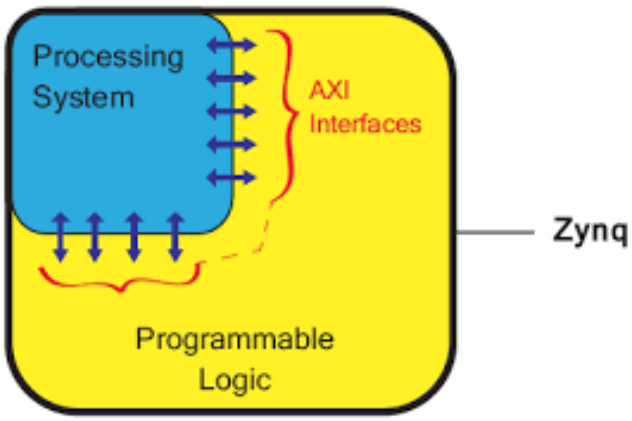



Figure. IV.1: A simplified Model of the Zynq Architecture

als and high-speed communication interfaces. The PL section is ideal for implementing high-speed logic, arithmetic and data flow subsystems. While the PS supports software routines and/or operating systems, meaning that the overall functionality of any designed system can be appropriately partitioned between hardware and software. Links between the PL and PS are made using industry standard Advanced eXtensible Interface

(AXI) connections.

#### IV.2.1.1 Processing System (PS)

All Zynq devices have the same basic architecture, and all of them contain, as the basis of the processing system, a dual-core ARM Cortex-A9 processor [CEES14]. Figure IV.2 depicts the PS architecture. This is a 'hard' processor, it exists as a dedicated and

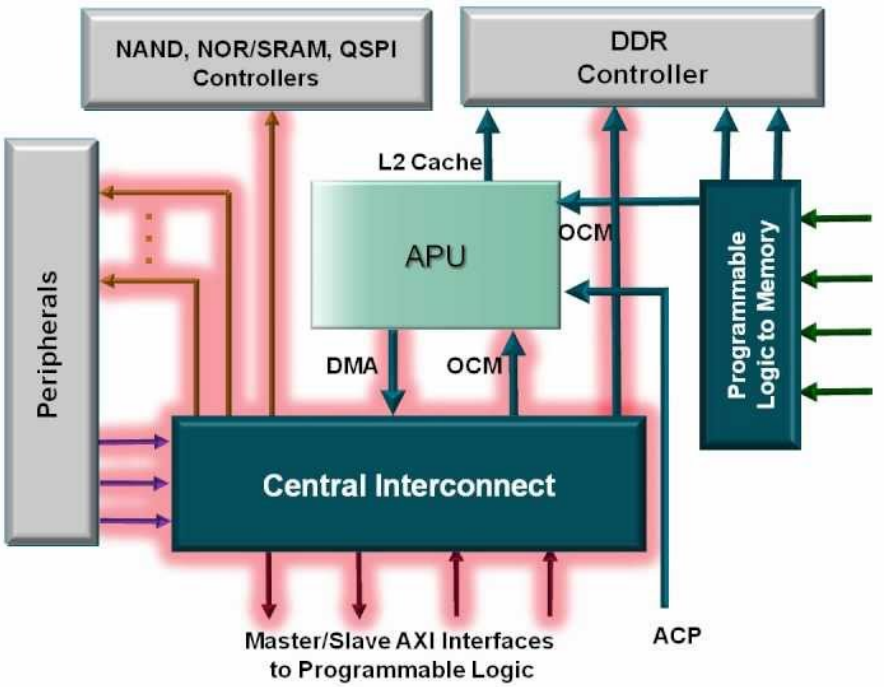



Figure. IV.2: Zynq Processing System

optimised silicon element on the device. For comparison purposes, the alternative to a hard processor is a 'soft' processor like the Xilinx MicroBlaze, which is formed by combining elements of the PL fabric. The implementation of a soft processor is therefore the equivalent of any other IP block deployed in the logic fabric of an FPGA. In general, the advantage of soft processors is that the number and precise implementation of processor instances are flexible. On the other hand, hard processors can achieve higher performance, as is the case with Zynq's ARM processor.

It is important to note that the Zynq processing system encompasses not only the ARM processor, but a set of associated processing resources forming an application processing unit, as well as other peripheral interfaces, cache memory, memory interfaces,

interconnect, and clock generation circuitry [CEES14].

#### IV.2.1.2 Programmable Logic (PL)

PL is the second principal part of the Zynq architecture [CEES14], which is based on the Artix®-7 and Kintex®-7 FPGA fabric. The PL part of the Zynq device is illustrated in Figure IV.3, with various features highlighted. The PL is mainly composed of a general purpose FPGA logic structure, which is made up of slices and Configurable Logic Blocks (CLBs), as well as Input/Output Blocks (IOBs) for interfacing. Indeed, CLBs are small, regular groupings of logic elements that are laid out in a two-dimensional array on the PL, and connected to other similar resources via programmable interconnects.

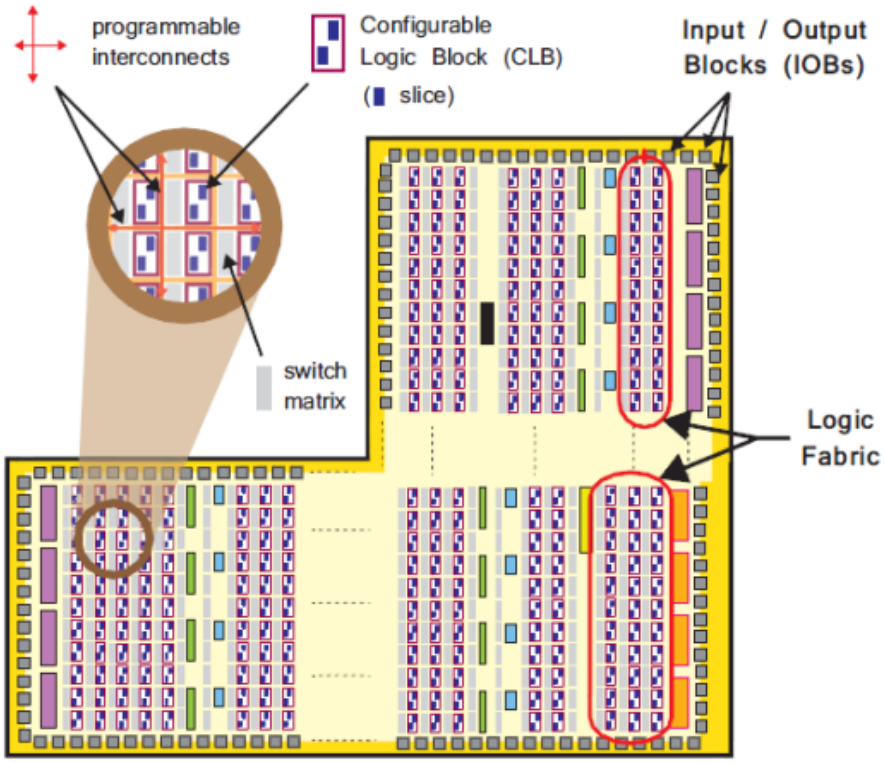



Figure. IV.3: Zynq Programmable Logic

Each CLB is located next to a switch matrix and contains two logic slices, as presented in Figure IV.4. Where, the slice is a subunit within the CLB, containing resources to implement combinatorial and sequential logic circuits. As shown in Figure IV.4, Zynq slices are composed of 4 Look-Up-Tables (LUTs), 8 Flip-Flops (FFs), and other logic. LUTs are a flexible resource capable to implement a logic function of up to six inputs.

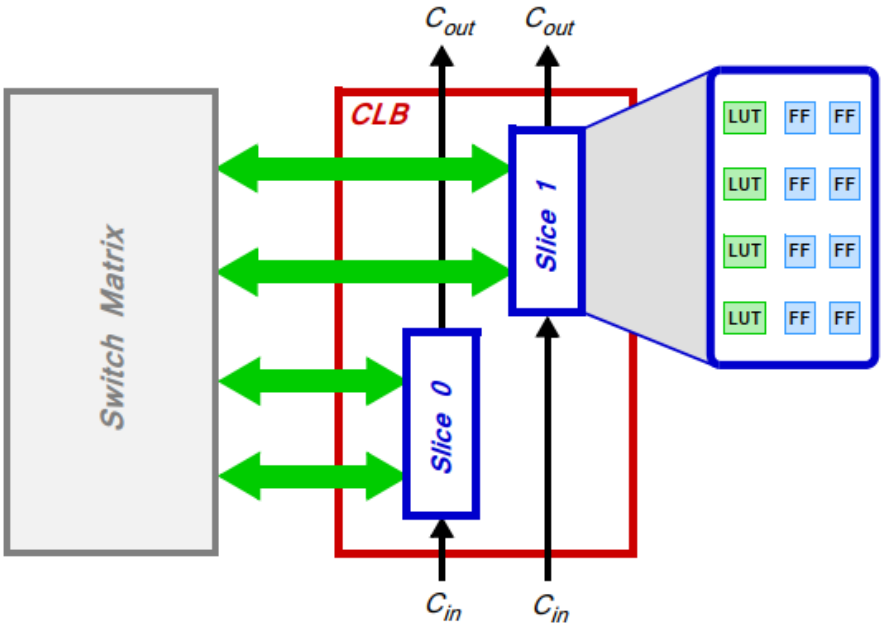



Figure. IV.4: Block CLB

They can be combined together to form larger logic functions, memories, or shift registers, as required. While, the FF is a sequential circuit element implementing 1-bit register, with reset functionality. One of the FFs can optionally be used to implement a latch. Carry logic is also an arithmetic circuit that requires intermediate signals to be propagated between adjacent slices, and this is achieved via carry logic. The carry logic comprises a chain of routes and multiplexers to link slices in a vertical column. Finally, IOBs are resources that provide interfacing between the PL logic resources, and the physical device 'pads' used to connect to external circuitry. Each IOB can handle 1-bit input or output signal. IOBs are usually located around the perimeter of the device.

#### IV.2.1.3 PS—PL Interfaces

In this section, we introduce the connections between the PS and PL and consider how they can be used. We start by introducing the AXI standard, on which most of these connections are adopted. AXI stands for Advanced eXtensible Interface, and the current version is AXI4, which is a part of the ARM AMBA®3.0 open standard. Many devices and IP blocks produced by third party manufacturers and developers are based on this standard. Support for AXI was first introduced into the Xilinx tool flow in release 12.3

of the ISE® Design Suite, and extensive support is now available in the Vivado Design Suite. AXI buses can be used flexibly, and in general are used to connect the processor to other IP blocks in an embedded system. In fact, there are three versions of AXI4, each representing a different bus protocol, as summarised below. The choice of AXI bus protocol for a particular connection depends on the desired properties of that connection.

In fact, AXI4 is for memory-mapped links and providing the highest performance; an address is supplied followed by a data burst transfer of up to 256 data words. AXI4-Lite is a simplified link supporting only one data transfer per connection. AXI4-Lite is also memory-mapped; in this case an address and single data word are transferred. Then, AXI4-Stream is for high-speed streaming data and supporting burst transfers of unrestricted size. There is no address mechanism; this bus type is best suited to direct data flow between source and destination (non memory mapped). In addition,

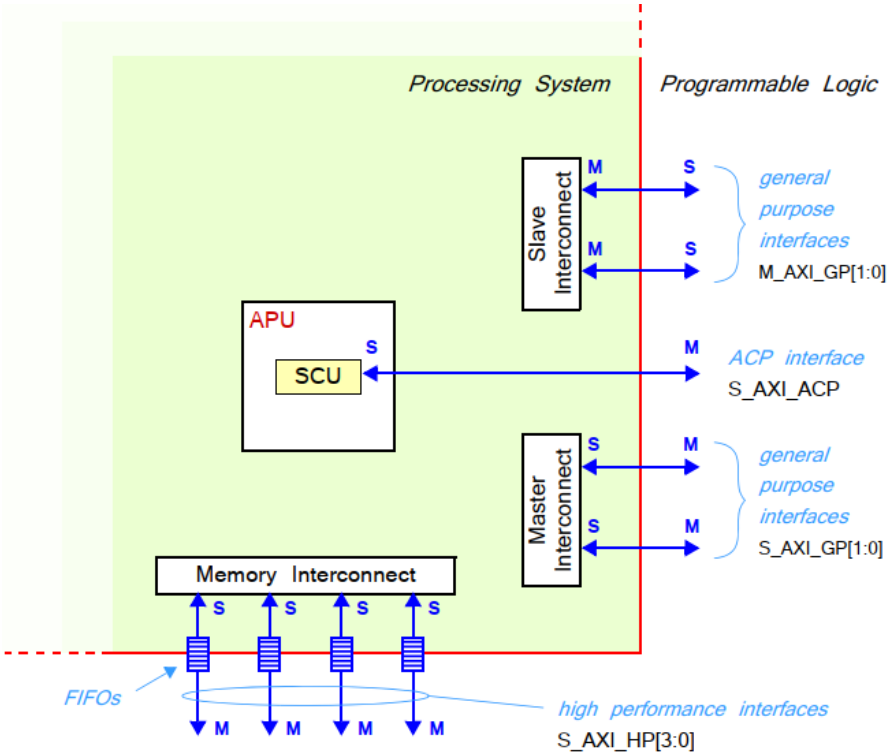



Figure. IV.5: AXI Interconnects and Interfaces

the main interface between the PS and the PL is via a set of nine AXI interfaces, each consisting of several channels. These establish dedicated connections between the PL and the interconnects within the PS, as indicated in Figure IV.5. It is useful to briefly

define these two important terms. An interconnect is actually a switch that manages and directs traffic between the connected AXI interfaces. There are several interconnects within the PS, some are directly interfaced with the PL (like Figure IV.5) and others are for internal use only. The connections between these interconnects are also formed using AXI interfaces. An Interface is a point-to-point connection for passing data, addresses, and handshaking signals between master and slave clients within the system. Other interface such as EMIO (Extended Multiplexed IO), DMA (Direct Memory Access) are also used.

### IV.2.2 Direct Memory Access (AXI DMA)

AXI DMA transfers data between memory and AXI4-Stream-type target peripherals [Joh14]. AXI DMA in Vivado provides high-bandwidth direct memory access between an AXI4 memory-mapped and an AXI4-Stream ports on IPs interfaces [Xil19a]. PYNQ supports the AXI central DMA IP with the PYNQ DMA class [PYN19]. DMA can be used for high performance burst transfers between PS DRAM and PL. It helps to offload data from the Central Processing Unit (CPU) in processor-based systems [Xil19a]. AXI DMA data movement between system memory and stream target is through the AXI4 Read Master to AXI4 memory-mapped to stream (MM2S) Master, and AXI stream to memory-mapped (S2MM) Slave to AXI4 Write Master.

### IV.2.3 FPGA-SoC: PYNQ-Z1

The PYNQ-Z1 FPGA is the chosen hardware platform, which is based on Xilinx ZYNQ SoC technology [Xil19b]. It provides a Python environment, to make it easier for designers to exploit the PL and PS of the FPGA board. Xilinx offers Python packages and associated libraries to facilitate the interaction with hardware modules based on Overlays. Overlays, or hardware libraries, are designed to be programmable and reusable FPGA designs to extend the user application from The PS into the PL of the ZYNQ. An overlay is a PL design class developed by hardware designers. PYNQ overlays can be customized the hardware platform for a certain application.

### IV.2.4 High Level Synthesis (HLS)

Vivado High Level Synthesis (HLS) allows functions written in C, C++, SystemC and OpenCL kernels to be synthesized into a Register-Transfer-Level (RTL) implementation [Xil14]. Vivado HLS provides a number of optional C libraries to enable higher productivity and high performance RTL design. These include arbitrary precision libraries allowing operations to be performed at any arbitrary precision. Video libraries allow us to easily implement many of the most popular OpenCV video functions on math, linear algebra and DSP libraries providing high quality RTL implementations. The output from Vivado HLS is a high quality RTL implementation in both verilog and VHDL languages. This RTL can be added into the Vivado IP catalog and used within IP integrator on Vivado or used in system generator for DSP. The Vivado HLS design flow is presented in Figure IV.6.

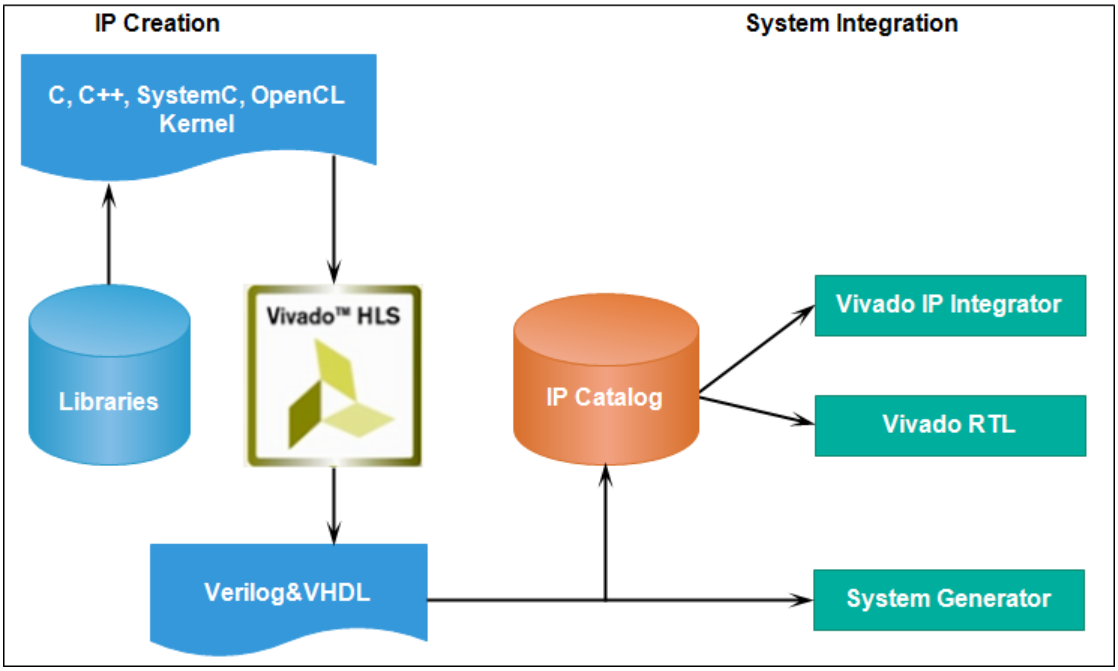



Figure. IV.6: Vivado HLS Design Flow

## IV.3 Proposed CNN Accelerator on FPGA-SoC

In this section, we introduce the proposed deep CNN-based CU partition for HEVC. Then, we accelerate our proposed deep CNN on PYNQ-Z1 FPGA board with a hardware-

software architecture.

### IV.3.1 Proposed Deep CNN Architecture

As mentioned in the previous chapter, the HEVC standard adopts a quadtree structure, known as CTU. CTU supports CU partition from $64 \times 64$ to $8 \times 8$ at four levels. The CU quadtree partition consumes much of the HEVC encoding complexity, due to the adoption of a wide variety of CU sizes at the RDO level. Therefore, the cost of computational complexity remains a critical issue that must be properly considered in the optimization task. To overcome this issue, we proposed a deep CNN architecture, as shown in Figure IV.7, aiming to predict the CU partition at HEVC inter-prediction [BKSA20a], already detailed in chapter II.

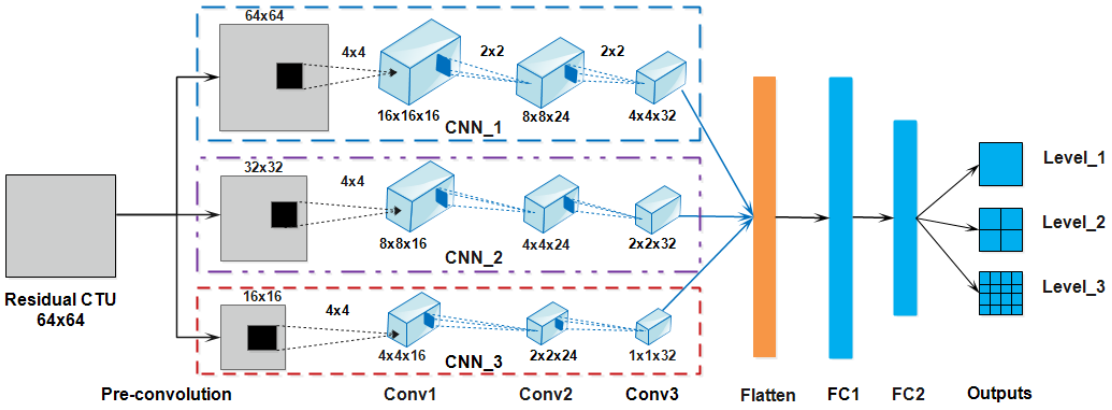



Figure. IV.7: Proposed Deep CNN-based CU Partition for HEVC

Our proposed Deep CNN architecture contains three interconnect CNN models (CNN_1, CNN_2, CNN_3) corresponding to three levels. These CNN models present the HEVC CU partition sized $64 \times 64$, $32 \times 32$ and $16 \times 16$. Each CNN network presented consists of three convolutional layers, flatten, two fully connected layers and output layer, as depicted in Figure IV.7.

The residual CTUs of size $64 \times 64$ are considered as an input image of a proposed model. Then, the input CTUs are down-sampled into three levels corresponding to $64 \times 64$, $32 \times 32$ and $16 \times 16$. At each level, all features maps are extracted through three convolutional layers. 16 filters of size $4 \times 4$ at the first convolutional layer of each level are used. In the same way, at the second and third convolutional layer, the extracted

features are convoluted with 24 filters and 32 filters sized $2 \times 2$. In addition, all feature maps are concatenated together and then flattened into a vector to move at the last layers of CNN model. At the end, three fully connected layers include two hidden layers and one output layer. The two first fully connected layers combine the feature vectors and the output layer generates the predicted CU partition at three levels. The outputs have 1, 4 and 16 elements on level 1, level 2 and level 3, serving as the predicted binary labels corresponding to $64 \times 64$, $32 \times 32$ and $16 \times 16$.

The proposed CNN has better performance compared to conventional techniques, but it requires higher computational complexity which is due to the large number of parameters in convolutional and fully connected layers. To solve this issue, this model will be accelerated on a hardware platform.

### IV.3.2 CNN Accelerator based on Vivado HLS

In this chapter, we have chosen to accelerate the first CNN model (CNN_1) corresponding to level 1 for a $64 \times 64$ CU partition. In the proposed deep CNN_1 model, three convolutional layers are used, each one is characterized by specific parameters such as the filters number and the kernel size, as mentioned in Table IV.1.

Table IV.1: CNN_1 Model Summary

| Layers | Input Size | Filters Number | Kernel Size | Output Size |
|---|---|---|---|---|
| Input CTU | $64 \times 64$ | / | / | / |
| CONV1 | $64 \times 64$ | 16 | $4 \times 4$ | $16 \times 16$ |
| CONV2 | $16 \times 16$ | 24 | $2 \times 2$ | $8 \times 8$ |
| CONV3 | $8 \times 8$ | 32 | $2 \times 2$ | $4 \times 4$ |
| Flatten | 2048 | / | / | / |
| FC1 | 2048 | / | / | 64 |
| FC2 | 64 | / | / | 48 |
| FC3 | 48 | / | / | 1 |

The Vivado HLS is used to design IPs for each convolutional and fully connected layers (CONV-IP, FC-IP) based hardware accelerator. The proposed method aims to reduce

the hardware cost and energy while optimizing the processing time. Figure IV.8 and Figure IV.9 depict the accelerated CONV-IP cores (CONV1_0, CONV2_0, CONV3_0) and FC-IP cores (FC_1, FC_2, FC_3), respectively. Each CONV-IP consists of the AXI communication bus that links the PL part (implemented in the FPGA part) by the PS part and it includes all memories and input-output port.

The AXI-Stream and AXI-Lite (S_AXI_Lite) HLS interfaces support the tasks of streaming the input/output data, and configure convolution parameters, respectively. In our case, AXI-Stream-based interface is used to transfer input, Kernel and output data of the CONV-IP core. While each layer is controlled by the PS through an S-AXI-Lite interface. The CONV-IP is implemented in the PL part and transfers a slave demand to the PS in order to get access to store all the parameters in the memory. Then, the input data size of CONV_1 is $64 \times 64$ and the kernel size used is $4 \times 4$ (16 filters in total). All the processed data pass through the CONV operation to extract the feature maps in a $16 \times 16$ format. For CONV_2, the input image sized to $16 \times 16$, is convoluted with $2 \times 2$ kernels to generate the $8 \times 8$ output data. Additionally, for CONV_3, the input image sized to $8 \times 8$ is convoluted with $2 \times 2$ kernels to extract the $4 \times 4$ output data. Following the above acceleration steps mentioned in the section IV.2.4, each CONV layer is accelerated using Vivado HLS tool and then exported as an RTL core (CONV1_0, CONV2_0, and CONV3_0). Therefore, these generated IP cores will be integrated and used in the Vivado design.

On the other hand, the proposed CNN_1 model contains three fully connected layers with different parameters, as shown in Table IV.1. We accelerate these three fully connected IP cores, as mentioned in Figure IV.9. For each FC_IP, the communication part that connects the PL by the PS part and that includes all memories and input-output access ports, is added. The HLS Interface (AXI-Stream and AXI-Lite Slave (S_AXI_Lite)) is added to the C++ sources, as presented in Figure IV.9. In this study, AXI-Stream-based interface is used to transfer input and output data of the FC IP core. While each layer is controlled by the PS through an S-AXI-Lite interface. Then, all features maps (2048) are concatenated and transferred, through FC computation, to generate the output data of size 64. For FC_2, the input data used is scaled to 64

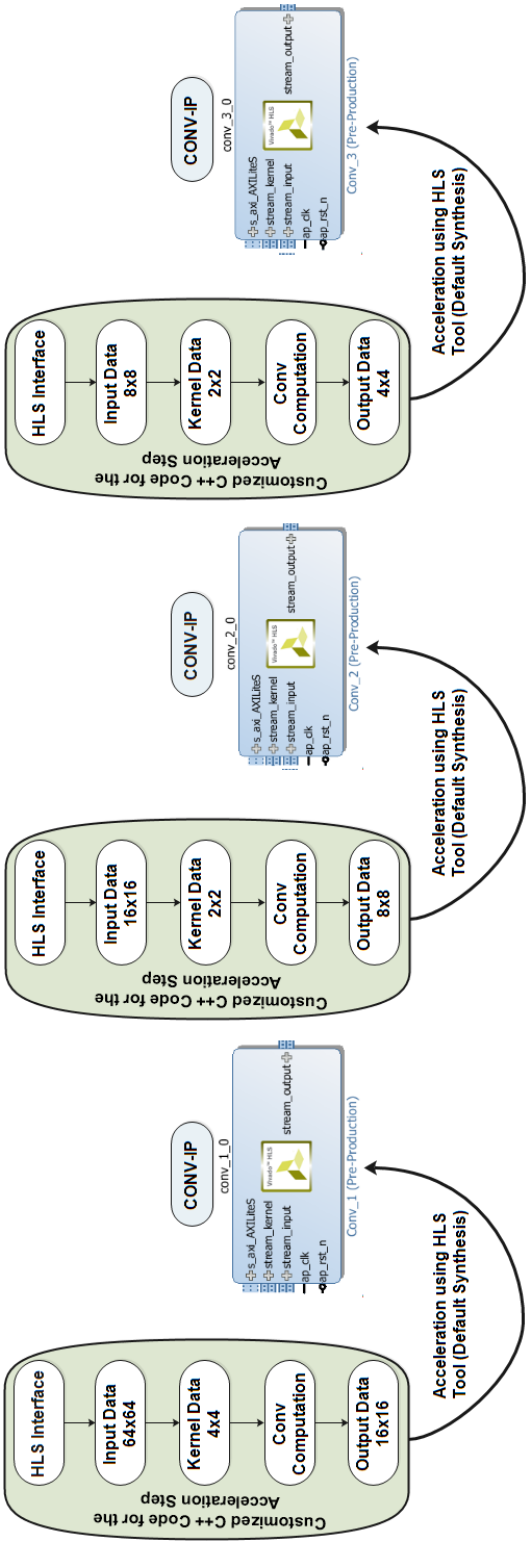



Figure. IV.8: CONV-IP Cores

followed by the FC computation operation to extract the output vector (48). While the FC_3 produces the output layer (1) through the input vector of size 48. Finally,

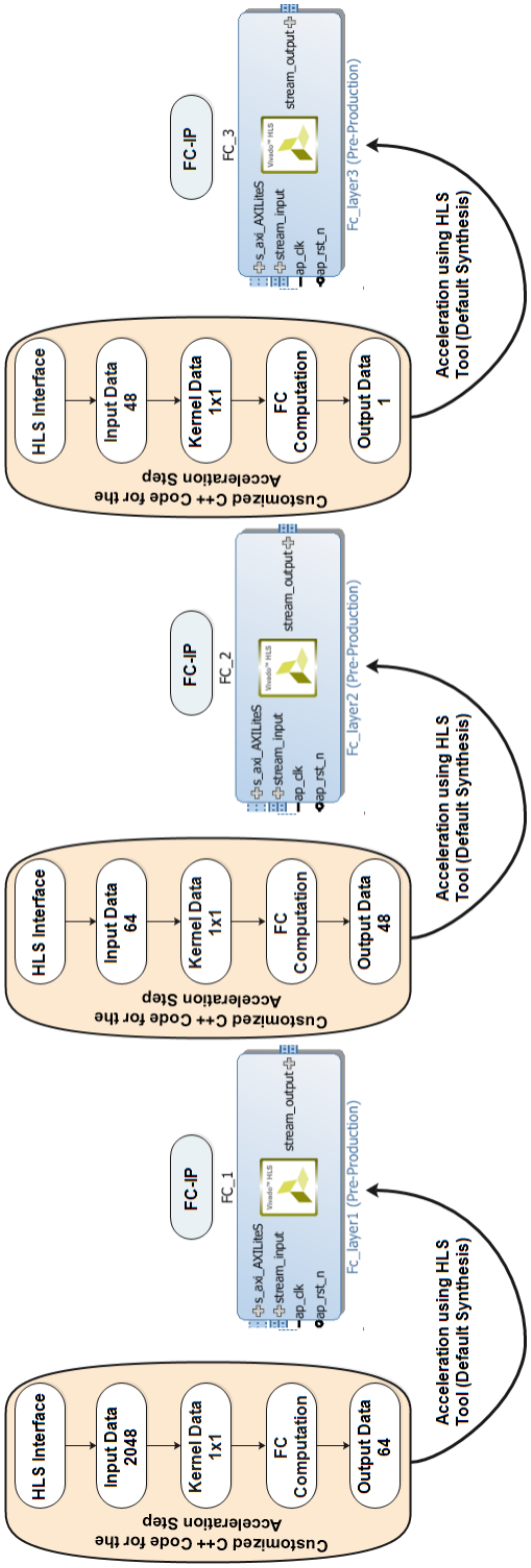



Figure. IV.9: FC-IP Cores

the accelerated FC layer is exported as an RTL IP core (FC_1, FC_2, and FC_3). Therefore, these generated IP cores will be integrated as a new repository to be used in the design.

### IV.3.3 Hardware-Software Co-Design for CNN on FPGA-SoC

Currently, there are two implementation CNN modes due to its hierarchical structure. The first one is the Streaming Architectures and the second is the Single Computation Engine. The former has the ability to allocate corresponding hardware resources to each (network layer) IP core (CONV-IPs and FC-IPs) and it has the following characteristics. Firstly, it can realize the inter-layer pipeline and flexible control and management within each IP core with a high customization degree. Secondly, it can be applied only to small network layers (minimum neurons by each layer), since, it is characterized by its higher demand for resources. The latter indicates that different network layers share the same accelerator (one CNN-IP) through resource reuse, which is a non-highly customized architecture, is more inflexible but it is easier. Considering all these advantages, we designed the system architecture in Streaming Architectures mode, as shown in Figure IV.10.

The CONV-IP and FC-IP accelerators, in the proposed co-design, are imported after the RTL exportation task, in Vivado HLS tool, as new repositories into the Vivado IPI. We can see that our co-design consists of two main components which are the PS part and the PL part. The former includes all tasks of control and data access into memories and AXI interface management. While the later consists of the hardware accelerators which are the CONV-IP and FC-IP cores. This choice aims to increase the design efficiency and flexibility by moving the most computational tasks to FPGA (PL) while keeping control and management to the PS system. As presented by Figure IV.10, the hardware-software architecture consists of many IPs. The PS part includes the external memory Double Data Rate (DDR), the Processor System Reset, and The Processing System (ZYNQ). While the PL part includes the AXI DMA, the AXI Interconnect, the Axi-General Purpose in/output (AXI-GPIO), the CONV-IP cores, and the FC-IP cores.

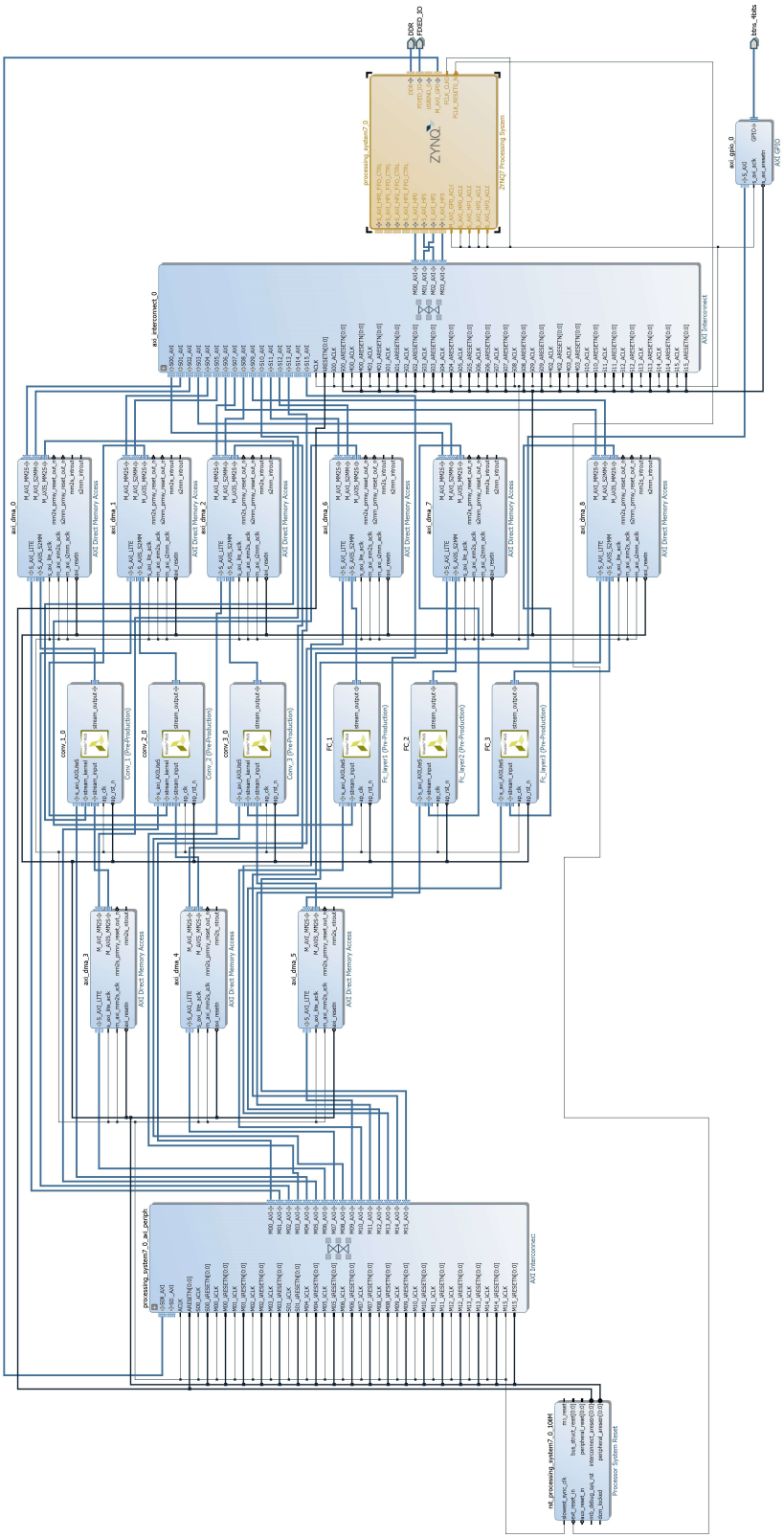


Figure. IV.10: Hardware-Software Co-Design

In this context, the initial processed data and weights are pre-stored in the external memory DDR. PS and PL are interconnected through the AXI bus. All accelerators (CONV-IP and FC-IP cores) receive configuration signals from the PS system through the AXI-Lite bus (e.g., performs standard convolution or depth-wise convolution, stride, convolution kernel size, etc...). Under the action of the DDR controller in the PS, the weight and the input data of the first layer (first CONV-IP core), required by the accelerator, are read from the DDR and are converted from the AXI-memory map format to the AXI-streaming format into the accelerator buffer (stream_input), through the AXI-peripherals Interconnect, under the action of the DMA. In this context, the DMA is used to process the data communication to/from the hardware components. The AXI-peripherals Interconnect has the ability to route transactions between several masters and slaves AXIs, allowing the DMA to communicate. While the AXI-stream interface is used between the DMA and the customized accelerator (CONV-IP core and FC-IP core) to provide data exchange. Moreover, the AXI-Lite is used between the PS system and the DMA to configure the DMA registers. The IP core (CONV-IP or FC-IP core) buffer uses the AXI-stream interface to receive data and weights from the DMA. After the processing task by the accelerator, this IP core will provide outputs that will be sent back to the DDR via the stream_output and AXI-DMA. Thus, this output data, from the current IP core, will be the input for the second layer (second IP core) with the corresponding weight and parameters. That is why each IP core in the co-design includes tow data exchanges which are the input_stream and the output_stream within the AXI-lite for the parameter configurations and the stream_kernel for kernel's values programming (only for CONV-IP cores). The above operation will be repeated until the data processing of the network model is completed. On the other hand, the PS reset core, within the AXI-gpio interfaces, generates a customized reset for the entire system, such as the AXI-Interconnect and peripherals.

## IV.4 Experimental Results

This section introduces the performance evaluation of the hardware accelerators and the full co-design system. The provided results are implemented on Vivado pack v2016.1 within PYNQ-Z1 board (Xilinx Zynq-7000 device).

### IV.4.1 IP Cores Hardware Resource

The three CONV-IP hardware resource occupation of the target FPGA are summarized in Table IV.2.

Table IV.2: Hardware Resource Occupation of the CONV-IP

| IP cores | Resource | BRAM_18k | DSP | FF | LUT |
|---|---|---|---|---|---|
| CONV1-IP | Total | 9 | 5 | 1277 | 1730 |
| | Available | 280 | 220 | 106400 | 53200 |
| | Utilization (%) | 3 | 2 | 1 | 3 |
| CONV2-IP | Total | 2 | 5 | 1443 | 2285 |
| | Available | 280 | 220 | 106400 | 53200 |
| | Utilization (%) | 0 | 1 | 2 | 4 |
| CONV3-IP | Total | 1 | 5 | 1501 | 2280 |
| | Available | 280 | 220 | 106400 | 53200 |
| | Utilization (%) | 0 | 2 | 1 | 4 |

Look-Up-Tables (LUTs), Digital Signal Processors (DSPs), Block RAMs (BRAMs), and Flip-Flops (FFs) are used in the proposed CONV-IP based hardware accelerator. The hardware CONV1-IP core occupies 3% of BRAMs to store the parameters, 3% of LUTs, 2% of DSP, while using only 1% of FFs. In addition, the CONV2-IP uses 1%, 2% and 4% of DSPs, FFs, and LUTs, respectively. For CONV3-IP core, the resources used are 2% of DSP, 1% of FFs, and 4% of LUTs. To sum up, the CONV1-IP presents 9% of hardware cost with a frequency of 150 MHz. While the CONV2-IP and CONV3-IP occupies 7% of hardware cost on the PYNQ-Z1 with a frequency of 150 MHz also. This is considered a low occupancy on the PYNQ-Z1 FPGA with high processing speed.

Table IV.3 shows the summary of FC-IP cores resource occupation. The number of

LUT slices consumed by our three accelerators, including FC1-IP, FC2-IP and FC3-IP is 3%. The hardware cores use 1% of FFs and 2% of DSPs. In addition, 1% of BRAMs are mainly used by the FC1-IP accelerator. These IPs operate at a frequency of 150 MHz.

Table IV.3: Hardware Resource Occupation of the FC-IP

| IP cores | Resource | BRAM_18k | DSP | FF | LUT |
|---|---|---|---|---|---|
| FC1-IP | Total | 4 | 5 | 1151 | 1618 |
| | Available | 280 | 220 | 106400 | 53200 |
| | Utilization (%) | 1 | 2 | 1 | 3 |
| FC2-IP2 | Total | 1 | 5 | 1151 | 1618 |
| | Available | 280 | 220 | 106400 | 53200 |
| | Utilization (%) | 0 | 2 | 1 | 3 |
| FC3-IP | Total | 0 | 5 | 1149 | 1618 |
| | Available | 280 | 220 | 106400 | 53200 |
| | Utilization (%) | 0 | 2 | 1 | 3 |

### IV.4.2 Hardware Cost of the Proposed Co-Design

This section presents the percentage of hardware resources consumed by the proposed co-design on the PYNQ-Z1 platform. After the implementation phase, the hardware resource occupancy of our proposed design is shown in Table IV.4.

Table IV.4: Hardware Cost

| Resource | LUT | LUTRAM | FF | BRAM | DSP | IO | BUFG |
|---|---|---|---|---|---|---|---|
| Total | 25026 | 1374 | 27979 | 22 | 30 | 4 | 1 |
| Available | 53200 | 17400 | 106400 | 140 | 220 | 125 | 32 |
| Utilization (%) | 47 | 8 | 26 | 16 | 14 | 3 | 3 |
| FPGA-Frequency (MHz) | 142 | | | | | | |
| PS7-Frequency (MHz) | 525 | | | | | | |
| Bitstream (Ko) | 3951 | | | | | | |

The resources used are a Global Buffer (BUFG), Input-Output, DSPs, BRAMs, FFs, Look-Up-Table RAM (LUTRAM), and LUT. From Table IV.4, the hardware design occupies 47% of LUTs, 8% of LUTRAM, 26% of FF, 16% of BRAM, 14% of DSP, and 3% of IO and BUFG. In addition, our proposed hardware design reaches 142 MHz working frequency of the FPGA with a 525 MHz of the PS. While the bitstream file represents 3951 Ko.

Figure IV.11 shows the power consumption of the hardware design on FPGA board. According to the power report, the static power of our proposed system is about $0.160W$ and the dynamic power is $1.539W$. However, the total on-chip power is $1.699W$ of the proposed architecture.

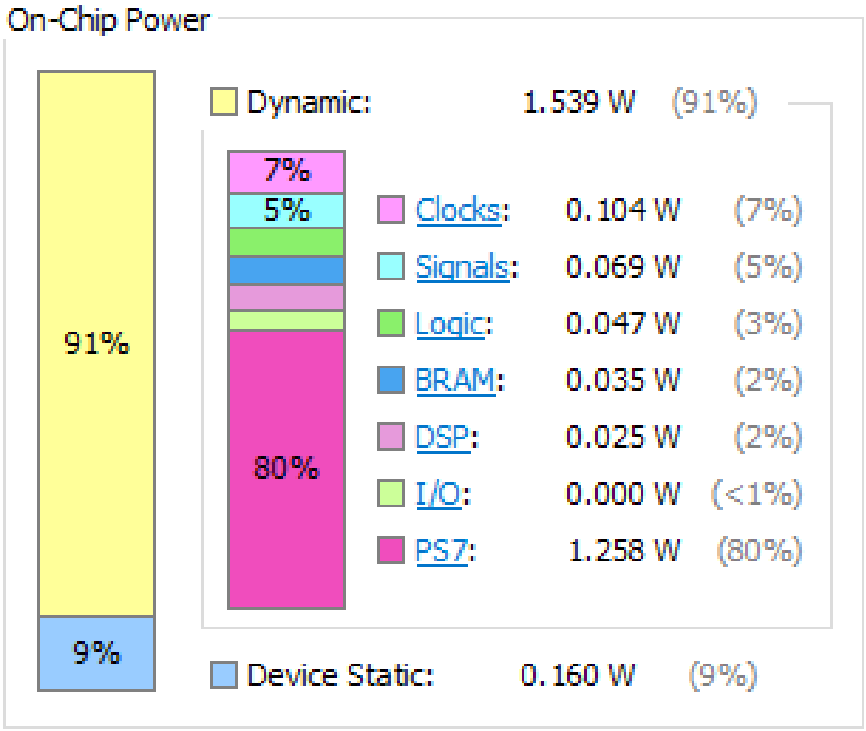



Figure. IV.11: Power Consumption

### IV.4.3 Comparative Study

The proposed design has also been compared to other related works. Table IV.5 shows the performance comparison, in which the relevant references are given. In the work cited in [LCX+19], authors propose a uniform design to accelerate both 2D and 3D

Table IV.5: Comparative Study

| | [LCX+19] | [ZWCL21] | [LZF+19] | Our Design |
|---|---|---|---|---|
| Platform | XC7VX690T | XC7Z035 | 7100 | XC7Z020 |
| Frequency (MHz) | 120 | 200 | 100 | 142 |
| LUTs | 62.9% | 48.4% | 51 | 47% |
| FFs | 50.2% | 31.7% | 38 | 26% |
| BRAMs | 26.6% | 74% | 46 | 16% |
| DSPs | 99.8% | 21.3% | 95 | 14% |
| Power (W) | 15.8 | 5.96 | 3.99 | 1.69 |

CNNs based on Xilinx VC709 board. This proposed method almost entirely uses the hardware resources of the FPGA, consuming almost all the DSPs, over 60% LUTs, 50% FFs, and more than 25% BRAMs. This design achieves an on-chip power consumption of 15.8 W under a 120 MHz working frequency of the FPGA. In reference [ZWCL21], an efficient hardware-implementation method for optical remote sensing object detection was proposed. However, this design has extremely high requirements for resource utilization, consuming more than 70% BRAMs. It achieves 5.96 W on-chip power consumption at a clock frequency of 200 MHz. In addition, the authors in [LZF+19] proposed a CNN accelerator, which accelerates the standard convolution and the depthwise separable convolution. This method has been implemented on the Xilinx ZYNQ 7100 hardware platform which achieves an on-chip power consumption of 3.99W at a clock frequency of 100MHz. Meanwhile, it consumed the highest hardware resources, using almost all DSPs, 50% LUTs, and over 40% BRAMs.

After this comparative study, we remark that our proposed design achieves high performance; low power consumption around $1.69W$ and occupies low hardware FPGAs resources. Therefore, our design can achieve a satisfactory balance between resource cost and power consumption and it is suitable for deployment on embedded devices with limited resource budget.

## IV.5 Conclusion

In this chapter, we have proposed a deep CNN based hardware-software design for HEVC CU prediction on FPGA-SoC. Our proposed work aims to accelerate the CNNs due to their computationally intensive. However, we have created a hardware IP core for each CNNs layer using the Vivado HLS tool. Then, we have designed a hardware-software architecture by importing the hardware IP cores based on the PYNQ-Z1 board. Compared to other designs, our architecture has shown clear advantages in terms of power consumption and hardware resources, which is suitable for deployment on embedded devices with limited resources.

The future work will be explored to accelerate the overall architecture of Deep CNN and implement it in a real-time on PYNQ-Z1 using customized Overlay.

# General Conclusion and Perspectives

The expansion of Internet coupled with the rapid introduction of UHD, HDR and 360° video contents in daily life have caused the explosion of video traffic. Recent study published in Cisco [CCH18] has predicted that video traffic will increase from 75% of the global IP traffic in 2017 to 82% in 2022. This increasing demand for video contents brings new challenges to compression, especially to enhance the coding efficiency and enable a high QoE of video services. Driven from these requirements, various technologies appeared as potential solutions for video coding deployment. In this thesis work, the main concern is reducing computational complexity and improving video quality for HEVC and VVC standards using artificial intelligence.

In this thesis, we have focused on video coding standards, such as HEVC and VVC. Particularly, we tackled the problem of complexity reduction and video quality using machine learning solutions. Four main contributions have been integrated in this work.

Firstly, we have reviewed the basic building blocks of a video compression system. A detailed description of the HEVC and VVC standards has been given, which are the current state-of-the-art video coding standards. Particularly, concepts and challenges related to video coding standard have been introduced. Afterwards, we have provided a background of the emerging video technologies, as artificial intelligence. Moreover, we have listed some related methods on HEVC and VVC complexity reduction.

To reduce the HEVC complexity, chapter II have proposed a fast CU partition algorithms based on machine learning tools. At inter-prediction, we have proposed an online SVM-based fast CU partition for reducing HEVC complexity. This chapter have also provided a deep CNN algorithm to predict CU partition at HEVC inter-mode. According to the achieved results, the latter does not explore the temporal correlation of the CU partition across neighboring frames. For this reason, we have proposed another deep learning approach, which combined the CNN and the LSTM networks. Therefore, HEVC encoding performance were improved, when we replaced classic RDO search with CNN-LSTM network to decide CU splitting in inter-mode. In summary, the proposed scheme saved a significant encoding complexity, compared to other previous approaches-based machine learning tools.

The third chapter have introduced a deep learning technique to improve VVC video quality while enhancing the user's services. To alleviate the coding artifacts, such as ringing, blocking, and blurring, the proposed WSE-DCNN technique has been integrated into VVC standard to replace the traditional in-loop filtering. The proposed VVC filtering technique has been used in the context of smart city based on M-IoT scenarios to contribute to the centralized cloud which attempts to meet the user's required video quality. Compared to the traditional VVC-based filters, the proposed framework has achieved the best compression performance in terms of objective and subjective quality.

Deep learning algorithms, such as CNNs, have significant higher accuracies than traditional algorithms, but they require huge amounts of computational resources and memory access due to the large number of parameters in the layers operation, which represents a computational challenge. Towards this end, Chapter IV have proposed a hardware-software architecture based on CNN accelerator for HEVC CU partition on FPGA-SoC. To achieve this goal, for each CNN layers, an IP cores has been created using Vivado HLS. Then, an hybrid architecture has been designed based on PS, PL, and CNN IP cores using Xilinx Vivado. Compared to others designs, our proposed architecture has obvious advantages in terms of power consumption and hardware resources,

which is suitable for deployment on embedded devices with limited resource.

Despite the advances mentioned throughout this manuscript, the video coding field remains an open topic of research and several improvements can be envisaged. Principal limitations to be considered and the perspectives are presented as follows.

- All the machine learning solutions proposed in Chapter II will be implemented in the VVC standard. This aims to predict the QuadTree plus Multi-type Tree (QTMT)-based CU partition, for drastically accelerating the encoding process of inter-mode VVC.

- Furthermore, the development of a new deep learning-based architecture that merges the reduction of complexity and the improvement of video quality will be subject to future research. Specifically, one algorithm predicts the VVC CU partition at inter prediction for reducing VVC complexity, and another replaces the original VVC filters to enhance visual quality.

- Finally, from another perspective, the overall architecture of Deep CNN proposed in chapter IV will be accelerated and implemented in real-time on PYNQ-Z1 FPGA using a customized OS. We will also consider the implementation of the HEVC or VVC standard with these accelerated algorithms on an embedded platform to have an optimized encoder.

# Bibliography


[AAB+16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[AM05] F Gregory Ashby and W Todd Maddox. Human category learning. *Annu. Rev. Psychol.*, 56:149–178, 2005.

[ARM18] Muhammad Amjad, Mubashir Husain Rehmani, and Shiwen Mao. Wireless multimedia cognitive radio networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 20(2):1056–1103, 2018.

[B+01] Leo Breiman et al. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, 16(3):199–231, 2001.

[B+13] Frank Bossen et al. Common test conditions and software reference configurations. In *JCTVC-L1100*, volume 12, 2013.

[BAH18] Olfa Ben-Ahmed and Benoit Huet. Deep multimodal features for movie genre and interestingness prediction. In *2018 international conference on content-based multimedia indexing (CBMI)*, pages 1–6. IEEE, 2018.

[BBL+19] Frank Bossen, Jill Boyce, X Li, V Seregin, and K Sühring. Jvet common test conditions and software reference configurations for sdr video. *Joint Video Experts Team (JVET) of ITU-T SG*, 16, 2019.

[BCL19] B Bross, J Chen, and S Liu. Jvet-m1001: Versatile video coding (draft 4). In *Joint Video Experts Team (JVET), 14th Meeting: Geneva, SW, Tech. Rep*, 2019.

[BCO+21] Benjamin Bross, Jianle Chen, Jens-Rainer Ohm, Gary J Sullivan, and Ye-Kui Wang. Developments in international video coding standardization after avc, with an overview of versatile video coding (vvc). *Proceedings of the IEEE*, 2021.

[Bjo01] Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. *VCEG-M33*, 2001.

[BKMS21] Soulef Bouaafia, Randa Khemiri, Amna Maraoui, and Fatma Elzahra Sayadi. Cnn-lstm learning approach-based complexity reduction for high-efficiency video coding standard. *Scientific Programming*, 2021, 2021.

[BKS+20] Soulef Bouaafia, Randa Khemiri, Fatma Ezahra Sayadi, Mohamed Atri, and Noureddine Liouane. A deep cnn-lstm framework for fast video coding. In *International Conference on Image and Signal Processing*, pages 205–212. Springer, 2020.

[BKS21] Soulef Bouaafia, Randa Khemiri, and Fatma Ezahra Sayadi. Rate-distortion performance comparison: Vvc vs. hevc. In *2021 18th International Multi-Conference on Systems, Signals & Devices (SSD)*, pages 440–444. IEEE, 2021.

[BKSA20a] Soulef Bouaafia, Randa Khemiri, Fatma Ezahra Sayadi, and Mohamed Atri. Fast cu partition-based machine learning approach for reducing hevc complexity. *Journal of Real-Time Image Processing*, 17(1):185–196, 2020.

[BKSA20b] Soulef Bouaafia, Randa Khemiri, Fatma Ezahra Sayadi, and Mohamed Atri. Svm-based inter prediction mode decision for hevc. In *2020 17th International Multi-Conference on Systems, Signals & Devices (SSD)*, pages 12–16. IEEE, 2020.

[CAAdSC14] Guilherme Correa, Pedro A Assuncao, Luciano Volcan Agostini, and Luis A da Silva Cruz. Fast hevc encoding decisions using data mining. *IEEE transactions on circuits and systems for video technology*, 25(4):660–673, 2014.

[CCH18] S Cicero, C Cromwell, and E Hunt. Cisco predicts more ip traffic in the next five years than in the history of the internet. *URL: https://newsroom.cisco.com/press-release-content?type=webcontent&articleId=1955935*, 2018.

[CCWL20] Sijia Chen, Zhenzhong Chen, Yingbin Wang, and Shan Liu. In-loop filter with dense residual convolutional neural network for vvc. In *2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 149–152. IEEE, 2020.

[CEES14] Louise Helen Crockett, Ross Elliot, Martin Enderwitz, and Robert Stewart. *The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc.* Strathclyde Academic Media, 2014.

[CJH16] Yang Cao, Tao Jiang, and Zhu Han. A survey of emerging m2m systems: Context, task, and objective. *IEEE Internet of Things Journal*, 3(6):1246–1258, 2016.

[CK13] Seunghyun Cho and Munchurl Kim. Fast cu splitting and pruning for suboptimal cu partitioning in hevc intra coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(9):1555–1564, 2013.

[CKL06] Jian-Wen Chen, Chao-Yang Kao, and Youn-Long Lin. Introduction to h. 264 advanced video coding. In *Proceedings of the 2006 Asia and South Pacific Design Automation Conference*, pages 736–741, 2006.

[CMMC19] Gabriel Cebrian-Marquez, José Luis Martínez, and Pedro Cuenca. Adaptive inter cu partitioning based on a look-ahead stage for hevc. *Signal Processing: Image Communication*, 76:97–108, 2019.

[CSZ09] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[Da14] Kingma Da. A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[DB17] Kajaree Das and Rabi Narayan Behera. A survey on machine learning: concept, algorithms and applications. *International Journal of Innovative Research in Computer and Communication Engineering*, 5(2):1301–1309, 2017.

[DIV18] Div2k. *URL: https://data.vision.ee.ethz.ch/cvl/DIV2K/.*, 2018.

[FAA+12] Chih-Ming Fu, Elena Alshina, Alexander Alshin, Yu-Wen Huang, Ching-Yeh Chen, Chia-Yang Tsai, Chih-Wei Hsu, Shaw-Min Lei, Jeong-Hoon Park, and Woo-Jin Han. Sample adaptive offset in the hevc standard. *IEEE Transactions on Circuits and Systems for Video technology*, 22(12):1755–1764, 2012.

[FJK+20] Bossen Frank, Boyce Jill, Suehring Karsten, Li Xiang, and Seregin Vadim. Vtm common test conditions and software reference configurations for sdr video. *document JVET-T2010*, 2020.

[FSK+20] Yibo Fan, Heming Sun, Jiro Katto, Jing Ming'E, et al. A fast qtmt partition decision strategy for vvc intra prediction. *IEEE Access*, 8:107900–107911, 2020.

[Fuk88] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

[Ful08] John Fulcher. Computational intelligence: an introduction. In *Computational intelligence: a compendium*, pages 3–78. Springer, 2008.

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[HALK21] JiUn Hong, Saad Arslan, TaeGeon Lee, and HyungWon Kim. Design of power-efficient training accelerator for convolution neural networks. *Electronics*, 10(7):787, 2021.

[HBA+21] Wassim Hamidouche, Thibaud Biatek, Mohsen Abdoli, Edouard François, Fernando Pescador, Miloš Radosavljević, Daniel Menard, and Mickael Raulet. Versatile video coding standard: A review from coding tools to consumers deployment. *arXiv preprint arXiv:2106.14245*, 2021.

[HCP+18] Shujun Huang, Nianguang Cai, Pedro Penzuti Pacheco, Shavira Narrandes, Yang Wang, and Wayne Xu. Applications of support vector machine (svm) learning in cancer genomics. *Cancer Genomics-Proteomics*, 15(1):41–51, 2018.

[HLS20] Zhijie Huang, Yunchang Li, and Jun Sun. Multi-gradient convolutional neural network based in-loop filter for vvc. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2020.

[HSS18] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. the elements of statistical learning (pp. 485-585), 2009.

[JLLL19] Wei Jia, Li Li, Zhu Li, and Shan Liu. Residue guided loop filter for hevc post processing. *arXiv preprint arXiv:1907.12681*, 2019.

[Joh14] Jeff Johnson. Using the axi dma in vivado, 2014.

[KK19] Naito S Kawamura K, Kidani Y. Ce13-2.6/ce13-2.7: Evaluation results of cnn based in-loop filtering. In *Tech. Rep. JVET meeting, no. JVET-N0710. ITU-T, ISO/IEC, ITU-R*, 2019.

[KLM96] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[KLP20] Minseon Kang, Yongseok Lee, and Moonju Park. Energy efficiency of machine learning in embedded systems using neuromorphic hardware. *Electronics*, 9(7):1069, 2020.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[KZP07] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LCX+19] Zhiqiang Liu, Paul Chow, Jinwei Xu, Jingfei Jiang, Yong Dou, and Jie Zhou. A uniform architecture design for accelerating 2d and 3d cnns on fpgas. *Electronics*, 8(1):65, 2019.

[LLYY15] Yuchen Li, Yitong Liu, Hongwen Yang, and Dacheng Yang. Fast quadtree structure decision for hevc intra coding using histogram statistics. *KSII Transactions on Internet and Information Systems (TIIS)*, 9(5):1825–1839, 2015.

[LXT+21] Tianyi Li, Mai Xu, Runzhi Tang, Ying Chen, and Qunliang Xing. Deep-qtmt: A deep learning approach for fast qtmt-based cu partition of intra-mode vvc. *IEEE Transactions on Image Processing*, 2021.

[LZF+19] Bing Liu, Danyin Zou, Lei Feng, Shou Feng, Ping Fu, and Junbao Li. An fpga-based cnn accelerator integrating depthwise separable convolution. *Electronics*, 8(3):281, 2019.

[LZZ+19] Na Li, Yun Zhang, Linwei Zhu, Wenhan Luo, and Sam Kwong. Reinforcement learning based coding unit early termination algorithm for high efficiency video coding. *Journal of Visual Communication and Image Representation*, 60:276–286, 2019.

[MAS18] Amir Ahmad Mohammadi, Mohammad Sadegh Alizadeh, and Mohammad Sharifkhani. Jvet encoder complexity analysis. In *ISCAS*, pages 1–5, 2018.

[MBA+20] Seifeddine Messaoud, Abbas Bradai, Olfa Ben Ahmed, Pham Quang, M Atri, and M Shamim Hossain. Deep federated q-learning-based network slicing for industrial iot. *IEEE Transactions on Industrial Informatics*, 2020.

[MBB+20] Seifeddine Messaoud, Abbas Bradai, Syed Hashim Raza Bukhari, Pham Tran Anh Qung, Olfa Ben Ahmed, and Mohamed Atri. A survey on machine learning in internet of things: Algorithms, strategies, and applications. *Internet of Things*, page 100314, 2020.

[MBW+19] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre GR Day, Clint Richardson, Charles K Fisher, and David J Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Physics reports*, 810:1–124, 2019.

[MC19] Hui Ma and Turgay Celik. Fer-net: facial expression recognition using densely connected convolutional network. *Electronics Letters*, 55(4):184–186, 2019.

[ML94] C Montgomery and H Lars. Xiph. org video test media (derf's collection). *URL: https://media. xiph. org/video/derf*, 1994.

[MMC13] RS Mitchell, JG Michalski, and TM Carbonell. *An artificial intelligence approach.* Springer, 2013.

[MMS+21] Alexandre Mercat, Arttu Mäkinen, Joose Sainio, Ari Lemmetti, Marko Viitanen, and Jarno Vanne. Comparative rate-distortion-complexity analysis of vvc and hevc video codecs. *IEEE Access*, 9:67813–67828, 2021.

[Mod] High Efficiency Video Coding Test Model. 16.5 https://hevc.hhi.fraunhofer.de/svn/svn hevcsoftware.

[MWZ+15] Felix Mercer Moss, Ke Wang, Fan Zhang, Roland Baddeley, and David R Bull. On the optimal presentation duration for subjective video quality assessment. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(11):1977–1987, 2015.

[MZB20] Di Ma, Fan Zhang, and David R Bull. Bvi-dvc: a training database for deep video compression. *arXiv preprint arXiv:2003.13552*, 2020.

[NBF+12] Andrey Norkin, Gisle Bjontegaard, Arild Fuldseth, Matthias Narroschke, Masaru Ikeda, Kenneth Andersson, Minhua Zhou, and Geert Van der Auwera. Hevc deblocking filter. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1746–1754, 2012.

[NH10] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[NQA+20] Ali Nauman, Yazdan Ahmad Qadri, Muhammad Amjad, Yousaf Bin Zikria, Muhammad Khalil Afzal, and Sung Won Kim. Multimedia internet of things: A comprehensive survey. *IEEE Access*, 8:8202–8250, 2020.

[OS11] JR Ohm and G Sullivan. Vision, applications and requirements for high efficiency video coding (hevc). *document N11872*, 2011.

[OSS+12] Jens-Rainer Ohm, Gary J Sullivan, Heiko Schwarz, Thiow Keng Tan, and Thomas Wiegand. Comparison of the coding efficiency of video coding standards—including high efficiency video coding (hevc). *IEEE Transactions on circuits and systems for video technology*, 22(12):1669–1684, 2012.

[OW83] Roger J Osborne and Merlin C Wittrock. Learning science: A generative process. *Science education*, 67(4):489–508, 1983.

[PJ19] Saroj Kumar Pandey and Rekh Ram Janghel. Recent deep learning techniques, challenges and its applications for medical healthcare system: A review. *Neural Processing Letters*, 50(2):1907–1935, 2019.

[PK19] Sang-Hyo Park and Je-Won Kang. Context-based ternary tree decision method in versatile video coding for fast intra coding. *IEEE Access*, 7:172597–172605, 2019.

[PV17] Debajyoti Pal and Vajirasak Vanijja. A no-reference modular video quality prediction model for h. 265/hevc and vp9 codecs on a mobile device. *Advances in Multimedia*, 2017, 2017.

[PYN19] DIGILENT PYNQ. Python productivity for zynq (pynq) documentation release 2.2, 2019.

[Ric13] I Richardson. An introduction to high efficiency video coding. *V Codex Ltd. White Paper*, 2013.

[RN16] Stuart J Russell and Peter Norvig. Artificial intelligence: a modern approach. malaysia, 2016.

[RZ85] David E Rumelhart and David Zipser. Feature discovery by competitive learning. *Cognitive science*, 9(1):75–112, 1985.

[SOHW12] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Trans-*

*actions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

[SS94] David R Shanks and Mark F STJOHN. Characteristics of dissociable human learning-systems. *Behav Brain Sci*, 17(3):367–395, 1994.

[TCY+13] Chia-Yang Tsai, Ching-Yeh Chen, Tomoo Yamakage, In Suk Chong, Yu-Wen Huang, Chih-Ming Fu, Takayuki Itoh, Takashi Watanabe, Takeshi Chujoh, Marta Karczewicz, et al. Adaptive loop filtering for video coding. *IEEE Journal of Selected Topics in Signal Processing*, 7(6):934–945, 2013.

[THM+21] Alexandre Tissier, Wassim Hamidouche, Souhaiel Belhadj Dit Mdalsi, Jarno Vanne, Franck Galpin, and Daniel Menard. Machine learning based efficient qt-mtt partitioning scheme for vvc intra encoders. *arXiv preprint arXiv:2103.05319*, 2021.

[TTAA19] Muhammad Tahir, Imtiaz A Taj, Pedro A Assuncao, and Muhammad Asif. Fast video encoding based on random forests. *Journal of Real-Time Image Processing*, pages 1–21, 2019.

[VTM20] Vtm 4.0 software, available at:. *URL: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/tree/VTM-4.0*, 2020.

[WBAK20] M Arif Wani, Farooq Ahmad Bhat, Saduf Afzal, and Asif Iqbal Khan. *Advances in deep learning*. Springer, 2020.

[WHB+20] Adam Wieckowski, Gabriel Hege, Christian Bartnik, Christian Lehmann, Christian Stoffers, Benjamin Bross, and Detlev Marpe. Towards a live software decoder implementation for the upcoming versatile video coding (vvc) codec. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3124–3128. IEEE, 2020.

[Wie15] Mathias Wien. High efficiency video coding. *Coding Tools and specification*, 24, 2015.

[Wit74] Merlin C Wittrock. Learning as a generative process. *Educational psychologist*, 11(2):87–95, 1974.

[WLM+16] Shanshe Wang, Falei Luo, Siwei Ma, Xiang Zhang, Shiqi Wang, Debin Zhao, and Wen Gao. Low complexity encoder optimization for hevc. *Journal of Visual Communication and Image Representation*, 35:120–131, 2016.

[WWG+19] S Wan, MZ Wang, H Gong, CY Zou, YZ Ma, JY Huo, YF Yu, and Y Liu. Ce10: Integrated in-loop filter based on cnn (tests 2.1, 2.2 and 2.3). In *Tech. Rep. JVET meeting, no. JVET-O0079. ITU-T, ISO/IEC, ITU-R*, 2019.

[XDLW14] Mai Xu, Xin Deng, Shengxi Li, and Zulin Wang. Region-of-interest based conversational hevc coding with hierarchical perception model of face. *IEEE Journal of Selected Topics in Signal Processing*, 8(3):475–489, 2014.

[Xil14] VH Xilinx. Vivado design suite user guide-high-level synthesis, 2014.

[Xil19a] Xilinx. Axi dma controller. *URL https://www.xilinx.com/products/intellectual-property/axi_dma.html*, 2019.

[Xil19b] PYNQ Xilinx. Python productivity for zynq. *URL http://www. pynq. io*, 2019.

[XLW+18] Mai Xu, Tianyi Li, Zulin Wang, Xin Deng, Ren Yang, and Zhenyu Guan. Reducing complexity of hevc: A deep learning approach. *IEEE Transactions on Image Processing*, 27(10):5044–5059, 2018.

[XLWM13] Jian Xiong, Hongliang Li, Qingbo Wu, and Fanman Meng. A fast hevc inter cu selection method based on pyramid motion divergence. *IEEE transactions on multimedia*, 16(2):559–564, 2013.

[YFY+18] Jiahui Yu, Yuchen Fan, Jianchao Yang, Ning Xu, Zhaowen Wang, Xinchao Wang, and Thomas Huang. Wide activation for efficient and accurate image super-resolution. *arXiv preprint arXiv:1808.08718*, 2018.

[ZKH+19] Yousaf Bin Zikria, Sung Won Kim, Oliver Hahm, Muhammad Khalil Afzal, and Mohammed Y Aalsalem. Internet of things (iot) operating systems management: Opportunities, challenges, and solution, 2019.

[ZKW+15] Yun Zhang, Sam Kwong, Xu Wang, Hui Yuan, Zhaoqing Pan, and Long Xu. Machine learning-based coding unit depth decisions for flexible complexity allocation in high efficiency video coding. *IEEE Transactions on Image Processing*, 24(7):2225–2238, 2015.

[ZWCL21] Ning Zhang, Xin Wei, He Chen, and Wenchao Liu. Fpga implementation for cnn-based optical remote sensing object detection. *Electronics*, 10(3):282, 2021.

[ZWH+21] Qiuwen Zhang, Yihan Wang, Lixun Huang, Bin Jiang, and Xiao Wang. Fast cu partition decision for h. 266/vvc based on the improved dag-svm classifier model. *Multimedia Systems*, 27(1):1–14, 2021.

[ZZK+17] Linwei Zhu, Yun Zhang, Sam Kwong, Xu Wang, and Tiesong Zhao. Fuzzy svm-based coding unit decision in hevc. *IEEE Transactions on Broadcasting*, 64(3):681–694, 2017.

[ZZP+17] Linwei Zhu, Yun Zhang, Zhaoqing Pan, Ran Wang, Sam Kwong, and Zongju Peng. Binary and multi-class learning based low complexity optimization for hevc encoding. *IEEE Transactions on Broadcasting*, 63(3):547–561, 2017.

# List of Publications

### Journal Articles

**Bouaafia Soulef**, Randa Khemiri, Amna Maraoui, and Fatma Elzahra Sayadi. Cnn-lstm learning approach-based complexity reduction for high-efficiency video coding standard. *Scientific Programming*, Vol. 2021, 2021.

**Bouaafia Soulef**, Randa Khemiri, Seifeddine Messaoud, Olfa Ben Ahmed, and Fatma Ezahra Sayadi. Deep learning-based video quality enhancement for the new versatile video coding. *Neural Computing and Applications*, pages 1–15, 2021.

**Bouaafia Soulef**, Randa Khemiri, Seifeddine Messaoud, and Fatma Elzahra Sayadi. Complexity analysis of new future video coding (fvc) standard technology. *International Journal of Digital Multimedia Broadcasting*, 2021, 2021.

**Bouaafia Soulef**, Randa Khemiri, Fatma Ezahra Sayadi, and Mohamed Atri. Fast cu partition-based machine learning approach for reducing hevc complexity. *Journal of Real-Time Image Processing*, 17(1):185–196, 2020.

**Bouaafia Soulef**, Seifeddine Messaoud, Randa Khemiri, and Fatma Elzahra Sayadi. Vvc in-loop filtering based on deep convolutional neural network. *Computational Intelligence and Neuroscience*, Vol. 2021, 2021.

### International Conferences

**Bouaafia Soulef**, Randa Khemiri, and Fatma Ezahra Sayadi. Rate-distortion performance comparison: Vvc vs. hevc. In *2021 18th International Multi-Conference on Systems, Signals & Devices (SSD)*, pages 440–444. IEEE, 2021.

**Bouaafia Soulef**, Randa Khemiri, Fatma Ezahra Sayadi, and Mohamed Atri. Svm-based inter prediction mode decision for hevc. In *2020 17th International Multi-Conference on Systems, Signals & Devices (SSD)*, pages 12–16. IEEE, 2020.

### Book Chapters

**Bouaafia Soulef**, Randa Khemiri, Fatma Ezahra Sayadi, Mohamed Atri, and Noureddine Liouane. A deep cnn-lstm framework for fast video coding. In *International Conference on Image and Signal Processing*, pages 205–212. Springer, 2020.

# Appendix A

# Convolutional Neural Networks

CNNs have achieved remarkable success in the fields of image processing and computer vision [BKSA20a, BKMS21]. Indeed, the key terminology and operations involved in CNNs architecture, including convolution, normalization, pooling, activation functions, and fully connected layers have been introduced [PJ19].

## A.1 Convolutional Layer

Convolutional Layers are the basic building blocks, which are the most computationally intensive. These are the first layers of the CNN architecture. The convolution operation is used to create a feature extraction map from an input image. Yet, a convolution operation is the multiplication and accumulation of input feature maps $I$ with shape $C \times H \times W$ and convolution kernels (W) of size $k \times k$, to yield an output feature map O with shape $M \times E \times F$. The convolution operation is calculated by equation A.1. Figure A.1 shows an example of the convolution operation.

$$\mathbf{O}[m][e][f] = \sum_{c=1}^{C} \sum_{i=1}^{K} \sum_{j=1}^{K} \mathbf{I}[C][e \times S + i][f \times S + j] \times \mathbf{W}[m][C][i][j] \tag{A.1}$$

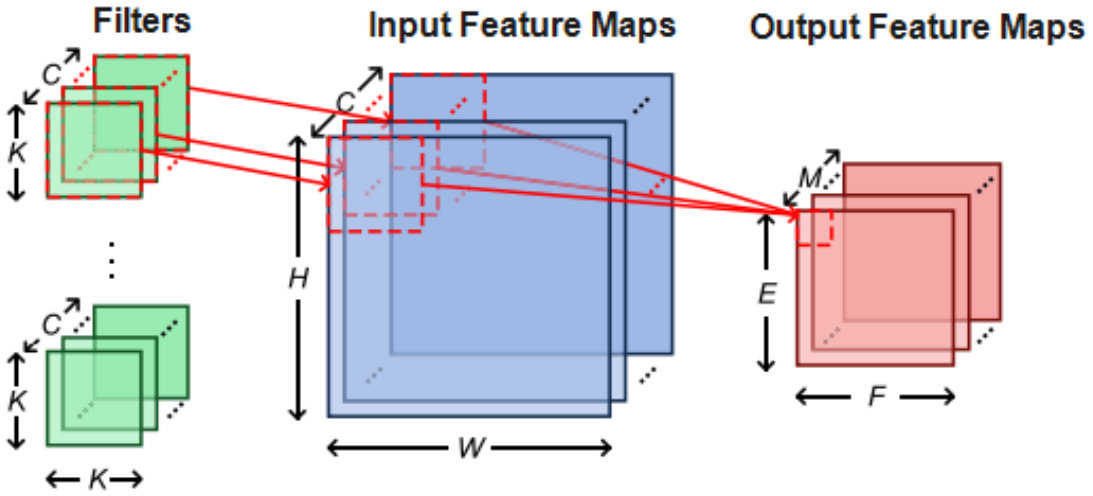



Figure. A.1: Convolution Operation

## A.2 Activation Functions

The commonly used activation function is the Rectified Linear Unit (ReLU) which clips all negative values to zero. It is a non-linear activation function used after the convolutional layer, which can be defined as.

$$f(x) = max(0, x) \tag{A.2}$$

where the weighted sum of the neuron inputs denoted by $x$. The goal of *ReLU* is to converge faster in training and has low computational complexity compared to other functions. Besides, *sigmoid* and *tanh* functions are another popular activation function, which can be given as.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{A.3}$$

$$f(x) = tanh(x) \tag{A.4}$$

Figure A.2 shows the non-linear activation functions.

## A.3 Pooling Layers

Pooling Layers reduces the spatial dimensions (sub-sampling) of the feature maps. The number of parameters and computation in CNNs are minimized depending on the Pool-

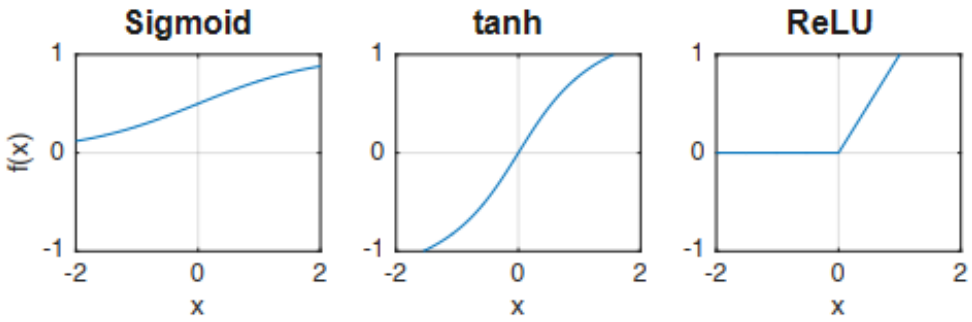



Figure. A.2: The Non-Linear Activation Functions

ing layer. Typically, it is applied after the convolution layers. Indeed, the most popular forms used are avg-pooling and max-pooling. Another pooling units are used in some CNNs, like MIN, and SUM operations. Figure A.3 represents the max and avg pooling layers.

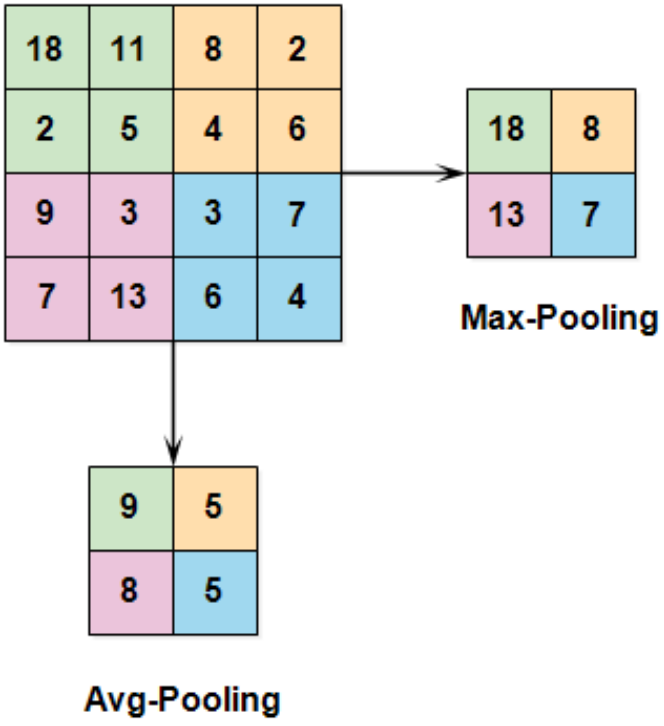



Figure. A.3: Max vs Average pooling

## A.4 Fully Connected Layer

Fully Connected Layers are typically employed as the last layers of a CNN, also known as inner-product layers. Yet, it is considered to be the classification layer where all input features are connected to all output features via synaptic weights, as shown in Figure A.4. The outputs of the fully connected layer pass to the next layer via the ReLU-based activation function or directly to a Softmax activation function which converts them to probability in the range (0, 1). The last layer of accuracy compares the upper probability labels of the softmax layer with the actual label and provides the accuracy of the CNN network.

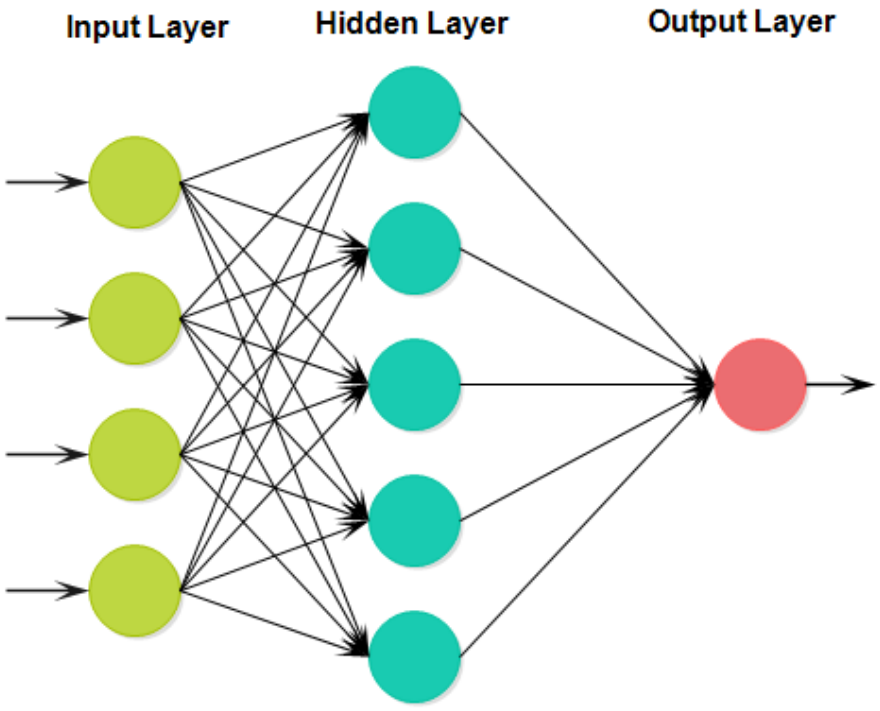



Figure. A.4: Fully Connected Layer

## A.5 Backpropagation Algorithm

The backpropagation algorithm is resumed using four equations. In order to see this, we must first establish some useful notation. We will assume that there are $L$ layers in the network with $l = 1, ..., L$ indexing the layer. Denote by $w^l_{jk}$ the weight for the connection from the $kth$ neuron in layer $l-1$ to the $jth$ neuron in layer $l$. The bias

of this neuron is denoted by $b_j^l$. By construction, in a feed-forward neural network the activation $a_j^l$ of the $jth$ neuron in the $lth$ layer can be related to the activities of the neurons in the layer $l-1$ by the equation A.5.

$$a_{\mathrm{j}}^{\mathrm{l}} = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right) = \sigma\left(z_{\mathrm{j}}^{\mathrm{l}}\right), \tag{A.5}$$

where the linear weighted sum is defined in equation A.6.

$$\mathrm{z}_{\mathrm{j}}^{\mathrm{l}} = \sum_k w_{jk}^l a_k^{l-1} + b_j^l \tag{A.6}$$

By definition, the cost function $E$ depends directly on the activities of the output layer $a_j^l$. It of course also indirectly depends on all the activities of neurons in lower layers in the neural network through iteration of Eq A.5. Let us define the error $\Delta_j^L$ of the $jth$ neuron in the $Lth$ layer as the change in cost function with respect to the weighted input $a_j^L$.

$$\Delta_{\mathrm{j}}^{\mathrm{L}} = \frac{\partial \mathrm{E}}{\partial \mathrm{z}_{\mathrm{j}}^{\mathrm{L}}} \tag{A.7}$$

This definition is the first of the four backpropagation equations. After that, we can analogously define the error of neuron $j$ in layer $l$, $\Delta_j^l$, as the change in the cost function with respect to the weighted input $z_j^l$:

$$\Delta_{\mathrm{j}}^{l} = \frac{\partial \mathrm{E}}{\partial \mathrm{z}_{\mathrm{j}}^{l}} = \frac{\partial \mathrm{E}}{\partial \mathrm{a}_{\mathrm{j}}^{l}} \sigma'(\mathrm{z}_{\mathrm{j}}^{\mathrm{l}}) \tag{A.8}$$

where $\sigma'(\mathrm{x})$ denotes the derivative of the non-linearity $\sigma(.)$ with respect to its input evaluated at $x$. Notice that the error function $\Delta_j^l$ can also be interpreted as the partial derivative of the cost function with respect to the bias $b_j^l$, since

$$\Delta_{\mathrm{j}}^{\mathrm{l}} = \frac{\partial \mathrm{E}}{\partial \mathrm{z}_{\mathrm{j}}^{l}} = \frac{\partial \mathrm{E}}{\partial \mathrm{b}_{\mathrm{j}}^{l}} \frac{\partial \mathrm{b}_{\mathrm{j}}^{l}}{\partial \mathrm{z}_{\mathrm{j}}^{l}} = \frac{\partial \mathrm{E}}{\partial \mathrm{b}_{\mathrm{j}}^{l}} \tag{A.9}$$

where in the last line we have used the fact that $\frac{\partial \mathrm{b}_{\mathrm{j}}^{l}}{\partial \mathrm{z}_{\mathrm{j}}^{l}} = 1$. This is the second of the four backpropagation equations. We now derive the final two backpropagation equations using the chain rule. Since the error depends on neurons in layer $l$ only through the

activation of neurons in the subsequent layer $l+1$, we can use the chain rule to write:

$$\Delta^{\mathrm{l}}_{\mathrm{j}}=\frac{\partial \mathrm{E}}{\partial \mathrm{z}^{l}_{\mathrm{j}}}=\sum_{\mathrm{k}}\frac{\partial \mathrm{E}}{\partial \mathrm{z}^{l+1}_{\mathrm{k}}}\frac{\partial \mathrm{z}^{l+1}_{\mathrm{k}}}{\partial \mathrm{z}^{l}_{\mathrm{j}}}=\sum_{\mathrm{k}}\Delta^{\mathrm{l+1}}_{\mathrm{k}}\frac{\partial \mathrm{z}^{l+1}_{\mathrm{k}}}{\partial \mathrm{z}^{l}_{\mathrm{j}}}=\left(\sum_{\mathrm{k}}\Delta^{\mathrm{l+1}}_{\mathrm{k}}\mathrm{w}^{\mathrm{l+1}}_{\mathrm{kj}}\right)\sigma'(\mathrm{z}^{l}_{\mathrm{j}}) \tag{A.10}$$

This is the third backpropagation equation A.10. The final equation can be derived by differentiating of the cost function with respect to the weight $w^l_{jk}$ as defined in A.11.

$$\frac{\partial \mathrm{E}}{\partial w^{l}_{\mathrm{jk}}}=\frac{\partial \mathrm{E}}{\partial \mathrm{z}^{l}_{\mathrm{j}}}\frac{\partial \mathrm{z}^{l}_{\mathrm{j}}}{\partial w^{l}_{\mathrm{jk}}}=\Delta^{\mathrm{l}}_{\mathrm{j}}\mathrm{a}^{l-1}_{\mathrm{k}} \tag{A.11}$$

The four backpropagation equations are defined by the equations A.8, A.9, A.10, and A.11, relating the gradients of the activations of various neurons $a^l_j$, the weighted inputs $z^l_j$ in A.6, and the errors $\Delta^l_j$. This algorithm consists of a forward pass from the bottom layer to the top layer where one calculates the weighted inputs and activations of all the neurons. One then backpropagates the error starting with the top layer down to the input layer and uses these errors to calculate the desired gradients. This description makes clear the incredible utility and computational efficiency of the backpropagation algorithm. We can calculate all the derivatives using a single "forward" and "backward" pass of the neural network. This computational efficiency is crucial since we must calculate the gradient with respect to all parameters of the neural network at each step of gradient descent.