POZNAN UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTING AND TELECOMMUNICATIONS
INSTITUTE OF MULTIMEDIA TELECOMMUNICATION

Doctoral Dissertation

Block partitioning in video encoding with the use of artificial neural network

Podział na bloki w kodowaniu wizji z wykorzystaniem sztucznych sieci neuronowych

Mateusz Lorkiewicz

Supervisors: prof. dr hab. inż. Marek Domański
dr hab. inż. Olgierd Stankiewicz, prof. PP

Table of contents

Abstract

This dissertation explores the field of video encoding, where moving images are compressed for to be efficiently transmitted in television systems, Video-On-Demand platforms, and similar applications. The research presented in this dissertation is focused on designing video encoder control algorithms. Special emphasis is put on CTU (Coding Tree Unit) partitioning, the most computationally intensive part of the encoding process. The research aims to develop a partitioning algorithm that significantly reduces the computational complexity of the encoder while preserving the compression efficiency, compared to the existing solution in reference encoder. Thus, the research explores Artificial Neural Network (ANN)-based approaches for partitioning algorithms.

The dissertation begins with a description of CTU partitioning in HEVC, along with the contextuality of the decisions. A survey of the existing partitioning algorithm is presented, pinpointing the aspects that require improvements. Methods that offer control over the trade-off between coding time and coding efficiency are reviewed, pointing out the complexity of controlling this trade-off in existing solutions.

Two approaches to the partitioning problem are explored: one for decision-making at Coding Unit (CU) scope and one for joint Coding Unit (CU) scope and Prediction Unit (PU) scope. ANN with non-trivial decision algorithms is introduced. Proposed ANNs are designed to jointly estimate depth-level probabilities for individual CTU subareas. A custom training dataset has been prepared to train the ANN models. The ANN architecture development process is described, and a detailed training and evaluation results analysis of the final models is presented.

This dissertation proposes original, non-trivial decision algorithms that utilize probabilities determined by the ANN. These decision algorithms are defined in two variants: hard-decisive and soft-decisive. It is demonstrated that one of the proposed algorithms allows a straightforward control of the trade-off between coding time and coding efficiency through a single parameter.

Comparative analysis with state-of-the-art solutions demonstrates that the proposed partitioning algorithms offer the best trade-off between encoding time reduction and coding efficiency.

Considering control over the trade-off between encoding time and coding efficiency, the proposed method provides the best results and the most straightforward control among other state-of-the-art solutions. Additionally, a new metric for rapid comparison of such methods is introduced, which coincides with well-established evaluation approaches.

Lastly, this dissertation explores the impact of contextual effects on partitioning decisions. The key achievements are:

- The author's experiment on determining the impact of the encoding context on decisions.
- Modifications to the proposed ANN model for processing contextual information.
- The author's method of training an ANN with augmented ground truth data.
- The author's method for global partitioning patterns optimization which introduces a negligible increase in encoding time.

All experiments were conducted using a modified version of HEVC reference model software developed by the author. This modification, which enables the rapid implementation of ANN-based partitioning algorithms, has been released under open-access terms.

Streszczenie

Podział na bloki w kodowaniu wizji z wykorzystaniem sztucznych sieci neuronowych

W niniejszej rozprawie przedstawiono wyniki badań w dziedzinie kodowania wideo, gdzie obrazy tworzące sekwencję wizyjną są poddawane kompresji, co wykorzystywane jest w systemach telewizyjnych, platformach VoD (Video-on-Demand) udostępniających filmy na żądanie i podobnych zastosowaniach. Badania przedstawione w tej rozprawie skupiają się na rozwoju algorytmów sterowania koderem. Szczególny nacisk położono na wyznaczanie podziałów bloku CTU (Coding Tree Unit), co jest najbardziej złożonym obliczeniowo etapem kodowania. Celem badań jest opracowanie algorytmu wyznaczania podziałów bloku CTU, który znacznie zmniejszy złożoność obliczeniową kodera, przy jednoczesnym zachowaniu wydajności kompresji w odniesieniu do rozwiązania zastosowanego w oprogramowaniu referencyjnym. W związku z tym zbadano możliwość wykorzystania sztucznych sieciach neuronowych (ANN) w algorytmie podziału bloku CTU.

Na początku rozprawy przedstawiono opis procesu podziału bloków CTU w technice HEVC. W opisie uwzględniono kontekstowość w podejmowaniu decyzji. Przedstawiony został przegląd istniejących algorytmów wyznaczania podziałów bloku CTU, wskazując aspekty, które wymagają ulepszeń. Równocześnie dokonano przeglądu metod pozwalających na sterowanie kompromisem pomiędzy czasem kodowania i wydajnością kodowania, gdzie wskazano na skomplikowanie sterowania tym kompromisem w istniejących rozwiązaniach.

W ramach badań rozważano dwa podejścia do problemu podziału bloków CTU: jeden rozważający decyzję o podziale na poziomie bloków Coding Unit (CU), oraz drugi rozpatrujący decyzje na poziomie zarówno bloków Coding Unit (CU) jak i Prediction Unit – PU). Proponowane sieci neuronowe zaprojektowano do łącznego estymowania prawdopodobieństw głębokości podziału dla poszczególnych obszarów w CTU. Do wytrenowania takiej sztucznej sieci neuronowej przygotowano dedykowany zbiór danych uczących. W rozprawie opisano proces tworzenia architektury sztucznych sieci neuronowych, wraz ze szczegółową analizą wyników trenowania i wydajności ostatecznych modeli.

Następnie, w niniejszej rozprawie, przedstawiono autorskie algorytmy decyzyjne, wykorzystujące prawdopodobieństwa estymowane przy pomocy sztucznych sieci neuronowych. Algorytmy te zdefiniowano w dwóch wariantach: twardo decyzyjnym i miękko decyzyjnym. W rozprawie wykazano, że jeden z proponowanych algorytmów pozwala na prostą kontrolę kompromisu między czasem kodowania a wydajnością kodowania używając tylko jednego parametru.

Analiza porównawcza z najnowszymi rozwiązaniami wykazała, że proponowane algorytmy partycjonowania oferują najlepszy kompromis między skróceniem czasu kodowania a wydajnością kodowania.

Biorąc pod uwagę kontrolę nad kompromisem między czasem kodowania a wydajnością kodowania, spośród metod znanych w literaturze, zaproponowana metoda zapewnia najlepsze wyniki i najprostszą kontrolę nad tym kompromisem. Dodatkowo wprowadzono autorską metrykę do szybkiego porównywania takich metod, której wyniki pokrywają się z innymi, dobrze znanymi metodami.

Na końcu rozprawy, zbadano wpływ efektów kontekstowych na decyzje dotyczące podziału bloku CTU. Kluczowe osiągnięcia w tym obszarze to:

- Zaproponowany autorski eksperyment do określania wpływu kontekstu kodowania na decyzje dotyczące podziału bloku CTU.
- Modyfikacje zaproponowanej sztucznej sieci neuronowej do przetwarzania informacji kontekstowych.
- Opracowana przez autora metoda trenowania sieci neuronowej poprzez rozszerzenie etykiet uczących (ground truth augmentation).

- Autorska metoda globalnej optymalizacji decyzji o podziałach bloków CTU, nieistotnie zwiększająca czas kodowania.

Wszystkie eksperymenty przedstawione w niniejszej rozprawie zostały przeprowadzone przy użyciu opracowanej przez autora zmodyfikowanej wersji oprogramowania modelowego dla techniki HEVC. Modyfikacja ta, umożliwiająca szybką implementację algorytmów podziału bloków CTU opartych na sztucznych sieciach neuronowych, została udostępniona na zasadach otwartego dostępu (open-access).

List of symbols, notations, abbreviations and terms

α	-	Control parameter for AlgIdx decision algorithm in soft-decisive variant (Subsection 7.3.2.1)
β	-	Control parameter for AlgPrb decision algorithm in soft-decisive variant (Subsection 7.3.2.2)
{ }	-	a set of elements
$[a; b]$	-	closed interval of integer numbers, such that { x | a<=x<=b }
$(a; b)$	-	open interval of real numbers, such that { x | a<x<b }
$\langle a; b \rangle$	-	closed interval of real numbers, such that { x | a<=x<=b }
(i, j)	-	vector of indices for the first two dimensions coordinates of the division matrix/tensor
$\leftarrow$	-	assignment operation
d	-	the index used for the third dimension coordinate of the tensor, related to depth level value (Section 2.3.3)
Iv()	-	the Iverson function [Fo99], such that Iv$(true) = 1$ and Iv$(false) = 0$
ArgMax	-	arguments of the maxima
cdl	-	current depth level
p.p.	-	percentage points
$PSNR$	-	Peak Signal-to-Noise Ratio
$BD\text{-}RATE$	-	Bjøntegaard metric [Bj01] used for bitrate comparison (expressed in percents), given the same quality
$BD\text{-}PSNR$	-	Bjøntegaard metric [Bj01] used for quality (in dB), given the same bitrate
TS	-	**Time Savings**, metric used for assessment of encoding time reduction (expressed in percents)
T_{ANN}	-	contribution of network processing time to coding time (expressed in percents)
FoM	-	Figure of Merit [Na20, He20]
DM	-	**Division Matrix**, used for storage of partitioning decisions in HM [HM], indexed: DM[i,j]
DT	-	**Division Tensor**, the output of the proposed ANN, indexed: DT[i,j,d]
CPU	-	Central Processing Unit
GPU	-	Graphical Processing Unit
NPU	-	Neural Processing Unit
MPEG2	-	MPEG2 video encoding standard [MPEG2]
AVC	-	Advanced Video Coding, video encoding standard [AVC]
HEVC	-	High Efficiency Video Coding, video encoding standard [HEVC]
VVC	-	Versatile Video Coding, video encoding standard [VVC]
AV1	-	AOMedia Video 1, video encoding standard [AV1]
VP9	-	VP9 video encoding standard [VP9]
JPEG	-	JPEG image encoding standard [JPEG]
CTU	-	Coding Tree Unit
CU	-	Coding Unit
PU	-	Prediction Unit

TU	-	Transform Unit
SRU	-	Smallest Representable Unit
ANN	-	Artificial Neural Network
CABAC	-	Context-Adaptive Binary Arithmetic Coding [HM]
MPM	-	Most Probable Mode [HM]
HM	-	HEVC reference model software [HM]
RD Optimization	-	Rate-Distortion Optimization
QP	-	Quantization Parameter [HM]
CTC	-	Common Test Conditions
2D	-	2-dimensional
3D	-	3-dimensional
ML	-	Machine Learning
ETvsCE trade-off	-	Encoding Time vs Coding Efficiency trade-off
ANN	-	Artificial Neural Network
CM	-	Confusion Matrix
MPEG	-	Moving Picture Experts Group
RD curves	-	Rate-Distortion Curves
JCT-VC test sequences	-	test sequences, defined in CTC for HEVC
MAC	-	Multiply And Accumulate
DIV2k	-	uncompressed images dataset [Ag17] used for the creation of the training dataset
Modified HM	-	author's modification of the HM software (Section 4.7)
AlgIdx	-	Index-based decision algorithm (Section 7.2.1 and Subsection 7.3.2.1)
AlgPrb	-	Probability-based decision algorithm (Section 7.2.2 and Subsection 7.3.2.2)
SVM	-	Support Vector Machines
HMM	-	Hidden Markov Model
Partitioning	-	division of block into smaller blocks.
Partitioning pattern	-	a specific division of block into smaller blocks.
Basic Approach	-	One of the proposed approaches to partitioning process, considering division into CU blocks only
ANN architecture for Basic Approach	-	the final ANN architecture, used for training models for Basic Architecture (Figure 5.1, Section 5.1)
Basic Architecture	-	a set ANN models with architecture presented in Figure 5.1, and trained for *QP* values [22, 27, 32, 37], trained using dataset defined in Section 4.2
Extended Approach	-	One of the proposed approaches to partitioning process, considering division into CU and PU blocks
ANN architecture for Extended Approach	-	the final ANN architecture, used for training models for Extended Architecture (Figure 5.1, Section 5.1)
Extended Architecture	-	a set of ANN models with architecture presented in Figure 6.1 and trained for *QP* values [22, 27, 32, 37], trained using dataset defined in Section 4.2
Coding block	-	a block of samples encoded in an encoder, a general term for such blocks used despite the specific encoding technique

Bit cost	-	number of bits required for encoding a set of data in an encoder
Dataset, Training Dataset	-	dataset used for training ANN models
Training Subset	-	the subset of the training dataset used for ANN model training
Validation Subset	-	the subset of the training dataset used for ANN model validation
Hyperparameter/s	-	ANN architecture parameters, such as: number of layers, type of layers, layer-related parameters, etc.
Hyperparameter tuning	-	Process of hyperparameter adjustment
Model weights	-	All trainable and non-trainable parameters of ANN model
Training sample	-	a single element of the training dataset, consisting of ANN input data and expected output (ground truth)
Ground truth	-	the expected output of the ANN, given a specific input
Batch	-	set of samples from the training dataset, used for weights update in ANN training
epoch	-	a single iteration over all training samples
Batch learning	-	use of all samples from the training dataset for single weight update
Minibatch learning	-	use of multiple sample batches during a single epoch
Layer	-	the functional block of the ANN architecture
Feature map	-	the output of hidden layers in ANN
Feature map size	-	size of the first two dimensions of a 3D feature map
Feature map channels	-	size of 3rd dimension of 3D feature map
Kernel size	-	a kernel of operation performed in a layer
Stride	-	shift of kernel during the layer processing, expressed in the number of feature map samples
Padding	-	extension of the feature maps by adding samples on the edges

1 Introduction

1.1 Preamble

Nowadays, video is almost everywhere: from TV, streaming platforms, and video storage to video conferences and social media. In global internet traffic, visual content takes up a significant share [For1]. This significance would not be possible without video and image compression techniques, especially lossy ones. Such techniques significantly reduce the data required for transmitting or storing video and make multiple use cases possible [Ri03A, Do10, Su12, Ka19, Br21A]. The transfer capabilities of telecommunication networks are still increasing, but these may not be sufficient for the rising demand for video content [Er24]. Thus, the research on video coding technologies is a very important topic.

Currently, there are multiple techniques available for video compression. Most utilize hybrid encoder architecture, which divides video sequence frames into specific sections called coding blocks. Such coding blocks, composed of pixels, are then predicted using temporal or spatial information. In most of the techniques, the prediction is done separately for each color component. Such prediction error is then quantized to remove redundant information and entropy encoded to achieve a high compression ratio [Ri03A, Do10, Su12, Ka19, Br21A, Br21B].

Among the organizations concerned with video compression, the most noteworthy is the Motion Picture Experts Group (MPEG) working on behalf of ISO/IEC. The research and standardization process of MPEG yielded several techniques, such as MPEG2 [MPEG2], AVC [AVC], HEVC [HEVC], and VVC [VVC]. The group focuses on further video coding research, which is currently aggregated in ECM software [Co23]. Along with MPEG, there are other initiatives focused on the development of new video coding techniques. Among others, the most popular ones are VP9 [VP9], developed by Google, and AV1 [AV1], which is the effect of the Alliance for Open Media consortium works.

The aforementioned video coding techniques employ sets of procedures, tools, semantic and syntax, with corresponding parameters defined for several use cases (profiles, levels). Generally, a standard for such techniques describes two vital elements: the semantic and syntax of the bitstream and the decoding procedure. The syntax defines how to interpret consecutive bits from a bitstream by defining a set of flags and fields. The decoder is defined as a set of processes, a set of tools used for video decoding, and techniques used to suppress coding artifacts. Standardization of a technology in such a way described above ensures that encoded video bitstream is interpreted identically on standard-compliant devices. The implementation of an encoder is not restricted by any means as long as it produces a bitstream compliant with the syntax defined in the standard.

The description of standardized technique specifies technology, but not the exact way to use it. In a video encoder the appropriate choice of coding block sizes is made considering the use of available tools and coding modes. This process highly impacts the overall coding performance of the video coding technique – wrong decisions may decrease the compression efficiency. Choosing block sizes, modes, and tools is a complicated optimization process. Given that newer and newer video coding techniques can encode a block in more and more ways, such optimization becomes an increasingly complex and challenging problem [Zo04, Su12, Vi12, St16, Br21B]. In numbers, the encoding time increase between consecutive generations of the video encoding techniques can be estimated to be around 100 times [Ri03B, To19, Si20A, Is21, Me21A].

The abovementioned process is the core of an encoder. Depending on the use case, this process must be controlled accordingly. One can define two aspects of encoder control:

1. **Definition of parameters and conditions for encoding process**. Multiple variables define how the encoding process should be performed: starting from organization of a sequence (e.g., in Group of Pictures – GoP), through division of the frames into smaller units (e.g., Slices, Tiles, CTUs in HEVC/VVC), setting the range of sizes for different types of coding blocks,

defining the amount of quantization applied to data, enabling available coding tools, ending with target bitrate. Overall, these parameters set conditions for optimization process and are used by the user to control an encoder.

2. **Description of the optimization process and definition of the decision-making process.** Every algorithm that allows estimation of encoding decision subsets may be called an encoder control algorithm. Such an algorithm may estimate the prediction mode, coding block sizes, types of transformations or decide if a coding tool should be applied. In modern video encoders, control algorithms determine their decisions among quintillions of possibilities [Bo12, Sa20].

Reference models are available for most encoding techniques. The term 'reference model' relates to software demonstrating the encoder and decoder for a given technique. These implementations of encoders focus on showcasing the highest possible compression efficiency for a given quality. Given that, the reference models employ extensive encoder control algorithms, which consider a large number of combinations and choose the most efficient one. As the enormous number of combinations, counted at least to quintillions [Bo12] (Section 2.2, Formula 2.1), is considered, the computing complexity of the reference models is too high for the reference model to be directly applied to consumer devices such as personal computers, TV studio equipment, or camera-equipped mobile devices (e.g., smartphones). Especially for smartphones, the compute complexity of the encoder is crucial due to limited computing power and low energy consumption requirements. In practice, simpler control algorithms are used at the cost of reduced compression efficiency [Sw10, Mi13, La20, Vi22].

All the abovementioned facts outline the major scientific problem: a search for low-complexity encoding control algorithms with possibly best compression efficiency. Work under this problem requires expert knowledge in multiple fields such as signal processing, data encoding, and video encoding, and a deep understanding of the target technique is needed, as the task is non-trivial to handle. Research on such encoding control algorithms is frequently addressed, especially for modern video coding techniques, e.g., HEVC [Ur23, Kh24, Fa24]. Problems related to this research field still need better-adapted solutions for practical use, such as accurate and fine-grained control over the Encoding Time vs. Compression Efficiency (ETvsCE) trade-off. This problem cannot be solved by adjusting encoder parameters, and currently available solutions require complicated handling.

A way to face the scientific challenge presented earlier is using Artificial Neural Network (ANN). Currently, huge advancements in Machine Learning are observed. The rising popularity of ANNs in multiple applications is caused by advances in software and hardware dedicated to these applications [Go16, Le20, Ga22, Pr23, Sa23]. The ANN-based solutions became practically viable, and the galloping development will increase their accessibility [For2, For3, For4]. The newest consumer devices offer extended ANN capabilities by including a Neural Processing Unit (NPU). These NPUs employ hardware that is dedicated to matrix operation computation. Most current technological companies that offer System on Chips (SoC) or Central Processing Units (CPUs), such as Snapdragon, Apple, Intel, or AMD, are currently offering NPU-featured devices for mobile devices [Or22, Ri24, In24, Ma23A]. With such hardware support, using ANN as part of the encoder control algorithm becomes a viable solution [Rh12, Ce21] and may vastly increase the integration of new video encoding techniques.

1.2 Scope of the dissertation

The dissertation is focused on the encoder control algorithm, precisely the partitioning process. The partitioning is referred to as a process of dividing of samples into smaller blocks. The partitioning is the most time-consuming process in encoding, constituting up to 95% of encoding time [Bo12, Du18, Me21A]. The set of blocks resulting from the partitioning algorithm will be called the partitioning pattern. The goal of the research is developing a partitioning algorithm characterized by two key aspects: low complexity along with maintaining the compression ratio. A reference partitioning algorithm is required to assess these aspects, which will be indicated later in this section. Other elements of the encoder control algorithm, like a selection of angular modes or coding tools, are out of the scope.

Major aspects of the dissertation scope are the prediction mode and standardized encoding technique used in research. The following paragraphs will present the relevant choices and their reasoning.

The scope of this dissertation is restricted to partitioning algorithms dedicated to Intra mode. The reasoning behind such a restriction is strictly connected with the importance of the Intra mode. Firstly, the Intra mode utilizes dependencies within the current encoding frame, while the Inter mode utilizes temporal and spatial dependencies in video sequence. The first frame of a sequence is always encoded in Intra mode, so these decisions will influence the encoding of the subsequent frames. Additionally, the Intra mode is used to compress images. Therefore the use case of the proposed method is extended along with the availability of the ANN training datasets.

The final argument in favor of focusing on Intra mode stemmed from a literature survey conducted at the beginning of the research works. The survey concluded that research focused on partitioning algorithm using ANN for Intra mode is more promising [Mo15, Ch18, Li18, Li21A, Am21, Li22C, Ab22, Ba22, Xu18A, Xu18B, Ki19, Wa21] than for Inter mode [Rh12, Sh12, Lu13, Le15, Pa14, Du15, Ki16B, Ta16, Ta17, Li17, Du18, Ta19, Tu19, Lu20, Ku20A, Zh21A, Hu21A, Ci22, Li22A, Ta22, Li22B, Ni22].

A significant aspect of the dissertation is dedicated to choosing a standardized video encoding technique to be used as the base for research. Among multiple standardized techniques for video encoding, the High Efficiency Video Coding (HEVC) was chosen. The rationalization of this choice is presented further in this section. Nevertheless, other video encoding techniques created by MPEG were considered. The AVC [AVC] is relatively common but much less efficient regarding compression efficiency. The successor of HEVC [HEVC], namely VVC [VVC] was still under development at the beginning of the research. Even though VVC technology now is standardized [VVC], it is still marginally popular and lacks hardware encoders and decoders. Video encoding techniques proposed by other initiatives like VP9 [VP9] and AV1 [AV1] are not that popular in the scientific field, and their performance does not overcome HEVC much. This dissertation is focused on Intra prediction mode, so image compression techniques such as JPEG [JPEG] could be used. Despite its popularity, it is being replaced by appropriately adapted solutions based on video encoding techniques, e.g., HEVC-based (the HEIF format).

At the beginning of the research, HEVC was the most recent technique for video encoding. The complexity of the reference encoder for HEVC is high but low enough to make the research possible with available computational power. Using the VVC reference encoder for experiments will extend the encoding time 100 times compared to HEVC, which would be impossible to overcome due to reasons explained further in the research methodology (Section 3.1). Still, HEVC is a cornerstone for newer techniques, and achievements for HEVC may be transposed to newer VVC with minor adjustments. HEVC is becoming one of the most significant video encoding techniques as hardware encoders and decoders have become broadly available in recent years [Ap24].

In summary, the scope of the dissertation includes the Intra mode in HEVC. The partitioning algorithms used in the reference encoder of HEVC employ an extensive estimation of partitioning patterns, resulting in an arduous search for the best one. Multiple patterns are tried in this process, but just one is finally used. Within the scope of this dissertation are partitioning methods that indicate a narrow range of partitioning patterns to consider or even point to only one. Such an approach vastly reduces the computational overhead of the partitioning algorithm.

The reference encoder of HEVC will be used in the research to assess the proposed partitioning methods. Within the scope of this dissertation are techniques implemented in the reference encoder of HEVC. The proposed partitioning algorithm will be tested as a replacement for the reference partitioning algorithm within the reference encoder of HEVC. Other decisions made by the modified encoder should be made the same way as those made by the reference encoder. Methods that impact other aspects of the

reference encoder or implement partitioning algorithms in other HEVC encoder implementations are out of this dissertation scope.

This dissertation is focused on partitioning algorithms that use the ANN. In particular, the use of the ANN to estimate whole partitioning patterns or sets of patterns is addressed. The search for the low-complex architecture of the ANN suitable for the presented task is one of the main topics of this dissertation. Thus, the Fully Connected ANNs are considered, which is rationalized in Section 4.3. The input data of the ANNs may be a set of samples from CTU. The subject of the contextually of the encoding process is within the scope of this dissertation. Thus, additional input data are considered as a way to include the encoding context in ANN processing. Furthermore, the ANN training data preparation, taking into account the contextually of encoding, is also explored in this dissertation. One of the topics in this dissertation is the use of the proposed partitioning algorithm in the global optimization of partitioning patterns. The ANNs considered in this dissertation estimate an output, which is a set of probability values.

The scope covers the methods that use the ANN and the non-trivial decision algorithm. Such a decision algorithm covers the conformance control of the partitioning pattern and the interpretation of the ANN output. The abovementioned partitioning algorithms proposed in this dissertation may indicate a single partitioning pattern or a set of partitioning patterns. Within the scope of this dissertation are decision algorithms that allow flexible control over the incidence of multiple partitioning patterns consideration. The development of decision algorithms directly relates to another major topic of this dissertation: the control over Encoding Time vs Compression Efficiency trade-off.

1.3 Research goals and theses

The research on video encoding control algorithms is presented in this dissertation. The focus is primarily on partitioning coding blocks in HEVC. Developing a partitioning algorithm aims to reduce the encoding time compared to the reference solution while maintaining a compression efficiency as close to the reference model as possible. The partitioning algorithm uses ANN to derive a decision. Such a network would be trained to mimic the partitioning decisions of the HEVC reference model. The targeted ANN should be as simple and computably low-complex as possible.

The ANN used in the partitioning algorithm generates outputs, which can be interpreted as probabilities of the depth level values for certain subareas of the coding block. In this dissertation, it is proposed to process the such output by a decision algorithm. Another goal of the research presented in this dissertation is to design a decision algorithm that yields the best compression efficiency for HEVC and ensures that the partitioning pattern conforms with the HEVC syntax. Another aspect of the targeted partitioning algorithm is the training of the ANN. An ANN model is trained with some error, which means that such a model will not always be sure of a particular partitioning. Considering this, the goal is to exploit this phenomenon, called the model uncertainty, by designing a soft-decisive algorithm. Soft-decisiveness will be used to improve the effectiveness of the developed partitioning algorithm. Additionally, such uncertainty can be used to fluently control the Encoding Time vs. Compression Efficiency (ETvsCE) trade-off. Such control is desired to be as simple as possible.

Considering the abovementioned research goals, the following theses were stated :

T1: The utilization of the Artificial Neural Network with a decision algorithm can significantly decrease the computational complexity of the video encoder as compared to HEVC reference model encoder.

T2: The employment of Artificial Neural Network with a soft- decision algorithm enables a single parameter control over the Encoding Time vs Compression Efficiency trade-off.

1.4 Overview of the Dissertation

The organization of this dissertation is presented in Figure 1.1. Chapter 2 discusses state of the art. It presents the main aspects of the HEVC technology, like the encoder control algorithm and partitioning rules and restrictions. Further in that chapter, a review of the partitioning methods in the literature is presented. The main aspects of such methods are described and broken down into non-neural and neural methods. These methods are summarized at the end of the chapter, and potential improvements and research directions are highlighted.

Chapter 3 presents the methodology used in research presented in this dissertation. The assessment of the video encoder modification impact is presented. The test sequences are presented along with the rules for experiments with a video encoder. The training methodology is presented with the choice of the dataset. Chapter 4 presents the main idea of the proposed partitioning algorithm. There, the overall approach is discussed, the definition of a problem for ANN training is set, the inputs and outputs of ANN are defined, and the training procedure is shown. Additionally, the details of the implementation of the partitioning algorithm into the Modified HM (Section 4.7) are presented.

Chapters 5 and 6 present two main approaches for partitioning algorithms: Basic and Extended. Detailed architectures of ANN are presented, and training and evaluation results are analyzed. Additionally, the choice of hyperparameters is discussed. In Chapter 7, a detailed description of the decision algorithms is presented. The hard-decisive variants are defined. The viability of soft-decisiveness is verified, and soft-decisive variants of the decision algorithms are defined. Proposed decision algorithms are evaluated. The control over Encoding Time vs. Compression Efficiency (ETvsCE) trade-off is proposed. Chapter 8 presents the comparison of the best-proposed partitioning algorithms with state-of-the-art solutions.

Chapter 9 presents exploration experiments for the proposed partitioning algorithm. It contains several approaches that aim to improve the performance of the Basic and Extended approaches or present minor achievements of the research. The improvement of the ANN is presented, and the utilization of contextuality of the encoding process in HEVC is discussed. Finally, the original method for global optimization of CTU partitioning with the use of Hidden Markov Model (HMM) is proposed. The direction of future research is shown. Chapter 10 contains the dissertation summary.

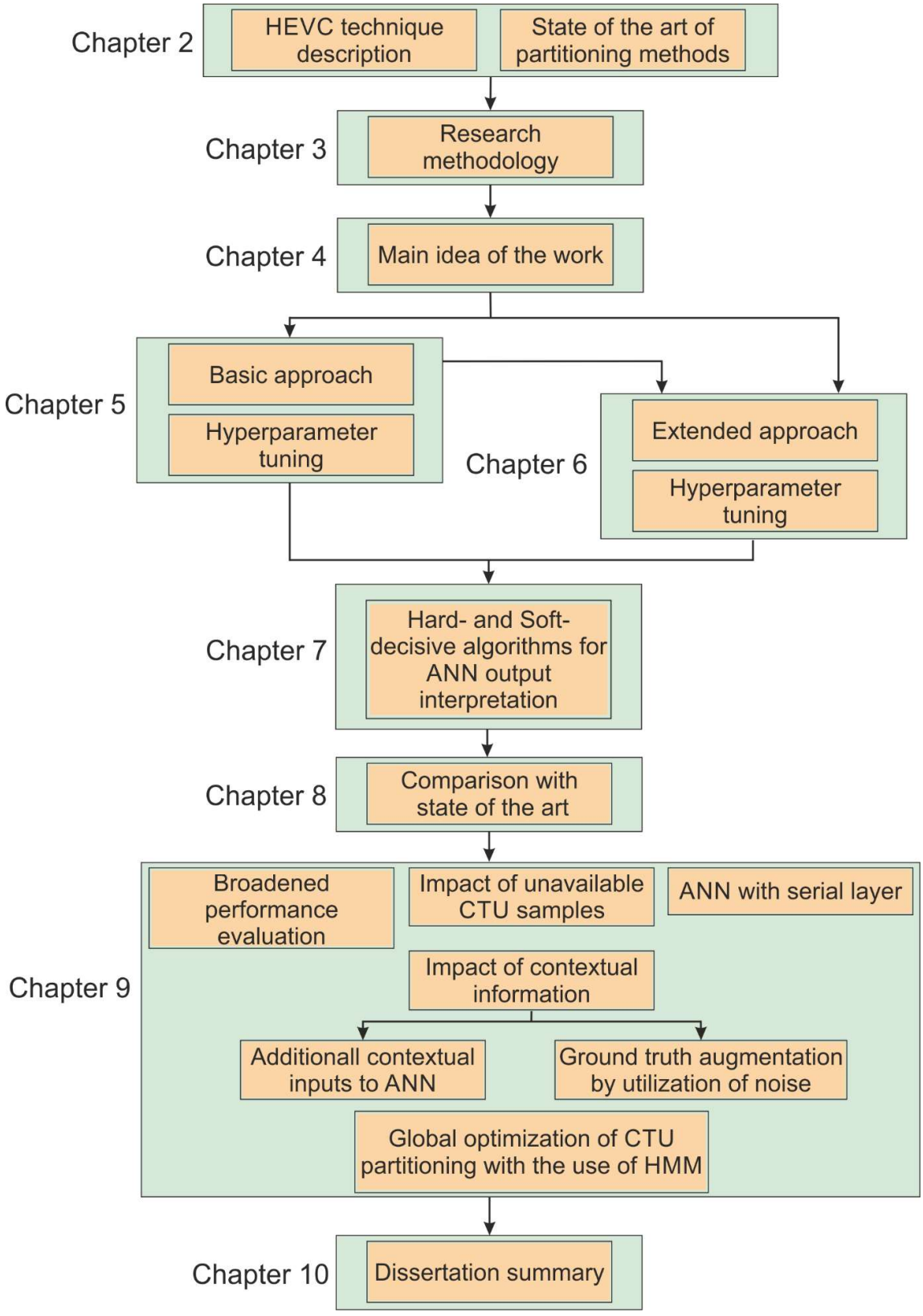

Figure 1. 1 Organization of the dissertation with the description of the main subjects of the chapters.

2 State of the art

2.1 High Efficiency Video Coding (HEVC) technology

Due to the MPEG works, in 2013, the High Efficiency Video Coding (HEVC) standard was released. This technique is also known as ITU.T H.265 or MPEG-H Part 2 [HEVC]. The coding efficiency of HEVC is about 50% better than its predecessor [Zo04, Su12, St16], the AVC [AVC]. The coding efficiency improvement came, for example, from increasing the number of ways a block of samples may be encoded. Compared to AVC [AVC], the complexity of the decoder remains similar [Zo04, V12], but the encoder, which requires a much more complicated encoder control algorithm, requires much more computations [Bo12, Co12, St16]. The simplified diagrams of the HEVC encoder and decoder are presented in Figures 2.1 and 2.2.

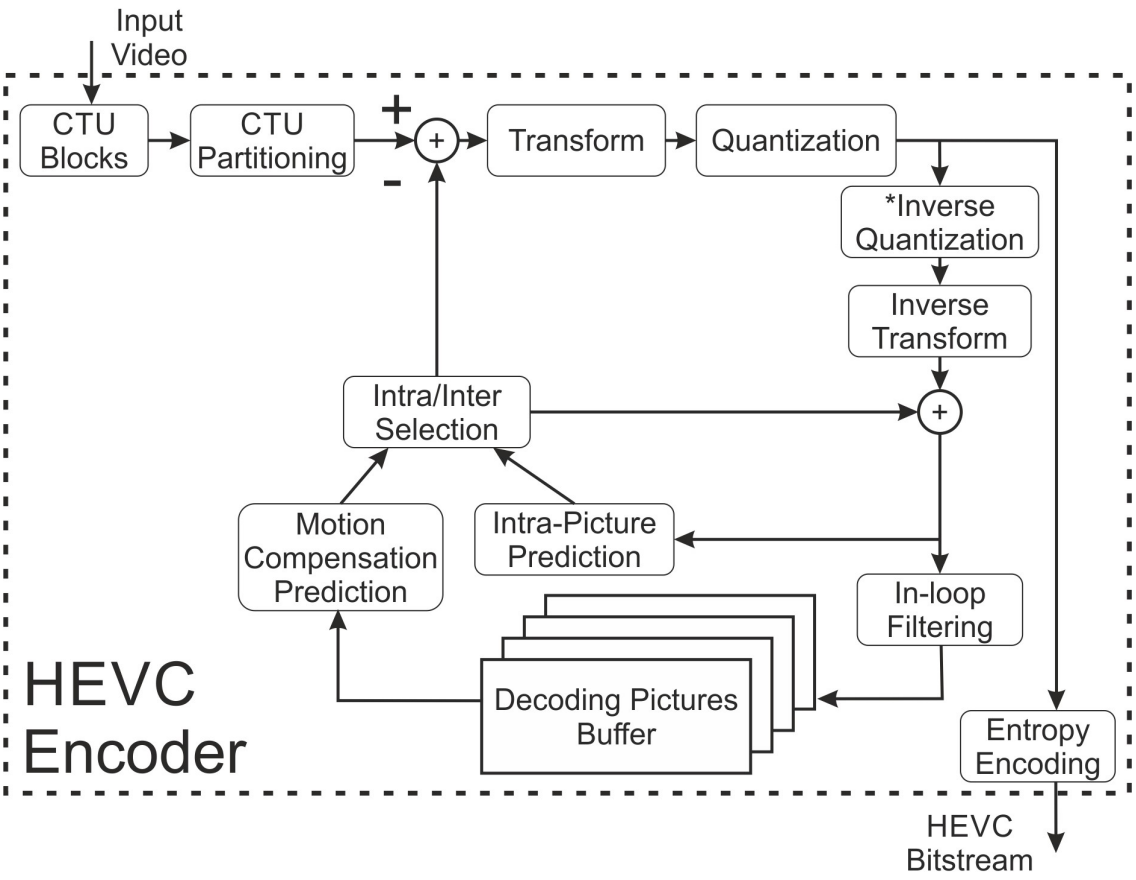

*Figure 2. 1. Simplified block diagram for HEVC encoder. Block "*Inverse quantization" refers to recovery of the original scale of the signal (quantization is lossy and cannot be inversed).*

The HEVC standard comes with multiple improvements compared to its predecessor, AVC [AVC]. Here are the most important ones from the standpoint of this dissertation. Firstly, macroblocks [AVC, Wi03A] were replaced by Coding Tree Units (CTUs) [HEVC, Su12]. The CTU block can be further divided into smaller blocks via partitioning [Su12, HEVC]. The partitioning is the most computationally demanding process, as it can constitute 95% of the encoding time [Bo12, Du18, Me21A]. The introduction of the CTU allows a better adaptivity of encoding decisions to video content, so better encoding effectiveness. A detailed description of the CTU is included in the next section. Secondly, the number of available options for encoding a block in Intra mode is significantly increased [HEVC].

Examples are the number of angular prediction modes or transform options. Thirdly, the more advanced CABAC encoder is used [HEVC, Su12, Ka19]. The improvements in CABAC come from better organization of binary sub-streams and more effective statistic modeling of the binary symbols. These made CABAC more impactful on multiple coding decisions in consecutive CTU blocks [Ka19]. The presented components of HEVC significantly affect the complexity of the algorithm controlling the encoder and, consequently, the computational complexity of the encoder. Those components will be further discussed in the dissertation.

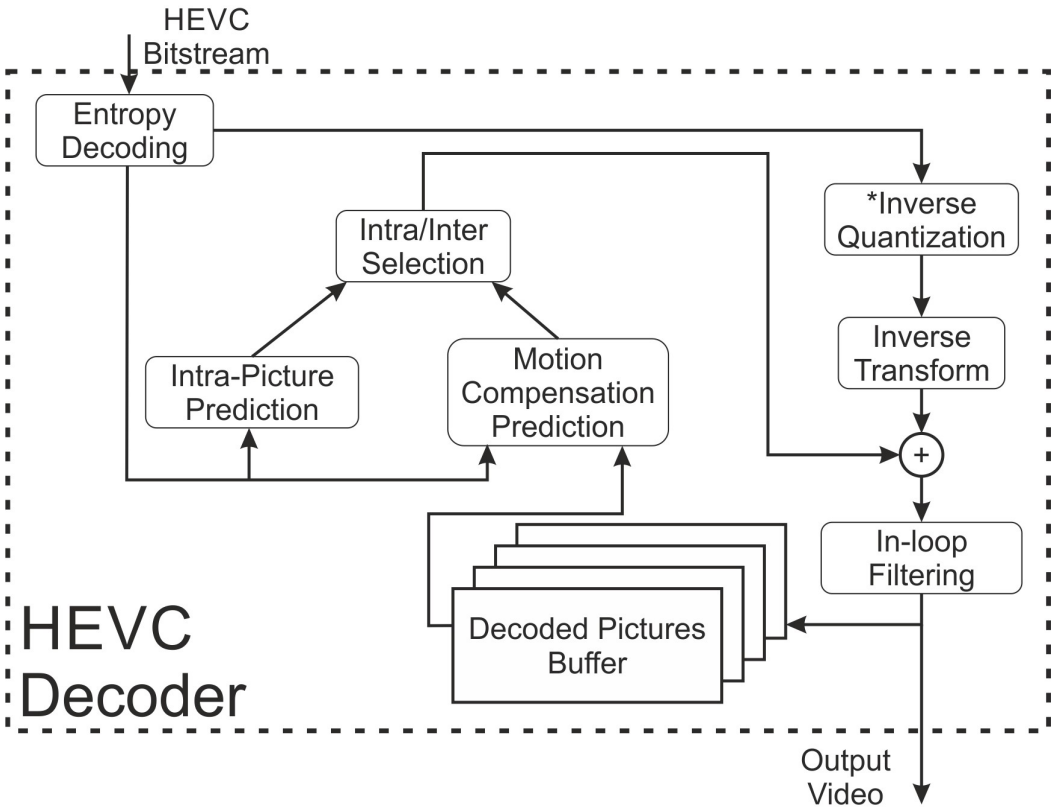

*Figure 2. 2 Simplified block diagram for the HEVC decoder. Block "*Inverse quantization" refers to recovery of the original scale of the signal (quantization is lossy and cannot be inversed).*

Along with the standard, the MPEG group shared the reference model for HEVC. This encoder is the HEVC Reference Model Software [HM, Mc14], commonly called HM. This software proves the high compression efficiency achievable by the technique but at the cost of the computational complexity. The applied encoder control algorithm performs an extensive search over available coding options. Still, some fast algorithms were used in less crucial areas. One such algorithm is the Most Probable Mode (MPM) mechanism [HM], which reduces the number of candidates for prediction mode. Despite that, a large number of available coding options are tried during encoding. A more detailed description of the encoding control algorithm used in HM is presented in Section 2.3.

The MPEG group standardized a more advanced encoding technique – VVC [VVC]. This technique may be described as a set of improvements for HEVC [Br21A, Br21B]. It means that multiple solutions for HEVC may be transferred with minor adjustments to VVC.

2.2 The partitioning of the Coding Tree Units (CTU) in HEVC

The partitioning is the most time-consuming process in the encoder [Bo12, Du18, Me21A]. As mentioned in the previous section, the Coding Tree Unit (CTU) is a block that can be further divided into smaller blocks via a partitioning process [Su12, HEVC]. The CTU organizes luma samples with corresponding chromas depending on the sampling scheme. CTU is always square, and the maximum size of the CTU is 64×64 luma samples with corresponding chromas samples.

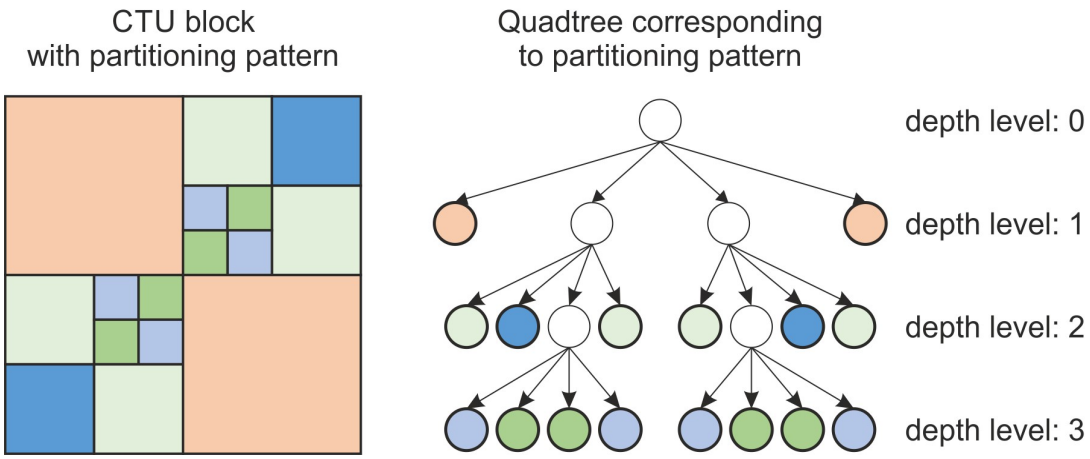

Figure 2. 3. Exemplary CTU partitioning with corresponding quadtree. Value in quadtree leaves indicates the block in CTU. White circles refer to internal nodes, and color circles to the leaves of quaternary tree.

The CTU partitioning arranges the samples into Coding Units (CUs). Such arrangement is done by consecutive block splits following a quaternary tree scheme. The set of blocks resulting from the division of the CTU is called a partitioning pattern. The partitioning pattern may be represented, e.g., by a quaternary tree, as shown in Figure 2.3. The depth of the leaf in the quaternary tree, referred to as depth level, is strictly connected to the CU size. In HEVC, the maximum depth level is 3 (indexed from 0). The exact interpretation of depth level as block size depends on the encoder configuration, the CTU size, and the minimum CU size precisely. The CUs in CTU are encoded following the Z-order scan [HEVC].

The calculation of the number of possible partitioning patterns may be done recursively. Here, it will be presented using the example. Let us assume that for a particular encoder configuration, the CTU size is $4 \cdot N$, and the minimum CU size is N. The number of possible partitioning patterns is determined using the following reasoning:

- A block of size $4 \cdot N$ may remain intact or be divided into four blocks of size $2 \cdot N$, according to the quaternary tree pattern.
- A block of size $2 \cdot N$ may remain intact or be divided into four blocks of size N, according to the quaternary tree pattern. Moreover, each block of size $2 \cdot N$ block can be partitioned independently.
- The block of size N cannot be further divided. It means that the block of size $2 \cdot N$ can be partitioned only in $1 + 1 = 2$ ways.
- Therefore, for the division of block of size $4 \cdot N$: possible partitioning patterns are no division (single $4 \cdot N$ block) or any variations of 4 blocks partitioned independently (2 possible partitioning patterns).

- The number of partitioning patterns for a block of size $4 \cdot N$ is $1 + 2^4 = 17$: no partitioning plus the number of variations with repetition for four blocks with two possible partitioning patterns.

Considering the above example, the formula for the number of possible partitioning patterns for HECV can be defined as:

$$PartNum_N(M) = \begin{cases} 1 + (PartNum_N(M/2))^4, & if\ N \neq M \\ 1, & if\ N = M \end{cases}, \tag{2.1}$$

where N is the minimum CU block size, and M is the CTU size, both defined for a given encoder configuration. A single value may represent CU block size, as CU is always square in HEVC. Using formula 2.1, for the most common HEVC configuration [HM], where $M = 64$ and $N = 8$, the number of possible partitioning patterns is 83 522.

Each CU block can optionally be divided into Prediction Units (PU). Then, PUs are divided into Transform Units (TU) blocks. Figure 2.4 presents the division scheme. The purpose of the PU is to use the same prediction mode and tools for the samples within a unit. A PU may have a rectangular shape, as shown in Figure 2.5. Restrictions for Intra mode are presented later in this section. Each division mode has a unique index. The residual signal derived by prediction is divided into TUs organized in the tree structure.

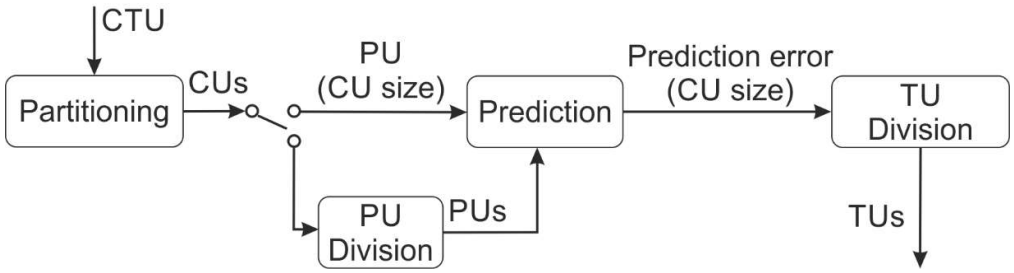

Figure 2. 4. Division of CUs into PUs and TUs.

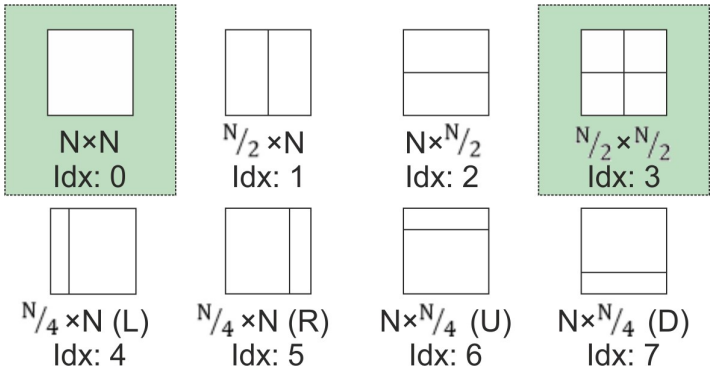

Figure 2. 5. Division modes for splitting CU into PUs with the corresponding indices (Idx). Only highlighted (green) modes are available in Intra prediction mode. N represents the size of the CU block.

Considering further division possibilities for CUs, PUs, TUs, and multiple prediction modes and coding tools, the partitioning algorithm may consider at least quintillions of ways to encode a single CTU block (formula 2.1). The encoder chooses the same partitioning and divisions both for luma and

chromas. The encoding control algorithm decides on the partition using the bit cost (Subsection 2.3.1) estimated for all components.

The HEVC syntax restricts available PU division modes in Intra mode. The only possible division mode for CU block sizes bigger than 8×8 is N×N (Idx: 0). This means that PU division is not considered. For CU block size 8×8, the available PU division modes are N×N (Idx: 0) and N/2×N/2 (Idx: 3) – highlighted in Figure 2.5. One can conclude that in Intra mode, a division into CUs and PUs may be considered jointly. In this case, PU division may be interpreted as the additional depth level of the quaternary tree. Considering this assumption, the number of possible CTU partitioning patterns, calculated using formula (2.1), increases to 48 663 522 406 470 666 257.

2.3 Rate-Distortion Optimization

2.3.1 General Description

The control over the HEVC encoder is not imposed by the standard [HEVC]. The HM [HM] uses Rate-Distortion Optimization (RD Optimization) as a base for the algorithm of video encoding control [Su98, HM, Do10, Ka19]. The RD Optimization considers available coding options through a trade-off between quality loss for the block versus its bit cost. In general, the optimization problem in RD Optimization is the minimization of cost function J [Do10, Ka19], defined as follows:

$$min\, J = D_{block}(mode, Q) + \lambda\, R_{block}(mode, Q), \tag{2.2}$$

where: D_{block} is the distortion of the block and R_{block} is bit cost – both for given $mode$ and given quality factor Q. The λ is the weighting coefficient, which value was estimated by thorough work of the MPEG group during the standardization procedure [HEVC, Ka19]. The simplified graphical interpretation of the search for optimal point is shown in Figure 2.6.

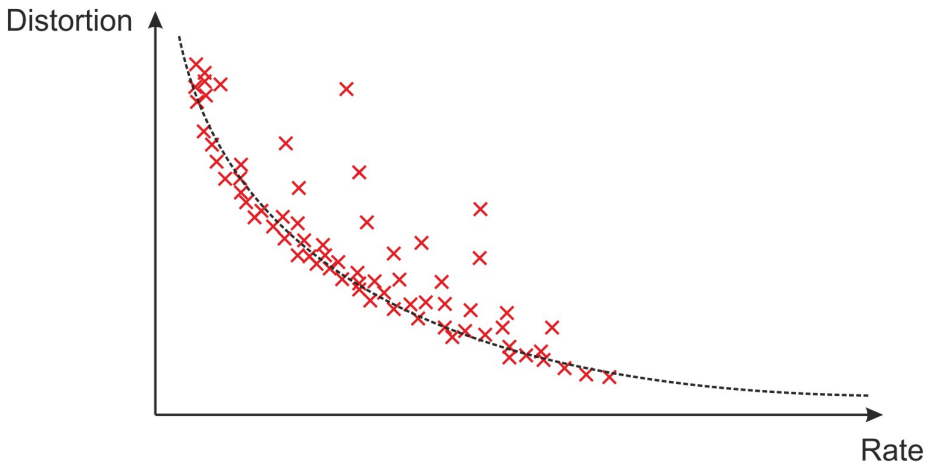

Figure 2. 6. Graphical interpretation of RD Optimization. Optimization aims to find a point (red × – different way of encoding) closest to the reference curve (dashed) for given optimization conditions.

Due to amount of possible coding options for a single CTU, such an optimization process is a very complex problem. Moreover, in video encoders such as [Bo12, Br21A, Br21B], the optimization should be performed slice-wise to ensure the best decisions. That is due to the dependencies between consecutive blocks within a single context of CABAC [HEVC, HM, Su12]. Moreover, calculating binary symbols statistics makes the decision process very sensitive. Furthermore, decisions should be fine-tuned with multiple passes of optimization. Therefore, all practically used encoder control algorithms apply suboptimal decision sets.

2.3.2 RD Optimization in HEVC Test Model

RD Optimization is the most complex and time-consuming process in the entire HM [Vi12, St16, Si20A, Is21, Me21A]. Still, the applied encoding control algorithm is a substantially simplified version of RD Optimization presented in the previous subsection. The optimization is done for CTU blocks, one by one, following the raster scan scheme. A particular CTU blocks influence decisions in further CTUs, e.g., by the CABAC context. Particular decisions for a CTU are made according to hierarchy, presented in Figure 2.7. The presented hierarchy corresponds to case when only Intra mode is allowed. In the research, the encoder was run in constant QP mode, so the *QP* adjustment steps made in constant bitrate mode are omitted. The CTU partitioning is the most important decision in the hierarchy (up to 95% of encoding time [Bo12, Du18, Me21A].), which makes it the most impactful on the complexity of the encoder.

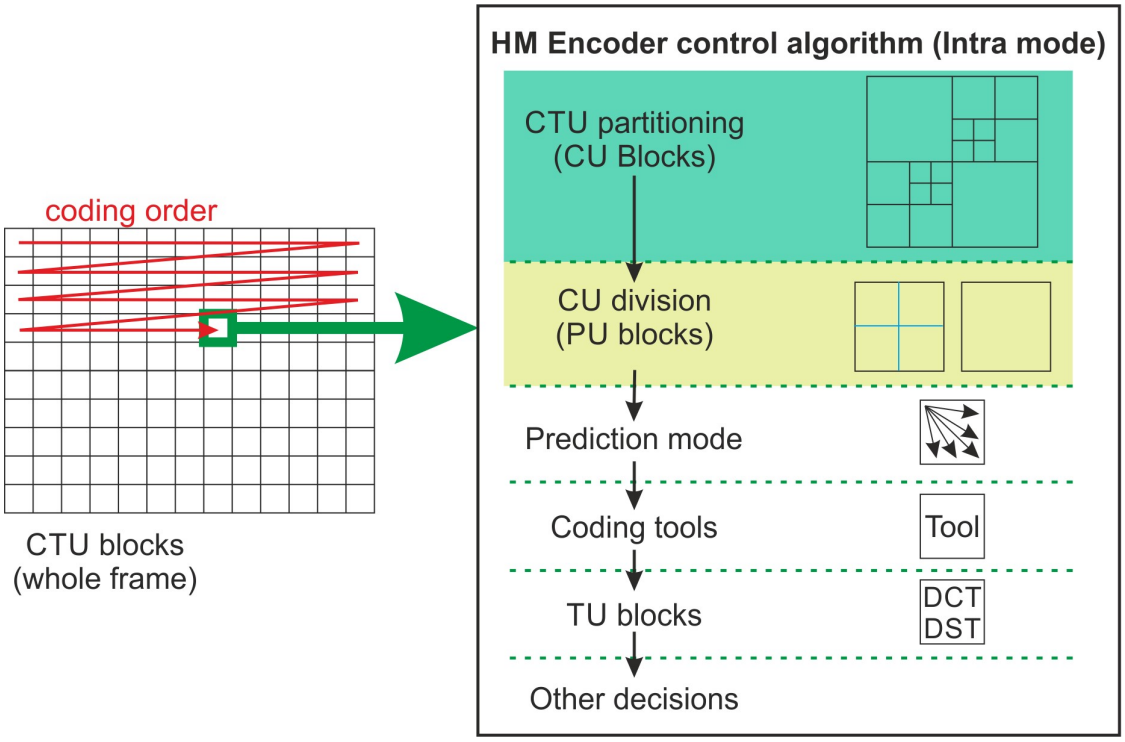

Figure 2. 7. Hierarchy of decisions for CTU in HM (Intra mode only). Highlighted sections of the hierarchy are within the scope of this dissertation.

The partitioning procedure is shown in Figure 2.8. Starting from the biggest possible CU (CTU size), the algorithm executes the following steps:

1. If not estimated earlier, find the cost (Formula 2.2) of encoding the block in its current size.
    a. If allowed, divide into PU blocks. Find prediction mode/modes.
    b. Decide on coding tool usage.
    c. After estimating the prediction error, decide on the TU sizes and transform type.
    d. Run CABAC to estimate bit cost for the current set of decisions.
2. Estimate the bit cost of encoding using a set of smaller blocks according to the quaternary tree. The procedure from step 1 is applied.
3. Compare the cost of encoding in the current CU size with the total cost using a smaller CU size.
    a. If a division is unprofitable – use the current CU size
    b. Otherwise, repeat the procedure for smaller CUs in Z-order.

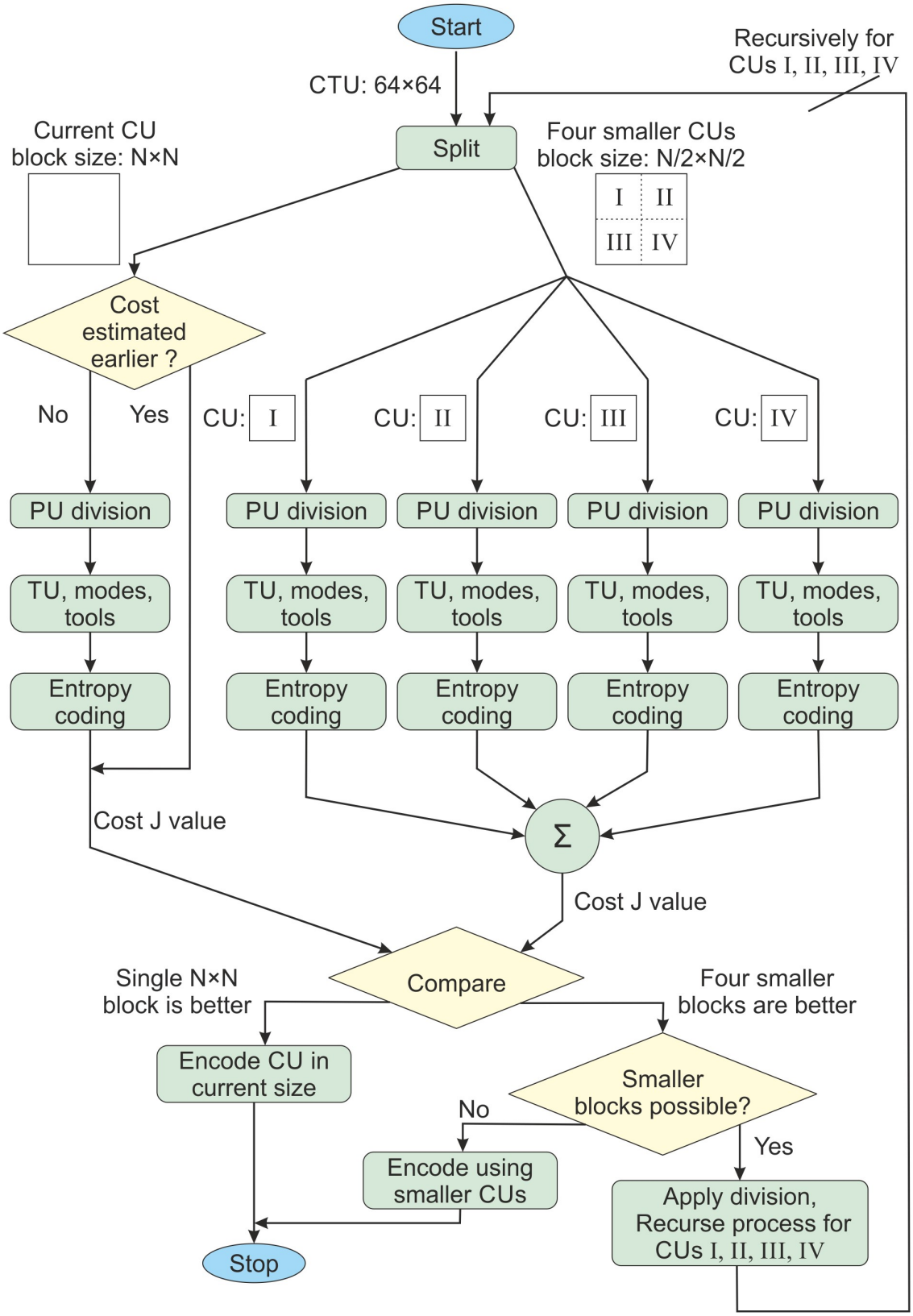

Figure 2. 8. CTU partition algorithm in HM, simplified for Intra mode only coding. "Cost J" is calculated with Formula 2.2

Two main compromises in the optimization may be highlighted from the described HM partitioning algorithm. Firstly, the algorithm greedily makes decisions. If the result for the current depth level is locally the best, further divisions are not considered. Termination of partitioning limits computational complexity, but better partitioning patterns may be missed. Still, the complexity is high, as each block size comes with a repetition of the decision process for lower hierarchy levels. A significant part of computations is checking blocks that will not be used in the final encoding. Checking block sizes from the biggest possible one increases computational complexity, especially when smaller blocks should be used.

Secondly, the algorithm does not revoke its decisions. The chosen partitioning patterns are not further modified. The only exception is adjusting the partitioning pattern to syntax restrictions, discussed in the following subsection. Decisions are made locally while the CTU is processed, but, as mentioned earlier, the partitioning is influenced by previous decisions. This influence will be called the contextual effect or contextuality of encoding. Contextuality comes from, for example, prediction, CABAC context, and mechanisms like MPM [HM]. Nevertheless, there is no mechanism to change previous decisions.

2.3.3 Storage of decisions in HM software and partitioning syntax restrictions

The decisions of the encoding control algorithm are stored in the HM encoder to be further used in the final bitstream composition, e.g., to provide relevant signaling. Depending on the decision type, the storage of decisions is slightly different. From the standpoint of this dissertation, the partitioning pattern (CU blocks) and the PU division mode storage are the most important.

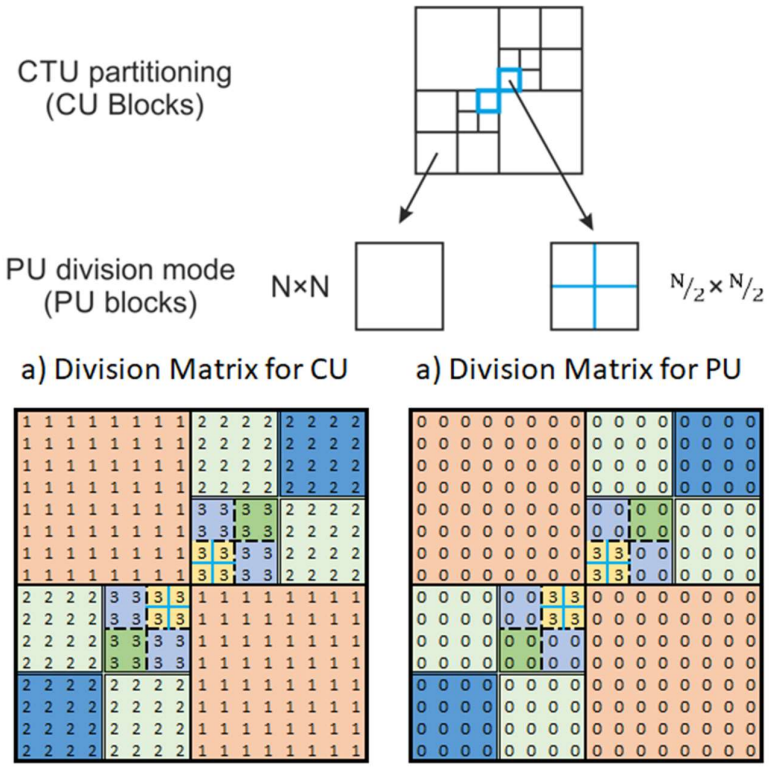

Figure 2. 9. Division matrices for CU and PU for exemplary partitioning pattern.

In general, the HM software stores the partitioning patterns as a square division matrix (Fig. 2.9). The Entire division matrix corresponds to the area of a given CTU, and each matrix field corresponds

to the division smallest representable unit/block. Therefore, the size of such a matrix depends on the size of CTU (in pixels) and the size (in pixels) of the smallest representable unit/block. In this dissertation, the smallest representable unit/block is denoted as SRU. In a typical configuration, the CTU size is 64x64, and the SRU is TU of size 4x4. Therefore, the size of the division matrix is 16x16.

The matrix for storage of CTU partitioning decisions (CU block sizes) is called the Division Matrix for CU (DM_{CU}) [HM], defined as follows:

$$DM_{CU}[i,j] = d \quad \text{where} \quad i,j \in [0;15], d \in [0;3]. \tag{2.3}$$

The d is the depth level value of the CU block that contains the area of the SRU. In DM_{CU}, the depth level value 0 corresponds to 64×64 CU, and the depth level value 3 corresponds to 8×8 CU. Decisions for PU divisions are stored in the Division Matrix for PU (DM_{PU}) [HM] defined as follows:

$$DM_{PU}[i,j] = p \quad \text{where} \quad i,j \in [0;15], p \in [0;7]. \tag{2.4}$$

The p is the index of the PU division mode. As the block is encoded in Intra mode, in DM_{PU} only two modes are allowed (indices 0 and 3). Exemplary partitioning patterns with PU division modes and corresponding Division Matrices are shown in Figure 2.9.

The CTU block size is always constant throughout the frame. It applies even when the resolution of the frame is not divisible by the size of the CTU. In such cases, the following procedure will be applied on CTU during the partitioning process:

1. If the resolution is divisible by the smallest available CU block (8x8 by default), go directly to step 2). Otherwise, the „Conformance window mode" is applied. The encoder extends the frame to a resolution divisible by the smallest available CU block. Missing samples are filled with the nearest left (for missing samples on the right side of the image) or the nearest top sample (the missing samples at the bottom) [HM].
2. All unavailable samples (their values are undefined) are grouped in isolated CUs. Such CUs are not encoded but are signalized in syntax [HM].

The abovementioned procedure is visualized in Figures 2.10 and 2.11.

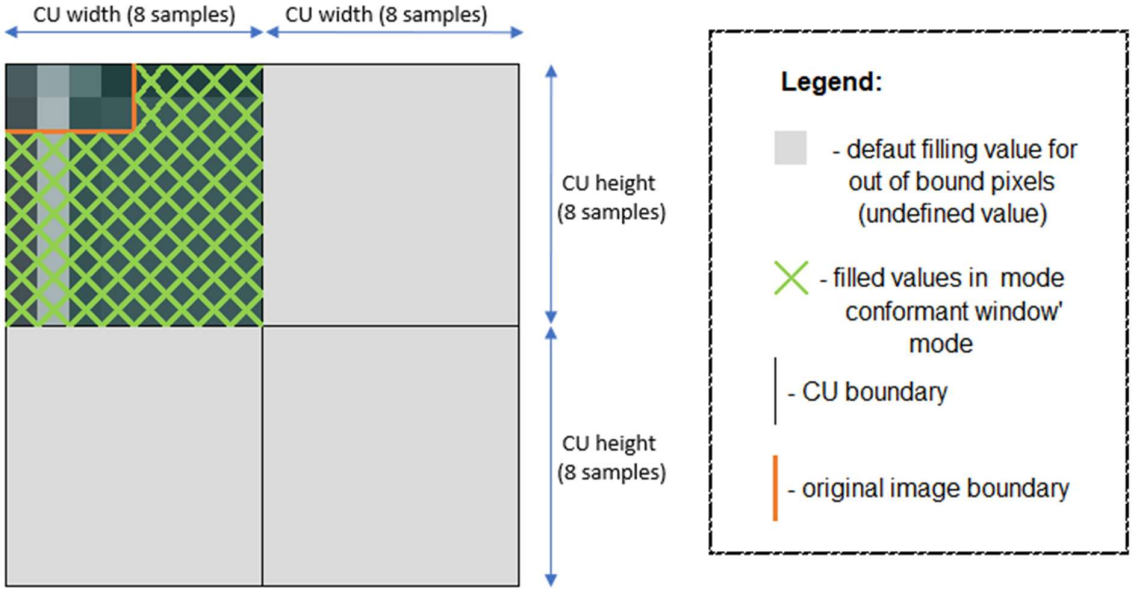

Figure 2. 10. Application of the Conformance Window Mode in HM software.

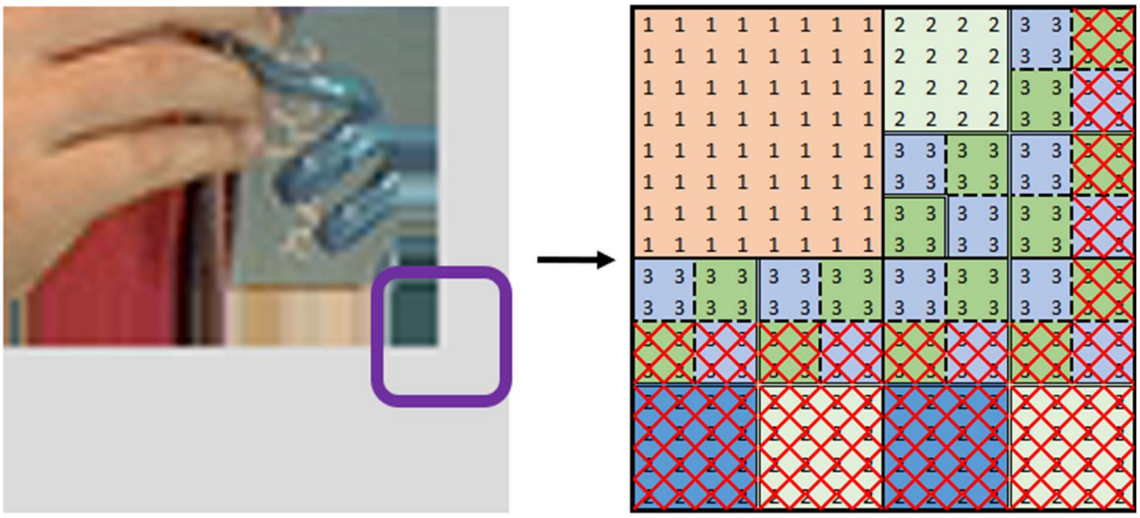

Figure 2. 11. Example of estimated partitioning pattern for CTU with unavailable samples (grey). The purple envelope marks the image fragment presented in Figure 2.10. Fragments of partitioning pattern marked with red crosshatch are not encoded but only signaled in syntax.

2.4 Partitioning methods

As presented in the previous sections, the encoding control algorithm in HM Software produces suboptimal decisions, although in terms of final bitrate is very efficient [Co12]. Despite that, the wide range of searches makes this approach very time-consuming (up to 95% of the encoding time [Bo12, Du18, Me21A]) and computationally complex. The complexity makes the HM not suitable for practical usage. That is why more HEVC encoding control algorithms were developed by multiple researchers [Rh12, Ce21]. These methods aim to reduce the complexity while maintaining the bitrate and quality of the decoded sequence. This dissertation is focused on CTU partitioning and PU division mode methods, which may be considered jointly in Intra mode. In this dissertation, such methods are called partitioning methods.

For most cases, the partitioning methods utilize the RD Optimization from HM Software to make the comparison fair. Methods for other levels of hierarchy (defined in Section 2.3.2), e.g. for angular prediction mode [Ya12, Ch13, So17, Ry18, Ja19, Xu19, Hu21B, Li24] or TU size [Lu20, Hu21A] were proposed in literature. Sometimes, these methods [Ch13, Xu19, Li24] are an internal part of the encoder control algorithms that estimate decisions for multiple hierarchy levels (Figure 2.7). The methods that focus directly on decisions for PU prediction mode and lower decisions in the hierarchy [Ya12, So17, Ry18] achieve ~2% of the time encoding time reduction with ~1% bitrate increase.

Less complex partitioning algorithms, compared to partitioning algorithms from HM software, employ the following methods of partitioning pattern estimation:

1. Hierarchical estimation of split flags of the quaternary tree (referred to as hierarchical approach) [Li16A, Xu18A, He21, Fe21].
2. Early termination of block size estimation in quaternary tree search [Ki13, Fe22].
3. A reduction of the search range, e.g., by setting depth levels to examine [Qi16, Lu19].
4. Estimation of the whole partitioning pattern at once [Kh13, Cr16].
5. Fast estimation of the bit cost of the block of a certain size [Ta17, Na24].

It is worth noticing that most of the methods proposed in the literature are suited for the compression of typical video content, represented in test sequences defined in Common Test Conditions for HEVC [CTCHEVC]. As the HEVC technique has multiple profiles for specific use cases, appropriately tailored

non-reference partitioning algorithms are proposed. Examples of these tailored profiles are 3D-HEVC [Ch19A, Ba22] and Screen Content Coding [Ku20B].

In the literature, multiple partitioning algorithm approaches were presented. To clarify the presentation, the survey of partitioning algorithms organizes these methods into two groups: Non-ANN and ANN. Each category will be discussed in a separate subsection.

2.4.1 Non-ANN-based approaches

Among non-ANN-based algorithms, the most straightforward method is the early termination of partitioning by data analysis [Ki13, Fe22]. Such a method frequently estimates other encoder decisions (Figure 2.7). The most popular solution is the hierarchical estimation of split flags [Zh14, Co15, Ce15, Li16A, Ga16, Ki16A, Zh18, Ya20A, Wa20, Li20A, Zh20B, Zh20, Ce21, We22]. The algorithms find the partitioning by estimating split flags in a top-bottom fashion. The estimation of the split flag may be considered uncertain [Li16A], and then the RD Optimization decides on the split. Two solutions were found [Kh13, Cr16], where the decision for partitioning was performed in a bottom-top fashion. Such an approach tries to estimate a partitioning pattern by assuming division into the smallest possible blocks and then joining them into bigger ones. Some researchers proposed different solutions for the partitioning problem. Instead of a hard-decisive algorithm, it is proposed to estimate block size ranges for a given CTU [Sh13, Qi16, Lu19, Me21B]. Such an algorithm may control the encoder to consider partitioning patterns, compared to a hierarchical estimation of split flags, but is more resilient to algorithm mistakes, so the bitrate increase is smaller compared to HM.

The type of input data differentiates the non-ANN-based algorithms. Most algorithms analyze the texture to decide within the partitioning process. Some methods classify the texture into a complexity category [Sh13, Zh18, Li20A, Zh20B, He21]. Other methods analyze the smoothness of the texture [Zh14, Li16A]. The smoother the texture is, the more probable that a bigger block should be used. Some algorithms determine keypoints, e.g., SIFT [Lo99], in the CU [Co15, Ki16A, Ya20A, We22]. Further, texture energy estimation [Zh20C, Me21B], Histograms of Oriented Gradients [Cr16], or Visual Saliency [Qi16] were employed to decide on the partitioning process.

Among the types of input data for the partitioning algorithms, the bit cost of the block was considered. A thresholding of such was presented as a viable solution [Ki13, Fe22]. Other methods consider the correlation with neighboring blocks [Sh13, Ce15, Lu19, Wa20, Me21B] as a context of the encoder influences the decisions. Another approach to leverage contextuality is to analyze the statistics of the decisions for previously encoded CTUs and frames [Ga16, Ki16A]. Such statistics may be estimated only offline [Ga16] or be updated during encoding [Ki16A]. Lastly, a partitioning algorithm may quickly estimate block bit cost and use it for split decisions.

The non-ANN partitioning algorithm may be categorized whether they employ ML or not. Among non-ML techniques, one can list texture analysis [Sh13, Zh14, Cr16, Qi16, Zh20B, Na24], context analysis [Ce15, Lu19, Me21B], and statistical analysis [Ga16, Ya20A]. The ML-based solutions worth mentioning are classifiers such as Decision Tree/Random Forest [Co15, Zh20B, He21, We22, Ts22], Bayesian [We22], or SVN [Li16A, Zh18]. Among ML-based algorithms, the ANN should be enlisted. Such approaches are described in the next subsection.

Determining the performance of the non-ANN partitioning algorithm may be difficult as they often combine decisions not only for CTU partitioning and PU division [Cr16, Ya20A, Wa20, Zh20B, Ja19]. Considering approaches only for CTU partitioning or CTU partitioning and PU division, the bitrate increase is relatively low, mostly in a range of $\langle 0.8\%; 1.2\% \rangle$, estimated with Bjøntegaard Delta [Bj01]. Regarding decreasing the encoding time, referred to HM, 10% to 53% was reported. In general, the higher the reduction in encoding time was, the more significant a bitrate increase was observed.

2.4.2 ANN-based approaches

ANN-based algorithm differs from typical approaches for partitioning algorithms. Instead of using expert knowledge to design the algorithm carefully, the ANN is trained to model the problem as precisely as possible. Considering the partitioning problem, this is done using a training dataset in most cases.

The definition of the problem is a crucial part of designing ANN-based algorithms. The ANN methods for the partitioning algorithm consider only CTU partitioning [Li17B, Fe18A, Xu18A, Ka18, Am18, Fe18B, Re19, Am20, Am21, Fe21] or CTU partitioning with PU division jointly [Wa18B, Sh19, Li22D, Zh22]. Some algorithms do not estimate PU divisions by the ANN but rather apply additional algorithms that use the ANN output. As this additional algorithm, the following were used: Naïve-Bias [Hu21B], Laplacian Transparent Composite Model [Am18], or statistical analysis [Zh23A]. An approach was proposed to encode the first frame with HM RD Optimization, and for the rest, ANN was used along with an analysis of spatial-temporal features [Zh23A].

The methods may be categorized by method type. Similarly to non-ANN-based approaches, the most popular method is the hierarchical estimation of split flags. Such methods may assume two patterns of operation. The first one is to run the ANN whenever the decision is needed [Yu15, Li16B, Li16C, Li17B, Ka18, Am18, Wa18B, Sh19, Ch20, Am20]. Most of the time, a different model is used for each depth level. The second operation pattern is to estimate all split flags a priori to the CTU encoding [Xu18A, Ya20B, Pa20, Am21, Li22D, Zh22]. Such flags are then used to encode the CTU, and decisions are stored in Division Matrices. The second operation pattern employs only one ANN model but with multiple outputs for each division level. The split flag may be considered uncertain [Xu18A, Hu21B], and then the RD Optimization decides on the split.

For other types of partitioning methods, there were proposed ANNs designed to estimate the range of the block sizes, which shall be considered during RD Optimization. Lastly, with the use of the ANN, it is possible to mimic the decisions of HM RD Optimization to estimate the whole partitioning pattern at once [Re19, Fe21, Li22C]. It is important to underline that such methods started appearing in literature parallel to the research presented in this dissertation [Lo21].

The ANNs employed in the partitioning algorithm, in most cases, process the CTU samples, specifically the luma component [Yu15, Li16B, Li16C, Fe18A, Xu18A, Fe18B, Sh19, Re19, Ch20, Am20, Am21, Li22D, Fe21, Zh22]. Some consider samples from neighboring CTUs as additional data [Ka18] or use only the partitioning of neighboring CTUs [Ka18]. Some authors utilized the bit cost of the partitioning pattern estimated by RD Optimization in HM [HM] as a variable in the loss function used during the training of model.

Partitioning algorithm with ANN uses multiple types of network architecture. The most popular ones are variations of the CoffeeLeNet[Le98] or AlexNet[Kr12]: [Yu15, Li16B, Li16C, Fe18A, Fe18B, Am20]. One of the modifications of such a network is an application of two [Hu21B] or three [Li17B, Xu18A] parallelly processing convolutional layer tracks whose outputs are concatenated before fully connected layers. With multiple convolutional layers tracks, asymmetric convolution kernels were proposed [Sh19, Ch20]. In other approaches, additional fully connected layers were added to provide multiple outputs from the model to simultaneously estimate split flags for all depth levels [Am21, Li22D, Zh22]. The method was found [Am18] that uses the fully connected ANN architecture. Simultaneously with the research presented in this dissertation [Lo21], a fully convolutional network became used in partitioning algorithms [Re19, Li22D, Fe21, Li22C, Zh23A]. More advanced architectures of convolutional networks, such as ResNets [Li21B, Zh22] and DenseNet [Zh21B], were employed in the partitioning algorithm. Still, the decrease in encoding time was not significant due to the complexity of the models.

For most method types, the ANN model decision may be interpreted in a way that always conforms with the HEVC syntax. Most of the time, the algorithms that estimate whole partitioning patterns at once have to adjust the output to the HEVC syntax. Such adjustments are made by application of the ArgMax function on ANN output [Li22C] or thresholding [Re19, Fe21].

The training dictates the performance of ANN-based approaches, which significantly depends on the training dataset. The most common approach is the use of uncompressed images or sequences. Many authors [Yu15, Li16B, Fe18A, Ka18, Wa18B, Fe18B, Re19] employ the JCT-VC dataset [CTCHEVC]. However, the results are biased as the same dataset is also used for evaluation. Other datasets used for ANN training are: RAISE [Da15] (~2000 images) [Li17B, Xu18A], DIV2k [Ag17] (900 images) [Li22D], CDVL[Pi13] or multiple datasets at once [Sh19, Ch20, Hu21B, Am21]. In method [Fe21], a part of the training images was downsampled to obtain lower-resolution images.

ANNs proposed in the mentioned method were trained in a supervised manner in most cases. A single method was found [Am18] that employed online training. In most methods, the CTU size was set as 64×64. The authors of [Ka18] used a CTU size of 32×32. The ANN architectures in [Zh22] take the *QP* as the additional input. However, in most methods, separate models are trained for each *QP*. The size of the networks differs among proposed methods. Models for the hierarchical estimation of split flags or block size range estimation, which use Alex-Net-like architecture, are quite big (~1 M weights) [Li17B]. More recent models for the hierarchical estimation of split flags or estimation of whole partitioning consist of tens of thousands of weights [Re19, Ch20]. Some methods are designed for direct application in hardware encoders [Li16B, Li16C] to highlight the practical aspect of the partitioning algorithm. Others, such as [Ch20], emphasize the parallelization of the partitioning process in software implementation.

There are multiple challenges regarding the comparison of ANN-based methods with non-ANN ones. One of them is the parallelization of ANN computation, which, with concurrent estimation of decisions for other levels of decision hierarchy [Ch20], makes the complexity comparison with single-treaded HM software [HM] difficult or even unfair. Another challenge is the use of different training datasets [Li17B, Sh19, Fe21] (size, content) or the test sequences used for the method evaluation [Am20]. Nevertheless, ANN-based methods offer encoding time savings of 40-70%, while the bitrate increase range is 1-5%, estimated with Bjøntegaard Delta [Bj01]. As with non-ANN, the higher the reduction in encoding time was, the more significant the bitrate increase. Due to the abovementioned factors, ANN-based methods are currently superior to non-ANN-based methods when considering the Intra mode of HEVC.

2.5 Methods of Encoding Time vs. Compression Efficiency trade-off control in the encoding process

The non-reference partitioning algorithms presented in the previous section offer a reduction of the encoding complexity but at the cost of increased bitrate. Such a relation is considered an Encoding Time vs. Compression Efficiency (ETvsCE) trade-off. Most methods offer a constant trade-off. Control over it can be useful, e.g., in multiple coding scenarios (in the server that encodes multiple video streams and the number of streams is varying) or when an encoder tries to fit the restriction of frame encoding time during transmission [Hu23]. In such situations, the change of *QP* may change the encoding time, but the quality changes, and the control is very coarse. Thus, it is beneficial that the partitioning algorithm allows control over the ETvsCE trade-off.

Few partitioning methods in literature allow control over the ETvsCE rate trade-off. Among non-ANN partitioning methods, authors of [Co16] proposed such a method by controlling the Pareto-based decision rule. Another approach, presented in [De16], uses spatial-temporal analysis of decisions to adjust the search ranges of the encoding control algorithm.

Considering the ANN-based partitioning algorithms, one method [Ch20] predefines sets of thresholds for splitting decisions. Such thresholds are estimated using an evolutionary algorithm as Pareto optimal points. Another method [Li20B] considers pruning the ANN weights to decrease the computational complexity. The last one [Hu21A] provides a heuristic model for control over multiple decision hierarchy levels.

Each of the presented methods has at least one important disadvantage. For some, the ETvsCE trade-off control is complicated [De16, Ch20, Li20B]. Others require multi-step, complicated, and precise setups of multiple parameters to offer control over such a trade-off [Co16, Ch20, Hu21A].

2.6 Summary of the state of the art

HEVC is a cornerstone video coding technique that is gaining importance nowadays. The gigantic number of ways to encode a CTU block requires a complex algorithm to control the encoding process. Another aspect that complicates the process is the impact of the coding context on the control algorithm. Such an algorithm performs a simplified RD Optimization to find suboptimal decisions. The most complex part of the optimization process is the CTU block partitioning. The CTU partitioning constitutes up to 95% of the encoding time. For Intra mode, the decision on the PU division may be considered an additional level of the quaternary tree. Thus, the PU division may be incorporated into the partitioning process.

In HM software, the reference model for the HEVC technique, the encoding control algorithm is very complex but achieves the best bitrate considering the state of the art. As this software and the encoding control algorithm are unsuitable for practical use, multiple approaches with a much less complex encoding control algorithm were proposed. Such approaches offer a particular Encoding Time vs. Compression Efficiency (ETvsCE) trade-off. Those approaches were surveyed and split into two categories: non-ANN and ANN-based. Considering the Intra mode, it was concluded that the ANN-based methods have a better potential for complexity reduction of the encoding control algorithm.

By the start of the research, the following aspects of partitioning algorithms have been spotted:

1. The ANN models used in literature are relatively big. There is significant potential for future studies to develop a smaller, simpler, less complex architecture.
2. When ANN is employed for the decision, the output of model is processed fairly simply, e.g., by thresholding or applying the ArgMax function. In this dissertation, more extensive decision algorithms are considered to provide a better partitioning algorithm.
3. CTU partitioning and PU division are rarely estimated using a single ANN. Modeling these decisions using a single ANN may result in a significant reduction in complexity.
4. Most partitioning algorithms utilize only CTU samples, especially the luma component. As the coding context impacts the decisions, the additional data input to the network may improve the partitioning algorithm.
5. Commonly, in the hierarchical estimation of split flags approaches, the output of the ANN estimates the probability of the split. When the estimation of whole partitioning is considered, the ANN may also be used to estimate values in the meaning of probability. In such cases, the output may be interpreted soft-decisively, so instead of one partitioning pattern, a precisely defined set of partitioning patterns may be checked to get a better bitrate. Another way to use such probabilities is to treat them as weighting coefficients in a global (frame/slice scope) partitioning optimization algorithm.
6. The RD Optimization algorithm used in HM gives a set of suboptimal decisions. Most ANN-based partitioning algorithms try to mimic such decisions in the training process. Considering that the model is always trained to with some error, and the contextuality of decisions in HM, the consecutive decisions from the ANN might be mismatched. The result

of this may be an increased bitrate. An ANN may be trained using multiple sets of suboptimal decisions to prevent this phenomenon.

7. The subject of control over the ETvsCE trade-off was discussed. Partitioning methods known from the literature that offer such a trade-off are challenging to control or require a complicated setup, e.g., an extensive process of operating point estimation. The need for a simply controlled method that does not require setup may be satisfied with the proper use of ANN, which outputs values in the meaning of probability and a tailored soft-decisive algorithm.

The most complex part of the optimization process is the partitioning of the CTU block, as it constitutes up to 95% of the encoding time. Further in this dissertation, research on developing the partitioning algorithm is presented. The Intra mode will be considered. A main aspect of the research is using the ANN as a part of such an algorithm.

3 Research methodology

3.1 Assessment of video encoder modification

Research in this dissertation focuses on video coding techniques. In particular, new partitioning methods are proposed. In practice, assessment of such methods generally requires modifying the encoder to integrate the proposed approach. The modified encoder is then evaluated using a set of test sequences to measure its performance. This methodology is employed also in this dissertation (Figure 3.1). This whole process is conducted within the Intra mode of the encoder.

Another challenge in the dissertation is that artificial neural networks (ANNs) are employed as the foundation for the developed partitioning methods. These ANNs are trained before use, based on a specifically prepared dedicated Training Dataset.

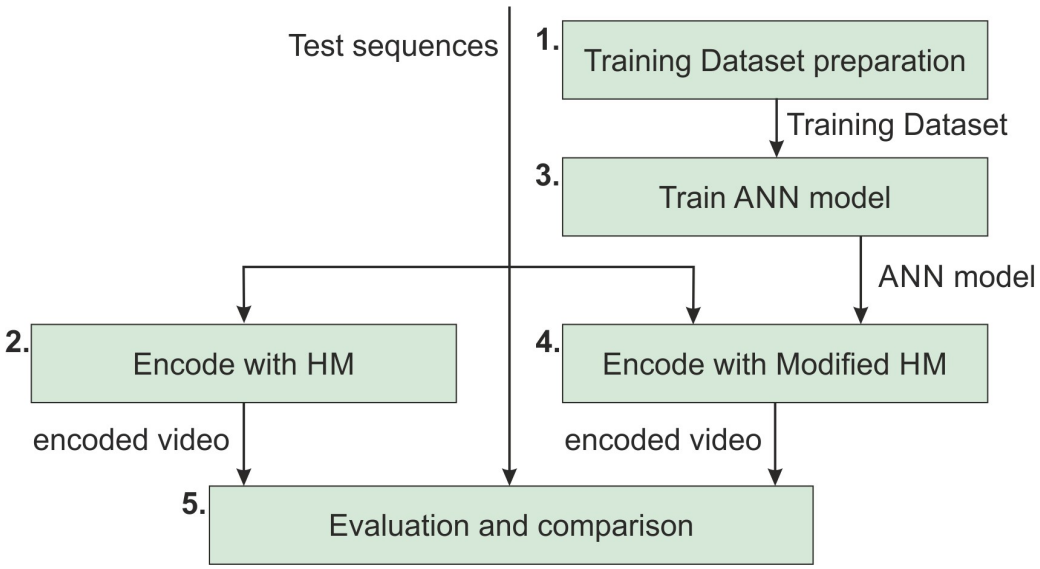

Figure 3. 1. Procedure for assessment of video encoder modification impact.

Therefore, the procedure outlined in Figure 3.1 was followed for assessment of the video encoder modifications. It consists of the following steps:

1. Prepare the Training Dataset (Section 4.2) for ANN training. The Training Dataset consists of CTU samples as input features and their corresponding partitioning patterns as ground truth labels.
2. Use HM [HM] to encode test sequences (described in Section 3.5). The version 16.23 of the HM was selected for this purpose.
3. Train the ANN model (described in Section 4.6)
4. Encode test sequences using Modified HM Software with trained ANN model. The modification of the HM with the implementation of ANN is described in Section 4.7.
5. Evaluate the results of encoding using Modified HM with the trained model and compare them with the result of encoding using HM. Evaluation and comparison are done for encoded sequence quality and encoding time. The ANN processing time is also evaluated for Modified HM with trained ANN models. Coding quality assessment requires test sequences. Details of coding quality assessment are presented in Section 3.2, and encoding time and ANN processing time in Section 3.3.

This assessment approach involves multiple ANN training, encoding, and decoding test sequences. Multiple repetitions of this process make experimenting laborious and very time-consuming. Step 1 of the presented procedure may be done once for multiple ANN models that share the same input data. Thus, the repetition of this step may be limited. Despite that, the requirements for proper encoding time assessment are rigorous and require specific hardware configuration, so precise execution of the process is demanding and time-intensive. Further discussion over the strategy of the experiment can be found in Section 3.7,

3.2 Coding quality assessment

A widely used metric for visual quality assessment is the Peak Signal-to-Noise Ratio (PSNR) [Bj01, Do12, Ka19, Dz22]. It is commonly applied in video coding research and is also utilized in HM software in Rate-Distortion (RD) optimization algorithm [HM, Wi03A, Su12, Br21A, Br21B, Co23]. In this dissertation, PSNR is employed as the primary metric for evaluating visual quality. It measures the change in image quality relative to a reference image, expressed in decibels (dB). This metric is calculated component-wise and is defined by the following formulas:

$$PSNR\ [dB] = 10\log_{10}\frac{[(2^{N_b}-1]^2}{MSE}, \tag{3.1}$$

$$MSE = \frac{1}{W \cdot \mathrm{H}}\sum_{i}^{H}\sum_{j}^{W}[f_{i,j} - r_{i,j}]^2, \tag{3.2}$$

where: N_b is the bit depth of the component samples, W and H are the width and the height of the image, $f_{i,j}$ and $r_{i,j}$ refers to sample values of assessed and reference images for i and j coordinates. For the video sequences, the $PSNR$ calculated for each frame is averaged.

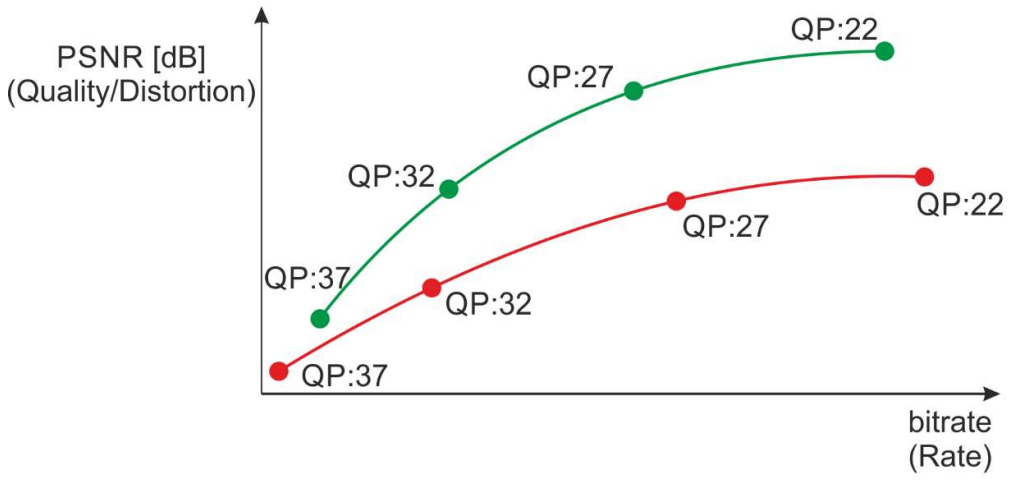

Figure 3. 2. Comparison of exemplary RD curves obtained using two video encoders with varying quantization parameter (QP) settings. Regarding quality-bitrate performance, the encoder represented by the green curve outperforms the one represented by the red curve.

As the quality of the decoded sequence is adjustable with the QP, a $PSNR$ metric is not sufficient to compare different encoding control algorithms. The Rate-Distortions curves (RD curves) [Do12, Ka19] are used to do so. The relative position of the corresponding RD curves is assessed to compare two encoders. An example of RD curves with a comparison of two encoders is presented in Figure 3.2.

The RD curves are a valuable tool in analyzing the impact of the modifications on specific sequences, but they are not so suitable for evaluating the broader effect across multiple sequences. The numerical comparison is made using the Bjøntegaard metrics [Bj01]. The Bjøntegaard metrics measure the difference between reference and tested methods. Two metrics are defined:

- $BD\text{-}RATE$ – metric dedicated to bitrate comparison (percentage), given the same quality of the decoded sequence. A positive value means the tested method produces a bigger bitrate than the reference method.
- $BD\text{-}PSNR$ – metric dedicated to quality comparison (in dB) given the same bitrate of encoded sequence. A negative value means worse quality for the tested method, given the same bitrate.

The visual interpretation of Bjøntegaard metrics is presented in Figure 3.3. Please note that while the calculation of both $BD\text{-}RATE$ and $BD\text{-}PSNR$ may seem redundant, it intended to identify potential evaluation artifacts, such as intersections of the RD curves or significant gaps between them. When such artifacts do not occur, the use of a single Bjøntegaard metric, e.g. sole $BD\text{-}RATE$, is sufficient for a reliable assessment. Such simplification is applied throughout the dissertation.

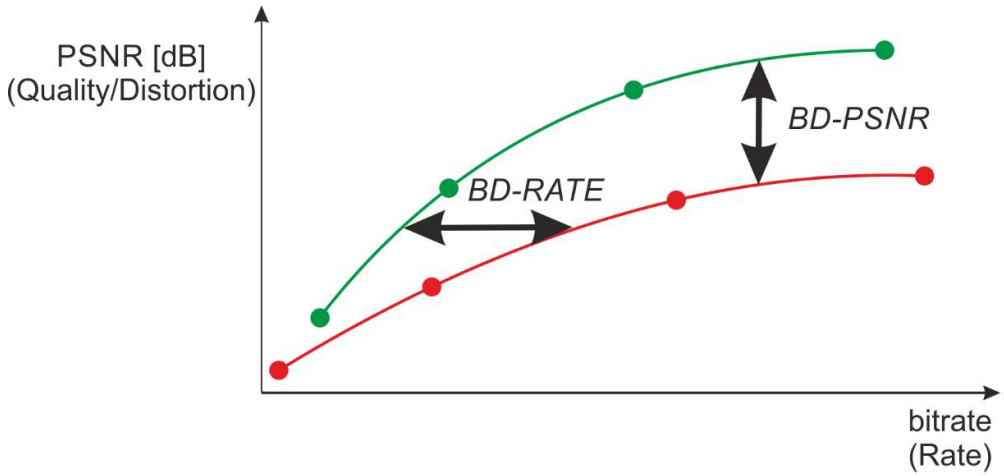

Figure 3. 3. Visual interpretation of Bjøntegaard metrics.

Bjøntegaard metrics are commonly used by researchers and are widely recognized; therefore, their description is considered common knowledge. Nevertheless, further in this dissertation (Section 3.4), author's novel metrics, derived from Bjøntegaard metrics, are proposed. Thus, the steps for calculating $BD\text{-}RATE$ and $BD\text{-}PSNR$ are presented hereafter:

a) For $BD\text{-}PSNR$: logarithm the input data points, for $BD\text{-}RATE$ do not modify input data.
b) Find the overlapping (shared) intervals of the rate and quality between RD points for reference and tested methods.
c) Find the approximation of the curves in overlapping intervals for reference and tested methods based on their RD points. The curve approximation differs among multiple variants of the metric. In this dissertation, the piecewise cubic spline interpolation [Ba97] is used.
d) Calculate the area under each of the curves. The sought-average differences are calculated through the mean of integrals.
e) Express the result: [%] for $BD\text{-}RATE$ and [dB] for $BD\text{-}PSNR$.

3.3 Encoding time measurement and assessment

The encoding time assessment is done with the Time Savings (TS) metric. The metric value determines the percentage reduction in time relative to the reference solution. The formula for TS is defined as follows:

$$TS = \left(1 - \frac{T_{tested}}{T_{reference}}\right) \cdot 100\%, \tag{3.3}$$

where T_{tested} is a processing time for the tested method and $T_{reference}$ is a processing time for the references method. The TS can return values in the range (-∞;100%). Negative values indicate an increase in complexity, the positive indicates a reduction,

To estimate the contribution of network processing time to coding time (T_{ANN}), expressed as a percentage, the following formula was used:

$$T_{ANN} = \left(\frac{T_{model}}{T_{tested}}\right) \cdot 100\%, \tag{3.4}$$

where T_{model} is an accumulated processing time of each ANN call and T_{tested} is a processing time for the tested method.

The computational complexity of the encoding process is commonly assessed by measuring the encoding time. Frequently, the assessment compares the encoding methods, and an important aspect is the use of comparable implementation of encoders. The measurement of execution time is not a trivial task [St06, Si20B]. When considering software executed on modern operating systems, one can list factors that impact such measurement:

- Operating system background task management.
- Operating system hidden tasks.
- Hardware platform.
- Hardware encumbrance.
- Time measurement method.

Considering the abovementioned factors, only one machine should be used to get the most accurate results, with one encoder instance at once. Simultaneously, the operating system should be monitored to prevent the use of hardware by applications other than the encoder. Such an approach would be very time-consuming. The specific arrangement for time-measurement experiments was used to accelerate the computations:

- Using the same machine to run the reference and tested method.
- Running multiple instances of the encoder parallelly as multiple tasks.
- Tasks for reference and tested methods run in parallel.
- For the encoding time measurement, the encoder build-in method was used (same in reference and tested method).
- The joint ANN processing time measurement was done by accumulating the processing time of each ANN call using the tested method. Time measurement for a single ANN call uses C++ standard library tools (std::chrono).

Time measurements collected using the presented procedure were compared with the single-machine and single-encoder instance approaches. The number of outlying results was noticeably more prominent using the proposed procedure, although the averaged results converge with the single-machine single encoder instance approach. Still, even with the proposed procedure, the time assessment

remains quite time-consuming. A computing cluster was used for experiment computation, containing multiple devices in multiple hardware configurations. Due to limited access to cluster (other users), the single-machine and single-encoder instance approach was still used in some cases.

3.4 Comparison of encoder control methods concerning Encoding Time vs. Compression Efficiency trade-off

3.4.1 Existing comparison techniques

Most encoder control algorithms reduce the complexity in some way but also increase the bitrate of the encoded video. As described in Section 2.5, this is referred to as Encoding Time vs. Compression Efficiency (ETvsCE) trade-off. The best way to compare encoder control algorithms regarding the ETvsCE trade-off is to use a graphical method similar to the RD curve. But in this case, the horizontal axis would refer to $BD\text{-}RATE$, and the vertical axis refers to TS.

A more numerical way to compare multiple methods, in terms of Encoding Time vs Compression Efficiency (ETvsCE) trade-off, is the Figure of Merit (FoM) metric [Na20, He20]. The formula for the metric is defined as follows:

$$FoM = \left|\frac{BD\text{-}RATE}{TS}\right| \cdot 100\%, \tag{3.5}$$

where $BD\text{-}RATE$ is the Bjøntegaard metric for bitrate delta, and TS is the Time Saving metric. For metric calculation, the values of $BD\text{-}RATE$ and TS for test sequences are averaged. A FoM is calculated separately for each method. FoM can have values in the range $(0; \infty)$, and in general, the smaller the value, the better.

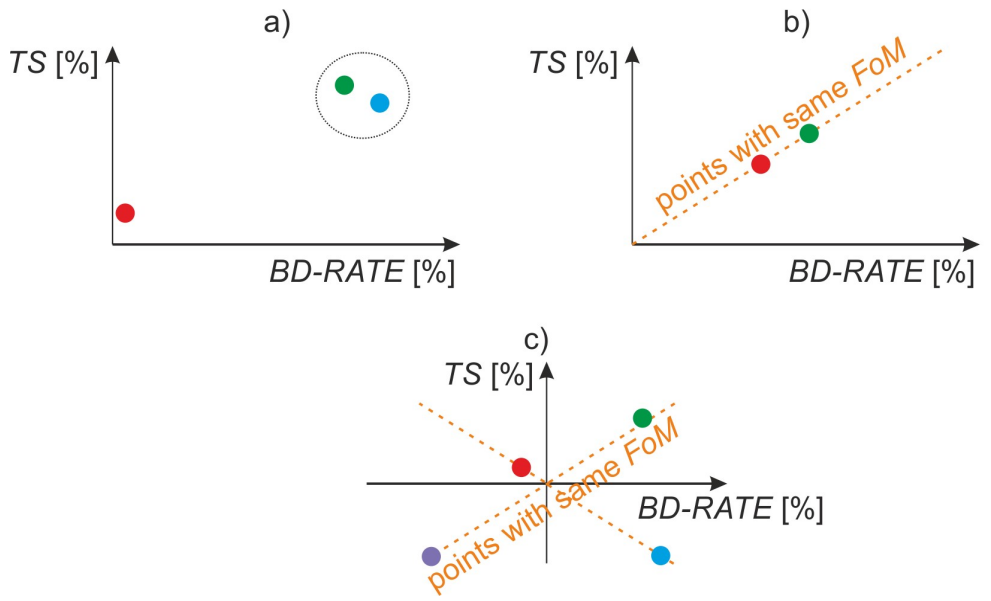

Figure 3. 4. Graphical representation of issues with FoM metric. Part a) shows that the outlying red point should not be compared with other points. Part b) shows two points with the same value of FoM, each of them may be superior depending on the use case. Part c) shows the FoM problem with negative values od TS and BD-RATE.

Mathematically, FoM represents the slope coefficient of a line, expressed in percentages. The absolute value is present due to $BD\text{-}RATE$ and TS value ranges. One can enlist problems corresponding to this metric:

a) A use of FoM for comparison is proper only when the compared methods are within relatively close ranges of $BD\text{-}RATE$ and TS. Otherwise, the comparison result may be misleading.
b) There is a possibility that two separate points may have the same FoM value. In such cases, this metric may ambiguously point out similar performance of compared methods. These methods should be considered separately depending on the specific use case.
c) FoM is misleading when one or both methods have a negative value of $BD\text{-}RATE$. A similar problem is spotted for negative values of TS.

The graphical representation of these problems is shown in Figure 3.4.

Still, FoM may be considered a viable metric when the encoding control algorithm offers a constant Encoding Time vs Compression Efficiency (ETvsCE) trade-off. For methods that provide control over such trade-off, the previously presented problems make this metric unsuitable for this task. A method that offers control over the ETvsCE trade-off will generate multiple points with possibly different FoM values, making the comparison difficult.

The graphical method may be used to perform a comparison. However, a comparison based only on visual clues (Fig. 3.4) may be imprecise and questionable. Also, it does not provide a mathematical metric to perform objective comparisons.

3.4.2 Proposed novel Encoding Time vs. Compression Efficiency trade-off metrics

To address the aforementioned challenges and enable an objective comparison of encoding control algorithms that offer a variable ETvsCE trade-off, **two novel metrics** are proposed in this dissertation:

- $\Delta BD\text{-}RATE|_{TS}$ as average $BD\text{-}RATE$ coding performance change between two curves while maintaining the same time saving TS.
- $\Delta TS|_{BD\text{-}RATE}$ as average time saving TS change between two curves while maintaining the same $BD\text{-}RATE$ coding performance.

The calculation of these metrics originates from Bjøntegaard metrics presented in Section 3.2. A detailed description of the metric value calculation will be presented here. Let us consider two datapoint sets, $Reference$ and $Test$, which consist of $BD\text{-}RATE$ values (BR for short) and TS values, representing two curves being compared (Figure 3.5). The $Reference$ datapoint set is defined as $\left(BR_{Ref_r}, TS_{Ref_r}\right)$, for $r \in [1; R]$. The $Test$ dataset is defined as $\left(BR_{Test_t}, TS_{Test_i}\right)$, for t $\in [1; T]$ (see Figure 3.5).

First, overlapping (shared) intervals BR_{int} and TS_{int} are found for values of the two compared datasets:

$$BR_{min} = Max(Min_r\left(BR_{Ref_r}\right), Min_r(BR_{Test_r}), \tag{3.6}$$

$$BR_{max} = Min(Max_r\left(BR_{Ref_r}\right), Max_r(BR_{Test_r}), \tag{3.7}$$

$$BR_{interval} = \langle BR_{min}; BR_{\max}\rangle, \tag{3.8}$$

$$TS_{min} = Max(Min_t\left(TS_{Ref_t}\right), Min_i(TS_{Test_t}), \tag{3.9}$$

$$TS_{max} = Min(Max_t\left(TS_{Ref}{}_t\right), Max_i(TS_{Test}{}_t), \tag{3.10}$$

$$TS_{interval} = \langle TS_{min}; TS_{max} \rangle. \tag{3.11}$$

Then, each curve is represented with the use of piecewise cubic spline interpolations [Ba97] in the form of:

$$\widetilde{BR_k}(TS) = a + b_k \cdot TS + c_k \cdot TS^2 + d_k \cdot TS^3, \tag{3.12}$$

$$\widetilde{TS_k}(BR) = e_k + f_k \cdot BR + g_k \cdot BD^2 + h_k \cdot BR^3, \tag{3.13}$$

where $a_k \ldots h_k$ are coefficient for each spline interpolation of segment k. Concatenation of these segments, yields piecewise spline representations of the $Reference$ and $Test$ curves, in the function of TS and in the function of $BD\text{-}RATE$ (BR for short), respectively: $\widetilde{BR_{Ref}}(TS)$, $\widetilde{BR_{Test}}(TS)$, $\widetilde{TS_{Ref}}(BR)$, and $\widetilde{TS_{Test}}(BR)$.

Finally, the sought average differences are calculated through the mean of integrals (|interval| refers to the length of the interval):

$$\Delta BD\text{-}RATE|_{TS} = \frac{1}{|TS_{interval}|} \int_{TS_{interval}} \widetilde{BR_{Test}}(\tau) - \widetilde{BR_{Ref}}(\tau)\, d\tau, \tag{3.14}$$

$$\Delta TS|_{BD\text{-}RATE} = \frac{1}{|BR_{interval}|} \int_{BR_{interval}} \widetilde{TS_{Test}}(\rho) - \widetilde{TS_{Ref}}(\rho)\, d\rho. \tag{3.15}$$

The process does not involve logarithmization of the input data (as in the case of $PSNR$ in the Bjøntegaard metric). The unit of the proposed metrics is percent points (p.p.) as they express the difference between values expressed in percents (%).

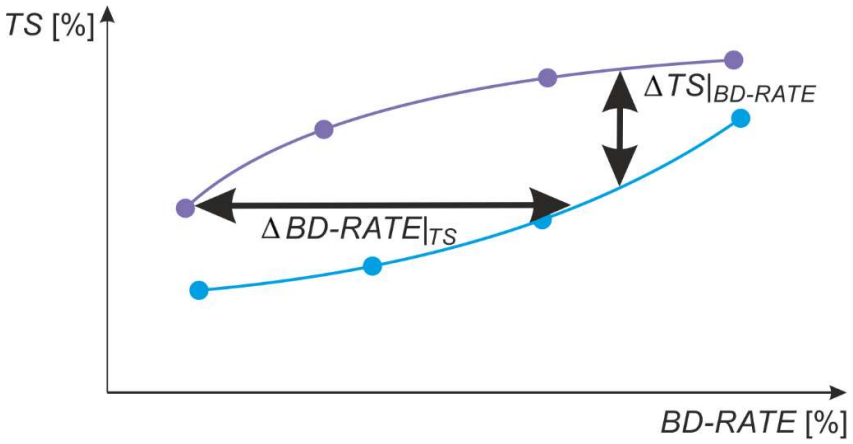

Figure 3. 5. Visual representation of proposed metrics: $\Delta BD\text{-}RATE|_{TS}$ and $\Delta TS|_{BD\text{-}RATE}$.

3.5 Test sequences

The Common Test Conditions (CTC) [CTCHEVC] for HEVC defines encoder configurations for coding scenarios: All Intra, Random Access, and Low Delay. This dissertation is focused on the All Intra scenario. The assessment for All Intra scenario presupposes conditions for standard video content

and encoding only in Intra prediction mode. Four *QP* values are defined for the coding quality assessment: {22, 27, 32, 37}.

The test sequences set defined in CTC for HEVC is widely known as the JCT-VC dataset [CTCHEVC] and was used for the development of multiple techniques by MPEG. In this dissertation, JCT-VC test sequences are referred to as the test sequence set or test sequences to differentiate it from datasets used for ANN training. The JCT-VC test sequence set for HEVC is composed of 20 sequences. The sample format is YCbCr, and the 4:2:0 chroma subsampling scheme is used. Test sequences are divided into five resolution classes. Sequences last from 5 to 10 seconds with different framerates. The bit depth is 8 bits, but for two sequences of higher resolution, the bit depth is 10. A detailed description of test sequences is presented in Table 3.1. Exemplary frames from the test sequences set are presented in Figure 3.6.

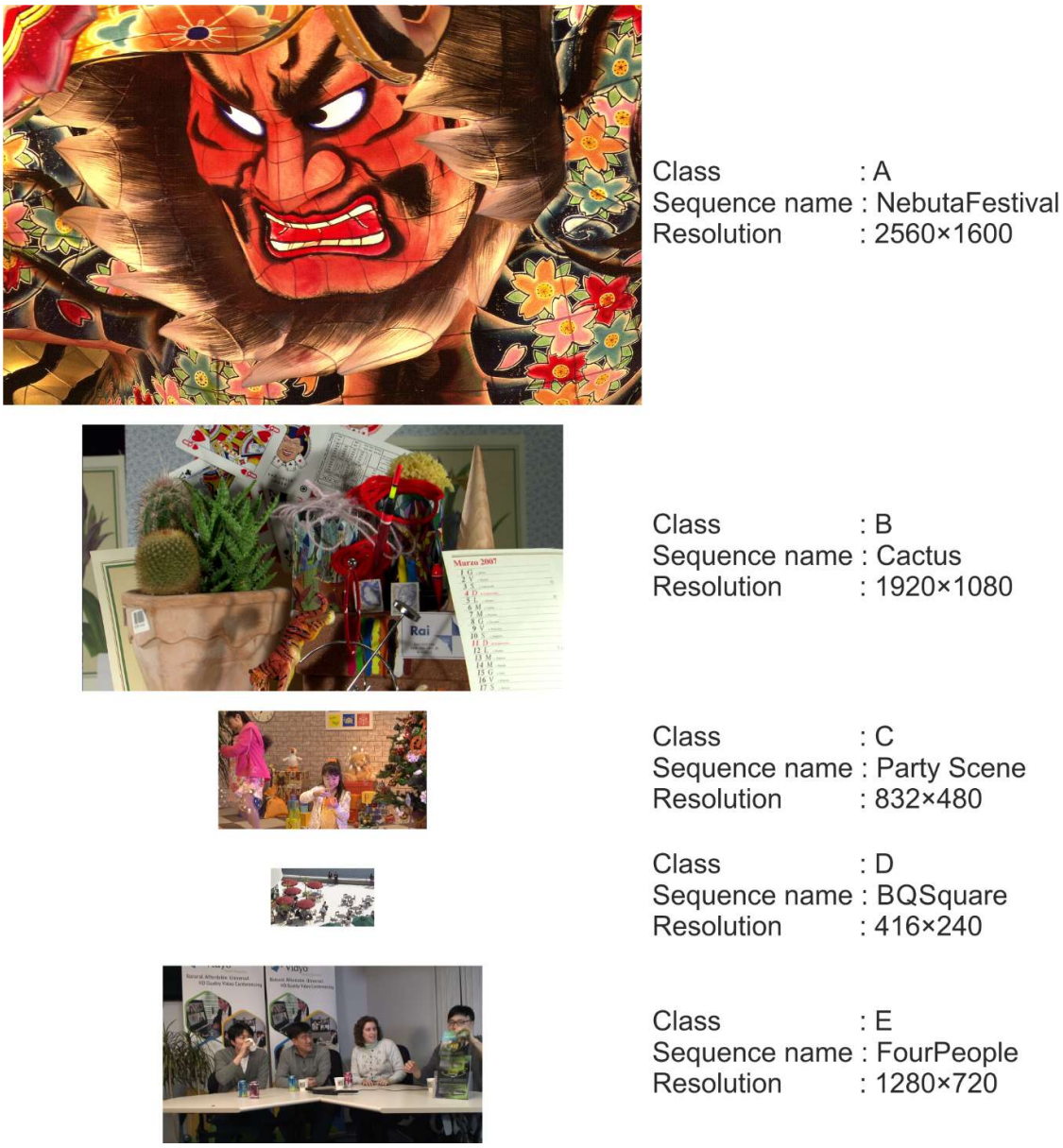

Figure 3. 6. Exemplary frames from the test sequence set.

The methods proposed in this dissertation were tested using two additional content types. The first type is 4k (3840×2160) resolution sequences. The CTC [CTCHEVC] for HEVC 4k resolution sequences for standard content. Thus, the test sequences for classes A1 and A2, defined in CTC for VVC [CTCVVC], were used. The second type of additional content is a set of sequences from the Screen Content Coding scenario. Such sequences are defined in class F of CTC for HEVC [CTCHEVC]. A detailed description of additional content test sequences is presented in Table 3.2. Use of sequences indicated as additional content is presented in Section 9.2.

Table 3. 1. Sequences in JCT-VC test sequence set.

JCT-VC class	Sequence Name	Resolution	Number of frames	Framerate	Bit depth
A	NebutaFestival	2560×1600	300	60	10
	PeopleOnStreet	2560×1600	150	30	8
	SteamLocomotiveTrain	2560×1600	300	60	10
	Traffic	2560×1600	150	30	8
B	BQTerrace	1920×1080	600	60	8
	BasketballDrive	1920×1080	500	50	8
	Cactus	1920×1080	500	50	8
	Kimono1	1920×1080	240	24	8
	ParkScene	1920×1080	240	24	8
C	BasketballDrill	832×480	500	50	8
	BQMall	832×480	600	60	8
	PartyScene	832×480	500	50	8
	RaceHorses	832×480	300	30	8
D	BasketballPass	416×240	500	50	8
	BlowingBubbles	416×240	500	50	8
	BQSquare	416×240	600	60	8
	RaceHorsesLow	416×240	300	30	8
E	FourPeople	1280×720	600	60	8
	Johnny	1280×720	600	60	8
	KristenAndSara	1280×720	600	60	8

Table 3. 2. Test sequences for evaluation of additional content.

JCT-VC class	Sequence Name	Resolution	Number of frames	Framerate	Bit depth
A1 (VVV)	Tango2	3840×2160	294	60	10
	FoodMarket4	3840×2160	300	60	10
	Campfire	3840×2160	300	30	10
A2 (VVC)	CatRobot	3840×2160	300	60	10
	DaylightRoad2	3840×2160	300	60	10
	ParkRunning3	3840×2160	300	50	10
HEVC F	ChinaSpeed	1024×768	500	30	8
	BasketballDrillText	832×480	500	50	8
	SlideEditing	1280×720	300	30	8
	SlideShow	1280×720	500	20	8

3.6 Artificial Neural Network models

3.6.1 Model training assessment

In this section, only metrics and methods for model training assessment are described. The training procedure and loss function will be presented in Section 4.6. The ANNs used in proposed partitioning algorithms are trained for multiclass classification problems and require Ground Truth data for training. These aspects will be rationalized in Chapter 4.

The assessment of trained models is done with a set of metrics. These metrics are calculated for both training and validation datasets. The first presented metric for model training assessment is the

model $Accuracy$ [Ha08, Go16]. An exact formula used for assessments of proposed modes will be defined in Subsection 4.6.2. The $Accuracy$ metric (4.10) originates from $Categorical\ Accuracy$, which is defined as follows:

$$Categorical\ Accuracy = \frac{1}{N}\sum_{n}^{N} \mathrm{Iv}(\mathrm{ArgMax}(y_n) = \mathrm{ArgMax}(y'_n)), \tag{3.16}$$

where: N – number of outputs, y_i – i-th ground truth value, y'_i - i-th predicted value, $\mathrm{Iv}(\cdot)$ is the Iverson function [Fo99], such that $\mathrm{Iv}(true) = 1$ and $\mathrm{Iv}(false) = 0$.

a)

Classifier Prediction (p) \ Ground Truth (g)

class index	0	1	2	3
0	555	274	162	9
1	91	734	104	71
2	21	37	932	10
3	139	183	258	420

b)

Classifier Prediction (p) \ Ground Truth (g)

class index	0	1	2	3
0	13.9	6.9	4.1	0.2
1	3.5	18.4	2.6	1.8
2	0.5	0.9	2.33	0.03
3	3.5	4.6	6.5	10.5

Figure 3. 7. Exemplary Confusion Matrix. a) Confusion Matrix with counts (Formula 3.17). b) Confusion Matrix with values expressed in percents (Formula 3.18)

In the case of classifiers, a commonly used assessment tool is Confusion Matrix (CM) [Ha08, Go16]. An exemplary matrix is presented in Figure 3.7. CM is a 2-dimensional square matrix of size equal to the number of classes. Values of the CM are determined with the formula:

$$CM_{count}[p, g] = \sum_{n}^{N} \mathrm{Iv}(\ g = \mathrm{Iv}(p = \ \mathrm{ArgMax}(y'_n)\ )\ ), \tag{3.17}$$

where, y'_n is a prediction of the classifier for the n-th sample, N is the number of samples in the dataset, p and g are successively the row and column indices of CM. The digit of CM may be referred to as the count of predictions of p-th class when the reference (Ground Truth) class was g-th. CM of the ideal classifier would be a diagonal matrix. When the number of samples in the dataset is huge, values in CM are normalized (divided by the number of samples) and expressed in percentage (multiplied by 100) for a more straightforward analysis. Thus the following formula is used :

$$CM_{\%}[p, g] = \frac{100\%}{N} \cdot\ CM_{count}[p, g]. \tag{3.18}$$

For CM analysis, a set of metrics is defined [Ha08, Go16]. Formulas for enlisted metrics are defined for multiclass classifiers. In this dissertation, the values of the presented metric are expressed in percentage (multiplied by 100%). The K refers to the number of classes. Here are presented metrics for CM analysis used in this dissertation:

- True Positive Count for class d – TP_d – counts correct predictions of class d.

$$TP_d = CM_{\%}[d, d] \cdot 100. \tag{3.19}$$

- False Positive Count for class d – FP_d – count of predictions of class d when Ground Truth indicates another class.

$$FP_d = \sum_{\substack{k \\ k \neq d}}^{K} CM_{\%}[d, k]. \tag{3.20}$$

- True Negative Count for class d – TN_d – count of correct predictions of another class when another class is indicated in Ground Truth.

$$TN_d = \sum_{\substack{l \\ l \neq d}}^{K} \sum_{\substack{k \\ k \neq d}}^{K} CM_{\%}[l, k]. \tag{3.21}$$

- False Negative Count for class d – FN_d – count of prediction of another class when the d class is indicated in Ground Truth.

$$FN_d = \sum_{\substack{k \\ k \neq d}}^{K} CM_{\%}[k, d]. \tag{3.22}$$

- Recall for class d - $Recall_d$ – metric for measuring the ability of a model to identify all instances of class d.

$$Recall_d = 100\% \cdot \frac{TP_d}{TP_d + FN_d}. \tag{3.23}$$

- Precision for class d - $Precision_d$ - metric for measures of the ability of a model to identify instances of a class d correctly

$$Precision_d = 100\% \cdot \frac{TP_d}{TP_d + FP_d}. \tag{3.24}$$

- F-score (balanced) for class d - $F\text{-}score_d$ - harmonic mean of $Precision_d$ and $Recall_d$

$$F\text{-}score_d = 2 \cdot \frac{Precision_d \cdot Recall_d}{Precision_d + Recall_d}. \tag{3.25}$$

- Micro-averaged metrics – metrics for assessment of a classifier across different datasets of different sizes (D defines the number of datasets)

$$Recall_{MicroAvg} = 100\% \cdot \frac{\sum_i^N TP_d}{\sum_i^N (TP_d + \mathrm{FN_d})}. \tag{3.26}$$

$$Precision_{MicroAvg} = 100\% \cdot \frac{\sum_i^N TP_d}{\sum_i^N (TP_d + FP_{\mathrm{d}})}. \tag{3.27}$$

- Macro-averaged metrics - metrics for assessment of a classifier across different datasets of the same size (N defines the number of datasets)

$$Recall_{MacroAvg} = \frac{1}{D}\sum_{d}^{D} Recall_d. \tag{3.28}$$

$$Precision_{MacroAvg} = \frac{1}{D}\sum_{i}^{D} Precision_d. \tag{3.29}$$

The model complexity is measured by model depth (number of layers), number of weights, and Multiply and Accumulate (MAC) operations count. These numbers are estimated by the software used for model training [Ab16].

3.6.2 Fundamentals for training dataset preparation

As mentioned in Section 3.1, the Training Dataset consists of CTU samples as input features and their corresponding partitioning patterns as ground truth labels (estimated by the HM encoder [HM]). In general, in ANN training, the bigger and more diversified the content of the training dataset, the better ANN can model the problem [Ha08, Go16]. Currently, ANN training is a very popular research task in computer vision. The most popular datasets are COCO [Li15] and ILSVRC (ImageNet) [De09]. These datasets contain hundreds of thousands to millions of images. Two issues, that apply to these (and many other) machine vision datasets, are: relatively small resolution of images and, more importantly, image compression using JPEG [JPEG].

Decisions of the encoder control algorithms highly differ depending on the resolution. More visually dense blocks generally are present in smaller-resolution pictures. Higher-resolution pictures tend to have more flat texture areas or complex textures in CTU. In such cases, the partitioning used will significantly differ. The second mentioned issue is the pre-encoding of the picture. Every lossy video coding technique introduce visual artifacts. The higher the compression – the more artifacts may occur. The ANN tends to find certain features in the data to use them in classification [Ha08, Go16]. Occurrences of such artifacts may highly impact the decisions of such models. As the information on encoding conditions for datasets is mostly unavailable, the negative impact of encoding artifacts on the training process would be extremely difficult to prevent.

The abovementioned issues with the most popular machine vision datasets make them unsuitable for the research presented in this dissertation. However, a few publicly available datasets are suitable for this task. Such candidates for ANN training are JCT-VC [CTCHEVC], RAISE [Da15], UCID [Sh04, Sh10] and DIV2k [Ag17].

The first one, the JCT-VC, was used in multiple approaches found in the literature [Yu15, Li16B, Fe18A, Ka18, Wa18B, Fe18B, Re19], but this dataset is used for encoder modification assessment. Using this dataset in training would make the comparison with state-of-the-art solutions unfair.

The RAISE [Da15] dataset consists of 8156 images, each with a robust description of the image source. The resolution of those images is always 4k (3840×2160). All images are available in two raw formats – straight from the camera sensor (Bayer matrix) and after simple processing. The bit depth of the image samples is 12 or 14 bits.

A similar dataset to RAISE is UCID, which consists of 1338 [Sh04] or 10 000 images [Sh10]. All images are raw output from the camera, without any postprocessing. Two cameras were used to obtain all images. Despite the large number of images, the content is not so diverse as multiple images represent the same scene. Moreover, the images' resolution is relatively small, ~500×300 samples.

The DIV2k dataset [Ag17] is divided into two subsets: training (800 images) and validation (100 images). Images in the dataset have multiple resolutions and vertical or horizontal orientations. The

resolution of the pictures is around 2000 in at least one dimension. Images are available in .png format. The bit depth of the samples is always 8.

Due to the abovementioned reasons, for the sake of ANN training, the DIV2k dataset has been chosen in this dissertation. Multiple resolutions better fit the task of developing the encoding control algorithm. Additionally, the content of the DI2k pictures is more diversified, e.g., landscapes, flora, fauna, and people (portraits, execution of activities). Figure 3.8 shows some exemplary pictures from the DIV2k dataset. Images are available in .png format and do not require additional preprocessing like for RAISE or UCID, which could impact the ANN training. This dataset should be large enough, as the training samples are CTUs.

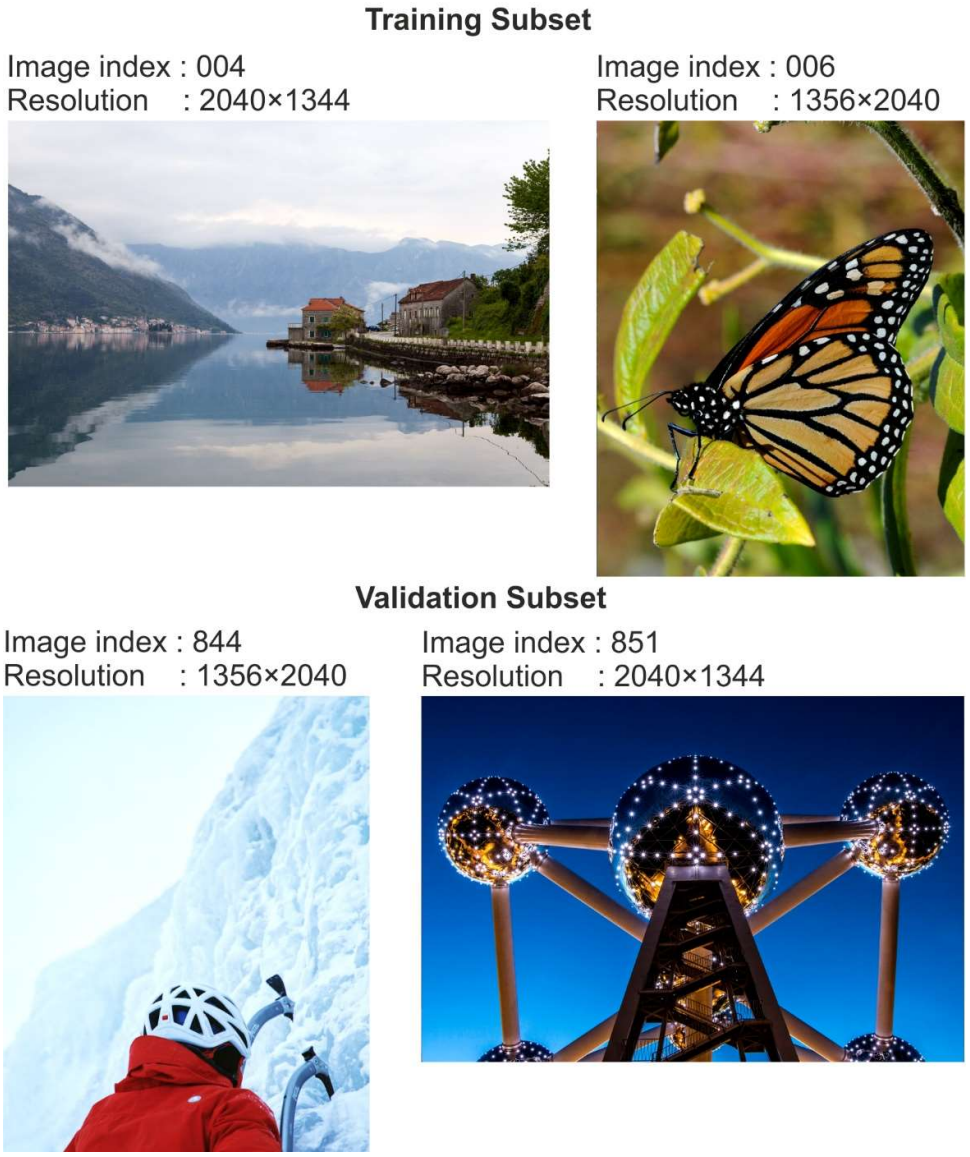

Figure 3. 8. Exemplary images from the DIV2k dataset.

Another viable option is using multiple datasets, like in [Sh19, Ch20, Hu21B, Am21, Fe21]. The lower-resolution images can be produced using subsampling [Fe21]. Due to hardware limitations, a bigger training dataset would be problematic for the research presented in this dissertation. Still, during

the development of the partitioning algorithms, the DIV2k was sufficient. A discussion about using a larger training dataset can be found in Section 9.7.

3.7 The strategy of the experiments

For test sequences (presented in Section 3.5) and training dataset (presented in Section 3.6.2), the training and evaluation last from 3 to 8 days using a single machine (all available threads) from a computing cluster. This time varies depending on the *QP* and the computational hardware. The acceleration of experiments using multiple machines was possible only in the encoder evaluation phase. The training of the ANN was always done using one machine.

During the development of an ANN model and hyperparameter tuning, the quality assessment may be more important than the precise encoding time assessment. Thus, to accelerate the experimenting, the same results of HM encoding were used for evaluation, so only steps 3 -5 of the procedure presented in Section 3.1 (Figure 3.1, Page 37), had to be performed. In this case, all available resources in the computing cluster could be utilized. Then, the whole procedure (Figure 3.1, Page 37), following constraints and recommendations for time assessment (Section 3.3), was performed for the most promising models. A crucial aspect is that the same hardware of the machines is required to split tasks into multiple machines, and the full access to the computing cluster was limited.

Thus, evaluating all combinations of models with decision algorithms and hyperparameters during doctoral research was practically impossible. That is why, in this dissertation, the greedy research strategy was applied:

- Preparation of a batch of experiments, with some ideas, to evaluate.
- Training of models and evaluation in the encoder without precise time assessment.
- Analysis of experiments results.
- Identifying currently best and promising ones and performing precise time assessments.
- Further development of most promising research directions.
- Repetition of the process.

Experimenting with the use of HEVC was a laborious and time-consuming process. At the beginning of the research, the VVC was considered to be used in the research. However, early experiments have shown that the encoding time is around 100 times longer. Such long encoding means that the research will not be performed in a given time, considering available computational resources.

4 General idea of the proposed solution

4.1 Proposed approach and general description of used Artificial Neural Networks

The partitioning algorithm which uses ANN is the core idea of the research presented in this dissertation. The partitioning is considered for CTUs of size 64×64 and All Intra mode. The innovative aspect of the proposed method is a non-trivial decision algorithm that processes the ANN output. The ANN is designed to output a tensor, which is then used to estimate the whole partitioning pattern at once. Such an ANN was novel by the time the research started [Lo21] and was developed simultaneously with other similar methods [Re19, Fe21, Li22C]. Another novelty in the dissertation is that the decision algorithm has the flexibility to indicate either a single partitioning pattern or a set of partitioning patterns to examine, depending on the certainty of the ANN model. Furthermore, this flexibility in the decision algorithm allows control over the Encoding Time vs Compression Efficiency (ETvsCE) trade-off. The control should be as simple as possible in terms of the number of adjustable parameters and the ease of setup to surpass the limitations of the current state-of-the-art techniques mentioned in Section 2.5. Details on decision algorithms presented in this dissertation are presented in Chapter 7.

The proposed partitioning algorithms presented in this dissertation are evaluated with the use of the Modified HM. A detailed description of the Modified HM implementation is provided in Section 4.7. The modification of the HM software ensures a proper assessment of the partitioning algorithm in reference to both HM software and other approaches found in the literature. The assessment of the proposed partitioning algorithms is presented in Figure 4.1.

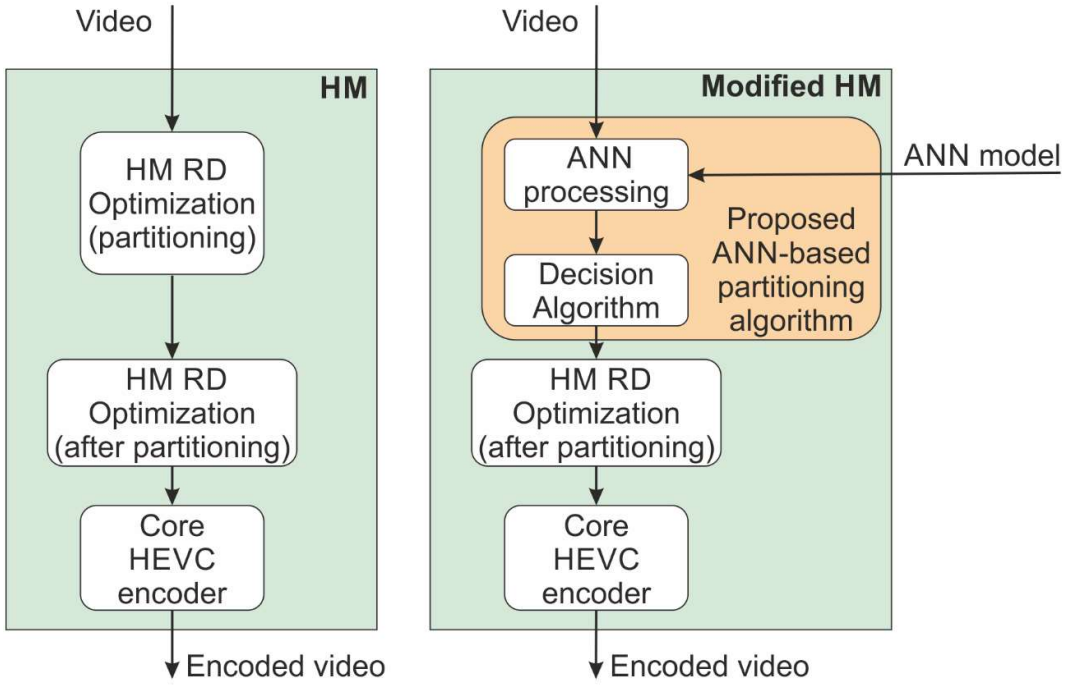

Figure 4. 1. Assessment of proposed partitioning algorithm in Modified HM encoder (Section 4.7) with regard to the Reference HEVC encoder (HM). The ANN model is trained using the methodology presented in Section 3.6, the training dataset presented in Section 4.2, and the training procedure presented in Section 4.6.

In this dissertation two approaches to the partitioning problem are defined, further impacting the ANN architecture. The difference between approaches to partitioning algorithms is the range of the decisions from the hierarchy shown in Figure 2.7. In the first approach, called the **Basic Approach**, the partitioning algorithm estimates only CTU partitioning: only CU block sizes are estimated, as shown in Figure 4.2. In the second approach for the partitioning problem, called the **Extended Approach**, CTU partitioning and PU division are considered jointly. The range of decisions is shown in Figure 4.3. As presented in Section 2.2, considering the Intra mode, the PU division can be applied only in CUs of size 8×8. Moreover, the available PU division modes are single 8×8 PU or four 4×4 PUs in this case. Therefore, the PU division can be treated as an additional level of the quaternary tree for CTU partitioning.

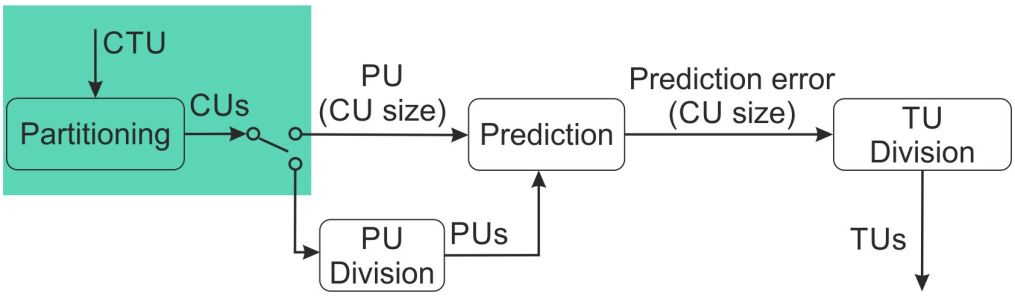

Figure 4. 2. Scope of encoding decisions (green) considered in Basic Approach.

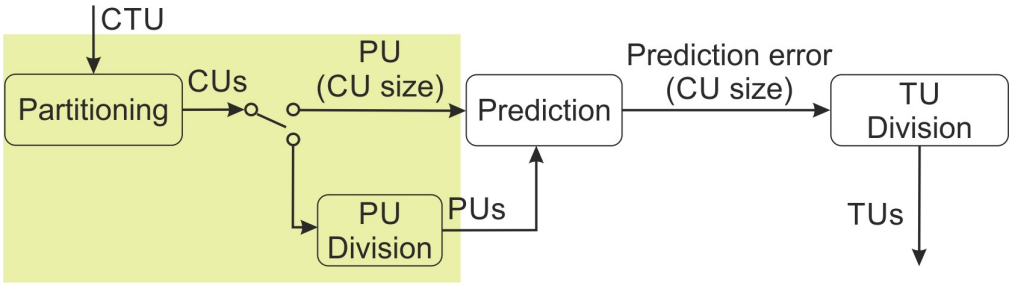

Figure 4. 3. Scope of encoding decisions (yellow) considered in Extended Approach.

The core of the proposed algorithms is the ANN. In this dissertation, an ANN is considered, which is trained to mimic the partitioning patterns estimated by the RD Optimization algorithm in HM (reference model software for HEVC) [HM]. The architecture of the ANNs is adjusted to the particular approach (Basic or Extended) used in the partitioning algorithm. Training of such models requires extracting partitioning patterns evaluated by HM, provided by the decoder in Modified HM (Subsection 4.7.1).

The ANNs proposed in this dissertation were designed considering arrangement of the architecture into two functional subnetworks (Figure 4.4), both combined and trained as one model. The first one, referred to as Subnetwork $\mathbb{A}$, is intended to extract the latent representation [Ha08, Go16] and reduce redundant information. Simultaneously, this subnetwork should reduce the shape of the resulting feature maps [Ha08, Go16]. The task of the second subnetwork (referred to as Subnetwork $\mathbb{B}$) is the classification of the desired block size for specific subareas of the CTU. This is done by a layer arrangement that reflects a quaternary tree. The output of Subnetwork $\mathbb{B}$ are values of probability meaning that is then used by the decision algorithm. Training data analysis and the discussion on the ANN learning problem are presented in Section 4.2. The assumptions for the ANN model are outlined in Section 4.3. The ANN inputs the same data as the HM algorithm, as described in Section 4.4. The output of the ANN is designed to estimate the whole partitioning pattern at once and simultaneously

deliver as much information as possible to process for the decision algorithm. The output of the ANN is discussed in Section 4.5. Then, in Section 4.6, the training procedure for the ANN models is presented.

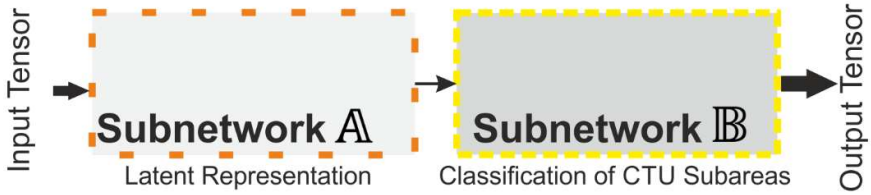

Figure 4. 4. Conceptual diagram of the proposed ANN architecture.

The motivation behind the proposed partitioning methods is to reduce the number of partitioning patterns that will be considered for CTU in the encoding process. The number of partitioning patterns to be considered is reduced from quintillions of possibilities (Section 2.2) to just one (when hard-decisive variants of the proposed algorithms are used – Section 7.2) or just a few (to address uncertainty of the ANN, with the use of soft-decisive variants of the algorithms – Section 7.3). Figure 4.5 visualizes proposed approaches with the hierarchy of decisions and the number of partitioning patterns to check.

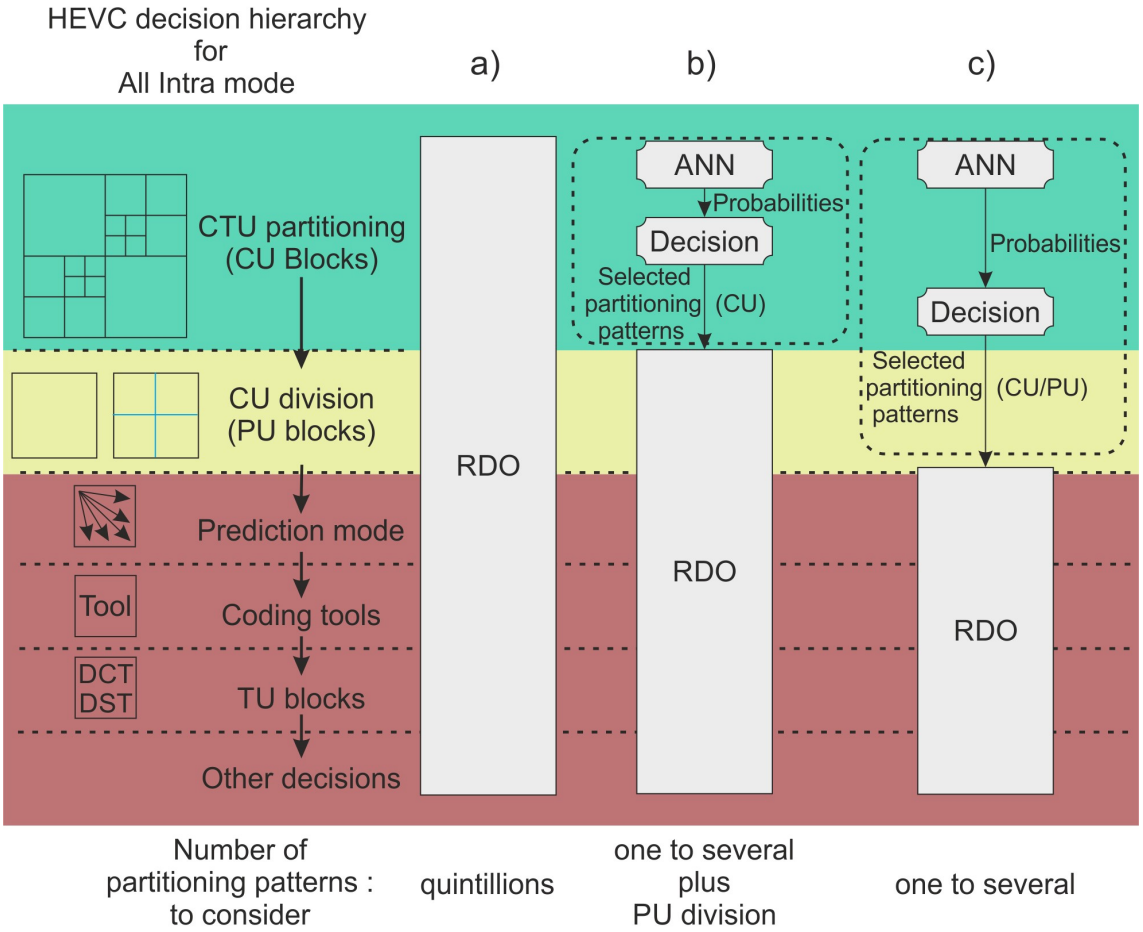

Figure 4. 5. Encoder control approaches matched up with decision hierarchy in HEVC encoder: a) classical RDO (Subsection 2.3.2), b) proposed Basic Approach, and c) proposed Extended Approach. Below the approaches, the number of partitioning patterns to consider in each approach is presented.

4.2 Training data preparation and analysis

The DIV2k dataset [Ag17] is used to prepare the training dataset. This dataset is composed of images in various resolutions, with both horizontal and vertical orientations. For some images, the resolution is either non-divisible by the size of the smallest possible CU (8×8) or is an odd number. When the image resolution is not divisible by CTU size, a CTUs spanning beyond the boundary of the image would be produced as described in Subsection 2.3.3. Despite the minority of such CTUs, their negative impact is expected, as is discussed in Section 9.4. Therefore, the preparation of the training image preparation is designed to exclude such CTUs.

The first step of training dataset preparation is cropping of the images into resolutions divisible by CTU size (64×64) to avoid training samples with CTU spanning beyond the boundaries of the picture. A cropped section is always in the middle of the image to remove the same amount of samples from each image side. The cropped section consists of the regular content of the image. The dataset of cropped DIV2k images is then used to prepare the training samples.

The second step of training data preparation is the image format conversion. The HM encoder [HM] is adapted to encode sequences in YCbCr color space. Test sequences [JCTVC] have also applied the chroma subsampling 4:2:0 [Do12, Ka19]. The chroma subsampling is very significant, as all components impact the decision of the RD Optimization in HM, as mentioned in Section 2.2. Thus, cropped DIV2k images were converted to the desired color space and chroma subsampling. The conversion followed the standard [BT709] and was done with state-of-the-art methods from the ffmpeg software [FFMPEG].

The third step is encoding the converted DIV2k images using the HM [HM]. The encoding is performed using the "All Intra" scenario with each QP enlisted in CTC [CTCHEVC]: {22, 27, 32, 37}. A separate training dataset has been prepared for each QP. Then, a decoder form Modified HM (described in Subsection 4.7.1) is used to extract the partitioning patterns for each encoded image (Division Matrices for CU - DM_{CU} - and Division Matrices for PU - DM_{PU}).

The last step is the setup of the training dataset for each QP. **A sample in the training dataset consists of a pair: CTUs luma component and corresponding partitioning pattern estimated by HM** [HM]. Partitioning patterns may include only DM_{CU} for Basic Approach or both DM_{CU} and DM_{PU} for Extended Approach From the DIV2k dataset is prepared a following training dataset (for single QP):

- Training Subset of 522 939 samples (extracted from 800 images).
- Validation Subset of 66 650 samples (extracted from 100 images).

Before the development of the actual ANN, the training dataset was first analyzed to define the learning problem for the model accordingly. As mentioned in the previous section, the ANN should be a classifier whose output can be used to estimate the whole partitioning pattern at once. To do so, one can consider each **partitioning pattern as a separate class** or the **depth level of a certain CTU subarea as a class**. These two options are discussed in further subsections.

4.2.1 Partitioning pattern as a separate class

As calculated in Section 2.2, the HEVC syntax allows using 83 522 different partitioning patterns for CTU size 64×64. This estimation is proper for the Basic Approach (Section 4.1). For the Extended Approach (Section 4.1), the number increases to quintillions (Section 2.2). Even for the Basic Approach, the sheer number of possible partitioning patterns discourages treating partitioning patterns as classes. Even so, the classifier training would be theoretically possible if the number of samples per class is sufficient. To check this, histograms of partitioning patterns were plotted for the Training Subset from training datasets.

Figure 4.6 presents the histogram for the Training Subset for QP=27. Partitioning patterns are sorted alphanumerically, with names as values of DM_{CU}. Due to DM_{CU} redundancy, the label of the partitioning pattern is composed of depth level values from every fourth row and column, as shown in Figure 4.7. For brevity, only one histogram was presented out of all eight possible (for each QP and subset of training dataset). Table 4.1 presents counts for exemplary partitioning patterns with their share in the partitioning pattern histogram. Table 4.2 shows some important parameters for subsets.

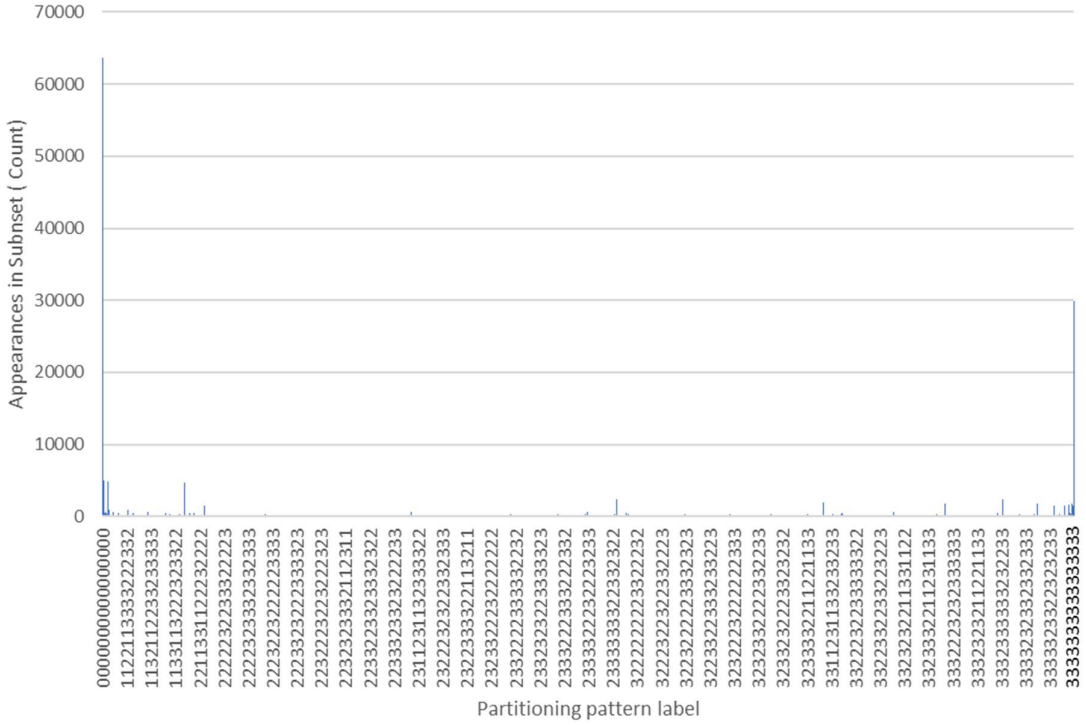

Figure 4. 6. Histogram of partitioning patterns considered a separate class for Training Subset, QP=27.

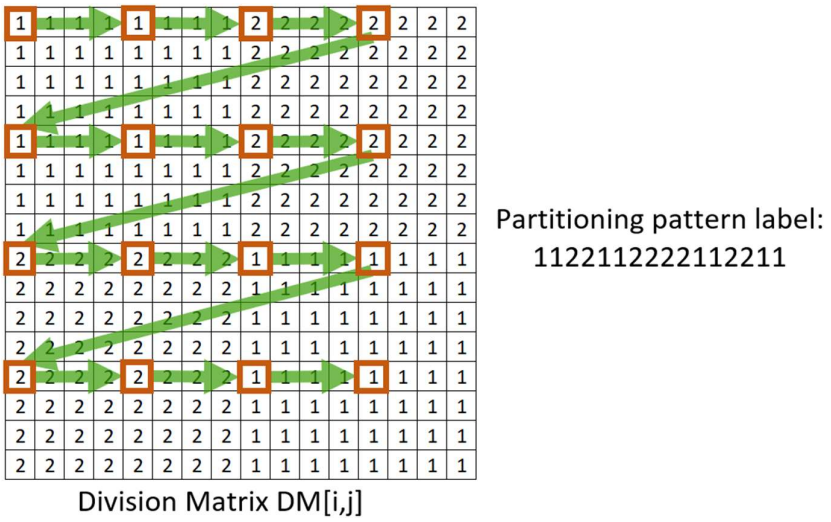

Figure 4. 7. Naming of the partitioning patterns. Due to the redundancy of Division Matrix for CTU partitioning, the name is composed of depth level values by indexing every fourth row and column.

The distribution of partitioning patterns count, presented in Figure 4.6, is strongly uneven. Among all possible partitioning patterns, most of them are chosen very rarely. Moreover, in the dataset, there are only 53 502 out of 83 522 possible partitioning patterns (Table 4.2). This observation applies to histograms of the Training and the Validation Subsets despite the *QP*.

The general rule for the ANN classifier to model the problem properly [Ha08, Go16] is independence and identical distribution of the classes (IID). The imbalance observed in the presented histogram indicates the Long Tail Distribution [Go16, Wa17]. Considering the number of classes for such a classifier, the model training would be challenging, even using methods dedicated to such a problem, like class-wise upsampling and downsampling or sample weighting [Wa17, Li20C, Zh23]. For each *QP*, the distribution differs, making the partitioning algorithm development even more difficult.

Table 4. 1. Exemplary partitioning pattern counts (considered a separate class), training dataset, for QP=27. The percentage refers to the number of partitioning patterns in the subset.

Exemplary partitioning patterns (labeled as values of DM_{CU} - raster scan)	**Training Subset**		**Validation Subset**	
	Count	**Percentage**	**Count**	**Percentage**
0000000000000000	63713	12.18	8373	12.56
1111111111111111	60937	11.65	6552	9.83
2222222222222222	515	0.10	56	0.08
3333333333333333	29917	5.72	3870	5.81

Table 4. 2. Analysis of the histogram (Figure 4.6) for the training dataset for QP=27.

Parameter	**Training Subset**	**Validation Subset**
Count of CTUs in Dataset	522939	66650
Minimal class count to cover 95% of CTUs in the dataset	30191	15178
Minimal class count to cover 99% of CTUs in the dataset	48273	17844
Count of occurring classes in the dataset	53502	**18510**

4.2.2 Depth level of a certain CTU subarea as a class

The second approach is to classify subareas of CTU by the depth levels represented in Division Matrices. Assuming the Basic Approach (Section 4.1), the values in DM_{CU} Correspond to SRU (Smallest Representable Unit) and identify the depth level on the quaternary tree. So, the ANN can estimate such depth levels for each SRU. When the Extended Approach (Section 4.1) is considered, the partitioning pattern is stored in DM_{CU} and DM_{PU}. The additional depth level value can indicate the depth level resulting from PU division (4). For the Basic Approach, the number of classes is 4, and for the Extended Approach is 5. Figures 4.8 to 4.11 present histograms of the depth level values for the training dataset. The statistics for considered all *QP* values:{22, 27, 32, 37}, are presented on histograms. The percentage of depth level appearance, referenced to all depth level values in the dataset, was used instead of counts for readability.

It can be observed that depth level values are not distributed evenly. However, compared to Figure 4.6, the distributions for depth levels are much closer to even, especially considering histograms for the Extended Approach. A substantial imbalance is observed only in the Basic Approach for *QP*=22 and *QP*=27. Depending on the *QP*, the distributions change. As expected, the percentages of depth level values for big CU blocks (small values) increase as the *QP* increases (quality drops). This phenomenon is visible, especially for the Extended Approach, where the PU division is considered. It can be noticed that the histograms for the Training and Validation Subsets are very similar. This observation applies to both Basic and Extended Approaches. Thus, evaluating the Validation Subset should appropriately reflect the model classification efficiency.

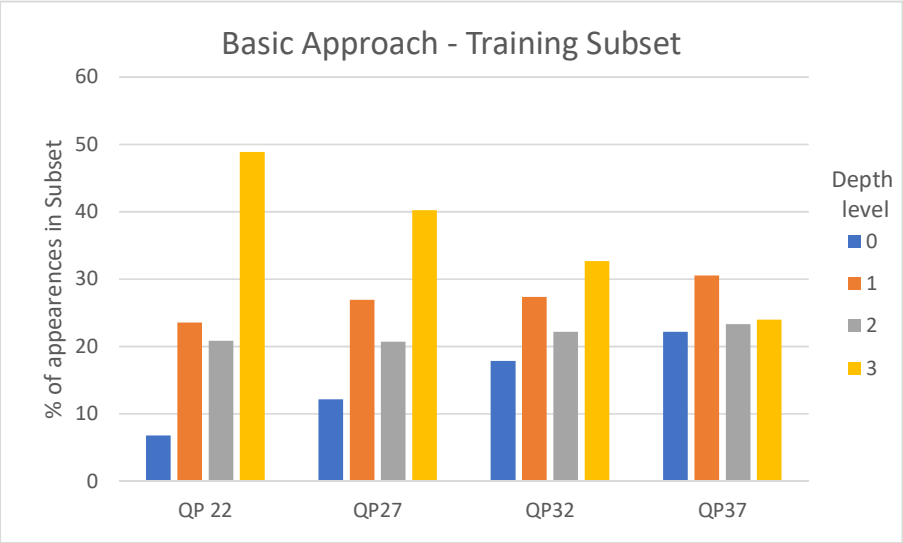

Figure 4. 8. Histograms of depth value distribution in Training Subset for Basic Approach

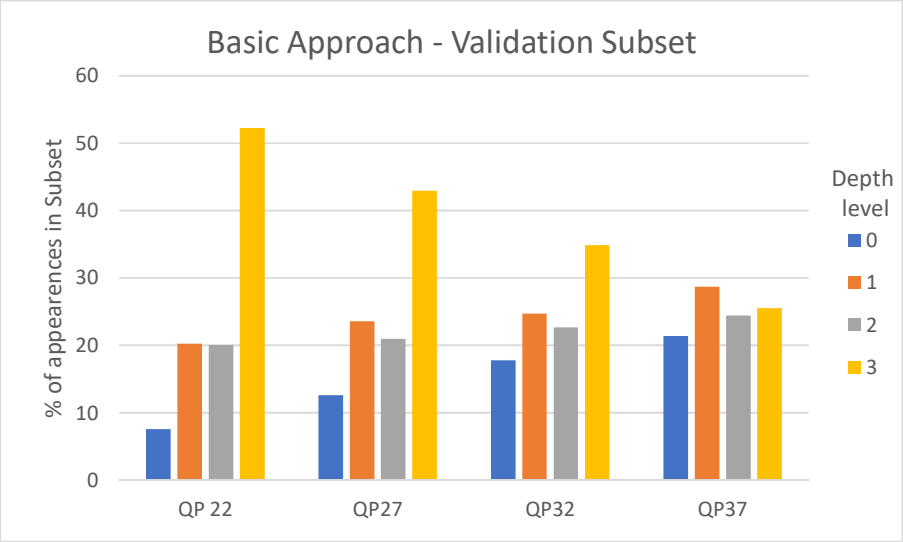

Figure 4. 9. Histograms of depth value distribution in Validation Subset for Basic Approach

It can be concluded that using the CTU subarea depth level as a class is much better, considering the IID criterion [Ha08, Go16], compared to the option presented in the previous subsection. As the depth level values indicate the classes, the learning problem is a multiclass classification with only 4 or 5 classes. It is important to underline that class imbalances, noticed during histogram analysis, may be an important factor that lowers the effectiveness of the model training. Different distributions for *QP* values imply that training separate models for each is the best idea. Additionally, the separation of the problem into different models for different *QP* values reduces its difficulty, so the smaller and effectively faster models are achievable. However, as the distributions change with the *QP* values, the sensibility of the distribution to configuration changes should be taken into account. The models trained for one encoder configuration used in different encoding configurations will perform worse.

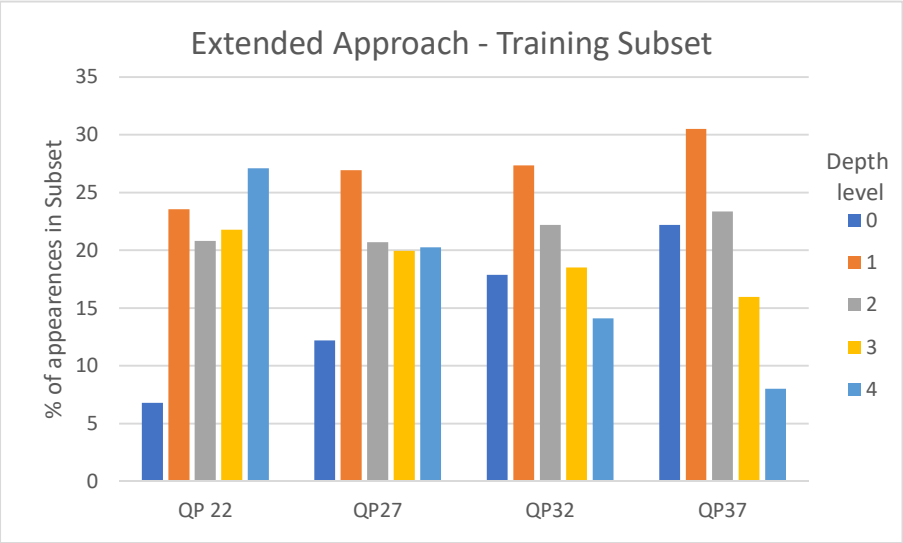

Figure 4. 10. Histograms of depth value distribution in Training Subset for Extended Approach

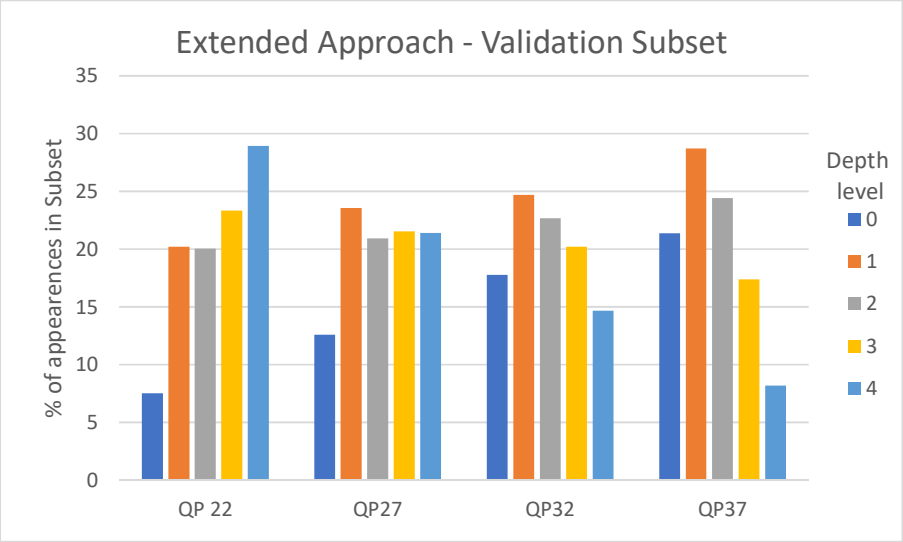

Figure 4. 11. Histograms of depth value distribution in Validation Subset for Extended Approach

4.3 Assumptions for Artificial Neural Network

As mentioned in previous sections, the idea for the partitioning algorithm is to use ANN that would mimic the decisions of the RD Optimization algorithm in HM [HM]. This goal can be achieved by training the ANN using three scenarios: supervised, unsupervised, or reinforced [Bi06, Ha08, Go16]. In this dissertation, supervised learning was chosen, as access to the Ground Truth data is not problematic. The other two scenarios are possible, but the partitioning problem may be too complex for those scenarios. Still, they may be a viable option for further research, having pre-trained models.

According to assumptions in Section 4.1, the partitioning algorithm is designed to reduce the encoder complexity. Therefore, the ANN used in the algorithm should be as low complex and straightforward as possible. Those aspects of the ANN are expressed by the number of layers, weights, and MAC (Multiply and Accumulate) operations count. In the following subsections, major assumptions made for the ANN architecture will be discussed, which concern the previously mentioned aspects.

4.3.1 Tensor format in the proposed ANNs

In the dissertation, a CTU is assumed to be a block of 64×64 samples as it is the most common setting used in HEVC (Section 2.2). CTU samples can be interpreted as a 3-dimensional (3D) data tensor. After analysis of the training dataset in the previous section, the network is a multiclass classifier that estimates the depth level for subareas of the CTU. In the HM software, the partitioning pattern is stored in the Division Matrix (DM) matrix. DM matrix contains a depth level for each SRU in the CTU. As the depth level values are represented by a single digit, the DM may also be interpreted as a 3D data tensor (with the size of the third dimension equal to 1). The depth level values can also be represented in the one-hot format, and the tensor will remain 3D. The exact format of the ANN output will be discussed further in Section 4.5.

4.3.2 Type of the proposed ANNs

As the input and output are 3D data tensors, the chosen ANN network architecture is a fully convolutional network [Ha08, Go16, Al17], more specifically with 2D convolutions. Such architecture is suitable for presented input/output configurations and should have much less weight than Alex-Net-like architectures [Le98, Kr12]. Fully convolutional networks are very popular in image processing [Kh16, Ca20, Va20, Jy24, Mo24] and well-researched [Fu69, F82, Le98, Le10, Pi15, Go16, Al17]. The important aspect is the availability of frameworks [Ji14, Ch15, Ab16, Pa19, On21] that offer optimized software for the computation and training of such networks. This is very important, especially for training, where optimized programming libraries dedicated to GPU computation can vastly decrease computation time. The main advantage of a fully convolutional network is the weight sharing [Ha08, Go16]. The same filters are used for the whole input or feature map. This means that each output of such a network is the product of the same classifier, at least in architectures without multiple processing tracks [Ha08, Go16, Al17]. The output of such a network is similar to multiple stacked classifiers [Ja07, Pi15].

4.3.3 Functional blocks used in the proposed ANNs

In the proposed ANN, the feature maps would be 3D tensors. In training, the feature maps in the Channel Last format are considered. The first two dimensions are referred to as size (height and width), and the last one as channels. Padding with zeros is used in proposed ANNs. Neuron activations are considered a separate layer. The number of training samples in a batch is called a batch size. In the proposed ANN architectures, the following classically known layers are used:

- 2D Convolution [Le98, Kr12, Al17], referred to as Conv2D.
- Batch Normalization [Io15], referred to as BatchNorm.
- Rectified Linear Unit [Fu69, Fu82, Gl10A], referred to as ReLU.
- Parametric ReLU [He15], referred to as PReLU – This activation layer has a trainable parameter, which is the slope coefficient of a line ($0 < a < 1$) for negative arguments. In the proposed networks, one slope coefficient is used for each channel of the input feature maps.
- Pooling Layers [Ya90, Ci12, Le18, Za22]– in particular Max Pooling (referred to as MaxPool) and Average Pooling (referred to as AvgPool).
- Softmax [Br89, Br90, Bi06, Go16, Ga17] – This layer is used always as the output layer in proposed ANNs. Softmax outputs values, with the meaning of probability, from multinoulli distribution [Bo68, Gi02, Go16].

4.4 The input of the Artificial Neural Network

The models proposed in this dissertation are processing the luma samples of the CTU. Chroma components are skipped due to two reasons. Firstly, the number of bits in bitstream that corresponds to the luma component is more significant than for chromas. Secondly, additional channels for chroma in the input tensor will increase the number of operations in the first layers of the ANN. These layers are the most complex part of the ANN in the proposed architectures (Section 5.1 and 6.1). In the early phase

of research, the use of all components was checked, but the results were no different than those of the luma alone. Therefore, the decision was made to use only the luma component further in the research.

The assumed size of the CTU block is 64×64 samples, so the shape of the input data tensor is 64×64×1. The architecture of Subnetwork $\mathbb{A}$ (Section 4.1) is determined by the input tensor size (first two dimensions), where the reduction of the feature map size is performed. Before imputing a tensor of samples to ANN, the data are preprocessed. During the research, the best results were obtained when the input samples were scaled to range $\langle 0; 1\rangle$. As the luma samples are always integer numbers in the range $\langle 0; 2^N\text{-}1\rangle$, where N is a bit depth, the preprocessed value of the sample $u_{i,j}$ is obtained using the formula:

$$u'_{i,j} = \frac{u_{i,j}}{2^N - 1}. \tag{4.1}$$

4.5 The output of the Artificial Neural Network and decision algorithm

The ANN output is a representation of the partitioning pattern. As mentioned in Subsection 4.3.1, the ANN can use a format similar to Division Matrix (DM) used by HM [HM]. Such a format would benefit the implementation of the ANN in the Modified HM (Section 4.7).

The general format of the DM was presented in Subsection 2.3.3. For assumed All Intra configuration, the CTU size is 64×64, and SRU (Smallest Representable Unit – Section 2.3.3) is 4×4. This results in the size of the DM_{CU} of 16×16. Considering the partitioning process alone, this representation of the pattern is redundant. Direct access to depth level value by the SRU is not required, and the quaternary tree directly implies how the partitioning must be done. Therefore, depending on the size of the smallest block in the given approach the size of DM can be reduced. **For the Basic Approach** (Section 4.1), the smallest CU block has the size of 8×8 samples, so **the DM may be reduced fourfold in each dimension**. **For the Extended Approach** (Section 4.1), the smallest PU block is 4×4 (same as SRU), so **the DM may be reduced twofold in each dimension**. The visual presentation of blocks for each depth level is shown in Figure 4.12.

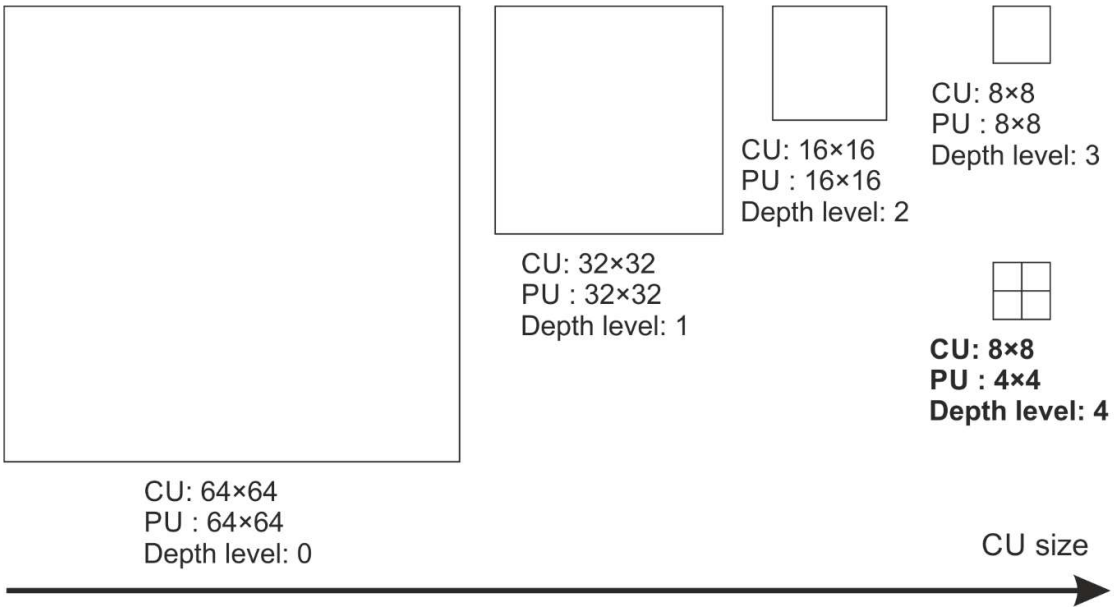

Figure 4. 12. Visual presentation of the CU/PU blocks with corresponding depth levels.

Following the above observations, one can define DM formats specific to each approach. In the following formulas, indices i and j are coordinates of the value in the DM, and d is the depth level value. The Division Matrix for the Basic Approach is defined as:

$$DM_B[i,j] = d \quad \text{where} \quad i,j \in [0;3], d \in [0;3]. \tag{4.2}$$

Similarly, the Division Matrix for the Extended Approach is defined as follows:

$$DM_E[i,j] = d \quad \text{where} \quad i,j \in [0;7], d \in [0;4]. \tag{4.3}$$

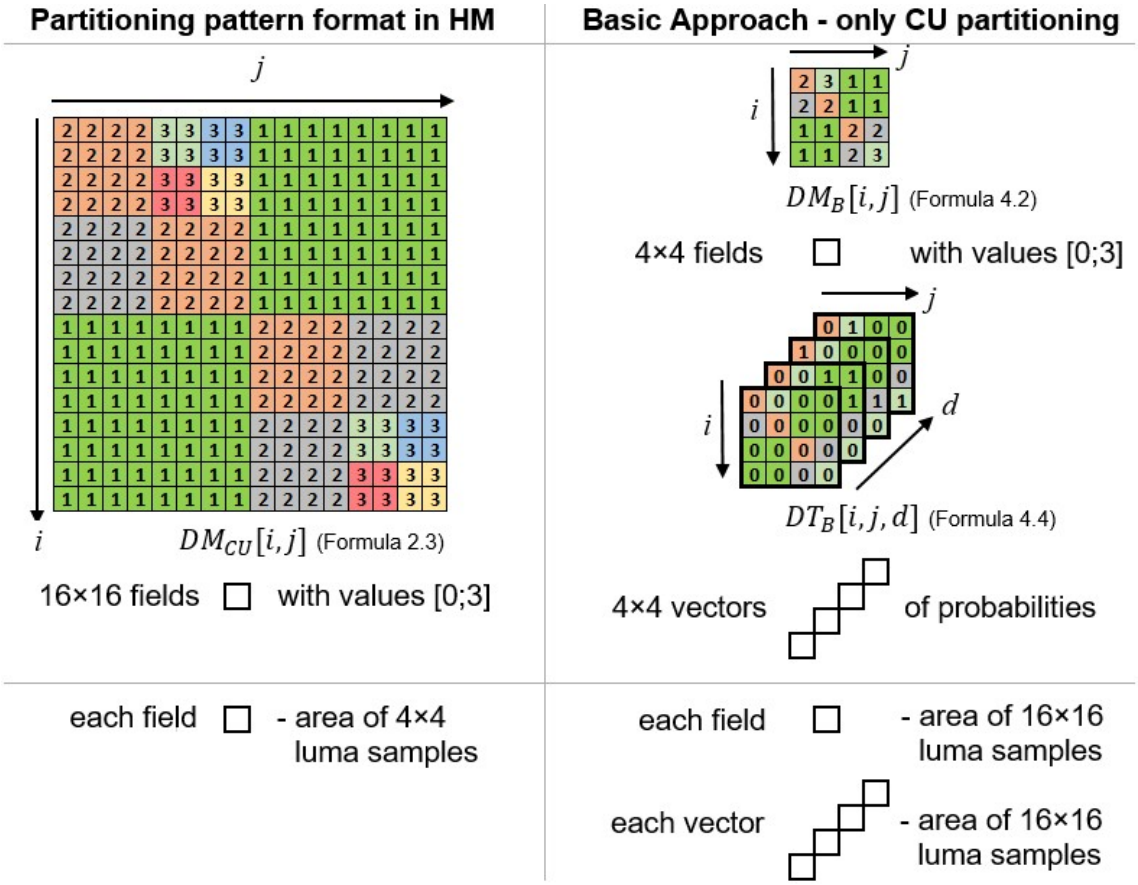

Figure 4. 13. Visual presentation of DM and DT for Basic Approach. Probabilities in DT are DM values converted to the one-hot format.

DM formats defined above are easily convertible to DM format used in HM [HM]. If the ANN would estimate directly the DM, the output would be indices of depth levels. Instead of this, the Softmax used as output layer of the ANN, returns d values that sum up to 1. As the training dataset reflects statistics of the population, the output of the ANN after training is expected to have probability meaning. Then the ANN is estimating the tensor with probabilities of depth levels for subareas of the CTU instead of a matrix with depth levels. Such a tensor would be related to as Division Tensor (DT). The Division Tensor for the Basic approach is defined as:

$$DT_B[i,j,d] = prob_{i,j,d} \quad \text{where} \quad i,j \in [0;3], d \in [0;3]. \tag{4.4}$$

Analogously, the Division Tensor for the Extended Approach is defined as follows:

$$DT_E[i,j,d] = prob_{i,j,d} \quad \text{where} \quad i,j \in [0;7], d \in [0;4]. \tag{4.5}$$

The visual presentation of the DM and DT for the Basic Approach is shown in Figure 4.13 and for the Extended Approach in Figure 4.14

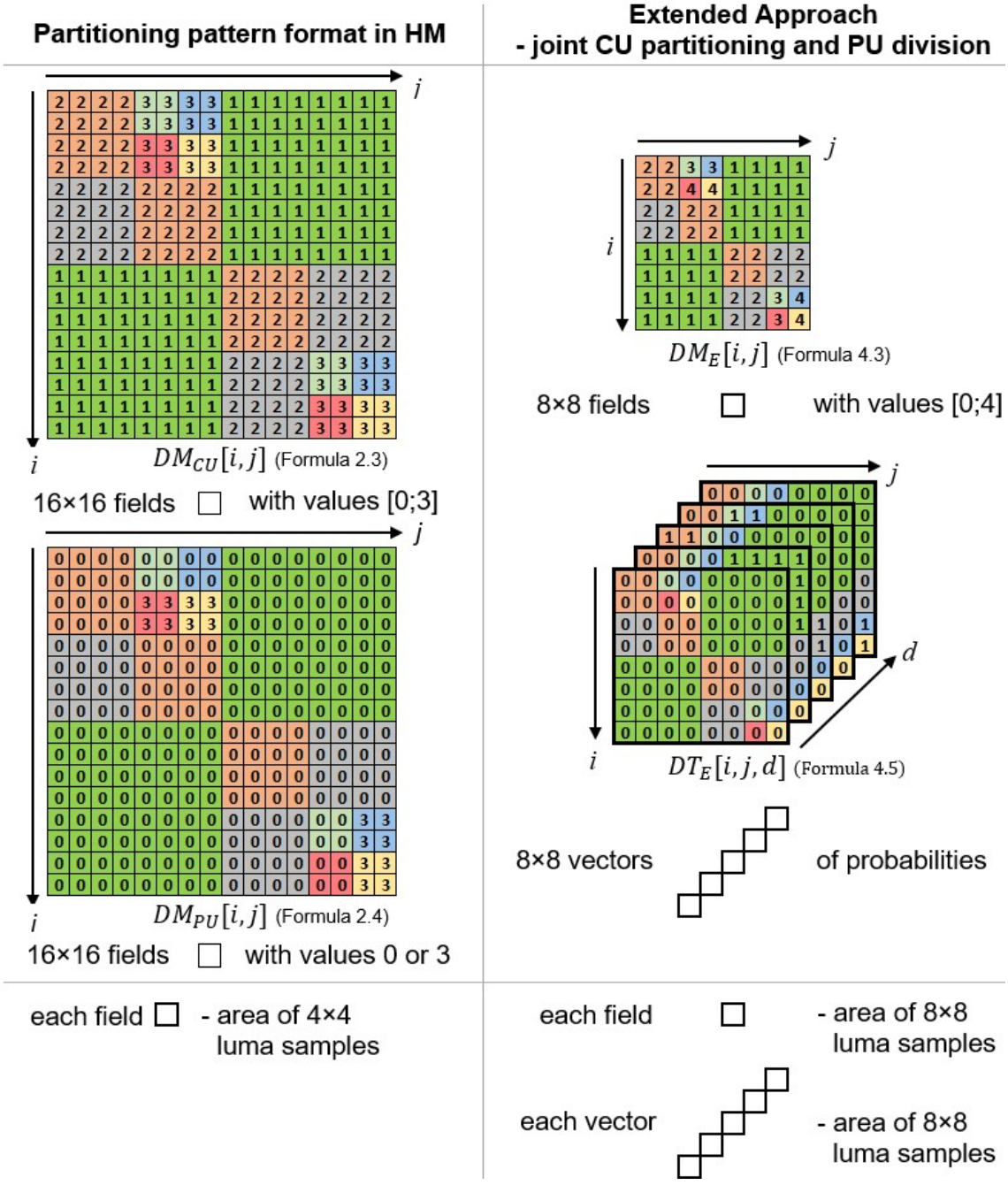

Figure 4. 14. Visual presentation of DM and DT for Extended Approach. Probabilities in DT are DM values converted to the one-hot format.

The use of DT as an output of the ANN requires adapting the training data, which is composed of Division Matrices (DMs) used in HM. First, the partitioning pattern is converted to DM, which is suitable for the approach taken. Then, the DM is converted to DT by conversion of the depth levels to

one-hot format, which is adjusted to the maximum depth level for the taken approach. The one-hot format is a representation of the class index in a vector, where all values are equal to 0 except the index of the class, for which a value is 1. Such a format means that the target division level has the maximal probability of what the ANN is to be trained.

Partitioning algorithms proposed in this dissertation return partitioning pattern or set of those. However, the ANN is trained as a multiclass classifier that estimates depth level values for multiple subareas in CTU at once. A raw output of the ANN is converted into a partitioning pattern or set of these by a decision algorithm. The subject of the decision algorithm is robustly discussed in Chapter 7. For the development of the ANN models in Chapters 5 and 6, a decision algorithm called Index-based (AlgIdx) was used (hard-decisive variant), which is presented in Subsection 7.2.1.

Summing up, the partitioning algorithm calculates the DT with the ANN. Then, DT is processed by a decision algorithm, which results in a DM specific to the applied approach. Finally, the DM should be converted to a partitioning storage format used by the HM. Described formats of the DM_B and DM_E precisely define a single partitioning pattern. However, the values outside the usable depth level range [0;4] may be set to define a subset of depths that should be considered. The Modified HM can interpret new possible values in DMs and apply appropriate computations. Using such values would be crucial for the implementation of soft-decisive variants of decision algorithms. This subject is further discussed in Section 7.3.

4.6 Model training

4.6.1 Loss function and learning rate optimizer

The ANN used in the partitioning algorithm is a fully convolutional network designed for the multiclass classification problem. For such ANN, as the loss function the $Categorical\ Cross\text{-}entropy$ [Ma03, Co06, Go16, Ma23B] was chosen. This loss function is a method of maximal likelihood estimation by minimizing Kullback-Leiber divergence [Ma03, Co06, Go16]. The $Categorical\ Cross\text{-}entropy$ is multiclass classification and the most popular loss function [Go16, Ma23B], was found as the best performing one among other tested. This $Categorical\ Cross\text{-}entropy$ function is defined with the following formula:

$$Categorical\ Cross\text{-}entropy = -\sum_{n}^{N} y_n^{true} \cdot \ln\left(y_n^{pred}\right), \tag{4.6}$$

where y_n^{true} is a Ground Truth value and y_n^{pred} is the predicted value, N is the number of training samples. The formula 4.6 is a general definition of this loss function dedicated to vector output. As in proposed ANNs the output is a Division Tensor (DT), the formula has to be appropriately adjusted. So, for the assumed output format, the $Categorical\ Cross\text{-}entropy$ is defined as:

$$Categorical\ Cross\text{-}entropy = -\sum_{n}^{N}\sum_{i}^{H}\sum_{j}^{W}\sum_{d}^{D} DT_n^{true}[i,j,d] \cdot \ln\left(DT_n^{pred}[i,j,d]\right), \tag{4.7}$$

where: H, W, and D are dimensions of the DT for the given approach (Section 4.5: DT_B for the Basic Approach, DT_E for the Extended Approach), and N is the number of training samples. The subscript n is the index of the training sample. Superscript $true$ refers to the Ground Truth sample, and superscript $pred$ refers to the prediction of the ANN.

Another important aspect of the ANN training is the choice of the learning rate optimization algorithm. For the ANNs training, the Adam [Ki14, Go16] optimizer was chosen. This optimizer adaptively adjusts the learning rate during the training process. The Adam algorithm is one of the most

popular optimizers [Ch19B, Su19], especially for complex problems like image classification [Su19, Sc20, Ha23]. Other optimizers, like SGD [Ro51, Bo98], AdaMax [Ki14], and Nadam [Do16], were considered, but the Adam optimizer proved to yield the best results in model training, as shown in Section 5.4. Additionally, the optimizer was restarted every 10th epoch of training, as this improved the training efficiency.

4.6.2 Adjustments of training assessment for the proposed model output

Similar to the loss function, the format of the ANN output influences the formulas used in the ANN training assessment. Firstly, $Accuracy$ metric is a wrapped version of the $Categorical\ Accuracy$ (3.16). Additionally, the $Accuracy$ is multiplied by 100% to express it in percentage. So, the formula is defined as follows:

$$Accuracy = \frac{100\%}{N \cdot H \cdot W} \sum_{n}^{N} \sum_{i}^{H} \sum_{j}^{W} \mathrm{Iv}\left(\mathrm{ArgMax_d}(DT_n^{true}[i,j]), \mathrm{ArgMax_d}\left(DT_n^{pred}[i,j]\right)\right), \qquad (4.8)$$

where: H and W are dimensions of DT for the given approach (Section 4.5: DT_B for the Basic Approach, DT_E for the Extended Approach), and N is the number of training samples. The subscript n is the index of the training sample. Superscript $true$ refers to the Ground Truth sample, and superscript $pred$ refers to the prediction of the ANN. Accordingly, the formula (3.18) for Confusion Matrix values (CM) changes to:

$$CM_{\%}[p,g] = \frac{100\%}{N} \sum_{n}^{N} \sum_{j}^{H} \sum_{j}^{W} \mathrm{Iv}\Big(g = \mathrm{Iv}\left(p = \mathrm{ArgMax_d}\big(DT_n^{pred}[i,j]\big)\right)\Big), \qquad (4.9)$$

where: H and W are dimensions of DT for the given approach (Section 4.5: DT_B for the Basic Approach, DT_E for the Extended Approach), N is the number of training samples, p and g are successively the row and column indices of CM. The digit of CM may be referred to as the count of predictions of p-th class when the reference (Ground Truth) class was g-th. The subscript n is the index of the training sample. Superscript $pred$ refers to the prediction of the ANN. As the problem was defined as multiclass, $Recall_{MicroAvg}$ and $Precision_{MicroAvg}$ CM analysis metrics will have the same value.

4.6.3 The strategy for training models

The ANN training process required many adjustments to achieve stability and convergence. To this end, many training strategies and tools were used and tested, except for the aforementioned optimizer and loss function. Further in this subsection, the most important methods and strategies that were incorporated into the final training procedure will be discussed.

The Early Stopping [Zh04, Ya07, Go16] regularization method was used in the ANN training process. Overfitting of a model is prevented by monitoring the loss function value for the validation subset. The training is terminated when the loss function is not improved throughout a certain number of training epochs (referred to as the patience coefficient). For the training of the models presented in this dissertation, a patience coefficient of 3 was used.

In most cases, the Early Stopping marks the termination of the training. However, an additional constraint**, the maximum number of epochs**, has been used to avoid redundant calculations. In the early stages of the research, when Early Stopping was not yet used, experiments were performed to estimate the maximum number of epochs. The conclusion was that surpassing 100 epochs does not benefit the model accuracy. This number of epochs was verified multiple times further in the research, and the conclusion was always the same.

Proposed models were trained with the stochastic gradient descent approach [Li14, Go16]. Therefore, instead of one weight update after processing the whole training dataset (batch method [Bi05, Ha10, Go16] – single batch per epoch), the weights are updated after a smaller portion of samples (called **minibatch method** [Wi03B, Li14, Go16]). Multiple batches per epoch are processed, so weights are updated multiple times. Such an approach provided better convergence of the training experiments and the regularization effect [Wi03B]. **The batch size of 64** training samples was used for training ANN models presented in this dissertation. The reasoning for this batch size is that training results were sufficient, and this value allowed the training on each available machine. The batch size is the power of 2 is justified by GPU construction [Go16]

A crucial aspect of the model training is **the weight initialization method**, as it highly impacts the further training of the model [Ha08, Sa14, Go16, Mi16]. As the author's architectures are proposed in this dissertation, models were trained from scratch. In the early stage of the research, multiple weight initialization methods were tested, e.g., random uniform or normal distribution, the He [He15] method with normal or uniform distribution, and the Glorot [Gl10B] method with normal or uniform distribution. The best-performing method was the Glorot method with uniform distribution, which was further used as a weight initialization method for all models presented in this dissertation. To avoid a bias in mini-batch samples used for gradient update [Bi05, Go16], **the training samples are shuffled at the start of each epoch and before division into mini-batches**

The non-determinism of the training procedure must be discussed here. Multiple aspects of the training procedure depend on randomness, e.g., weight initialization or data shuffling. Therefore, training the same architecture multiple times will always yield a slightly different model. During the research, it was observed that despite the same architecture and very close results in training assessment, for trained models, the evaluation in the encoder can vary up to 0.1 p.p. $BD\text{-}RATE$ bitrate reduction

A proper evaluation of single ANN architecture is a laborious process. It requires several repetitions of training and evaluation. As discussed in Section 3.7, a greedy experimentation strategy was adopted during research works due to the time complexity of the evaluation process. Thus, architecture or training procedure modifications were tested only onefold during the development of models for Basic or Extended approaches. If the result of the evaluation shows an improvement close to 0.1 p.p. $BD\text{-}RATE$ bitrate reduction, then such a model would be trained multiple times to check its convergence to this result. If a modification improved the results, it was adopted as a new best model.

4.6.4 The framework for training models

The author's framework for training the ANNs was prepared to perform training of the models. This framework consist of training dataset (described in Section 4.2) and the software. This software implements the following steps of ANN training: dataset loading, dataset preparation (preprocessing), model preparation for training, model training and assessment, model preparation for the encoder, and preparation of the tasks for evaluation of the ANN in the Modified HM. The steps of model preparation for training and encoder is further discussed in the next section. The software allowed the serialization of subsequent experiments. The setup for an experiment is done by definition of the training conditions (for each step) and network architecture in a configuration file. In addition to the trained models, the software returns the results of model training assessment.

Model preparation and training were done using the TensorFlow [Ab16, TENSORFLOW] library in Python [Ro09]. Among other frameworks for ANN training, e.g., PyTorch [Pa19], Tensorflow delivered the best training accuracy of the evaluated ANN architectures. Furthermore, by the start of the research, the library offered the broadest support of training techniques and tools. An additional feature was the best support for GPU training acceleration libraries. By the time of starting the research, Tensorflow offered an approach for processing tensors in Channel Last format (the last dimension of the tensor corresponds to channel) [TENSORFLOW], which is much more efficient in terms of

computational complexity in GPU [Nv23] for convolutional layers, thus significantly accelerate the training of the model.

4.7 Implementation of the model in the modified HEVC encoder

Evaluation of proposed partitioning algorithms requires a dedicated HEVC encoder. Additionally, the network training requires partitioning patterns estimated by the HEVC encoder. The HM [HM] is software created for the development of the HEVC technique. The HM was the only available encoder that offered an extensive search range of partitioning patterns. Other available HEVC encoders, e.g. x265 [X265], aggressively limit the number of considered partitioning patterns and potentially decrease the performance of proposed algorithms. Thus, the HM was chosen as software to modify and support the proposed partitioning algorithms.

The HM software is a huge project (~93 thousand code lines in 192 files in C++ [CPP]) that contain implementation of an encoder, a decoder, and other utility programs, e.g., for a bitstream analyzer. The modification of this software requires a deep knowledge of not only the standard but also data structures and specific algorithms implemented within the software. Additionally, the software features cmake [CMAKE] scripts for building projects and the inclusion of dependencies. The modification of this software requires not only the implementation of the method in code but also the appropriate placement in the software structure while maintaining the capabilities in code compilation provided by HM.

During the research works related to this dissertation, multiple modifications to HM software were made to test the proposed partitioning algorithms. To distinguish from the original HM, these modifications are referred to as Modified HM. The modifications of the HM are: support for the ANN processing library, extraction of the partitioning pattern in the decoder, implementation of control over the partitioning process, and implementation of partitioning methods. The Modified HM originates from the HM [HM] version 16.23. This version of HM was also used in evaluation of the proposed partitioning algorithms (Section 3.1).

The presented Modified HM keeps the signal flow of the HM software. Thus, all modifications were implemented in a way that preserve the consecutive CTUs' serial processing using a single thread. After the partitioning pattern or patterns are estimated, the actual encoding is performed, as described in Subsection 4.7.3. Such implementation ensures a fair comparison of the partitioning methods, especially with the HM.

In the software, the user can set up the Modified HM. Generally, the HM provides an extensive configuration of the encoding process. Commonly, it is done using the configuration files. The setup of the Modified HM was incorporated into the configuration mechanism of the HM. Such a solution eases the use of the software, especially for researchers familiar with HM. This aspect was raised as the Modified HM was released along with the paper [Lo24].

4.7.1 Support for the ANN processing library

The support for the ANN processing library raised several issues. The Tensorflow [Ab16, TENSORFLOW] library was used in the training process. During the development of the modification of HM, it was observed that integrating the Tensorflow in the encoder was possible but problematic. The integration would require significant interference with the compilation procedure of the encoder, which would make the use of Modified HM troublesome. The more suitable option for ANN implementation in the HM was the LibTorch library. The LibTorch is a C++ version of the PyTorch library [Pa19]. The LibTorch (and PyTorch), similar to TensorFlow, is a popular library for ANN processing.

The LibTorch requires the linkage of static libraries and access to dynamic ones. Those libraries are available in precompiled versions both for Windows and Ubuntu. Incorporating the LibTorch in the Modified HM does not disturb the organization of the software, as it was added in the cmake code of

the HM. This means that the original software use (project preparation, compilation) remains the same, and the LibTorch is included and linked automatically during software deployment.

The LibTorch library offers easy control over the ANN'a compute resource restrictions (such as cores and threads). This feature is crucial for a fair assessment of the HM modification as HM Software is a single-threaded application. The Python version (PyTorch) of model is not directly compatible with LibTorch. Though, the conversion from the Python model to the C++ one is featured by the PyTorch JIT format [PYTORCH, LIBTORCH]. Tests proved that the output of the converted model, evaluated with the LibTorch JIT module [LIBTORCH], is numerically identical to its source given the same input, so the model accuracy is maintained. Modified HM reads the model from the path indicated by the user. In this dissertation, an ANN is restricted to using only one processor core and one logical thread in experiments.

In the previous section, Tensorflow was chosen as the library for ANN training. Models are prepared to process tensors in Channel Last format. Unfortunately, by the time of the development of Modified HM, the LibTorch did not feature stable processing tensors in such a format. Thus, to incorporate the trained model into the encoder, it must be first converted into a PyTorch corresponding model, which processes data in Channel First (the first dimension of the tensor corresponds to the channel). Conversion between the model from Tensorflow format to PyTorch format is tricky. The difference in the implementation of particular layers prevents a direct conversion. Sufficient converters were not available at the time of the software development.

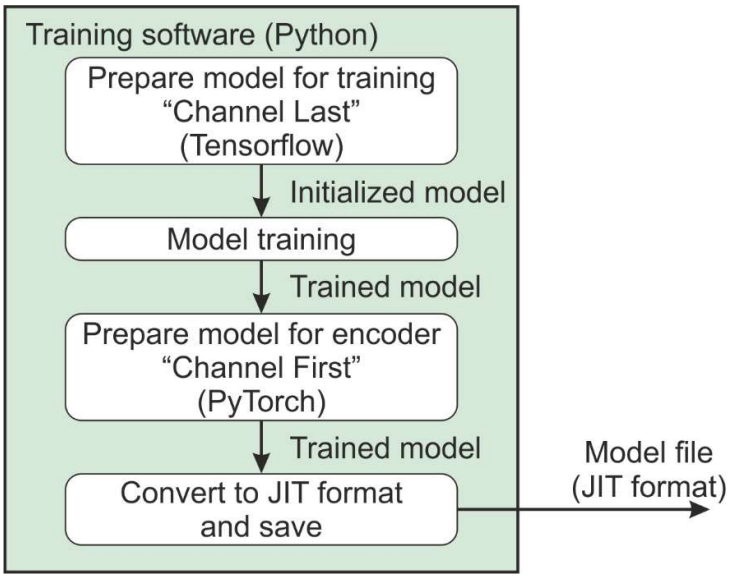

Figure 4. 15. Visualization of the model conversion between Tensorflow and LibTorch formats.

The software for the simultaneous preparation of matching models for TensorFlow and PyTorch was developed to solve the conversion problem. The models are built based on configuration, where the architecture and hyperparameters of the layers are defined. The TensorFlow model is created during the model preparation for the training. Then, after the completion of the training process, the PyTorch model is created. The first step of preparing the model for evaluation involves extracting the weights of the trained model and converting them to the Channel First format. Then, the converted weights are applied to the PyTorch version of the model. Lastly, the model is converted from Python to C++ format and saved for future use in Modified HM. The described process is visualized in Figure 4.15. It was observed that after loading the model in the encoder, running the model with random data in the encoder

initialization phase is beneficial. This step proves to accelerate the network computation later in the encoding phase.

4.7.2 Extraction of the partitioning pattern in the decoder

The extraction of the partitioning patterns was the first modification applied to the HM software. It was crucial for the creation of the training dataset. In the decoder, after extraction of the CTU from the bitstream, the DM_{CU} and DM_{PU} are available. Then, the Division Matrices (DMs) are saved in the text files. This procedure is shown in Figure 4.16.

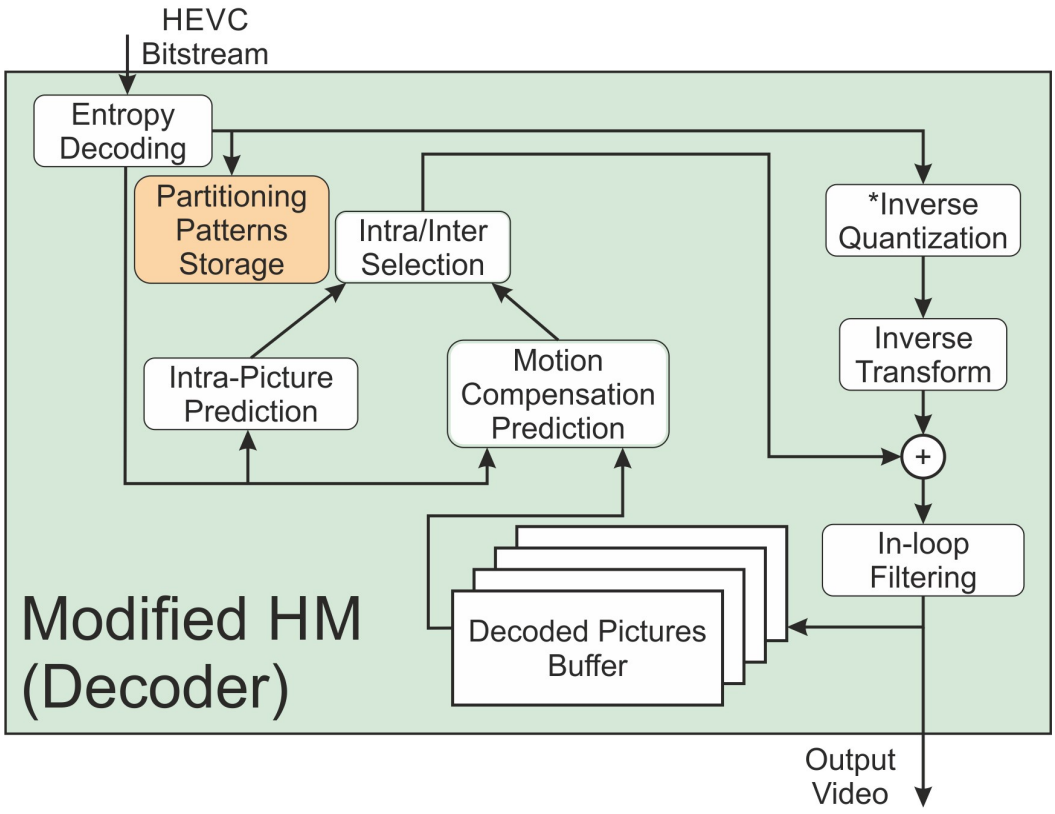

*Figure 4. 16. Modification of the Modified HM (Decoder) for partitioning patterns extractions. After the entropy decoding, the partitioning patterns for all CTUs are saved to a text file. Block "*Inverse quantization" refers to recovery of the original scale of the signal (quantization is lossy and cannot be inversed).*

4.7.1 Implementation of the control over the partitioning process

The partitioning process and CTU encoding are incorporated in the HM, as presented in Subsection 2.3.2 (Figure 2.8). The encoder starts from CTU size CU and searches for the best partitioning pattern following the hierarchy of the quaternary tree. All encoding decisions (Figure 2.7) for specific block sizes are estimated in the partitioning algorithm along with the cost value. After the decision for a block to remain at a certain depth level, the block is compressed.

The control over the partitioning process was implemented by skipping computations for unwanted depth levels. Firstly, the partitioning algorithm estimates the partitioning pattern for the current CTU. The partitioning pattern is stored in DMs. Then, the decision estimation is run. Before any computation, the current depth level is compared with the DM value corresponding to the top left SRU (Smallest Representable Unit) in the currently considered block. The partitioning algorithm may indicate one or two consecutive depth levels to consider. If the current depth level is indicated in the DM, the encoder

makes the rest of the decisions and estimates the bit cost (Formula 2.2) for the block. In the opposite case, the calculations are skipped, and the bit cost is set at the highest possible value. The modified algorithm for estimating the decision stage is shown in Figure 4.17.

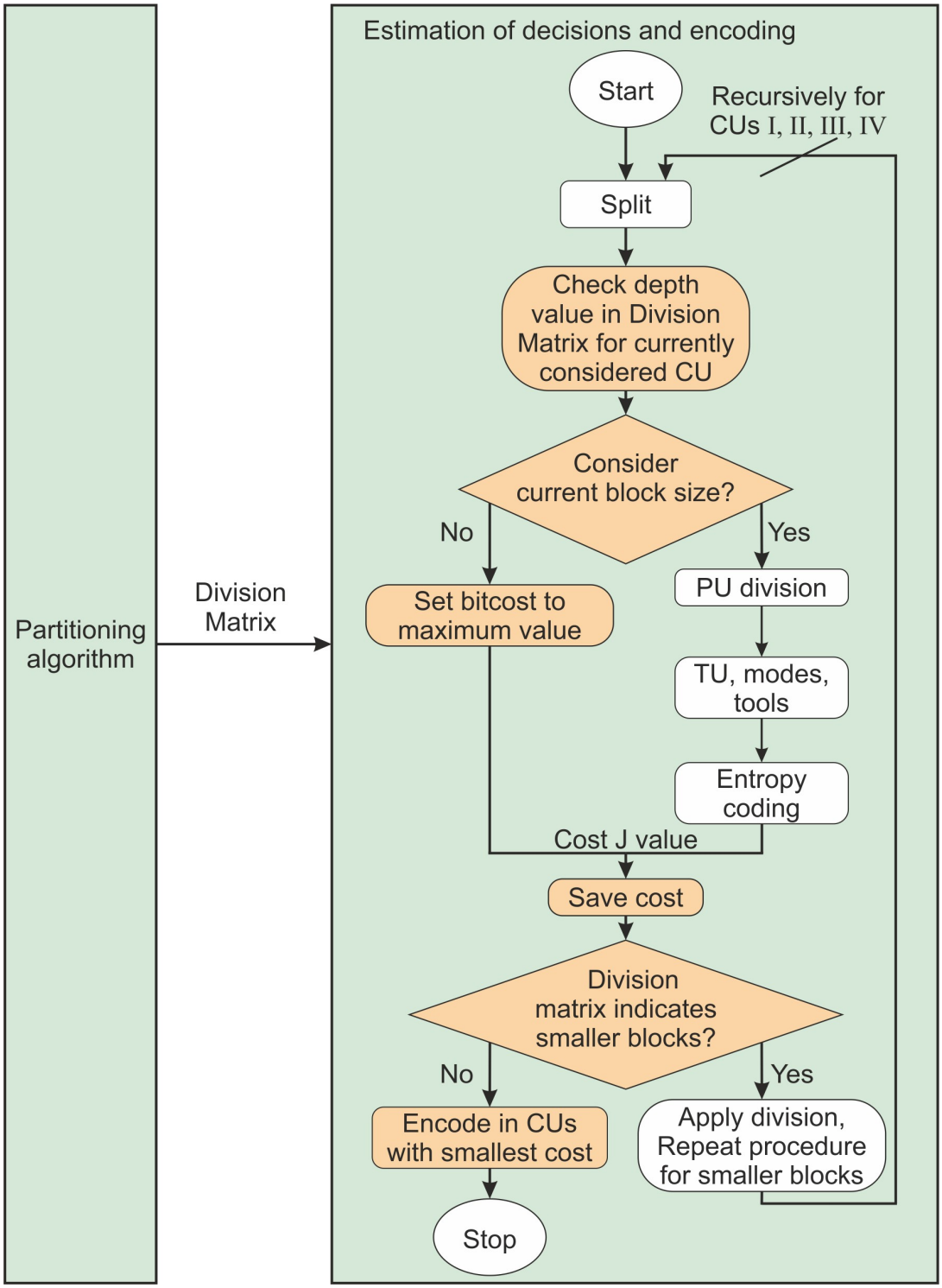

Figure 4. 17. Algorithm for encoding CTU in Modified HM (encoder). White blocks originate from the partitioning algorithm from HM, presented in Figure 2.8.

Such an approach has several advantages. Firstly, the encoder performs computations only for blocks indicated in the partitioning pattern. Secondly, such implementation does not disturb the internal mechanism of the encoder (e.g., the context of the CABAC or Most Probable Mode tool [HM]). Thirdly, the comparison between the two indicated depth levels can be applied at any depth level without further modification of the encoder. Lastly, it does not constrain the moment of estimation of the partitioning. By default, the partitioning pattern is estimated right at the start of coding the CTU. If part of the samples is unavailable, like in the case of CTUs spanning beyond the boundaries of the frame, the missing ones are set to zeros. However, in specific cases, as described in Section 9.8, the encoder can estimate partitioning patterns for the whole frame and then apply them in CTUs.

4.7.2 Implementation of partitioning methods

The implementation of the partitioning algorithm consists of three steps (Figure 4.18).The first step is gathering all the necessary data the core partitioning algorithm requires. It is done by accessing data available in the encoder data structures. For ANN-based algorithms, CTU luma samples will be accessed. If the algorithm needs access to additional data like in Section 9.6, access to these is granted in this step.

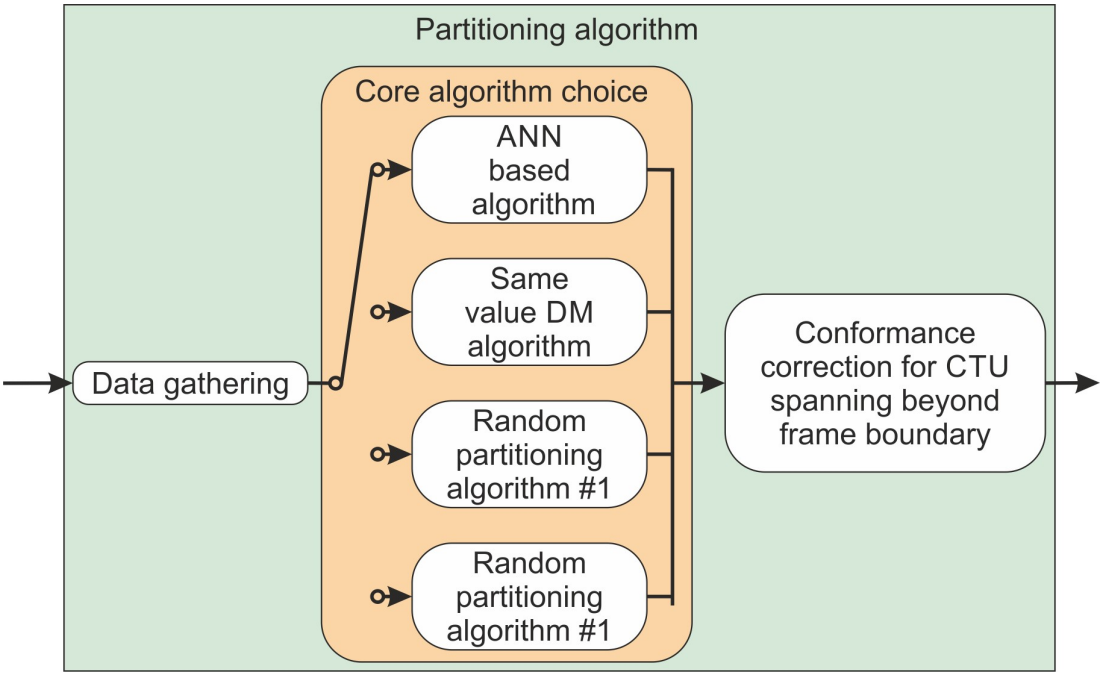

Figure 4. 18. Implementation of proposed partitioning algorithms in Modified HM (encoder).

The second step is processing the data using the core algorithm. In the initialization of the encoder, the core algorithm is chosen according to the configuration. This core algorithm is then used for the whole encoding process. The most important one, the ANN-based core algorithm, is described in Subsection 4.7.5. Three additional simple core algorithms are implemented in the Modified HM. The first one allows the sequence to be encoded using a DM filled with the same value. This partitioning algorithm was used to inspect the influence of specific block sizes, or sets of block sizes, on the computational complexity of the encoder and bitrate of the encoded sequence. The other two partitioning algorithms implement random partitioning methods. One of them estimates the partitioning pattern by randomly deciding on splits of the quaternary tree, starting from the root. Another one draws a CTU partitioning pattern from a list of all possible ones. When used in the Extended Approach (Section 4.1), each time the block size is 8×8, the algorithm randomly decides on PU division. Presented random

partitioning algorithms were used to ensure that the proposed approaches with ANNs are working correctly.

The last step of the partitioning algorithm implementation is conditional. In the case of CTUs spanning beyond the frame boundaries, the conformance with HEVC syntax is inspected and adjusted accordingly.

4.7.3 ANN-based core partitioning algorithm

Implementation of the core partitioning ANN-based algorithm consists of four steps. These steps are shown in Figure 4.19. The first one is input ANN preparation. The CTU samples are copied to the LibTorch-specific data structure, and then preprocessing is applied. The second step is processing the data by ANN. The time of this step is measured and accumulated. Then, the DT outputted from ANN is processed by the decision algorithm. Further, the algorithm returns the DM that specifies the partitioning pattern or their set. Lastly, the DM outputted by the decision algorithm is extended to the same size as used in HM. The user of the Modified HM can configure ANN input preprocessing, ANN model, compute resources available for ANN, chosen approach (Basic or Extended), and decision algorithm with its parameters.

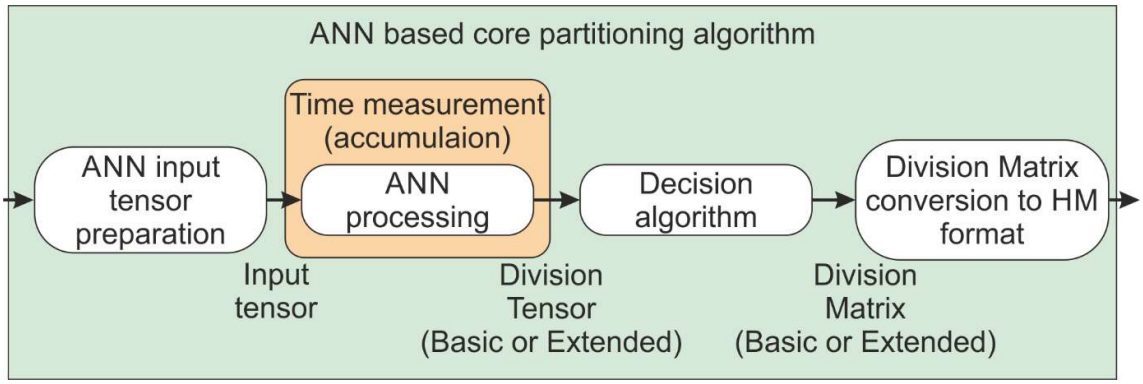

Figure 4. 19. Steps of ANN-based core partitioning algorithm in Modified HM (encoder).

5 ANN model for the Basic Approach

5.1 Detailed description of ANN architecture

As discussed in Section 4.1, the proposed ANN architecture is divided into two subnetworks, combined and trained as one model. Subnetwork 𝔸 is dedicated to the reduction of feature map size and expansion of feature map channels. The Subnetwork 𝔹 outputs the probability values. This subnetwork utilizes a layers arrangement that reflects a quaternary tree structure. The ANN architecture used in the Basic Approach (Section 4.1) is presented in Figure 5.1. The ANN architecture and hyperparameter values are the result of long-lasting research and a methodical amendment process. Selected directions of the model development are discussed in Section 5.4.

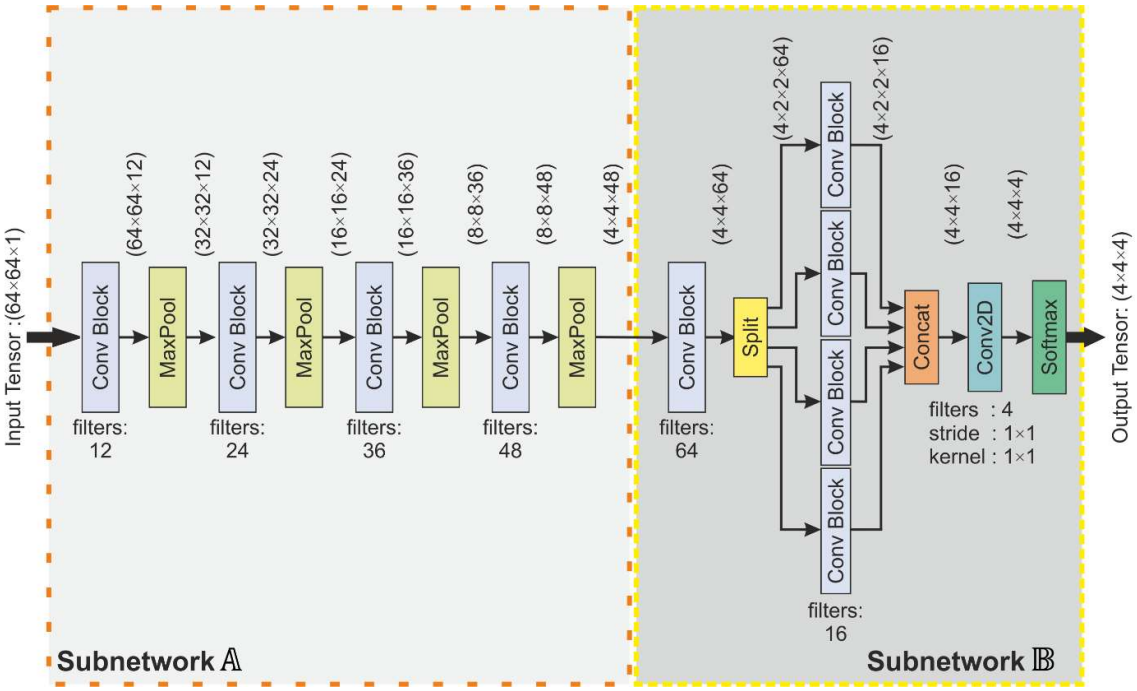

Figure 5. 1. The ANN architecture used in the Basic Approach. The "Conv Block" refers to block of layers, presented in Figure 5.2.

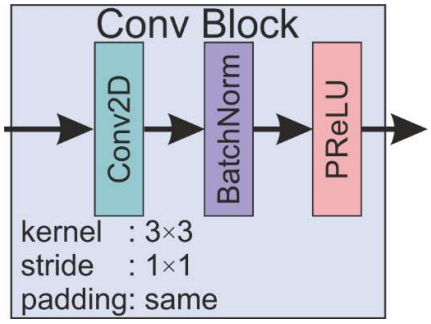

Figure 5. 2. Visualization of the Conv Block.

The proposed architecture comprises convolutional blocks, named Conv Blocks, for short. A Conv Block (Fig. 5.2) is composed of three layers: Conv2D (filter size 3×3, stride of 1×1, padding to get the same output feature maps size as input), BatchNorm, and PReLU. The number of filters in Conv Block corresponds to the number of filters applied in the Conv2D layer.

The Subnetwork $\mathbb{A}$ contains four Conv Blocks. The number of filters increases in consecutive Conv Blocks and is equal to 12, 24, 36, and 48. The output of each Conv Block is connected to the MaxPool layer (Pool size of 2×2, stride of 2×2). The size of feature maps outputted by pooling layers decreases twofold in each dimension.

The arrangement of layers in Subnetwork $\mathbb{B}$ recalls the quaternary tree and mimics the architecture of the partitioning in HEVC. Similarly like in the HM partitioning algorithm, the feature maps are processed in subareas according to consecutive block sizes. The Subnetwork $\mathbb{B}$ is composed as follows:

- Firstly, feature maps from Subnetwork $\mathbb{A}$ are processed by Conv Block, whose number of filters is 64. This Conv Block corresponds to the consideration under splitting a block of size 64×64.
- Then, the outputted feature maps are split into four tensors in a quaternary tree manner. Each resulting tensor is processed by a separate Conv Block, whose number of filters is 16. Using this Conv Blocks set corresponds to the considerations of splitting 4 blocks of size 32×32. Those Conv Blocks do not share weights, as the split decision may be taken differently in different subareas of the CTU.
- Following the quaternary tree manner, each the resulting four feature maps should be further split and processed is separate four Conv Blocks. However, the tensor size would be then 1×1×16. During the research, an architecture with a set of 16 separate Conv2D layers was tested. However, the training of such models was not converging and demonstrated weak performance in the evaluation.
- Instead of processing in 16 separate Conv Blocks in Subnetwork $\mathbb{B}$, the feature maps are concatenated to recover 4×4×16 feature maps. The setup of the feature maps during concatenation aims to reproduce the feature map alignment before the split. The concatenated tensor is processed by the Conv2D layer, with a kernel size of 1×1 and stride 1×1. This layer corresponds to the consideration of splitting 16 blocks of size 16×16.
- Lastly, feature maps are processed by the Softmax layer to estimate the output tensor of probabilities, which is DT_B.

The ANN architecture for the Basic Approach ensures a stable training process that yields a model that works very similarly with each new training. Presented Conv Block and layers arrangement with corresponding hyperparameters were found by the try and check method. This process is described in Section 5.4. Further in this dissertation, a set ANN models with architecture presented in Figure 5.1, and trained (according to the description in Section 4.6) for QP values {22, 27, 32, 37}, is referred to as the **Basic Architecture.** The detailed analysis of training results is presented in Section 5.2 and evaluation results are discussed in Section 6.3.

5.2 Training results for the Basic Architecture

5.2.1 Assessment of training accuracy

The training of the Basic Architecture (Section 5.1) was performed according to the description in Section 4.6. According to Section 4.2, four models were achieved as a result of training, one for each QP from CTC [CTCHEVC]: 22, 27, 32, and 37. The $Accuracy$ (Formula 4.8) values for trained Basic Architecture are presented in Table 5.1. Please note that the $Accuracy$ is presented both for the Training and the Validation Subsets of the training dataset.

Depending on the QP value, the $Accuracy$ value varies in the range $\langle 69.7; 74.4 \rangle$. The differences in $Accuracy$ values for Training and Validation Subsets do not exceed 1 p.p. This means that the models are not overfitted. The bigger the QP value, the smaller the $Accuracy$ value. In one case, the $Accuracy$ value is larger for a Validation Subset (QP=22). The bigger the QP, the smaller the Accuracy value, with around 5 p.p. difference between QP values 22 and 37. Such a big difference is connected to the statistics of the training datasets and is further analyzed in this section with Confusion Matrices (CM).

Table 5. 1. Training Accuracy values for Basic Architecture for QP values.

QP Value	*Accuracy* [%]	
	Training Subset	Validation Subset
22	74.2	74.4
27	73.0	72.9
32	72.5	72.4
37	70.5	69.7

The mentioned range of the achieved *Accuracy* values, $\langle 69.7; 74.4 \rangle$, may suggest a relatively poorly trained models. However, in the *Accuracy* metric (formula 4.8), each outputted set of probabilities is individually compared with a one-hot vector from Ground Truth. There are two reasons to consider this range of *Accuracy* value as sufficient. Firstly, the model may estimate a similar probability for at least two depth level values for a certain CTU subarea. The result of $\mathrm{ArgMax_d}$ in the *Accuracy* formula (4.8) may indicate a different depth level than in Ground Truth. Secondly, when the partitioning decisions in HM are made by comparing bit costs (Section 2.3.2), the difference in estimated bit costs is minimal in some cases. Thus, the CTU samples contain features that may be assigned to more than one depth level by the ANN. However, the partitioning pattern estimated by HM lacks information on the superiority margin for the indicated depth level. The model may then estimate a higher probability value for a depth level different from Ground Truth. The negative impact on *Accuracy* value may be more significant in partitioning patterns that comprise bigger CUs. In such cases, all ANN outputs corresponding to that CTU subarea should imply the same depth level, which may be difficult due to the presented reasoning.

Figure 5.3 presents an example of a learning curve (*QP*=27). The models do not overfit during the training. The *Accuracy* achieves high value relatively quickly, and then the values increase quite slowly. The curve for the Training Subset steadily rises. The curve for the Validation Subset fluctuates but is close to the curve for the Training Subset. Analysis of training curves for *QP* values showed a faster termination of training for bigger *QP* values.

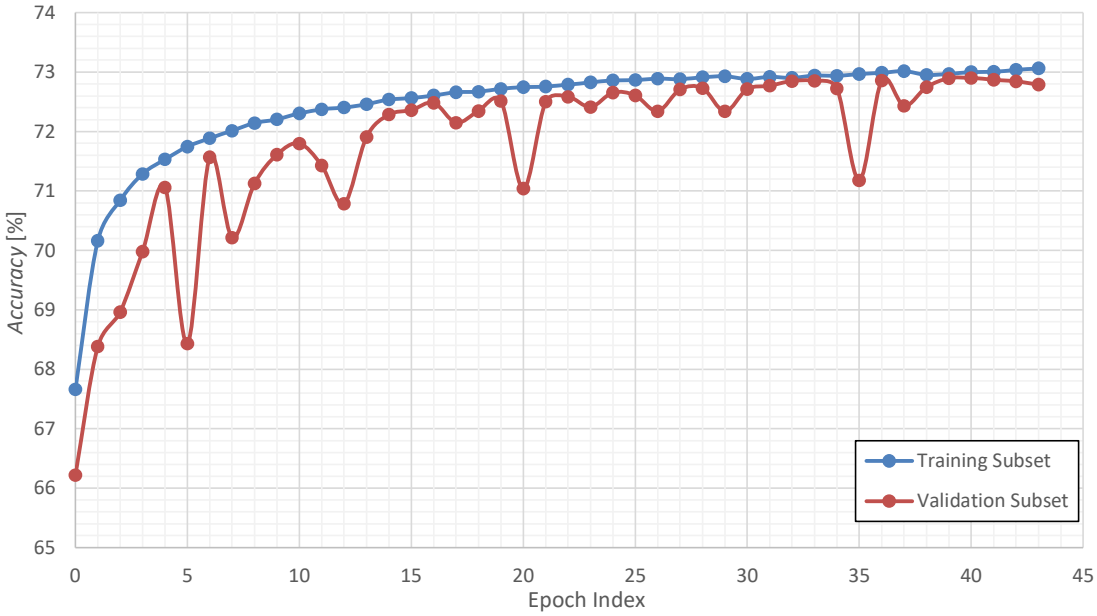

Figure 5. 3. Learning curves (Accuracy) of Basic Architecture for Q= 27. The blue curve represents the results for the Training Subset, and the red curve represents the results for the Validation Subset.

5.2.2 Analysis of Confusion Matrices

The Basic Architecture (Section 5.1) is trained to estimate the depth level of a certain CTU subarea as a class (Subsection 4.2.2). The assessment of such multiclass classifiers with Confusion Matrices (Subsection 3.6.1) was used to analyze the differences in $Accuracy$ (Formula 3.8) results between QP values observed in the previous subsection. Confusion Matrix allows checking the effectiveness of the model prediction for depth level values (classes of classifier). Additionally, it shows which depth level values are confused and to what extent. Confusion Matrices for each model were calculated for both the Training and the Validation Subsets. Tables 5.2 to 5.9 present values of calculated Confusion Matrices (Formula 4.9). The number of samples (*N)* for Formula 4.9 is calculated by multiplying the number of samples in the given subset (Section 4.5) by size of the Division Tensor:

- $522939 \cdot 4 \cdot 4 = 8367024$ for the Training Subset;
- $66650 \cdot 4 \cdot 4 = 1066400$ for the Validation Subset.

Values on the diagonal of the Confusion Matrix (True Positive Count – Formula 3.19) are bolded for readability.

Table 5. 2. Confusion matrix for Basic Architecture (QP=22), Training Subset, normalized. Values expressed in [%].

		Ground Truth (HM)			
	Depth value	**0**	**1**	**2**	**3**
ANN output ($ArgMax_d$)	**0**	**2.208**	4.172	0.244	0.167
	1	2.094	**17.426**	1.804	2.211
	2	0.568	5.321	**7.897**	7.030
	3	0.010	1.481	2.421	**44.947**

Table 5. 3. Confusion matrix for Basic Architecture (QP=27), Training Subset, normalized. Values expressed in [%].

		Ground Truth (HM)			
	Depth value	**0**	**1**	**2**	**3**
ANN output ($ArgMax_d$)	**0**	**6.930**	4.822	0.262	0.169
	1	3.754	**19.206**	1.947	2.014
	2	0.231	4.301	**9.491**	6.668
	3	0.019	1.258	2.607	**36.321**

Table 5. 4. Confusion matrix for Basic Architecture (QP=32), Training Subset, normalized. Values expressed in [%].

		Ground Truth (HM)			
	Depth value	**0**	**1**	**2**	**3**
ANN output ($ArgMax_d$)	**0**	**12.137**	5.055	0.485	0.182
	1	3.177	**19.722**	2.883	1.544
	2	0.209	4.442	**12.000**	5.521
	3	0.052	1.549	3.411	**27.632**

Table 5. 5. Confusion matrix for Basic Architecture (QP=37), Training Subset, normalized. Values expressed in [%].

		Ground Truth (HM)			
	Depth value	**0**	**1**	**2**	**3**
ANN output ($ArgMax_d$)	**0**	**15.514**	5.734	0.688	0.243
	1	3.581	**21.467**	3.802	1.649
	2	0.244	4.646	**13.433**	5.023
	3	0.060	1.120	3.337	**19.458**

Similar Confusion Matrices are observed for both Training and Validation Subsets. The best Confusion Matrices are observed for the depth level value 37. For these matrices, the values on the diagonal are always the highest, considering each row (the depth level value indicated by ANN output). The worst Confusion Matrices are observed for the depth level value 22, as the smallest values on the diagonal are observed. These results are rationalized with the distribution of the depth level values presented in Subsection 4.2.2. As the *QP* increases, the distributions of the depth level values are closer to uniform. The observations for Confusion Matrices are counterintuitive to $Accuracy$ results (Subsection 5.2.1), where for the best achieved $Accuracy$ was for *QP*=22 and the worst for *QP*=37.

The results in Confusion Matrices for *QP*=22 indicate a poor prediction of the depth value 0 (interpreted as the biggest block size). Additionally, for smaller *QP* values 22 and 27, the low relevance for depth levels 0 and 2 is observed. Overall, all models predict the most accurate for the depth levels 1 and 3. All these observations coincide with statistics of depth level values, shown in Figure 4.8 and 4.9 (Subsection 4.2.2).

Table 5. 6. Confusion matrix for Basic Architecture (QP=22), Validation Subset, normalized. Values expressed in [%].

		Ground Truth (HM)			
	Depth value	**0**	**1**	**2**	**3**
ANN output (ArgMax$_d$)	**0**	**3.133**	4.022	0.172	0.193
	1	1.714	**14.855**	1.444	2.189
	2	0.381	4.912	**7.218**	7.517
	3	0.009	1.658	2.396	**48.187**

Table 5. 7. Confusion matrix for Basic Architecture (QP=27), Validation Subset, normalized. Values expressed in [%].

		Ground Truth (HM)			
	Depth value	**0**	**1**	**2**	**3**
ANN output (ArgMax$_d$)	**0**	**7.897**	4.211	0.245	0.209
	1	3.379	**16.178**	1.864	2.144
	2	0.190	3.980	**9.509**	7.260
	3	0.019	1.322	2.776	**38.818**

Table 5. 8. Confusion matrix for Basic Architecture (QP=32), Validation Subset, normalized. Values expressed in [%].

		Ground Truth (HM)			
	Depth value	**0**	**1**	**2**	**3**
ANN output (ArgMax$_d$)	**0**	**12.726**	4.338	0.464	0.225
	1	2.496	**17.397**	2.986	1.810
	2	0.178	4.176	**12.344**	5.991
	3	0.042	1.473	3.776	**29.577**

Table 5. 9. Confusion matrix for Basic Architecture (QP=37), Validation Subset, normalized. Values expressed in [%].

		Ground Truth (HM)			
	Depth value	**0**	**1**	**2**	**3**
ANN output (ArgMax$_d$)	**0**	**15.160**	5.183	0.722	0.312
	1	2.944	**19.732**	4.124	1.893
	2	0.230	4.667	**14.145**	5.352
	3	0.059	1.180	3.703	**20.594**

To further study the trained Basic Architecture, the Confusion Matrices were assessed using metrics described in Subsection 3.6.1. The results are shown in Table 5.10 for the Training Subset and in Table 5.11 for the Validation Subset. The Basic Architecture performs similarly for Training and Validation Subsets. A detailed comparison of the results indicates better performance for the Validation Subset. This confirms a proper generalization of the modeled problem in ANNs.

The best metrics values for each QP are observed for the depth level value 3. The high $Recall$ (Formula 3.23) and $Precision$ (Formula 3.24) values are observed, which results in $F1\text{-}score$ (Formula 3.25) have a high value, in the range $\langle 77.29; 87.10 \rangle$ for both Training Validation Subsets. The classification of depth level 1 is almost identical despite the QP considering the $F1\text{-}score$ (~66%). A significant difference (0.2) between $Recall$ and $Precision$ values is observed only for QP=22. The depth level 0 for QP=22 has the worst results among all depth levels: $F1\_score$ is as low as 37.83% for the Training Subset, and for the Validation Subset, it is slightly better: 0.49. However, for higher QP values, the model achieves similar results for depth level value 0 as observed for depth level values 1 and 3. The worst results were achieved for depth level value 2. The $F1\_score$ value never exceeds 0.6. Additionally, the difference between $Recall$ and $Precision$ surpasses 0.2 for QP values 22 and 27.

Models for different QP values classify the depth level values 1 and 3 similarly and are significantly better in identifying the class instance ($Recall$) than identifying the class correctly ($Precision$). The opposite observations are for depth level values 0 and 2, where the classification is better for models trained for bigger QP values. Additionally, the $Recal$ is always worse than $Precision$. Therefore, the Basic Architecture is overfitted toward depth level values 1 and 3, especially the second one. This is further confirmed in results of $Recall_{micro}$/ $Precision_{micro}$ (Formula 3.26 and 3.27) and $Recall_{macro}$ (Formula 3.28) and $Precision_{macro}$ (Formula 3.29). The $Recall_{micro}$/ $Precision_{micro}$ have similar values for all QP values, around 70%. For the $Recall_{macro}$ and $Precision_{macro}$, the higher the QP, the higher the metric value. The difference between the values of those metrics is ~5 p.p. for QP=22 and ~1 p.p. for QP values 32 and 37.

Table 5. 10. Analysis of the Confusion Matrix with assessment metrics – Basic Architecture, Training Subset. Values expressed in [%].

QP	Depth value d	$Recall$	$Precision$	$F1\text{-}score$	$Recall_{macro}$	$Precision_{macro}$	$Recall_{micro}$ $Precision_{micro}$
22	0	32.51	45.24	37.83	59.12	63.29	72.48
	1	74.04	61.36	67.11			
	2	37.94	63.86	47.60			
	3	91.99	82.69	87.10			
27	0	56.88	63.38	59.96	66.11	68.76	71.95
	1	71.34	64.91	67.98			
	2	45.87	66.34	54.24			
	3	90.34	80.41	85.08			
32	0	67.96	77.93	72.60	69.73	71.29	71.49
	1	72.17	64.10	67.90			
	2	54.12	63.90	58.61			
	3	84.65	79.22	81.84			
37	0	69.95	79.97	74.63	69.76	70.51	69.87
	1	70.38	65.12	67.65			
	2	57.54	63.19	60.23			
	3	81.16	73.78	77.29			

The observations coincide with the histograms of the training dataset presented in Section 4.2 (Figures 4.7 and 4.8). For each QP, the depth level values 1 and 3 were always the most frequent in both Training and Validation Subsets. According to the histograms for QP=22, the depth level 0 appears significantly less frequently in the subsets, which is reflected in metrics. Considering the metrics values,

the best classifier (model) is trained for QP=37. The reasoning is the distribution of the depth level values closest to uniform for the QP=37.

Table 5. 11. Analysis of the Confusion Matrix with assessment metrics – Basic Architecture, Validation Subset. Values expressed in [%].

QP	Depth value d	$Recall$	$Precision$	$F1\text{-}score$	$Recall_{macro}$	$Precision_{macro}$	$Recall_{micro}$ $Precision_{micro}$
22	0	41.66	59.83	49.12	60.86	66.36	73.39
	1	73.53	58.38	65.08			
	2	36.04	64.27	46.18			
	3	92.22	82.96	87.35			
27	0	62.86	68.76	65.68	66.84	69.49	72.40
	1	68.66	62.97	65.69			
	2	45.41	66.07	53.83			
	3	90.41	80.15	84.97			
32	0	71.68	82.41	76.67	70.34	71.92	72.04
	1	70.46	63.53	66.82			
	2	54.41	63.08	58.42			
	3	84.83	78.66	81.62			
37	0	70.92	82.42	76.24	69.58	70.51	69.63
	1	68.77	64.14	66.38			
	2	57.98	62.33	60.08			
	3	80.65	73.16	76.72			

Considering the Basic Architecture has a multidimensional output, the results are promising. However, from the standpoint of the partitioning algorithm, the goal is to estimate the whole partitioning pattern. The assessment of the model for lower depth levels indicates worse performance than for higher depth levels. Multiple ANN outputs influence the decision for bigger blocks in CTU, and then the inaccuracy of the model for these depth level values may be compensated. Such compensation is less likely for the depth level value 2. When the ANN output suggests a significantly different partitioning pattern compared to the HM, the decision algorithm may still estimate the same or similarly efficient one.

5.3 Evaluation of the Basic Architecture in the encoder

Before the actual evaluation with test sequences, the Basic Architecture (Section 5.1) was evaluated with images used for training (cropped DIV2k dataset, Section 4.2). The evaluation of the models was performed according to description in Section 3.1. The achieved results are presented in Table 5.12. The presented results are relative to HM. Recalling the ~70% $Accuracy$ results for Basic Architecture, the results of the evaluation show that the proposed partitioning algorithm with such a trained network performs very well. The difference between Training and Validation Subsets is ~0.3 p.p. in $BD\text{-}RATE$ and 0.006 dB in $BD\text{-}PSNR$. This proves that the models were trained successfully. The encoding time reduction (Time Savings: TS, Formula 3.3achieved with the proposed model is almost the same, ~53%.

Table 5. 12. Evaluation of Basic Architecture used in Modified HM on training dataset images. Presented results are relative to HM.

	$BD\text{-}RATE$ [%]	$BD\text{-}PSNR$ [dB]	TS [%]
Training Subset images	1.45	-0.069	53.75
Validation Subset images	1.73	-0.075	52.65

The results of the Basic Architecture evaluation on test sequences (Section 3.5) are presented in Table 5.13. The evaluation of the models was performed according to description in Section 3.1. The presented results are relative to HM. For most of the test sequences, an increase of ~2% $BD\text{-}RATE$ is observed. Among test sequences, the worst results were achieved for the whole class E (talking heads

content), “BasketballDrive” (class B), and “BasketballDrill” (class C). The best metric results were achieved for sequences with the smallest resolution (class D). Observations for $BD\text{-}PSNR$ are the same as for $BD\text{-}RATE$. The models did perform well with sequences of bigger and smaller resolutions compared to those appearing in DIV2k. The overall results are a little worse than for the Validation Subset, especially compared to class B, which contains sequences of similar resolution. Sequences with the worst results have a similar type of content, the partially visible people. The reason is the insufficient number of images with such content in the training dataset.

The proposed partitioning algorithm with Basic Architecture reduced the encoding time compared to HM. In most cases, for higher resolution, better TS results are observed. For sequences “NebutaFestival”, “SteamLocomotiveTrain” or “Kimono1” the encoding time is reduced by at least 70%. For lower resolution sequences, the TS is not that impressive. For class D, the TS does not exceed 50% and may be as low as 38.99%. Sequences with smaller resolutions have smaller number of CTUs per frame, and the objects in sequence are smaller (proportionally to the resolution). In such cases, smaller blocks will be utilized more frequently. A similar impact of smaller CU blocks is observed for sequences in class A. two sequences in class A (“SteamLocomotiveTrain” and “Kimono1”), the evaluation showed a much better TS than for the rest of the sequences (“PeopleOnStreet” and “Traffic”). For “PeopleOnStreet” and “Traffic” sequences, the content presents multiple small moving objects, while for “SteamLocomotiveTrain” and “Kimono1” the content is much simpler. In rest resolution classes, the TS also varies between sequences, but not so significantly.

Table 5. 13. Evaluation of Basic Architecture used in Modified HM on test sequences. Presented results are relative to HM.

JCT-VC class	Sequence Name	$BD\text{-}RATE$ [%]	$BD\text{-}PSNR$ [dB]	TS [%]
A	NebutaFestival	1.40	-0.101	72.53
	PeopleOnStreet	2.22	-0.127	50.99
	SteamLocomotiveTrain	2.12	-0.061	75.03
	Traffic	2.22	-0.120	53.72
B	BQTerrace	1.47	-0.077	52.50
	BasketballDrive	3.04	-0.089	62.76
	Cactus	2.26	-0.082	59.94
	Kimono1	1.93	-0.067	72.21
	ParkScene	1.73	-0.077	62.06
C	BasketballDrill	2.60	-0.127	48.53
	BQMall	1.65	-0.092	52.15
	PartyScene	0.49	-0.035	38.30
	RaceHorses	1.56	-0.091	53.28
D	BasketballPass	1.47	-0.089	49.30
	BlowingBubbles	0.45	-0.030	44.18
	BQSquare	0.64	-0.051	38.99
	RaceHorsesLow	1.15	-0.072	48.00
E	FourPeople	2.59	-0.149	56.34
	Johnny	3.47	-0.145	65.38
	KristenAndSara	2.71	-0.138	65.39

The above observations conclude that the TS is highly dependent on the content of the sequences. In the previous section, the assessment of the Basic Architecture indicated the best classification results for depth level 3 and worse for the rest, especially depth level 0. The evaluation results show that the more complex and condensed the sequence content, the lower the TS. Therefore, the best TS is achieved when smaller CUs are used more excessively in the estimated partitioning patterns.

The impact of the small CU blocks was indicated as the cause of the lower results of TS. Other factor affecting the encoding time is the ANN processing. To check this, the T_{model} (aggregated time of ANN processing) was measured for all encoded sequences. The T_{model} for a given sequence was then divided by the total number of CTUs in the considered sequence. The average ANN time processing for a single CTU is similar despite the sequence or *QP*. For a machine used in time assessment of Basic Architecture, the statistics of average ANN time processing for a single CTU are:

- Mean value: 2.48 ms
- Standard deviation 55.48 μs

Simultaneously, mean time of CTU processing (over all sequences and *QP* values) was calculated: 29.99 ms for HM and 14.00 ms for Modified HM (Basic Approach). The ANN processing time is relatively constant. This fact is used to check the impact of the CU block size on the encoder complexity. As shown in statistics of the depth level values for the Basic Approach (Section 4.2.2), the smaller *QP* values favor smaller CUs, and bigger *QP* values favor bigger CUs. Thus, the changes in the contribution of network processing time to coding time T_{ANN} (Section 3.3, Formula 3.3) observed *QP* values is used to estimate the impact of the CU blocks in partitioning patterns on encoder complexity. Table 5.14 presents the statistics of T_{ANN} for sequence resolution classes and *QP* values.

Table 5. 14. Statistics of T_{ANN} of Basic Architecture for a given set of sequences and QP. The All class refers to the mean over all results for the given QP value.

JCT-VC class	*QP*	$\mu(T_{ANN})$ [%]	$\delta(T_{ANN})$ [%]
A	22	20.48	7.85
	27	24.70	8.83
	32	27.22	8.56
	37	31.85	7.73
B	22	15.11	5.77
	27	21.55	5.95
	32	26.65	6.22
	37	32.75	5.83
C	22	10.31	1.56
	27	13.27	2.35
	32	16.76	3.02
	37	22.42	4.64
D	22	10.23	1.84
	27	12.97	2.46
	32	15.69	3.19
	37	19.70	4.16
E	22	20.03	2.84
	27	24.42	3.48
	32	28.42	3.49
	37	33.55	3.25
All	**22**	**14.99**	**6.46**
	27	**19.24**	**7.41**
	32	**22.86**	**7.73**
	37	**28.02**	**7.94**

The bigger the *QP*, the higher the T_{ANN}. This relation is observed for each sequence class. Considering statistics for different sequence resolution classes, the smaller the resolution, the smaller the T_{ANN}. The highest standard deviation values are observed for sequence resolution class A, where a difference in the content complexity was indicated as a potential cause of lower TS. Therefore, it was confirmed, that the bigger the share of big CUs in partitioning patterns, the lower the complexity of the encoder. When smaller blocks are used more frequently, worse results of TS are expected. This

conclusion conforms with observations presented earlier in this section. Further experiments on the impact of the CU block size on encoder computation are presented in Section 6.3 and Subsection 7.3.1.

Further assessment of Basic Architecture is done with RD curves. For the readability of this dissertation, only curves for the best and the worst performing sequences are presented. The best value of the $BD\text{-}RATE$ was achieved for the class C sequence ”PartyScene”, which curves are shown in Figure 5.4. The worst value of the $BD\text{-}RATE$ was observed for the class E sequence “Johnny”, which curves are shown in Figure 5.5.

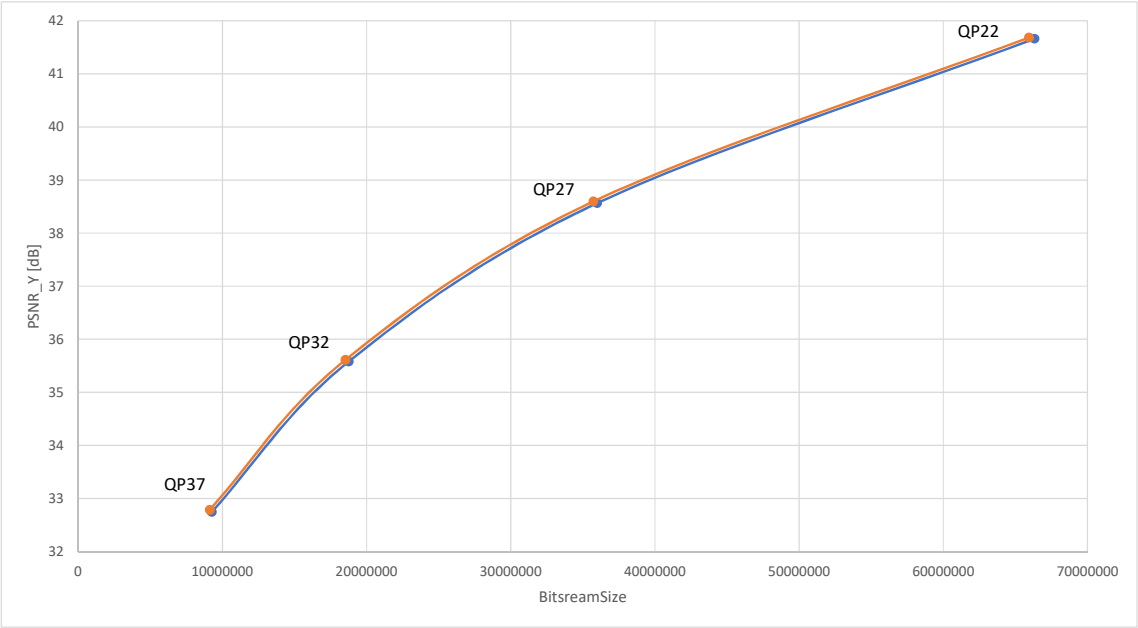

Figure 5. 4. RD curves for HM (orange) and Modified HM with Basic Architecture (Blue) for Test sequence “PartyScene”(Class C)

For the ”PartyScene” sequence, the curves for HM and Modified HM with the Basic Architecture coincide. The differences between curves are negligible. For the “Johnny” sequence, the distance between the curves is more noticeable yet still quite small. The distance between curves visually increases near points for higher QP values. However, the Euclidian distance between the corresponding points is bigger for the smaller QP values. Therefore, the bigger the QP, the closer the curves. This observation coincides with the conclusion from Section 5.2, where the model trained for QP=37 was indicated as the best.

RD curves for the rest of the test sequences were analyzed, and the observations were the same as those presented. The curves for any test sequence were intersecting. Given that, the assessment of only $BD\text{-}RATE$ would be sufficient for the Basic Architecture.

Lastly, the presented Basic Architecture is compared with random partitioning algorithms. As described in Subsection 4.7.4, two random partitioning methods are implemented in Modified HM. The first one estimates the partitioning pattern by thresholding the random value from the uniform distribution. The second one draws the partitioning pattern from the list of all possible ones. The results of the comparison of random methods with the proposed Basic Architecture are shown in Table 5.15. The presented results are relative to HM. Values labeled “All” are calculated by averaging the results for all test sequences.

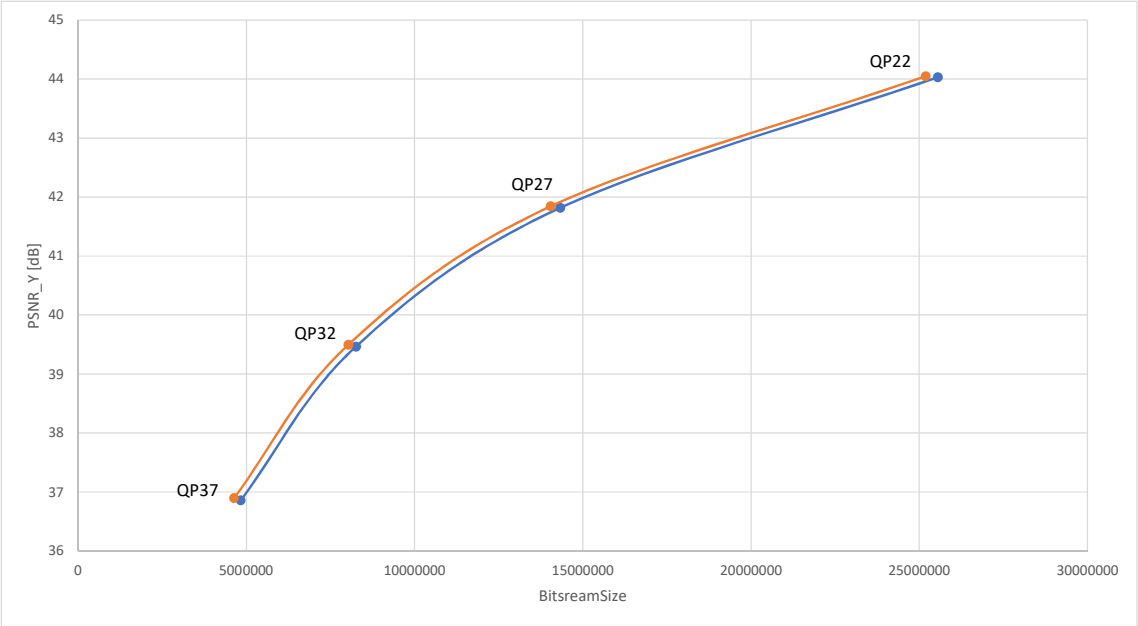

Figure 5. 5. RD curves for HM (orange) and Modified HM with Basic Architecture (Blue) for Test sequence "Johnny"(Class E)

The proposed algorithm is better, by at least 11 p.p, in terms of $BD\text{-}RATE$, than random partitioning algorithms. However, in terms of TS the random algorithms are better, especially the first one. Such difference in $BD\text{-}RATE$ proves that the evaluation results of the proposed method are the effect of adequately estimated partitioning patterns and not the rest of the optimization done in the Modified HM.

In the first random partitioning algorithm, the random split flags more frequently indicate partitioning patterns composed of bigger CUs. In the second one, the small CUs are more frequent, the CTU can be divided into more combinations of small blocks (Section 2.2), and partitioning patterns are equally likely. The second random partitioning algorithm is better than the first one by 3 p.p. in terms of $BD\text{-}RATE$. Therefore, bigger blocks are less computationally complex to encode but not as efficient in bitrate. This observations are further explored in Section 7.3.1.

Table 5. 15. Comparison of proposed partitioning algorithm (Basic Architecture) with two random partitioning algorithms implemented in Modified HM. Presented results are relative to HM.

JCT-VC class	Modified HM with Basic Architecture)			Modified HM with random partitioning algorithm (1)			Modified HM with random partitioning algorithm (2)		
	***BD-RATE* [%]**	***BD-PSNR* [dB]**	***TS* [%]**	***BD-RATE* [%]**	***BD-PSNR* [dB]**	***TS* [%]**	***BD-RATE* [%]**	***BD-PSNR* [dB]**	***TS* [%]**
A	1.99	-0.102	63.07	11.22	-0.568	80.11	13.25	-0.624	62.54
B	2.09	-0.079	61.90	12.38	-0.445	79.42	12.39	-0.443	62.26
C	1.57	-0.086	48.07	16.87	-0.904	79.35	10.32	-0.575	63.17
D	0.93	-0.061	45.12	14.72	-0.936	78.39	9.71	-0.633	61.91
E	2.93	-0.144	56.57	24.55	-1.132	79.89	19.85	-0.925	62.79
All	**1.86**	**-0.091**	**56.08**	**15.34**	**-0.763**	**79.41**	**12.73**	**-0.616**	**62.51**

The proposed partitioning algorithm with the Basic Architecture proved to perform well in the Modified encoder despite the training $Accuracy$ of ~70%. The method significantly reduces the encoding time while maintaining a very similar bitrate to HM.

5.4 Basic Architecture tuning

The Basic Architecture (Section 5.1) is the result of long-lasting research and a methodical iterative process. The architecture development was time-consuming and resource-costly, as it consisted of verification of the ideas in a try-and-check procedure. Following the greedy strategy, several hundred model variants were tested. This section presents a snippet of the selected most interesting directions of architecture development and attempts to overcome problems with training procedure and data (presented in Section 4.2). All presented tuning approaches were discarded due to rules defined in Subsection 4.6.3 (converge of the training process and increase of evaluation result by 0.1 p.p. in terms of $BD\text{-}RATE$). All model discussed in this section were trained according to the description in Section 4.6.The evaluation of the models was performed according to description in Section 3.1. The architecture tuning presented in this section is divided into two categories, depending on the impact on the complexity of the model: non-affecting and affecting.

5.4.1 Complexity-non-affecting tuning

The modifications considered in this category are: layer type adjustment, training data preparation, and the choice learning rate optimization algorithm. The complexity change after removing the normalization layer is negligible, so it was considered as complexity non-affecting.

The results of evaluation on test sequences (Section 3.5) for applied tunings are presented in Table 5.16. Presented results are relative to HM. Presented tuning are indexed for clarity of description. The term label corresponds to the class label, the depth level value in one-hot format. Section 9.3 presented another exploration experiment for the ANN architecture for Basic Approach (Figure 5.1, Section 5.1), which was considered a viable option for memory-restricted applications.

Table 5. 16. Assessment of selected complexity non-affecting tuning – ANN architecture for Basic Approach. "No modification" refers to results for Basic Architecture. Presented results are relative to HM.

Modification Type	Idx	Description modification for ANN model for Basic Approach	$BD\text{-}RATE$ [%]	$BD\text{-}PSNR$ [dB]
Layer Type Adjustment	1	ReLU as activation	1.85	-0.091
	2	LeakyReLU as activation (0.1)	1.83	-0.090
	3	LeakyReLU as activation (0.2)	1.85	-0.091
	4	Removing BatchNorm from the network	1.95	-0.096
Training data preparation	5	CTU partitioning category samples restriction (max 100)	2.05	-0.100
	6	Label smoothing – method 1 – coef: 0.01	1.82	-0.090
	7	Label smoothing – method 1 – coef: 0.20	1.82	-0.089
	8	Label smoothing – method 2 – coef: 0.01	1.82	-0.090
	9	Label smoothing – method 2 – coef: 0.20	1.90	-0.094
	10	Label weighting method 1	1.79	-0.087
	11	Label weighting method 2	1.81	-0.089
	12	Label weighting method 3	1.81	-0.089
Learning rate optimization algorithm	13	SGD (LR = 0.001)	2.15	-0.106
	14	NAdam (LR = 0.001)	1.84	-0.090
	15	Adamax (LR = 0.001)	1.86	-0.092
None		**No modification (Basic Architecture)**	**1.86**	**-0.091**

During the development of the ANN architecture for Basic Approach, multiple setups of layers in the Conv Blocks (Figure 5.2, Section 5.1) were tested. As the applied order of layer has been established, changes in activation and normalization were tested. The ReLU (Idx 1) and LeakyReLU (Idx 2 and 3) were tried. The LeakyReLU is a variation of the PReLU, where the slope coefficient is set up as a constant value. Both of these activations showed a slight improvement, but the evaluation results are comparable with PReLU. Then, the application of the batch normalization was questioned. This layer

accelerates the training but may cause the model to be stuck in the local minimum of the gradient [Go16]. The models with removed BatchNorm layers (Idx 4) delivered worse evaluation results.

As mentioned in Section 5.2, the dataset imbalance negatively impacts the model training. Multiple methods were considered to prevent this. Among them, the down-sampling of training dataset [He09, Mo12, Ar22, Ab23], label smoothing [Go16, Mu19] and label weighting [Go16, By19, Xu21]. The evaluation results of models trained with these methods are presented in Table 5.16.

The down-sampling of the training dataset was applied as a restriction of training samples with the same partitioning pattern. The best-achieved result was for 100 randomly chosen training samples per partitioning pattern (Idx 5), but the evaluation results did not improve. Still, such a simple down-sampling method worked better than more complicated methods of drawing training samples from the training dataset.

Then, the label smoothing was tested. As the Ground truth data are tensors with the one-hot interpretation of the DM_B, this may be difficult for the model to train. Thus, sharing a small part of probability with other outputs may improve the training. Two methods of label smoothing were tested: sharing the same small amount of probability (Idx 6 and 7) or sharing the decreasing amount of probability around the indicated class (Idx 8 and 9). As shown in Table 5.16 the improvements in evaluation results are minor.

Further, label weighting was applied. According to the chosen classification problem presented in Section 4.2, the label weight should correspond to each ANN output that refers to a certain depth level value. Unfortunately, the chosen training framework [TENSORFLOW] does not directly support this. Therefore, the label weight was applied as a training sample weight applied during training [TENSORFLOW]. Two statistics were used to estimate the weight of the training sample: depth level values statistics and partitioning pattern statistics (Section 4.2.2). Eleven different formulas for sample weight were tested. Results for the best-performing three among the proposed sample weighting (SW) methods (Idx 10, 11, and 12) are presented in Table 5.16. Their formulas are as follows:

$$SW_1(DM_B) = PartProb(DM_B) \cdot \sum_{i=0}^{4}\sum_{j=0}^{4}\left(1 - \ln\left(\frac{DepthProb(DM_B[i,j])}{maxDepthProb}\right)\right), \tag{5.1}$$

$$\begin{aligned} SW_2(DM_B) = &\left(1 - \log10\left(\frac{PartProb(DM_B)}{maxPartProb}\right)\right) \\ &\cdot \sum_{i=0}^{4}\sum_{j=0}^{4}\left(1 - \log10\left(\frac{DepthProb(DM_B[i,j])}{maxDepthProb}\right)\right), \end{aligned} \tag{5.2}$$

$$SW_3(DM_B) = \left(\frac{1}{4} - PartProb(DM_B) + 1\right) \cdot \sum_{i=0}^{4}\sum_{j=0}^{4}\left(\frac{1}{4} - DepthProb(DM_B[i,j]) + 1\right), \tag{5.3}$$

where $PartProb()$ is the probability of the partitioning pattern (Subsection 4.2.1), $maxPartProb$ is is the maximum value of probability among partitioning patterns, $DepthProb()$ is the probability of the depth level (Subsection 4.2.2), and $maxDepthProb$ is the maximum value of probability among depth levels. All probability values are statistics of the Training Subset.

The evaluation of presented tunings turns out to be superior to the Basic Architecture but by a margin that is too small. Unfortunately, these results were obtained only once, as re-training of the models with the same conditions yielded much worse results. The applied label weighting was very sensitive to the randomness of the training process, discussed in Subsection 4.6.3.

Lastly, the training with different learning rate optimization algorithms was explored. Multiple algorithms were tested with several learning rate values. The learning rate value of 0.001 was the best for all methods. Among learning rate optimization algorithms, the best were SGD (Idx.13), NAdam (Idx 14), and Adamax (Idx 15). The NAdam achieved slightly better results than the Adam used in Basic Architecture training. Still, further experimentation did not confirm the superiority, so Adam was chosen for use in further training.

5.4.2 Complexity-affecting tuning

The development of the ANN architecture was made in two different directions: decreasing the model complexity while maintaining the $BD\text{-}RATE$, and increasing the model complexity while reducing the $BD\text{-}RATE$. The results of evaluation on test sequences (Section 3.5) for applied tunings are presented in Table 5.17. Presented results are relative to HM. The complexity is assessed with the Multiply and Accumulate (MAC) operations count (in millions: M) and the number of weights in the model. Presented tuning are indexed for clarity of description.

Table 5. 17. Assessment of selected complexity affecting tuning – ANN architecture for Basic Approach. "No modification" refers to results for Basic Architecture. Presented results are relative to HM.

Modification Type	Idx	Description modification for ANN model for Basic Approach	$BD\text{-}RATE$ [%]	$BD\text{-}RATE$ [dB]	MAC count [M]	Num. of weights
Model complexity reduction	1	Reduction of filter number: Subnetwork 𝔸: 6,12,28,24 Subnetwork 𝔹: 32,8,4	1.98	-0.097	1.82	23 194
	2	Additional Conv Blocks in Subnetwork 𝔸 (reduced filter number): 4 more blocks	1.97	-0.097	4.07	39 863
	3	Additional Conv Blocks in Subnetwork 𝔸 (reduced filter number): 8 more blocks	1.98	-0.098	5.78	46 568
	4	Adding bottleneck convolution layer: ½	1.92	-0.091	4.57	53 248
	5	Adding bottleneck convolution layer: ¼	2.06	-0100	2.54	25 926
	6	Adding bottleneck convolution layer: ¾	1.88	-0.090	6.56	75 442
Model complexity increase	7	Increase of filter number (by 12 in each layer except the last)	1.89	-0.091	15.34	122 140
	8	Increase of filter number (by 24 in each layer except the last)	1.93	-0.093	27.41	160 456
	9	Additional Conv Blocks in Subnetwork 𝔸: 4 more blocks	2.27	-0.112	15.76	120 442
	10	Additional Conv Blocks in Subnetwork 𝔸: 8 more blocks	1.79	-0.086	24.12	148 552
None		**No modification (Basic Architecture)**	**1.86**	**-0.091**	**6.76**	**91 600**

The ANN architecture for Basic Approach (Figure 5.1, Section 5.1) is low complex. Further complexity reduction was achieved by reduction of the filter number in layers. Most of the computations are done in the Subnetwork 𝔸, as the biggest feature maps are processed. The reduction of filter numbers was extensively explored. Multiple configurations were tested, but most of the time, a change in Subnetwork 𝔸 filter numbers resulted in a significant increase of $BD\text{-}RATE$. In Table 5.17 (Idx 1), it is shown that for the filter number halved in all layers (except the last convolution), the reduction of the MAC operation count was operations count around 4 times, but the $BD\text{-}RATE$ increased by 0.12. Unfortunately, further tries at training such a network yielded a much bigger increase in $BD\text{-}RATE$.

The potential of the complexity reduction was proven, but the model was too unstable in training. Thus, the number of computations was reduced by using a deeper network with a smaller number of filters in layers. Previously presented architecture with the filter number halved in all layers (except the last convolution) served as a base. As the primary source of complexity in the Subnetwork 𝔸 was identified, the additional Conv Blocks (Figure 5.2, Section 5.1) would be applied before each Conv

Blocks already in the architecture. The number of filters in additional Conv Blocks was adjusted to gradually increase towards the number of filters of Conv Block already in the architecture. An exemplary inclusion of additional Conv Block was presented in Figure 5.6. In Table 5.17 two best experiments were presented: with 4 (Idx 2) and 8 (Idx 3) additional Conv Blocks. The training of the modified model was stable, maintaining the $BD\text{-}RATE$ and $BD\text{-}PSNR$ of the model with a halved number of filters. The architecture with additional Conv Blocks is less complex than the Basic Architecture, but the $BD\text{-}RATE$ and $BD\text{-}PSNR$ are worse.

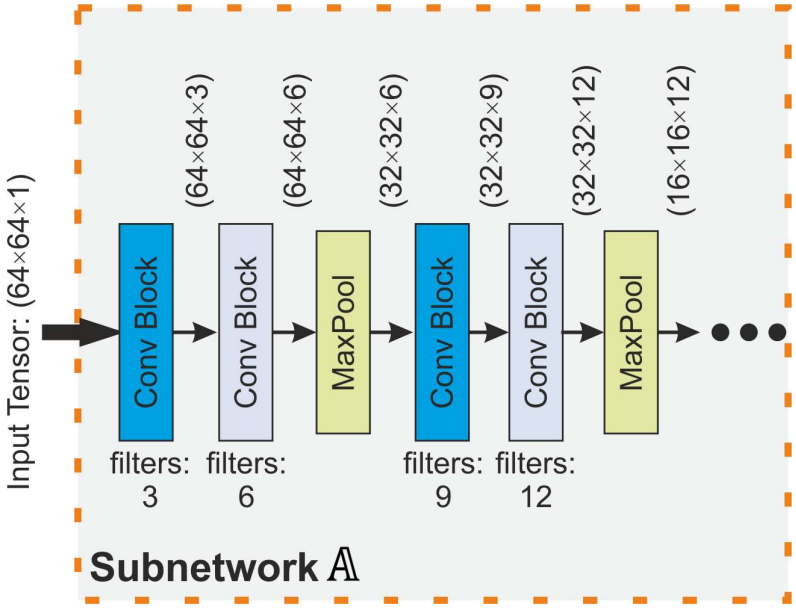

Figure 5. 6. Example of inclusion of additional Conv Blocks.

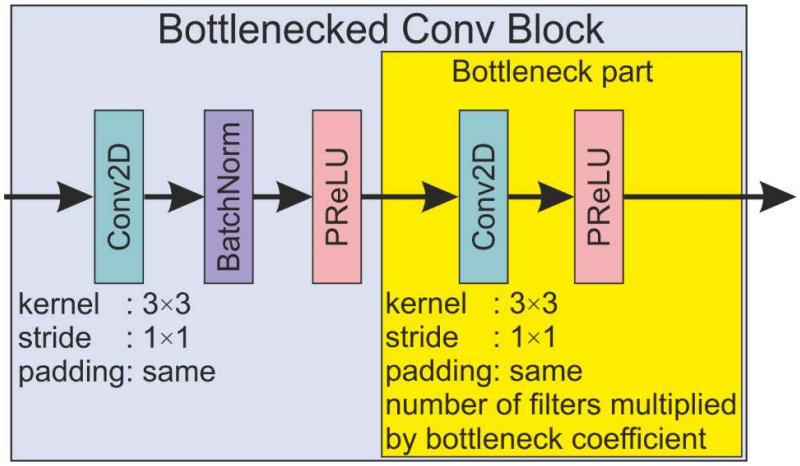

Figure 5. 7. Architecture of the Bottlenecked Conv Block.

The last approach to reduce the model complexity was applying the bottleneck layers in the model [Ti15, Sa19]. The bottleneck was applied as an extension of the Conv Block by additional Conv2D and PReLU layers, as shown in Figure 5.7. The number of filters in additional Conv2D equals the number of filters for a given block multiplied by the bottleneck coefficient. The best results were achieved for the bottleneck coefficients: $\frac{1}{2}$ (Idx 4), $\frac{1}{4}$ (Idx 5), and $\frac{3}{4}$ (Idx 6). All Conv Blocks are replaced by the

Bottlenecked Conv Blocks, except the ones after split in Subnetwork $\mathbb{B}$. The results of the evaluation of the model with Bottlenecked Conv Blocks are presented in Table 5.17. The evaluation results proved this modification to perform better than the application of the additional layers, considering the similar complexity of the model. Still, the results were not good enough to surpass the Basic Architecture.

The opposite direction of research on the model, increasing the model complexity while reducing the $BD\text{-}RATE$, was started by increasing the number of filters in Conv Blocks. Firstly, the experiments with an increased number of filters in each layer were performed. In Table 5.17, two selected experiments are presented that illustrate the observed tendency. The number of filters (except the output layer) was increased by 12 in the first experiment (Idx 7) and by 24 in the second experiment (Idx 8). Smaller increases in the number of filters did not change the evaluation result of the model. With a bigger increase, the evaluation $BD\text{-}RATE$ result got worse. The extensive complexity of the model was identified as a cause of poor training.

Lastly, the number of Conv Blocks was increased, but this time the non-modified ANN architecture for Basic Approach was used. The application of the additional layer was the same as shown in Figure 5.6. The evaluation results of selected experiments (Idx 9 and 10) are presented in Table 5.17. Adding layers has proven to be successful in one case. However, despite stable training and $BD\text{-}RATE$ improvement, the increase in the complexity was too significant to adopt this approach in further experiments.

6 ANN model for the Extended Approach

6.1 ANN architecture changes from the ANN architecture for the Basic Approach

The Extended Approach for the partitioning algorithm is the extension of the Basic Approach, as described in Section 4.1. In the Extended Approach, the partitioning algorithm estimates the CU partitioning and the PU division. The PU division is considered as an additional level of the quaternary tree, indexed as 4. Therefore, the ANN architecture must be appropriately modified.

Following adjustments were applied to the ANN architecture used in Basic Approach to produce DT_E:

- The output tensor size (first two dimensions) is doubled (from 4×4 to 8×8). To address this, the architecture of Subnetwork A was changed by removing the last MaxPool layer.
- The number of output channels has to increase to 5 (to represent PU division). The was attained by increasing the number of filters in the last Conv2D of the Subnetwork B.

The ANN architecture for the Extended Approach is presented in Figure 6.1. Conv Blocks are the same as in the Basic Architecture and described in Section 5.1 (Figure 5.2). Further in this dissertation, a set of ANN models with architecture presented in Figure 6.1 and trained (according to the description in Section 4.6) for *QP* values {22, 27, 32, 37} is referred to as the **Extended Architecture**. The detailed analysis of training results is presented in Section 6.2 and evaluation results are discussed in Section 6.3.

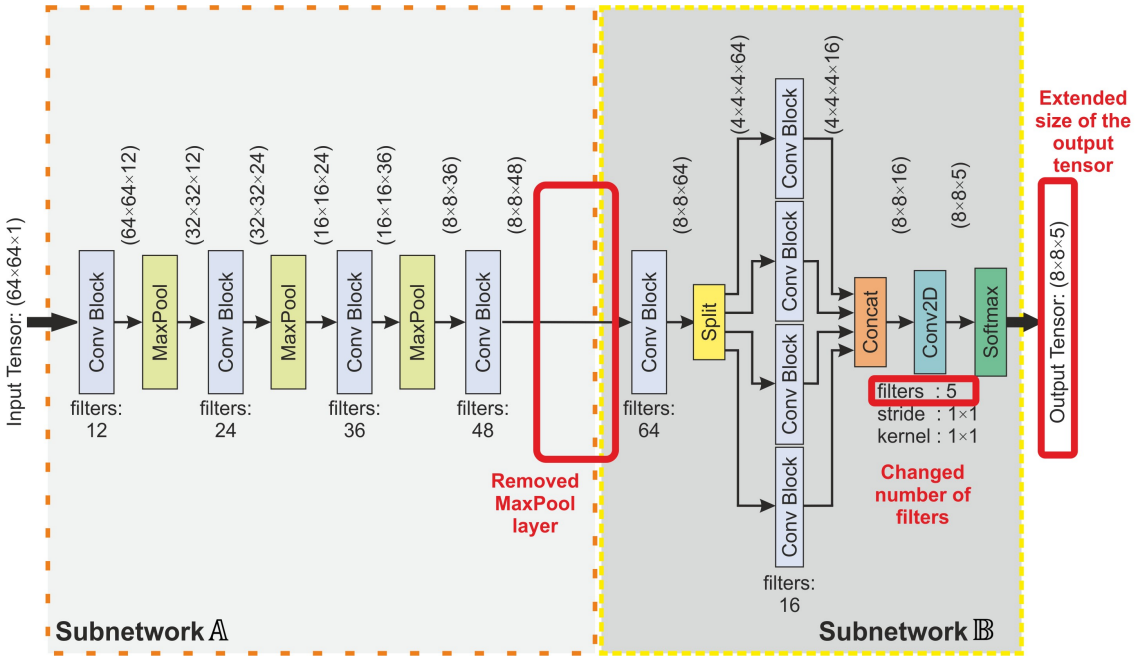

Figure 6. 1. The ANN architecture used in the Extended Approach. Changes to the ANN architecture for Basic Model (Figure 5.1, Section 5.1) are marked with red envelopes. The "Conv Block" refers to block of layers, presented in Figure 5.2 (Section 5.1).

The ANN architecture for the Extended Approach was derived from the ANN architecture for the Basic Approach with a minor set of applied changes. Nonetheless, the architecture is the best one found after a thoughtful development, similar to the Basic Approach. The results of this evaluation for the most

interesting architecture tuning are presented in Section 6.4. The most interesting tested modifications are: MaxPool layer removal choice, more complex network, or Subnetwork $\mathbb{B}$ extension for processing of an additional depth level.

6.2 Training results for the Extended Architecture

6.2.1 Assessment of training accuracy

Training of the Extended Architecture (Section 6.1) was performed according to the description in Section 4.6. For each *QP*: 22, 27, 32, and 37, a separate model was trained. The *Accuracy* (Formula 4.8) values for trained Extended Architecture are presented in Table 6.1. For comparison, *Accuracy* values for Basic Architecture (Section 5.1) are also presented, for the sake of comparison.

Table 6. 1. Training Accuracy values for Extended Architecture. For comparison, results for Basic Architecture are presented.

***QP* Value**	***Accuracy* [%]**			
	Training Subset		**Validation Subset**	
	Extended Architecture	**Basic Architecture**	**Extended Architecture**	**Basic Architecture**
22	60.7	74.2	60.5	74.4
27	62.2	73.0	61.7	72.9
32	63.9	72.5	63.1	72.4
37	64.3	70.5	64.3	69.7

The *Accuracy* values for Extended Architecture are from the range $\langle 61; 64 \rangle$. The results for the Training and the Validation subsets are nearly identical, proving no overfitting. The *Accuracy* value increases as the dataset for higher *QP* is used for the training. Such a relation indicates that the problem with model overfitting for certain depth level values, observed in Section 5.2, does not occur for the Extended Architecture. This will be further investigated with the Confusion Matrices in the next subsection. However, the *Accuracy* values are significantly smaller than those for Basic Architecture. The differences are ~5 p.p. for *QP*=37 and ~13 p.p. for *QP*=22.

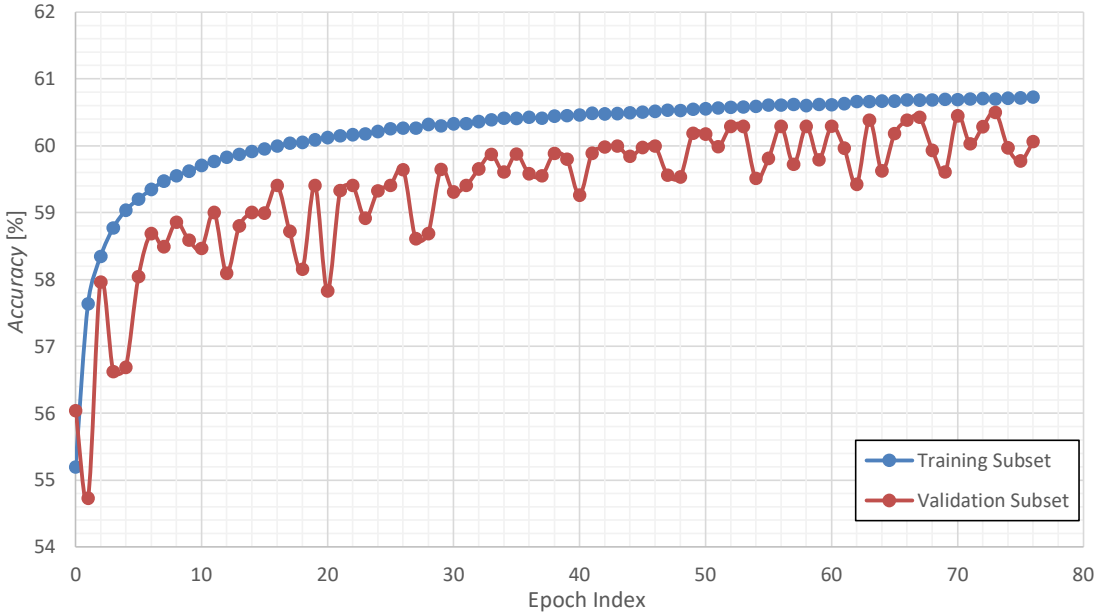

Figure 6. 2. Learning curves (Accuracy) of Extended Architecture for QP=22. The blue curve represents the results for the Training Subset, and the red curve represents the results for the Validation Subset.

Achieved A$ccuracy$ values are even smaller than observed for the Basic Architecture. It should be recalled that in the $Accuracy$ metric (Formula 4.8), each outputted vector of probabilities is individually compared with a one-hot vector from Ground Truth. The same reasoning is used to consider this range of $Accuracy$ value as sufficient, as discussed in Subsection 5.2.1 for Basic Architecture. Moreover, the training problem for the Extended Architecture is far more complex, as the number of outputs from the network increased more than fourfold. This is why the observed $Accuracy$ values are smaller as compared to Basic Architecture. Therefore, the Extended Architecture is expected to perform worse in the evaluation than the Basic Architecture.

Figure 6.2 presents an exemplary learning curve of the Extended Architecture. The curve for the Training Subset rises quickly at first, then steadily rises. The Curve for the Validation Subset follows a similar scheme, but significant fluctuations of the $Accuracy$ are observed. Curves are visually offset, but not by more than 0.5 p.p.

6.2.1 Analysis of Confusion Matrices

The Extended Architecture (Section 6.1) is trained to estimate the depth level of a certain CTU subarea as a class (Subsection 4.2.2). Confusion Matrices (Subsection 3.6.1) have been calculated to further evaluate whether the trained network model is able to classify depth levels reliably and to investigate the nature of inference errors, similarly like in Section 5.2. Confusion Matrices were calculated separately for the Training and Validation Subsets. Tables 6.2 to 6.9 present values of calculated Confusion Matrices (Formula 4.9). The number of samples (N) for Formula 4.9 is calculated by multiplying the number of samples in the given subset (Section 4.5) by size of the Division Tensor:

- $522939 \cdot 8 \cdot 8 = 33468096$ for the Training Subset.
- $66650 \cdot 8 \cdot 8 = 4265600$ for the Validation Subset.

Values on the diagonal of the Confusion Matrix (True Positive Count – 3.19) are bolded for readability.

Table 6. 2. Confusion matrix for Extended Architecture (QP=22), Training Subset, normalized. Values expressed in [%].

		Ground Truth (HM)				
	Depth value	**0**	**1**	**2**	**3**	**4**
ANN output (ArgMax$_d$)	**0**	**1.236**	5.217	0.250	0.062	0.026
	1	0.554	**19.733**	2.021	0.739	0.489
	2	0.082	6.013	**9.739**	2.920	2.061
	3	0.001	1.215	2.958	**10.041**	7.548
	4	0.000	0.706	1.426	4.726	**20.238**

Table 6. 3. Confusion matrix for Extended Architecture (QP=27), Training Subset, normalized. Values expressed in [%].

		Ground Truth (HM)				
	Depth value	**0**	**1**	**2**	**3**	**4**
ANN output (ArgMax$_d$)	**0**	**5.666**	6.111	0.318	0.061	0.028
	1	2.785	**20.811**	2.283	0.624	0.417
	2	0.089	4.795	**11.663**	2.502	1.641
	3	0.003	1.297	3.577	**9.525**	5.539
	4	0.001	0.671	1.500	3.601	**14.491**

Table 6. 4. Confusion matrix for Extended Architecture (QP=32), Training Subset, normalized. Values expressed in [%].

		Ground Truth (HM)				
	Depth value	**0**	**1**	**2**	**3**	**4**
ANN output ($ArgMax_d$)	**0**	**9.479**	7.781	0.448	0.102	0.047
	1	1.992	**21.767**	2.505	0.676	0.386
	2	0.044	5.358	**12.735**	2.672	1.362
	3	0.008	1.411	3.635	**9.493**	3.975
	4	0.003	0.514	1.153	2.869	**9.584**

Table 6. 5. Confusion matrix for Extended Architecture (QP=37), Training Subset, normalized. Values expressed in [%].

		Ground Truth (HM)				
	Depth value	**0**	**1**	**2**	**3**	**4**
ANN output ($ArgMax_d$)	**0**	**14.518**	6.909	0.615	0.112	0.026
	1	3.657	**22.978**	3.097	0.638	0.130
	2	0.169	6.217	**13.950**	2.475	0.535
	3	0.049	1.617	3.994	**8.656**	1.654
	4	0.019	0.528	1.037	2.299	**4.123**

The Confusion Matrices for both Training and Validation subsets are similar. The differences are insignificant, indicating that the training led to a good generalization of the learning problem. Extended Architecture predicts the depth level value 0 slightly worse than Basic Architecture (Section 5.1), especially for *QP* values 22 and 27. The best prediction is observed for depth level value 1, which is even better than for Basic Architecture. The depth level 2 is predicted by Extended Architecture significantly better, notably for smaller *QP* values.

Table 6. 6. Confusion matrix for Extended Architecture (QP=22), Validation Subset, normalized. Values expressed in [%].

		Ground Truth (HM)				
	Depth value	**0**	**1**	**2**	**3**	**4**
ANN output ($ArgMax_d$)	**0**	**2.267**	4.956	0.207	0.054	0.035
	1	0.419	**16.692**	1.772	0.789	0.530
	2	0.056	5.363	**9.170**	3.189	2.250
	3	0.001	1.286	3.037	**10.945**	8.061
	4	0.000	0.792	1.543	5.188	**21.396**

Table 6. 7. Confusion matrix for Extended Architecture (QP=27), Validation Subset, normalized. Values expressed in [%].

		Ground Truth (HM)				
	Depth value	**0**	**1**	**2**	**3**	**4**
ANN output ($ArgMax_d$)	**0**	**6.475**	5.680	0.300	0.063	0.045
	1	2.399	**17.786**	2.244	0.660	0.476
	2	0.067	4.545	**11.779**	2.793	1.754
	3	0.002	1.371	3.827	**10.569**	5.769
	4	0.000	0.721	1.598	3.972	**15.105**

Table 6. 8. Confusion matrix for Extended Architecture (QP=32), Validation Subset, normalized. Values expressed in [%].

		Ground Truth (HM)				
	Depth value	**0**	**1**	**2**	**3**	**4**
ANN output (ArgMax$_d$)	**0**	**10.226**	6.926	0.432	0.111	0.058
	1	1.547	**19.311**	2.625	0.778	0.429
	2	0.035	5.090	**13.177**	2.990	1.397
	3	0.007	1.429	4.008	**10.691**	4.064
	4	0.003	0.522	1.234	3.185	**9.725**

Table 6. 9. Confusion matrix for Extended Architecture (QP=37), Validation Subset, normalized. Values expressed in [%].

		Ground Truth (HM)				
	Depth value	**0**	**1**	**2**	**3**	**4**
ANN output (ArgMax$_d$)	**0**	**14.574**	6.015	0.625	0.133	0.029
	1	3.057	**21.177**	3.520	0.783	0.157
	2	0.143	6.100	**14.802**	2.796	0.552
	3	0.044	1.675	4.389	**9.587**	1.665
	4	0.014	0.511	1.059	2.476	**4.115**

The results for depth level 3 in presented Confusion Matrices are ambiguous. In half of the cases, models predict it correctly, but the second half are mistaken with a smaller or bigger block. For depth level 4, the best results are observed for smaller *QP* values. As the *QP* increases, the Extended Architecture is less accurate for depth level value 4, but still, the results are better than for depth level value 0. The better prediction of Extended Architecture, for some depth level values, is associated with depth level values statistics. As presented in Subsection 4.2.2, the histograms for the Extended Approach (Figure 4.10 and 4.11) are closer to even for each *QP* than for the Basic Approach (Figure 4.8 and 4.9).

An important observation is the tendency of mistaken predictions. For smaller *QP* values, the model tends to make mistakes by indicating bigger depth values, and for bigger *QP* values the opposite. This phenomenon was not observed for the Basic Architecture. This means that the Extended Architecture is better at mimicking the behavior of the partitioning algorithm from HM. The impact of the observed phenomenon will be further discussed in Section 7.3.

The Confusion Matrices calculated for the Extended Architecture were analyzed with metrics described in Subsection 3.6.1. Results are shown in Table 6.10 for the Training Subset and 6.11 for the Validation Subset. The results for the Training and Validation subsets are similar but slightly in favor of the latter. This similarity confirms a proper learning problem generalization during the model training.

The $F1\text{-}score$ (Formula 3.25) for almost all cases rises as the *QP* is increased. The exception here is the depth level 4, where the opposite tendency is observed. In most cases, the $Precision$ (Formula 3.24) is around 60%. Only for the depth level value 0 in *QP* values 32 and 37 the metric value exceed 80%. The $Recall$ (Formula 3.24)is mostly around 50% and increases as the *QP* increases. The best $Recall$ values are observed for depth level 1, which is around 80% despite the *QP*.

The worst classification results are assessed for depth level value 0 for *QP*=22. The very low $Recall$ value, 18.20%, indicates that for this *QP*, the model confuses the prediction. Fortunately, for the rest of the *QP* values, Extended Architecture performs better for this depth level value. The results are comparable with other depth levels and better than those observed for corresponding Basic Architecture. The model for the *QP*=37 has the best predictions, considering depth level value 0, despite not being the most probable depth level according to the histograms shown in Subsection 4.2.2. Depth levels 1 and 2 achieved comparable metric values as the Basic Architecture in most cases. Nevertheless, the

results for *QP* values 22 and 32 are noticeably better for Extended Architecture. The metrics for depth level 1 are the best for most *QP* values compared to other depth levels. The above observations show that Extended Architecture is better in the classification of bigger CUs.

Table 6. 10. Analysis of the Confusion Matrix with assessment metrics (Subsection 3.6.1) – Extended Architecture, Training Subset. Values expressed in [%].

QP	Depth value d	$Recall$	$Precision$	$F1\text{-}score$	$Recall_{macro}$	$Precision_{macro}$	$Recall_{micro}$ $Precision_{micro}$
22	0	18.20	65.96	28.52	53.93	61.27	60.99
	1	83.84	60.01	69.95			
	2	46.79	59.40	52.35			
	3	46.14	54.31	49.89			
	4	74.69	66.66	70.44			
27	0	46.51	66.32	54.67	59.89	62.46	62.16
	1	77.31	61.78	68.68			
	2	56.37	60.30	58.27			
	3	47.77	58.39	52.55			
	4	71.51	65.52	68.39			
32	0	53.08	82.24	64.52	61.86	65.20	63.06
	1	79.66	59.10	67.86			
	2	57.44	62.19	59.72			
	3	51.25	60.04	55.30			
	4	67.86	62.42	65.03			
37	0	65.46	78.85	71.53	61.25	65.04	64.22
	1	75.34	60.08	66.85			
	2	59.75	61.48	60.60			
	3	54.20	61.05	57.42			
	4	51.50	63.74	56.97			

Table 6. 11. Analysis of the Confusion Matrix with assessment metrics (Subsection 3.6.1) – Extended Architecture, Validation Subset. Values expressed in [%].

QP	Depth value d	$Recall$	$Precision$	$F1\text{-}score$	$Recall_{macro}$	$Precision_{macro}$	$Recall_{micro}$ $Precision_{micro}$
22	0	30.15	82.63	44.18	55.89	63.78	60.47
	1	82.62	57.38	67.73			
	2	45.78	58.30	51.29			
	3	46.91	54.28	50.33			
	4	73.98	66.30	69.93			
27	0	51.54	72.40	60.21	60.59	62.98	61.71
	1	75.48	59.08	66.28			
	2	56.26	59.65	57.90			
	3	49.07	58.53	53.39			
	4	70.60	65.25	67.82			
32	0	57.60	86.53	69.16	62.62	65.64	63.13
	1	78.21	58.03	66.63			
	2	58.08	61.36	59.67			
	3	52.93	60.22	56.34			
	4	66.30	62.05	64.11			
37	0	68.18	81.73	74.34	61.65	65.20	64.26
	1	73.81	59.69	66.00			
	2	60.68	60.68	60.68			
	3	55.23	60.77	57.87			
	4	50.34	63.13	56.02			

Metrics values for the depth level 3 are similar to depth level 2. For each *QP*, the $Recall$ value is around 10 p.p. worse than the $Precision$. This coincides with an earlier observation of the tendency for

mistaken predictions. For QP=22, the depth level value 4 is classified very well, as metric values are similar to depth level 1. But, as the QP increases, the results worsen. The metric values for depth level 4 are better than for depth level value 3.

Results for both depth level values 3 and 4 for Extended Architecture are significantly worse compared to depth level 3 for Basic Architecture. It is important to recall that the depth level values 3 and 4 in the Extended Approach practically cover the same range of decisions as depth level 3 in the Basic Approach. However, The results for depth levels 3 and 4 in Extended Architecture are close to results for other depth levels. Therefore, the overfitting is not observed for Extended Architecture, as it was for Basic Architecture.

Despite the overall better classification assessment of Extended Architecture, results of micro and macro averaged metrics (Formula 3.28 and 3.29) are worse than Basic Architecture. On average, the results are around 10 p.p. worse. The difference between $Recall_{macro}$ (Formula 3.28) and $Precision_{macro}$ (Formula 3.29) is slightly bigger for the Extended Architecture. The low results of $Recall_{micro}$ coincides with an earlier observation that the model output may ambiguously point to a particular depth level. This phenomenon is explored in Chapter 7.2.

The Extended Architecture estimates a multidimensional output, which is significantly bigger than that of Basic Architecture. CU blocks. From the standpoint of the partitioning algorithm, the goal is to estimate the whole partitioning pattern. It was observed that the Extended Architecture is better in the classification of bigger. Thus, the decision algorithm can better compensate the inaccurately indicated depth levels for bigger CU blocks. Despite worse overall training assessment for the Extended Architecture, the results are considered satisfactory.

6.3 Results of evaluation in encoder

Firstly, the Extended Architecture (Section 5.1) was assessed (Section 3.1) by encoding images used for training (cropped DIV2k dataset, Section 4.2). The achieved results are presented in Table 6.12. The results of $BD\text{-}RATE$ for both Training and Validation subsets are increased by 1 p.p. compared to Basic Architecture (Section 5.1). Similarly, the $BD\text{-}PSNR$ is worse by 0.4 dB. Considering the difference in $Accuracy$ (Formula 4.8) was close to 10 p.p., the difference in evaluation results is relatively small. Recalling the $Accuracy$ results of 60% for Extended Architecture, the $BD\text{-}RATE$ and $BD\text{-}PSNR$ results are satisfactory. Simultaneously, using the Extended Architecture resulted in 17 p.p. better encoding time reduction (Time Savings: TS, Formula 3.3). The reasons for such results are slightly better classification results for lower depth level values and the decision over PU division. These aspects are further investigated and discussed in Subsection 7.3.1.

Table 6. 12. Evaluation of Extended Architecture used in Modified HM on training dataset images. Presented results are relative to HM.

	***BD-RATE* [%]**	***BD-PSNR* [dB]**	***TS* [%]**
Training Subset images	2.31	-0.109	69.30
Validation Subset images	2.73	-0.115	69.04

Results of the evaluation on test sequences (Section 3.5) are presented in Tables 6.13 and 6.14. The first presents the detailed results, and the second one presents the averaged results. In both tables, the results for Basic Architecture were presented for comparison. The presented results are relative to HM.

The $BD\text{-}RATE$ varies in the range $\langle 1.68; 5.87 \rangle$. Within the resolution class, the $BD\text{-}RATE$ results for sequences may vary up to ~3.5 p.p., as in Class C. The exception is Class E, where results for all sequences are above 5% $BD\text{-}RATE$. Despite similar resolutions, results for three sequences from Class B are ~2 p.p. worse than for DIV2k dataset images. The results for sequences share the same tendency in results as observed for Basic Architecture: the same sequences have higher or lower $BD\text{-}RATE$ and

BD-PSNR values. Thus, all observations connected with sequence resolution and content made for Basic Architecture (Section 5.2) are viable for Extended Architecture. The averaged results in classes are around two times bigger for both *BD-RATE* and *BD-PSNR*, compared to Basic Architecture. Yet, the results are much better than for random partitioning methods presented in Section 5.2.

Table 6. 13. Detailed results of evaluation on test sequences for Extended Architecture in Modified HM. Results for the Basic Architecture are presented for comparison. Presented results are relative to HM.

JCT-VC class	Sequence Name	Extended Architecture			Basic Architecture		
		BD-RATE **[%]**	***BD-PSNR*** **[dB]**	***TS*** **[%]**	***BD-RATE*** **[%]**	***BD-PSNR*** **[dB]**	***TS*** **[%]**
A	NebutaFestival	1.77	-0.128	76.25	1.40	-0.101	72.53
	PeopleOnStreet	4.47	-0.254	69.36	2.22	-0.127	50.99
	SteamLocomotiveTrain	2.88	-0.083	80.14	2.12	-0.061	75.03
	Traffic	3.80	-0.203	70.43	2.22	-0.120	53.72
B	BQTerrace	3.80	-0.213	71.40	1.47	-0.077	52.50
	BasketballDrive	4.89	-0.143	72.57	3.04	-0.089	62.76
	Cactus	4.13	-0.148	72.27	2.26	-0.082	59.94
	Kimono1	2.53	-0.088	76.47	1.93	-0.067	72.21
	ParkScene	2.92	-0.130	73.59	1.73	-0.077	62.06
C	BasketballDrill	5.54	-0.267	68.56	2.60	-0.127	48.53
	BQMall	3.72	-0.207	69.58	1.65	-0.092	52.15
	PartyScene	2.06	-0.149	62.64	0.49	-0.035	38.30
	RaceHorses	3.62	-0.209	72.09	1.56	-0.091	53.28
D	BasketballPass	3.58	-0.215	67.41	1.47	-0.089	49.30
	BlowingBubbles	2.39	-0.165	59.74	0.45	-0.030	44.18
	BQSquare	1.68	-0.134	61.07	0.64	-0.051	38.99
	RaceHorsesLow	3.23	-0.199	67.50	1.15	-0.072	48.00
E	FourPeople	5.09	-0.290	71.73	2.59	-0.149	56.34
	Johnny	5.87	-0.245	73.98	3.47	-0.145	65.38
	KristenAndSara	5.21	-0.266	73.28	2.71	-0.138	65.39

Table 6. 14. Averaged results of the evaluation of Extended Architecture in Modified HM. Results for the Basic Architecture are presented for comparison. The 'All' class corresponds to the average calculated over all test sequences. Presented results are relative to HM.

JCT-VC class	***BD-RATE*** **[%]**		***BD-PSNR*** **[dB]**		***TS*** **[%]**	
	Extended Architecture	**Basic Architecture**	**Extended Architecture**	**Basic Architecture**	**Extended Architecture**	**Basic Architecture**
A	3.23	1.99	-0.167	-0.102	74.05	63.07
B	3.66	2.09	-0.144	-0.079	73.26	61.90
C	3.74	1.57	-0.208	-0.086	68.22	48.07
D	2.72	0.93	-0.178	-0.061	63.93	45.12
E	5.39	2.93	-0.267	-0.144	73.00	56.57
All	**3.66**	**1.86**	**-0.187**	**-0.091**	**70.49**	**56.08**

The Extended Architecture is superior to Basic Architecture in terms of encoding time reduction (Time Savings: TS). The results are in the range $\langle 59.74; 80{,}14\rangle$, which means that even a sequence with the worst TS is better than the average result for Basic Architecture. Same as for Basic Architecture, the tendency of smaller TS results for smaller resolution is observed. The TS varies no more than 10 p.p. within the resolution classes, so the TS for Extended Architecture is much more consistent. The most significant improvements in TS, compared to Basic Architecture, were observed for Class A sequences "PeopleOnStreet" and "Traffic", and whole Classes C and D. Compared to Basic Architecture, the Extended Architecture improves the TS by 14.14 p.p. on average. The most significant improvements (~20 p.p.) are observed for classes D and C.

The observed TS improvements can be explained by the content of sequences in the mentioned classes. The content in class A is complex (see Section 5.3), and the content in classes C and D is more condensed due to resolution. Both cases can be characterized by more frequent usage of smaller CU blocks, which are especially computationally expensive. This means that the control over PU division is a significant part of the encoder complexity.

One should recall that the Extended Architecture is a modified version of the Basic Architecture. One of the applied modifications is the removal of the last MaxPool from Subnetwork $\mathbb{A}$. This results in processing feature maps of twice the dimensions by the Subnetwork $\mathbb{B}$, increasing the model complexity. To assess the change in complexity for all encoded sequences, the T_{model} (aggregated time of ANN processing, expressed in seconds) was measured and divided by the number of CTUs of that sequence. Similarly, as for Basic Architecture, the averaged ANN time processing for a single CTU again is very similar despite the sequence or *QP*. For a machine used in time assessment of Extended Architecture, the statistics of average ANN time processing for a single CTU are as follows:

- Mean value: 2.31 ms
- Standard deviation 126.90 μs

Simultaneously, mean time of CTU processing (over all sequences and *QP* values) was calculated: 29.99 ms for HM and 8.99 ms for Modified HM (Extended Approach). Surprisingly, the average ANN time processing for a single CTU is slightly smaller compared to the Basic Architecture (2.48 ms, Section 5.3). The rationale for this phenomenon is the ANN framework. Processing a single layer by the framework is more time-consuming than processing bigger feature maps in Subnetwork $\mathbb{B}$. The standard deviation is over twice as big, but the ANN time processing for a single CTU is still considered constant.

Same as in Section 5.3, the T_{ANN} (Formula 3.3) is used to estimate the impact of partitioning patterns on encoder complexity. Table 6.15 presents the statistics of T_{ANN} for a given set of sequences and *QP*.

Table 6. 15. Statistics of T_{ANN} for Extended Architecture for a given set of sequences and QP

JCT-VC class	***QP***	$\boldsymbol{\mu(T_{ANN})}$ **[%]**	$\boldsymbol{\delta(T_{ANN})}$ **[%]**
A	22	26.21	5.39
	27	30.13	5.74
	32	34.41	4.79
	37	37.58	4.02
B	22	22.96	4.95
	27	29.12	4.70
	32	34.24	3.89
	37	38.37	2.69
C	22	16.47	2.78
	27	21.18	3.68
	32	25.69	4.37
	37	32.01	4.91
D	22	15.93	3.09
	27	19.10	3.82
	32	23.34	4.44
	37	29.17	4.65
E	22	29.02	2.49
	27	32.93	2.58
	32	36.72	2.10
	37	39.97	1.71
All	**22**	**21.82**	**6.40**
	27	**26.30**	**6.77**
	32	**30.76**	**6.63**
	37	**35.34**	**5.56**

Similarly, as for Basic Architecture, the bigger the QP, the higher the impact of the ANN processing on the encoding time. Other observations for Basic Architecture, described in Section 5.3, apply here, e.g., the lower the resolution, the smaller blocks are used more frequently. Apart from it, the T_{ANN} is bigger Compared to Basic Architecture, by ~6 p.p. on average. As the averaged ANN processing time of a single CTU is almost the same and the average TS was 14.5 p.p. better, the reduction of the encoding time is caused mainly by the decision of the PU division. Additionally, the improvements in the classification of smaller depth level values, observed with Confusion Matrices metrics, may also impact the TS, as bigger CU blocks are predicted more accurately. The effect of block size on encoding time will be analyzed in Subsection 7.3.1.

Lastly, the RD curves for sequences encoded with Extended Architecture were assessed. Again, only curves for the best and the worst $BD\text{-}RATE$ will be presented. Figure 6.3 presents the curve for "BQSquare" sequence (the best $BD\text{-}RATE$: 1.68%), and Figure 6.4 presents the curve for "Johnny" sequence (the worst $BD\text{-}RATE$: 5.87%).

The curves for sequence "BQSquare" almost coincide. The $BD\text{-}RATE$ of 1.68% is similar to the average results for Basic Architecture. For the "Johnny" sequence, the curves are visually distanced. Despite the visual impression, the smaller the QP, the bigger the distance between the curves is. The same effect was observed for the Basic Architecture. It should be mentioned that the $BD\text{-}RATE$ of more than 5% was observed for only four sequences.

Compared to Basic Architecture, the Extended Architecture performs worse in terms of $BD\text{-}RATE$, but much better in terms of encoding time (TS). Therefore, this approach is a viable solution as it offers a different ETvsCE tradeoff. Despite the slightly better classification for smaller depth level values (bigger blocks), the overall smaller $Accuracy$ resulted in worse assessment in the encoder. The $Accuracy$ smaller by 10 p.p. resulted in an increase of ~1.8 p.p. in $BD\text{-}RATE$ on average. The reasoning for this effect is associated with the impact of blocks of a specific size on the encoding efficiency. This subject will be discussed widely in section 7.3.1.

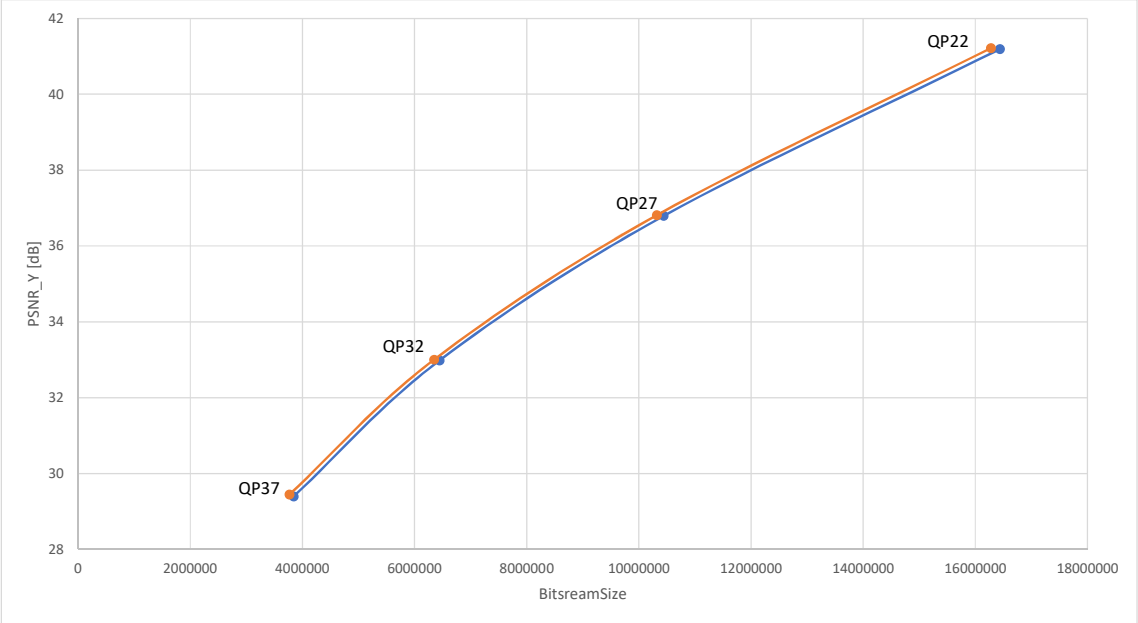

Figure 6. 3. RD curves for HM (orange) and Modified HM with Extended Architecture (Blue) for Test sequence "BQSquare"(Class C).

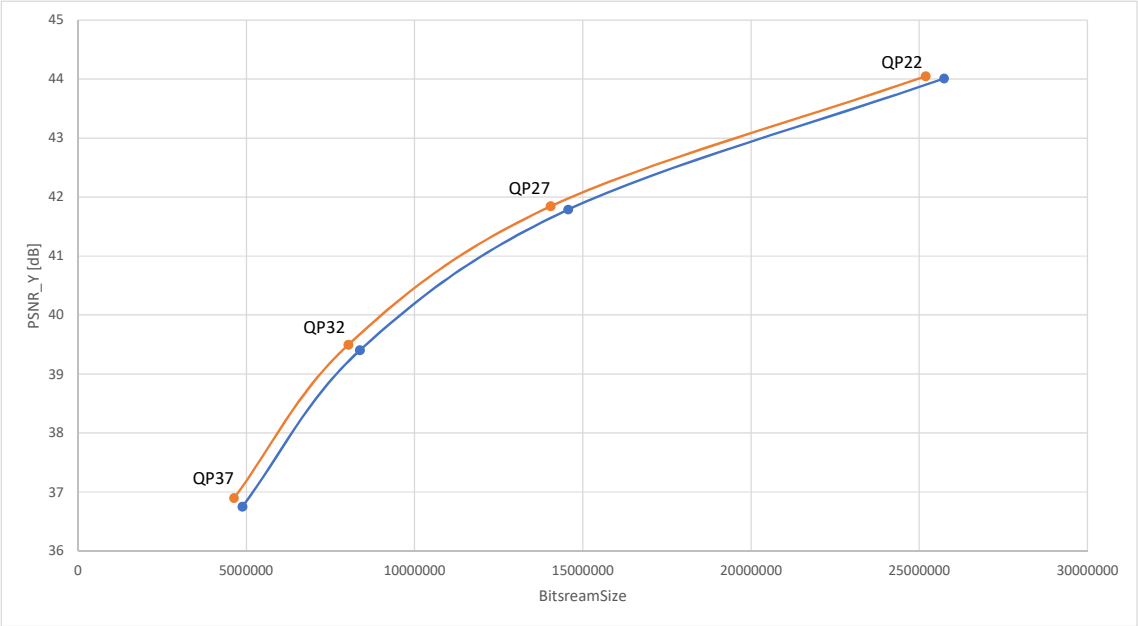

Figure 6. 4. RD curves for HM (orange) and Modified HM with Extended Architecture (Blue) for Test sequence "Johnny"(Class E).

6.4 Extended Architecture tuning

The ANN architecture for the Extended Approach (Section 6.1) was derived from the ANN architecture for the Basic Approach (Figure 5.1, Section 5.1) with a minor set of applied changes described in Section 6.1. Nevertheless, the ANN architecture development process was performed similarly to the Basic Architecture (Section 5.1). Thus, this process was time-consuming, resource-costly and consisted of verifying ideas in a try-and-check procedure. In this section, only the most interesting ones are presented. Multiple ideas presented in Section 5.4 were also tested in Extended Architecture. The examples are adjustment of layer types, label smoothing, label weighting, or modifications of Subnetwork $\mathbb{A}$ complexity (number of layers). The results were similar to those for ANN architecture for Basic Approach (Section 5.1), so the conclusions for ANN architecture for Extended Approach were the same.

One should recall that a greedy experimenting approach was applied, described in Section 3.7. All model were trained according to the description in Section 4.6. All tuning approaches presented in this section were discarded due to rules defined in Subsection 4.6.3 (converge of the training process and increase of evaluation result by 0.1 p.p. in terms of $BD\text{-}RATE$). The results for tunings of ANN architecture for Extended Approach were obtained with a single run of model training. The precise time assessment was not performed on tunings, so the complexity of the model is assessed using the Multiply and Accumulate (MAC) operations count (in millions: M) and the number of weights in the model. The evaluation of the models was performed according to description in Section 3.1.

The tunings of ANN architecture for Extended Approach presented in this section are grouped into three types: MaxPool removal location, the extension of the Subnetwork $\mathbb{B}$ to reflect PU division, and an increase in the number of filters in Conv2D layers. The evaluation results for chosen tunings are presented in Table 6.16. Presented results are relative to HM.

The first described tuning is the location of MaxPool layer removal. During the development of the architecture, all possible locations were considered. As shown in Table 6.16, the earlier the MaxPool is removed, the better the model evaluation results. The improvement of $BD\text{-}RATE$ was 0.11 p.p. for the removal of the first MaxPool, but the MAC operation count increased almost three times. Removing

further MaxPool layers resulted in minor improvements, with a significant complexity increase. This proves that removing the last MaxPool layer was the best option.

Table 6. 16. Assessment of selected tunings for architecture in Extended Approach. "No modification" refers to results for Extended Architecture (Section 5.3). Presented results are relative to HM.

Modification Type	Description modification for ANN model for Extended Approach	*BD-RATE* [%]	*BD-PSNR* [dB]	MAC count [M]	Num. of weights
MaxPool removal (vs Basic Architecture)	After the first Subnetwork 𝔸 Conv Block	3.56	-0.177	25.57	91 617
	After the second Subnetwork 𝔸 Conv Block	3.62	-0.186	17.53	91 617
	After the third Subnetwork 𝔸 Conv Block	3.68	-0.188	11.53	91 617
Deeper Subnetwork B	The additional level of quaternary tree mimic by Subnetwork 𝔹, Number of filters: 64,32,16,5	3.61	-0.190	9.43	203 489
	The additional level of quaternary tree mimic by Subnetwork 𝔹, Number of filters: 64,16,8,5	3.63	-0.189	8.61	110 521
More filters in convolution layers	More filters in Subnetwork 𝔸: 24,48,72,96	3.55	-0.179	27.76	197 613
	More filters in Subnetwork 𝔹: 128,32,4	3.63	-0.183	12.09	230 449
None	**No modification (Extended Architecture)**	**3.66**	**-0.187**	**8.54**	**91 617**

As described in Section 4.1, the Subnetwork 𝔹 architecture mimics the quaternary tree. The ANN architecture for Extended Approach inherits the Subnetwork 𝔹 from the ANN architecture for Basic Approach (Figure 5.1, Section 5.1) with minor modifications. As the Extended Approach (Section 4.1) considers one level deeper quaternary tree, the Subnetwork 𝔹 shall be adjusted accordingly. Thus, the Subnetwork 𝔹 was extended with additional layers, as shown in Figure 6.5. This tuning was thoroughly tested to find the best hyperparameters. Unfortunately, the modified architecture did not perform well in most cases, as the evaluation results were much worse than those from Extended Architecture (Section 5.1 and 5.3). In Table 6.16, results for the two best hyperparameter sets for this modification are presented. The presented evaluation results are slightly better than those of the Extended Architecture. The improvements in $BD\text{-}RATE$ and BD-PSNR were too minor to use in further research. The number of operations did not increase significantly, but, as observed in Section 6.3, additional layers increase the ANN processing time. Because of adding Conv Blocks (Figure 5.2, Section 5.1), the number of weights increased up to twofold. All additional weights are in Subnetwork 𝔹, and along with specific architecture, the gradient in backpropagation is too spread out. This influences the training of the Subnetwork 𝔸 and causes such evaluation results.

Lastly, results for two tuning experiments are presented where the number of filters is increased. For Subnetwork 𝔸 or 𝔹 the number of filters in each Conv Block was doubled. One should observe that the tuning for Subnetwork 𝔹 results in a minor decrease of $BD\text{-}RATE$, while the number of weights increased around 2.5 times and MAC operation count 1.5 times. The improvement of $BD\text{-}RATE$ for Subnetwork 𝔸 tuning is 0.11 p.p. but at the cost of a significant rise in complexity. The MAC operation count is increased 3.25 times, and the number of weights is doubled. These examples show the overall tendency: the relatively huge increase in model complexity slightly improves evaluation results. As the main advantage of the Extended Approach is the reduction in TS, this research direction was abandoned.

Concluding the results for hyperparameter tuning for both ANN models for Basic and Extended Approaches, any viable one was found despite the extensive search for better architectures. The improvement in $BD\text{-}RATE$ would require an increase in the complexity of the model, which would potentially not change or make worse the ETvsCE trade-off. One should recall that the ANN is just one part of the proposed partitioning algorithms. The second one, the decision algorithm, can significantly

increase the effectiveness of the partitioning algorithm. Please note that in the evaluation experiments up to this point of the dissertation, the hard-decisive variant of AlgIdx (Subsection 7.2.1) was used.

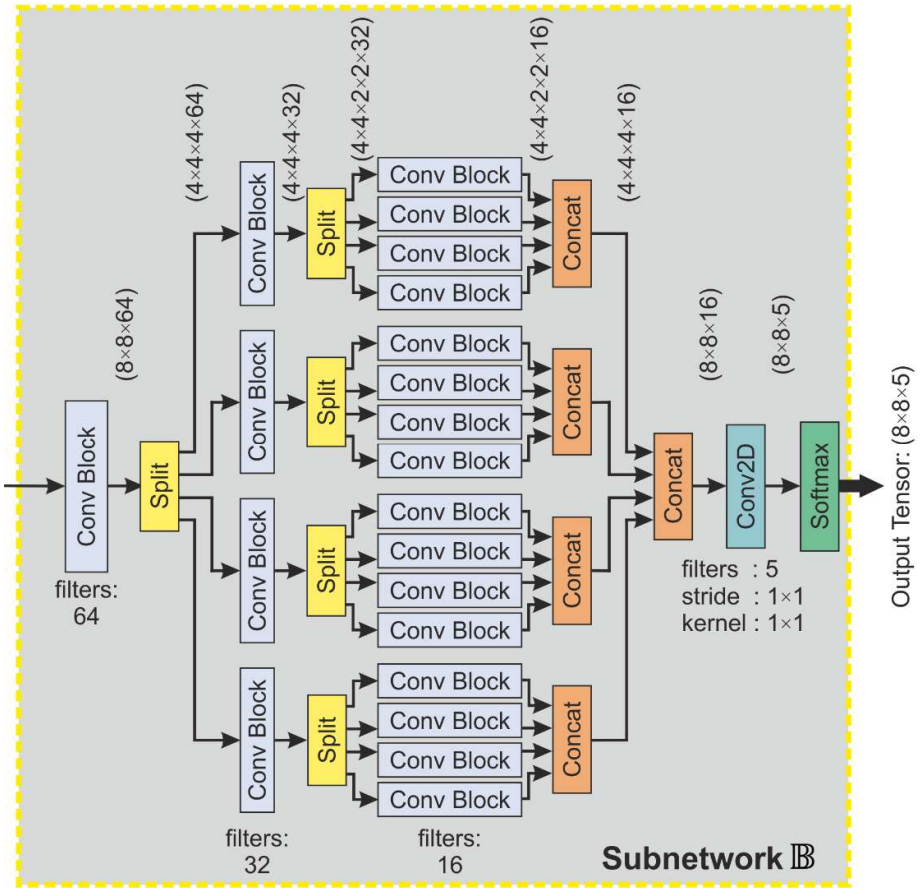

Figure 6. 5. Architecture of the Subnetwork 𝔹 with mimicking of additional depth level.

7 Decision algorithm for the ANN output

7.1 General description of the proposed decision algorithms

This chapter provides a description of the decision algorithms, which are among the most distinctive developments of this dissertation. Each of the proposed decision algorithms is processing the output of the ANN (the Division Tensor - DT), to produce a partitioning pattern (the Division Matrix - DM), conformant with the HEVC syntax. It should be noted that during the research, many experimental algorithms were developed and tested; however, only the final algorithms — the best-performing ones — are presented in the dissertation. For the sake of clarity, lets recall the requirement for the partitioning algorithm, described in Section 4.1: the whole partitioning algorithm must be as low complex as possible. Thus, proposed decision algorithms are so simple that their computing time is negligible compared to ANN processing time.

It is important to note that the output of the ANN may happen to be non-conformant with the HEVC syntax. To prevent such non-conformance, the proposed decision algorithms are designed to process the DT in a quaternary tree manner, and, as a result, to produce only HEVC-syntax-conformant decisions (Section 2.2). The overall scheme of all of the proposed decision algorithms is presented below.

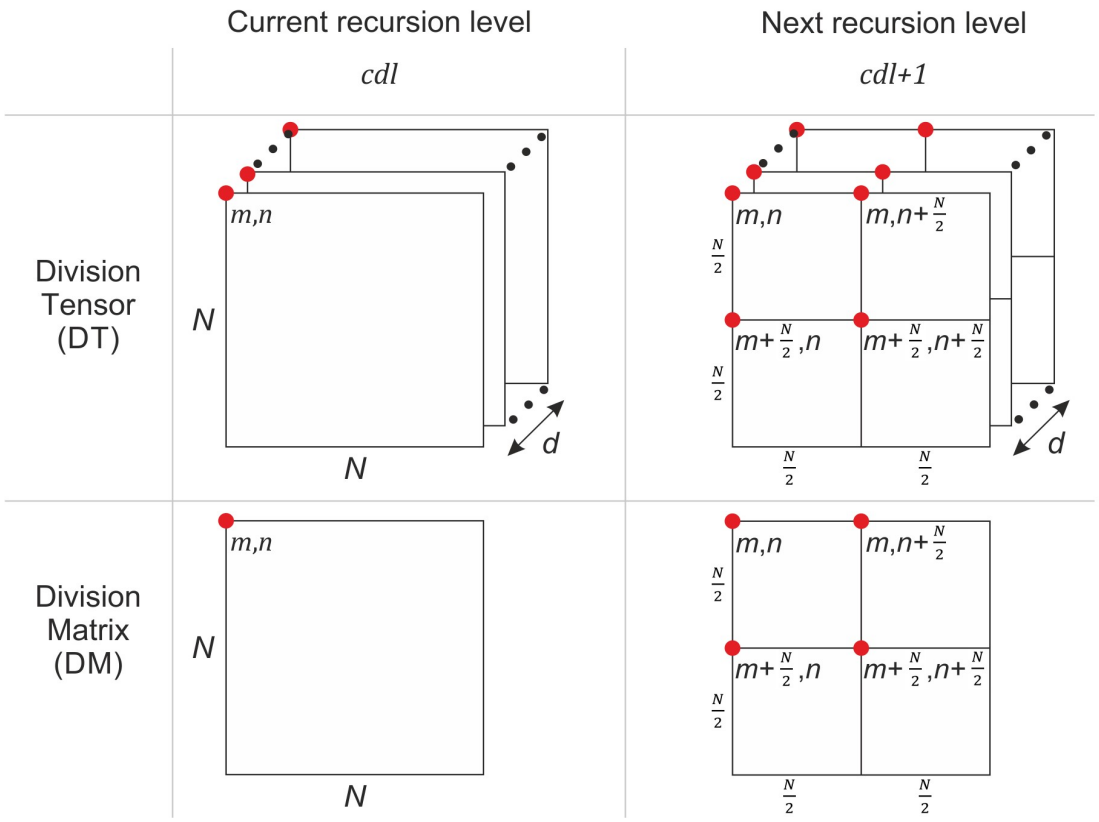

Figure 7. 1. Visualization of current block area $A_{m,n,N}$ (Formula 7.1) in Division Tensor/Matrix in two consecutive recursion levels.

The processing is done **recursively**, in a top-bottom fashion. It starts at the top level of the entire CTU and proceeds downward through progressively smaller CU blocks until reaching the smallest blocks (CU or PU) at the bottom. The recursion (Figure 7.1) is controlled by the following arguments:

- Current depth level, cdl, which starts from $cdl = 0$, and increments with each recursion level.

- Current block size, $N \times N$, expressed in Division Matrix/Tensor (DT/DM) indices and is halved with each recursion level. The initial value is equal to DT/DM size, which is $N = 4$ for Basic Approach and $N = 8$ for Extended Approach (Section 4.5).
- Starting indices of the current block, m, n, inside of Division Matrix/Tensor.

The recursive procedure is defined as follows:

Procedure**: DecisionAlgorithm(cdl, N, m, n):**

1. Denote the area $\mathrm{A}_{m,n,N}$ of the current block inside of $DT[i,j,\cdot]$ and $DM[i,j]$ as specified by the indices m, n and the size N:

$$\mathrm{A}_{m,n,N} = \{(i,j) \mid i \in [m; m+N-1],\ j \in [n; n+N-1]\}. \tag{7.1}$$

2. Analyze the DT values in the denoted area $\mathrm{A}_{m,n,N}$, e.g. $DT[i,j,\cdot]$, $(i,j) \in \mathrm{A}_{m,n,N}$.
3. Make decision according to the analysis of the considered DT area $\mathrm{A}_{m,n,N}$:
   - If certain conditions (specified for particular algorithm in Section 7.2 and 7.3) are met, fill the values in DM corresponding to the area of the current block with cdl :

$$DM[i,j] \leftarrow cdl, \qquad for\ (i,j) \in \mathrm{A}_{m,n,N}, \tag{7.2}$$

     where ← symbol stands for assignment operation. Terminate the recursion.
   - Otherwise: Recurse the procedure, for next depth level $cdl + 1$. Split the block following the quaternary tree (Figure 7.1) and perform recursive call for each subdivided block:
     - call **DecisionAlgorithm($cdl + 1$, $\frac{N}{2}$, m, n)**
     - call **DecisionAlgorithm($cdl + 1$, $\frac{N}{2}$, m, $n + \frac{N}{2}$)**
     - call **DecisionAlgorithm($cdl + 1$, $\frac{N}{2}$, $m + \frac{N}{2}$, n)**
     - call **DecisionAlgorithm($cdl + 1$, $\frac{N}{2}$, $m + \frac{N}{2}$, n+ $\frac{N}{2}$)**

(7.3)

The above procedure serves as a foundation for all decision algorithms presented in this chapter. In subsections corresponding to particular algorithms, the description includes the idea behind an algorithm, analysis, and processing of the DT, a definition of the factor used for decision, and conditions for application depth level or further analysis.

In this dissertation, two approaches to the decision algorithm are presented. The **hard-decisive approach** (Section 7.2) implies that the decision algorithm always outputs a single partitioning pattern. The partitioning pattern is used by the encoder without any changes, and the rest of the decisions are estimated by the RD algorithm only for indicated blocks. This applies identically to both Basic and Extended approaches (Section 4.1). In the **soft-decisive approach** (Section 7.3), a single or set of partitioning patterns is indicated. Therefore, in the case of a set of partitioning patterns, the RD algorithm is comparing them to choose the best-performing one.

7.2 Hard-decisive approach for ANN output interpretation

7.2.1 Index-based Algorithm (AlgIdx)

The name of the first proposed partitioning algorithm, the **Index-based algorithm** (**AlgIdx**), corresponds to indices of depth levels calculated with the ArgMax function. The algorithm follows the recursive procedure of the decision algorithm defined in Section 7.1, starting from current depth level $cdl = 0$, $N \times N$ equal the size of Division Tensor/Matrix, $m = 0$, $n = 0$. The following steps of the procedure are defined:

Procedure: $\mathbf{AlgIdx_{hard\text{-}decisive}}(cdl, N, m, n)$:

1. Denote the area $A_{m,n,N}$ of the current block inside of $DT[i,j,\cdot]$ and $DM[i,j]$ as specified by the indices m, n and the size N, as in Formula 7.1.

$$A_{m,n,N} = \{(i,j) \mid i \in [m; m+N-1],\ j \in [n; n+N-1]\}. \tag{7.1}$$

2. Analyze the DT values in the denoted area $A_{m,n,N}$, e.g. $DT[i,j,\cdot]$, $(i,j) \in A_{m,n,N}$.
   - If N is equal 1, set $DM[m,n] \leftarrow cdl$ and terminate the algorithm.
   - For the denoted area $A_{m,n,N}$, find the index of the most probable depth level $DM_{AlgIdx}[i,j]$ defined as follows:

$$DM_{AlgIdx}[i,j] = \mathrm{ArgMax_d}(DT[i,j,\cdot]), for\ (i,j) \in A_{m,n,N}. \tag{7.4}$$

     where $\mathrm{ArgMax_d}(F)$ is an argument for the maximal value of tensor F, along the d axis (channel).
   - Calculate the C value defined as follows:

$$C = \frac{1}{N^2} \cdot \sum_{(i,j) \in A_{m,n,N}} \mathrm{Iv}\left(DM_{AlgIdx}[i,j] = cdl\right), \tag{7.5}$$

     where $\mathrm{Iv}(\cdot)$ is the Iverson function [Fo99], such that $\mathrm{Iv}(true) = 1$ and $\mathrm{Iv}(false) = 0$.
3. Make decision according to the analysis of the considered DT area $A_{m,n,N}$:
   - If $C > 0.5$, fill the values in DM corresponding to the area of the current block with cdl (Formula 7.2), and terminate the recursion.
   - Otherwise: Recurse the procedure, for next depth level $cdl + 1$. Split the block following the quaternary tree (Figure 7.1) and perform recursive call for each subdivided block as in Formula 7.3 using $\mathbf{AlgIdx_{hard\text{-}decisive}}$ procedure.

A rationale for procedure steps is as follows. A straightforward method to make decisions on partitioning pattern, using values in the DT, is to count the number of outputs indicating the current depth level cdl in the analyzed area $A_{m,n,N}$ (Formula 7.1) for current block. As the ANN model estimates the probabilities of the depth levels for certain subareas in the CTU ($DT[i,j,\cdot]$, Section 4.5), the highest value should indicate the best one. Thus, the ArgMax function is used to find indices of the most probable depth levels. The result is the DM_{AlgIdx} (Formula 7.4): Division Matrix with partitioning pattern for the analyzed area ($A_{m,n,N}$) which may not be conformant with the HEVC. If the count of indices of the cdl is bigger than half the number of indices in the analyzed area (N^2), then the current depth level value is applied. Such a threshold was chosen to ensure that at least half the CTU areas ($DT[i,j,\cdot]$) in the analyzed area ($A_{m,n,N}$) indicate the cdl. Otherwise, the procedure is recursed considering smaller blocks ($cdl + 1$).

AlgIdx in hard-decisive variant was used as a decision algorithm during the development of Basic and Extended Architectures (Section 5.1 and 6.1). One of the reasons is the similarity of AlgIdx to the $Accuracy$ metric (Formula 4.8), which was a main metric used for model training assessment. Secondly, the simplicity of AlgIdx makes it the good reference point for further research of decision algorithms. Additionally, a similar approach was used by authors of [Fe21], where the ANN estimates indices, and a similar algorithm is used for conformance correction. This makes AlgIdx a well representation of the solutions found in the literature.

AlgIdx does not make full use of information in a DT outputted from the ANN:

- As mentioned in Section 5.2, the probabilities for a given CTU subarea ($DT[i,j,\cdot]$) outputted by ANN cannot always be explicitly translated to just one depth level value.
- In some cases, the ANN is outputting a very high probability value for a single depth level, which is translated as a very confident decision.
- In other cases, the result for a certain CTU subarea ($DT[i,j,\cdot]$) is ambiguous – the model is estimating similar, relatively high, probability values for at least two depth levels.
- The model is trained to estimate probabilities of depth level for each CTU subarea ($DT[\cdot,\cdot,\cdot]$) separately, so within a single DT, both mentioned cases may occur simultaneously.

Therefore, the use of the ArgMax removes crucial information about the level of model certainty. The discarded non-zero probability values can still be used in the decision algorithm, as described in the next section.

7.2.2 Probability-based algorithm (AlgPrb)

The **Probability-based algorithm** (**AlgPrb**) is aiming to make use of all probabilities estimated by ANN. The algorithm follows the recursive procedure of the decision algorithm defined in Section 7.1, starting from current depth level $cdl = 0$, $N \times N$ equal the size of DT, $m = 0$, $n = 0$. The algorithm defines the steps of the procedure as follows:

Procedure**: $\mathbf{AlgPrb}_{\mathbf{hard\text{-}decisive}}(cdl, N, m, n)$:**

1. Denote the area $\mathrm{A}_{m,n,N}$ of the current block inside of $DT[i,j,\cdot]$ and $DM[i,j]$ as specified by the indices m, n and the size N, as in Formula 7.1.

$$\mathrm{A}_{m,n,N} = \{(i,j) \mid i \in [m; m+N-1],\ j \in [n; n+N-1]\}. \tag{7.1}$$

2. Analyze the DT values in the denoted area $\mathrm{A}_{m,n,N}$, e.g. $DT[i,j,\cdot]$, $(i,j) \in \mathrm{A}_{m,n,N}$.
   - If N is equal 1, set $DM[m,n] \leftarrow cdl$ and terminate the algorithm.
   - For each depth level $L \geq cdl$, calculate S_L:

$$S_L = \sum_{(i,j)\in \mathrm{A}_{m,n,N}} DT[i,\mathrm{j},L]. \tag{7.6}$$

3. Make decision according to the analysis of the considered DT area $\mathrm{A}_{m,n,N}$:
   - If $S_{cdl} = \max(S_L)$, fill the values in DM corresponding to the area of the current block with cdl (Formula 7.2), and terminate the recursion.
   - Otherwise: Recurse the procedure, for next depth level $cdl + 1$. Split the block following the quaternary tree (Figure 7.1) and perform recursive call for each subdivided block as in Formula 7.3 using $\mathbf{AlgPrb}_{\mathbf{hard\text{-}decisive}}$ procedure.

The rationale for the steps of the above-mentioned algorithm steps is as follows: instead of counting indices in the analyzed area $\mathrm{A_{m,n,N}}$ (Formula 7.1) for current block, the probabilities are summed up ($L = cdl$, Formula 7.6). Moreover, the probabilities corresponding to all bigger depth level ($L > cdl$) values are also added up separately (Formula 7.6). If the sum for the current depth level cdl is the highest of all sums, then this depth level is applied in DM for the corresponding in the analyzed area $\mathrm{A_{m,n,N}}$. Otherwise, the procedure is recursed considering smaller blocks ($cdl + 1$).

7.2.3 Evaluation of the proposed hard-decisive algorithms

Table 6.1 presents the results of proposed partitioning algorithms evaluation with two presented hard-decisive algorithms: AlgIdx (Subsection 7.2.1) and AlgPrb. (Subsection 7.2.2) The evaluation was performed following the methodology presented in Chapter 3, with images from DIV2k [Ag17] used

for creating training datasets (Section 4.2) and test sequences (Section 3.5). Both Basic and Extended Architectures (Section 5.1 and 6.1) were tested, with proposed hard-decisive algorithms. The presented results are relative to HM.

Table 7. 1. Evaluation results of proposed partitioning algorithms with hard-decisive algorithms in Modified HM. Presented results are relative to HM.

		Training Subset Images		Validation Subset Images		Test sequences	
Arhitecture	**Decision Algorithm**	***BD-RATE* [%]**	***BD-PSNR* [dB]**	***BD-RATE* [%]**	***BD-PSNR* [dB]**	***BD-RATE* [%]**	***BD-PSNR* [dB]**
Basic	AlgIdx	1.43	-0.069	1.74	-0.075	1.86	-0.091
	AlgPrb	1.34	-0.066	1.78	-0.075	1.80	-0.089
Extended	AlgIdx	2.31	-0.109	2.73	-0.112	3.66	-0.189
	AlgPrb	2.18	-0.105	2.74	-0.113	3.44	-0.177

For the Basic Architecture, in most cases, AlgPrb is marginally better than AlgIdx in terms of compression efficiency. The differences do not exceed 0.1 p.p. $BD\text{-}RATE$, which was the value defined in Section 4.6 as the threshold for model mattering improvement. In this case, the improvement in the evaluation result comes solely from decision algorithms, so even slightly better value is a valuable improvement., For test sequences (which are the most important result) AlgPrb outperforms AlgIdx by 0.06 p.p. in terms of $BD\text{-}RATE$ and by 0.02 dB in terms of $BD\text{-}PSNR$.

In the presented evaluation, the Basic Architecture showed very similar results both for images used for creating training datasets and test sequences. This observation applies to both decision algorithms. This leads to the conclusion that the Basic Architecture is trained to generalize the problem quite well, and the model is quite certain of the outputted partitioning pattern.

In the case of Extended Architecture, the difference is much more significant. Considering images from DIV22k (used for creating training datasets), the results for both decision algorithms are very similar. For Validation Subset Images, AlgPrb performed negligibly worse (0.01 p.p. $BD\text{-}RATE$) than AlgIdx. The opposite is observed for test sequences. The differences are 0.22 p.p. in terms of $BD\text{-}RATE$ and 0.12 dB in terms of $BD\text{-}PSNR$ in favor of AlgPrb. This is twice the improvement threshold defined in Section 4.6 achieved just with the decision algorithm. The difference in decision algorithm performance, along with lower training $Accuracy$ of Extended Architecture, suggests that the models are less certain about partitioning. Thus, considering probability values estimated for multiple depth levels helps in deciding on better-performing partitioning patterns.

7.3 Soft-decisive approach for ANN output interpretation

In the previous section, hard-decisive algorithms were presented, where only single partitioning patterns is derived. However, as suggested earlier in this dissertation (Section 4.1), indicating a set of partitioning patterns to check may be a viable solution. Thus, more than one partitioning pattern is implied by the decision algorithm, according to the output of the ANN. Such an approach is referred to as **the soft-decisive approach**. Its core idea is to indicate a pair of block sizes depending on the certainty of ANN predictions. While this concept can be extended to consider three or four block sizes, such an expansion is beyond the scope of this dissertation, as the primary objective remains the reduction of encoding complexity (Section 4.1).

7.3.1 Viability of the soft-decisive approach

The experiment presented in this subsection has two main goals. The first one is to check the viability of the soft-decisive approach, which is considered in this section. The second one is to analyze the effect of CU/PU block size on encoding complexity and verify observations related to this from sections 5.3 and 6.3.

Table 7. 2. Evaluation of Modified HM with a limited number of possible CU and PU block sizes.. Presented results are relative to HM.

One indicated block size				Two considered block sizes			
Block size	***BD-RATE* [%]**	***BD-PSNR* [dB]**	***TS* [%]**	**Block sizes**	***BD-RATE* [%]**	***BD-PSNR* [dB]**	***TS* [%]**
CU: 64×64 PU: 64×64	15.82	-0.790	85.01	CU: 64×64 PU: 64×64 or CU: 32×32 PU: 32×32	12.97	-0.665	74.29
CU: 32×32 PU: 32×32	14.06	-0.717	86.09	CU: 32×32 PU: 32×32 or CU: 16×16 PU: 16×16	8.25	-0.442	73.75
CU: 16×16 PU: 16×16	11.01	-0.569	86.12	CU: 16×16 PU: 16×16 or CU: 8×8 PU: 8×8	5.20	-0.271	66.49
CU: 8×8 PU: 8×8	13.26	-0.627	78.89	CU: 8×8 PU: 8×8 or CU: 8×8 PU: 4×4	9.38	-0.425	36.20
CU: 8×8 PU: 4×4	36.14	-1.543	55.71				

The experiment consists of encoding test sequences (Section 3.5) consistently using the same block size or allowing the RD Optimization in HM to choose one block size from a pair of adjacent block sizes in a quadtree. The modified HM was used to perform the experiment, employing the partitioning algorithm, which allows the sequence to be encoded using a DM filled with the same value (Subsection 4.7.4). Firstly, values for single block size were used:

- 0 – use only block size: CU: 64×64 PU: 64×64.
- 1 – use only block size: CU: 32×32 PU: 32×32.
- 2 – use only block size: CU: 16×16 PU: 16×16.
- 3 – use only block size: CU: 8×8 PU: 8×8.
- 4 – use only block size: CU: 8×8 PU: 4×4.

Then, the Modified HM was adapted to restrict the RD Optimization in HM to choose one block size from a pair of adjacent block sizes in a quadtree. The best-performing block size is selected according to the calculated bit cost. The following values for DM were defined and used to indicate specific pairs of blocks:

- 5 – consider only block sizes: CU: 64×64 PU: 64×64 and CU: 32×32 PU: 32×32.
- 6 – consider only block sizes: CU: 32×32 PU: 32×32 and CU: 16×16 PU: 16×16.
- 7 – consider only block sizes: CU: 16×16 PU: 16×16 and CU: 8×8 PU: 8×8.
- 8 – consider only block sizes: CU: 8×8 PU: 8×8 and CU: 8×8 PU: 4×4 (effectively the same as depth 3 in Basic Approach).

The encoding results were assessed following the methodology presented in Chapter 3. The results of this experiment are presented in Table 7.2. Presented results are relative to HM.

7.3.1.1 Effect of single indicated block size on encoding complexity

Considering the use of single block size, for CU blocks 64×64 to 16×16, a tendency is observed: the bigger the block, the smaller $BD\text{-}RATE$ increase. This is not the case for CU block 8×8. When PU 8×8 is used, the $BD\text{-}RATE$ (13.26%) is comparable to CU 32×32 (14.06%), but for PU 4×4 the $BD\text{-}RATE$ increases greatly to 36.14%. This confirms the observation from Sections 5.2 and 6.3, made for the impact of block size on encoding efficiency. The hasty use of small block sizes has a much bigger negative impact on encoding efficiency than the use of bigger blocks. Such a phenomenon was observed on RD curves in Sections 5.2 and 6.2. The distance between corresponding points on point RD curves was bigger for smaller QP values. This concedes with the above observation, as statistics of depth values, presented in Section 4.2, showed a bigger share of small block sizes for smaller QP values.

The worse performance of Extended Architecture (Section 6.1) in terms of $BD\text{-}RATE$, compared to Basic Architecture (Section 5.1), was observed in Section 6.3. In the Extended Approach (Section 4.1), the model additionally decides on the PU division. As shown in Table 7.2, the over-extensive use of CU: 8×8 and PU: 4×4 (depth level value 4) significantly increases the $BD\text{-}RATE$. This means that mistaken indications of depth level 4 negatively influence the evaluation results. Furthermore, an observed tendency of mistaken prediction (in favor of small blocks for smaller QP values and in favor of big blocks for bigger QP values) impacts the result. Still, the average $BD\text{-}RATE$ for Extended Architecture was 3.66% (Section 6.3), which means that despite low $Accuracy$ (Section 6.2), models still estimate partition patterns quite well.

For results of encoding time reduction (Time Savings: TS, Formula 3.3) it was observed that in terms of encoding complexity, choosing any of CU block sizes: 64×64, 32×32, and 16×16 does not matter. For all these blocks, the TS is ~86%, which can be treated as the theoretical limit of the TS, without significant modifications of the HM. It can be observed that mistakenly choosing one of these block sizes would have an impact only on the $BD\text{-}RATE$. For the CU block 8×8, when the PU 8×8 is used, the TS of 78.89% is still close to bigger blocks, but when only the PU 4×4 is used, the TS drops to 55.71%.

Recalling observations from Section 6.3, Extended Architecture performed 17 p.p. better than Basic Architecture in terms of TS. A better classification was observed for bigger blocks (64×64, 32×32, and 16×16), which turned out to be similarly efficient in terms of complexity. Additionally, the tendency for mistaken predictions observed for the Extended Architecture (in favor of small blocks for smaller QP values and in favor of big blocks for bigger QP values) will impact TS negatively for smaller QP values and positively for bigger QP values.

7.3.1.2 Effect of considering two block sizes on encoding complexity

When the encoder is restricted to consider a pair of block sizes, it is observed that the $BD\text{-}RATE$ is reduced, as expected. That is compared to both cases of using only a single block size from the available block sizes. The smaller the possible block sizes are, the better the $BD\text{-}RATE$, except for the last pair. The result for the last pair: CU: 8×8; PU: 8×8 and CU: 8×8; PU: 4×4 is ~4 p.p. worse than for pair: CU: 16×16; PU: 16×16 and CU: 8×8; PU: 8×8. Still, the result of 9.38 % is much better than that for the use of a single block, especially recalling the result for CU: 8×8; PU: 4×4 (36.14).

The encoding time reduction (Time Savings: TS, Formula 3.3) measured for cases when the encoder considers only two block sizes has decreased, but not drastically in most cases. If only blocks: CU: 64×64; PU: 64×64, CU: 32×32; PU: 32×32, CU: 16×16; PU: 16×16 are considered, the decrease in TS not bigger than ~10 p.p.. The pair CU: 16×16; PU: 16×16 and CU: 8×8; PU: 8×8 results in 66.49% of TS, which is still quite a good result but a little higher value was expected. This means that when

smaller blocks are considered, making other decisions in the encoder is more complex. Complexity increase is even more noticeable for pair PU: 8×8 and CU: 8×8; PU: 4×4, where the TS is just 36.20%. As this reflects the smallest possible depth in the Basic Approach (Section 4.5), this explains worse TS results compared to the Extended Approach.

7.3.1.3 Conclusions

Concluding the above observations, considering a pair of block sizes instead of a single one can significantly improve the $BD\text{-}RATE$. The impact on the encoding complexity is not that high for bigger blocks. A much bigger increase in encoder complexity is observed for smaller block sizes, but it is still viable to consider two block sizes. That is because bad decisions on too small a block may have a huge impact on $BD\text{-}RATE$. All this means that the indication of a decision algorithm to check a pair of block sizes is a viable option. To reduce the complexity of the encoder, this should be done only when the ANN is not certain of the block size. Other times only one block size should be indicated. This should result in improvements of $BD\text{-}RATE$, while the encoding time reduction (Time Savings: TS, Formula 3.3) should decrease slightly or even remain the same. Such a mechanism can be applied in both proposed hard-decisive algorithms.

7.3.2 Proposed soft-decisive variants of the algorithms

7.3.2.1 Index-based decision algorithm

The previously described hard-decisive variant of AlgIdx algorithm (7.2.1) makes hard binary decision (to split or not to split) based on comparison of C value (Formula 7.5) with threshold of 0.5. The C value in some cases may be close to the threshold: slightly above or below. Such a situation may be interpreted as an uncertain decision, which indicates that more than one block size should be checked. To address the mentioned uncertainty problem, a **soft-decisive variant of the Index-based algorithm** (AlgIdx) is proposed. It follows the same recursive procedure of decision algorithm (defined in section 7.1) as hard-decisive variant, but includes additional case of uncertain decision. The range of C value, when a decision is considered uncertain, is controlled with a single parameter $\alpha \in \langle 0; 0.5)$. Therefore, when $C \in (0.5 - \alpha\,; 0.5 + \alpha)$ the decision is considered uncertain. The higher the α, the wider range of C, so the decision algorithm will indicate checking a pair of blocks more frequently.

The algorithm follows the recursive procedure of the decision algorithm defined in Section 7.1, starting from current depth level $cdl = 0$, $N \times N$ equal the size of DT, $m = 0$, $n = 0$. The algorithm defines the steps of the procedure as follows:

Procedure: $\mathbf{AlgIdx_{soft\text{-}decisive}(cdl, N, m, n)}$:

1. Denote the area $\mathrm{A}_{m,n,N}$ of the current block inside of $DT[i,j,\cdot]$ and $DM[i,j]$ as specified by the indices m, n and the size N, as in Formula 7.1.

$$\mathrm{A}_{m,n,N} = \{(i,j) \mid i \in [m; m+N-1],\ \ j \in [n; n+N-1]\}. \tag{7.1}$$

2. Analyze the DT values in the denoted area $\mathrm{A}_{m,n,N}$, e.g. $DT[i,j,\cdot]$, $(i,j) \in \mathrm{A}_{m,n,N}$.
   - If N is equal 1, set $DM[m,n] \leftarrow cdl$ and terminate the algorithm.
   - For the denoted area $\mathrm{A}_{m,n,N}$, find the index of the most probable depth level $DM_{AlgIdx}\,[i,j]$ as in Formula 7.4:

$$DM_{AlgIdx}[i,j] = \mathrm{ArgMax_d}(DT[i,j,\cdot]), for\ (i,j) \in \mathrm{A}_{m,n,N}. \tag{7.4}$$

     where $\mathrm{ArgMax_d}(F)$ is an argument for the maximal value of tensor F, along the d axis (channel).
   - Calculate the C value defined as in Formula 7.5:

$$C = \frac{1}{N^2} \cdot \sum_{(i,j) \in \mathrm{A}_{m,n,N}} \mathrm{Iv}\big(DM_{AlgIdx}\,[i,j] = cdl\big), \tag{7.5}$$

where Iv(·) is the Iverson function [Fo99], such that $\mathrm{Iv}(true) = 1$ and $\mathrm{Iv}(false) = 0$.

3. Make decision according to the analysis of the considered DT area $\mathrm{A}_{m,n,N}$:
   - If $C > 0.5$, fill the values in DM corresponding to the area of the current block with cdl (Formula 7.2), and terminate the recursion.
   - If $\boldsymbol{C \in (0.5 - \alpha\,; 0.5\, +\, \alpha)}$, fillthe values in DM corresponding to the area of the current block with $\boldsymbol{cdl + 5}$ (Formula 7.2). This corresponds to checking and comparing blocks for depth level values *cdl* and *cdl+1* with RD Optimization (Subsection 2.3.2). Terminate the recursion.
   - Otherwise: Recurse the procedure, for next depth level $cdl + 1$. Split the block following the quaternary tree (Figure 7.1) and perform recursive call for each subdivided block as in Formula 7.3 using $\mathbf{AlgIdx}_{\textbf{soft-decisive}}$ procedure.

7.3.2.2 Probability-based decision algorithm

In the previously described hard-decisive variant of AlgPrb algorithm (7.2.2) the decision are made using sums of probabilities, where the sum for the currently considered level is checked if it is maximal among all. Compared to AlgIdx, there is no threshold that could be used. However, if the difference between the sums for the current (S_{cdl} – formula 7.6) and the next depth level (S_{cdl+1} – formula 7.6) is small enough, then both corresponding block sizes should be checked. The condition is that one sum S_{cdl} or S_{cdl+1} is maximal among all sums for the current depth level. The absolute value of these sums is used to check if the difference is small enough. Additionally, this absolute value is normalized by the sum of these sums:

$$\frac{|S_{cdl} - S_{cdl+1}|}{S_{cdl} + S_{cdl+1}}. \tag{7.7}$$

Two block sizes should be checked if the resulting value is small enough. This can be controlled with the use of parameter $\beta \in \langle 0; 0.5)$. Therefore, when $(\max(S_L) = S_{cdl}\ or\ S_{cdl-1})$ and $\frac{|S_{cdl} - S_{cdl+}\ |}{S_{cdl} + S_{cdl+1}} \leq \beta$ the decision is considered uncertain. The higher the β, the higher this range, so the decision algorithm will indicate checking a pair of blocks more frequently.

The **soft-decisive variant of the Probability-based algorithm** (AlgPrb) follows the recursive procedure of the decision algorithm defined in Section 7.1. The algorithm starts from the current depth level $cdl = 0$, so that *N* is equal the size of DT, $m = 0, n = 0$. The algorithm defines the steps of the procedure as follows:

Procedure**: $\mathbf{AlgPrb}_{\textbf{soft-decisive}}(\boldsymbol{cdl, N, m, n})$:**

4. Denote the area $\mathrm{A}_{m,n,N}$ of the current block inside of $DT[i,j,\cdot]$ and $DM[i,j]$ as specified by the indices m, n and the size N, as in Formula 7.1.

$$\mathrm{A}_{m,n,N} = \{(i,j) \mid i \in [m; m+N-1],\ \ j \in [n; n+N-1]\}. \tag{7.1}$$

5. Analyze the DT values in the denoted area $\mathrm{A}_{m,n,N}$, e.g. $DT[i,j,\cdot]$, $(i,j) \in \mathrm{A}_{m,n,N}$.
   - If N is equal 1, set $DM[m,n] \leftarrow cdl$ and terminate the algorithm.
   - For each depth level $L \geq cdl$, calculate S_L, as in Formula 7.5.

$$S_L = \sum_{(i,j) \in \mathrm{A}_{m,n,N}} DT[i,\mathrm{j},L]. \tag{7.5}$$

6. Make decision according to the analysis of the considered DT area $A_{m,n,N}$:
   - If $(\mathbf{max}(\boldsymbol{S_L}) = \boldsymbol{S_{cdl}}\ or\ \boldsymbol{S}_{cdl-1})$ and $\frac{|\boldsymbol{S_{cdl}}-\boldsymbol{S_{cdl+1}}|}{\boldsymbol{S_{cdl}}+\boldsymbol{S_{cdl+1}}} \leq \boldsymbol{\beta}$, fill the values in DM corresponding to the area of the current block with $\boldsymbol{cdl+5}$ (Formula 7.2). This corresponds to checking and comparing blocks for depth level values *cdl* and *cdl+1* with RD Optimization (Subsection 2.3.2). Terminate the recursion.
   - If $S_{cd} = \max(S_L)$, fill the values in DM corresponding to the area of the current block with cdl (Formula 7.2), and terminate the recursion.
   - Otherwise: Recurse the procedure, for next depth level $cdl+1$. Split the block following the quaternary tree (Figure 7.1) and perform recursive call for each subdivided block as in Formula 7.3 using $\mathbf{AlgPrb}_{\mathbf{soft\text{-}decisive}}$ procedure.

7.3.2.3 Commentary

One can notice that for $\alpha, \beta = 0$ soft-decisive variants become hard-decisive. It should be noted that both soft-decisive variants of the algorithm have the same range of parameter value control: $\alpha, \beta \in \langle 0; 0.5)$. However, the values of these parameters have different meanings. This refers to different ways of impacting the number of soft decisions (check pair of blocks). Thus, the same value of a parameter cannot be referred to as equivalent.

7.3.3 Evaluation of the soft-decisive variants of the algorithms with Basic and Extended Architectures

The proposed soft-decisive algorithms were evaluated on test sequences (Section 3.5). Algorithms were tested on both Basic and Extended Architectures (Section 5.1 and 6.1). The evaluation was performed according to description in Section 3.1. The soft-decisiveness control parameter (α or β) was evaluated in the range $\langle 0.05; 0.45\rangle$, with a 0.05 step. As the evaluation results, the mean over all test sequences is presented for each parameter value. Full evaluation results (9 parameter values per model per decision algorithm) would decrease the readability of this section. In corresponding tables, the result for the hard-decisive variant (α or β = 0) is presented as a reference.

7.3.3.1 Evaluation of soft-decisive variant of Index-based decision algorithm

Table 7.3 presents evaluation results for the soft-decisive variant of AlgIdx (Subsection 7.3.2.1). When used with Basic Architecture (Section 5.1), the $BD\text{-}RATE$ and $BD\text{-}PSNR$ improvement are not so big (~0.2 p.p. and ~0.008 dB consecutively). Additionally, further improvements are not observed for $\alpha > 0.3$. For bigger α values, a pair of block sizes is checked more frequently, which is proven by decreasing the value of T_{ANN} (contribution of network processing time to coding time Section 3.3, Formula 3.3). That is because the ANN processing time for CTU is constant, as shown in Section 5.3. Surprisingly, the encoding time reduction (Time Savings: TS, Formula 3.3) remains almost the same despite the α value.

When soft-decisive AlgIdx is paired with Extended Architecture, the improvements in terms of $BD\text{-}RATE$ and BD-PSNR are almost three times better (~0.6 p.p. and ~0.03 dB consecutively). Still, the results of $BD\text{-}RATE$ are above 3%. Unfortunately, for $\alpha > 0.3$ both $BD\text{-}RATE$ and $BD\text{-}PSNR$ start to increase. Again, as α increases, T_{ANN} is steadily decreasing. The same was observed for TS, where the difference between $\alpha = 0.45$ and the hard-decisive variant is ~2.5 p.p..

Concluding the above observations, the soft-decisiveness mechanism is utilized relatively rarely. Falling T_{ANN} proves, that pair of blocks are considered more frequently with α value increase. The $BD\text{-}RATE$ and $BD\text{-}PSNR$ improvements are small but observable. However, increasing α value improves the evaluation results but only to a certain point. Further, α value increase gives no improvement and results in saddle effect. This means that for a range of soft decisiveness that is too big,

the algorithm may misinterpret the DT to use too big blocks. The TS has a very similar value for Basic Architecture despite the α value. This effect is investigated further in this section.

Table 7. 3. Evaluation results for the soft-decisive variant of AlgIdx with Basic Architecture (Basic) and Extended Architecture (Extended). Results for BD-RATE, BD-PSNR and TS are relative to HM, T_{ANN} is relative to encoding time of Modified HM.

Algorithm Variant	α value	*BD-RATE* [%]		*BD-PSNR* [%]		*TS* [%]		T_{ANN} [%]	
		Basic	Extended	Basic	Extended	Basic	Extended	Basic	Extended
Hard-decisive	0.00	1.86	3.66	-0.091	-0.187	56.08	70.50	21.27	28.55
Soft-decisive	0.05	1.80	3.50	-0.089	-0.179	54.97	69.29	19.36	27.94
	0.10	1.79	3.40	-0.088	-0.175	55.10	69.22	19.32	27.76
	0.15	1.78	3.32	-0.088	-0.171	54.84	68.99	19.28	27.62
	0.20	1.77	3.26	-0.087	-0.168	55.00	68.89	19.25	27.46
	0.25	1.72	3.15	-0.085	-0.163	54.97	68.63	19.11	27.17
	0.30	1.68	3.05	-0.083	-0.158	54.51	68.02	18.99	26.70
	0.35	1.68	3.04	-0.083	-0.158	54.64	67.98	18.97	26.54
	0.40	1.68	3.08	-0.083	-0.159	54.45	67.72	18.91	26.32
	0.45	1.68	3.10	-0.083	-0.160	54.66	67.46	18.84	26.02

7.3.3.2 Evaluation of soft-decisive variant of Probability-based decision algorithm

Results for the soft-decisive variant of AlgPrb (Subsection 7.3.2.2) are presented in Table 7.4. Compared to AlgIdx, results for Basic Architecture are much better. The $BD\text{-}RATE$ and $BD\text{-}PSNR$ are improved by ~0.6 p.p. and 0.028 dB for β = 0.45. What is more, $BD\text{-}RATE$ and $BD\text{-}PSNR$ improvements are monotonic as the β increase. Observations for encoding time reduction (Time Savings: TS, Formula 3.3) and T_{ANN} (contribution of network processing time to coding time Section 3.3, Formula 3.3) are the same as for the soft-decisive variant of AlgIdx. This means that a significant gain in $BD\text{-}RATE$ and $BD\text{-}PSNR$ was achieved without increasing the complexity of the encoding process.

Table 7. 4. Evaluation results for the soft-decisive variant of AlgPrb with Basic Architecture (Basic) and Extended Architecture (Extended) . Results for BD-RATE, BD-PSNR and TS are relative to HM, T_{ANN} is relative to encoding time of Modified HM.

Algorithm Variant	β value	*BD-RATE* [%]		*BD-PSNR* [%]		*TS* [%]		T_{ANN} [%]	
		Basic	Extended	Basic	Extended	Basic	Extended	Basic	Extended
Hard-decisive	0.00	1.80	3.44	-0.089	-0.177	54.20	69.22	19.41	28.44
Soft-decisive	0.05	1.69	3.10	-0.084	-0.160	55.72	69.01	19.29	27.38
	0.10	1.59	2.80	-0.079	-0.145	55.51	68.23	19.21	26.53
	0.15	1.51	2.55	-0.075	-0.132	55.45	67.31	19.14	25.81
	0.20	1.43	2.34	-0.071	-0.121	55.44	66.40	19.06	25.14
	0.25	1.37	2.15	-0.068	-0.111	55.30	65.36	19.08	24.60
	0.30	1.32	2.00	-0.066	-0.103	55.36	64.52	19.02	23.99
	0.35	1.27	1.87	-0.063	-0.097	55.52	63.56	18.94	23.40
	0.40	1.24	1.78	-0.062	-0.091	55.51	62.82	18.88	22.87
	0.45	1.23	1.71	-0.061	-0.088	55.5	62.02	18.83	22.33

The use Soft-decisive variant of AlgPrb with Extended Architecture results in great performance improvements. For β = 0.45 the $BD\text{-}RATE$ is 1.71% and $BD\text{-}PSNR$ is 0.088 dB, so results are twice better. This is similar results to the Basic Architecture with soft-decisive AlgPrb for β = 0.05. Both $BD\text{-}RATE$ and $BD\text{-}PSNR$ steadily improve as β increases. Unfortunately, this is occupied by ~8 p.p. decrease of TS. Still, the TS is better than for Basic Architecture by ~7 p.p.. T_{ANN} and TS steadily decreases as β increases.

According to the above observations, the soft decisions are applied more effectively for AlgPrb than for AlgIdx. This is proven by the results for Basic Architecture, where T_{ANN} is almost the same for both algorithms. It is not the case for the Extended Architecture, where for AlgPrb the T_{ANN} is much bigger. Two factors could have caused this: more frequent decisions to check a pair of blocks and uncertain decisions for the smallest blocks (CU: 8×8; PU: 8×8 and CU: 8×8; PU: 4×4) where the computations are the most complex. Overall, results for AlgPrb are much better for both Basic and Extended Architectures. This proves that the appropriate interpretation of the ANN output yields increased performance in terms of both encoding efficiency and time. Additionally, the changes in observed metrics are always monotonic as the β increases.

7.3.3.3 Conclusions on soft-decisive variants of the decision algorithms

The soft-decisiveness built into the decision algorithm proved to improve the performance of both algorithms. These performance improvements come only from a better interpretation of the ANN output. Bigger improvements are observed for Extended Architecture. Training results for Extended Architecture (e.g., $Accuracy$ ~61%, Formula 4.8), despite better classification for some depth level, were overall worse than for Basic Architecture (e.g., $Accuracy$ ~71%, Formula 4.8). Thus, better improvements are observed as the ANN output indicate a partitioning pattern less certainly.

Another reason of higher improvements observed for Extended Architecture is control over the PU division. As shown in Subsection 7.2.1, encoding blocks CU: 8×8; PU: 4×4 is the most complex among all other options. What is more, inappropriate use of this block size may result in a significant bitrate increase. So, the soft-decisive algorithm can spot an uncertain indication of a smaller block and decide to check if a bigger one will perform better. That may compensate for the increase in $BD\text{-}RATE$ or TS.

It has been demonstrated, that the soft-decisive algorithm compensates the worse training results. This means that a smaller and less complex model could be applied. In the case of the Extended Architecture, improvements of $BD\text{-}RATE$ come with a decrease of the encoding time reduction (Time Savings: TS, Formula 3.3). However, the ~3 p.p. and ~8 p.p. worse TS (for AlgIdx and AlgPrb consecutive) still makes the Extended Approach a viable solution. For Basic Architecture, the improvements in $BD\text{-}RATE$ did not affect the TS at all, but the T_{ANN} decreases as the α or β increase. This means that in some cases, the decision algorithm, instead of hardly indicating depth level 3 (checking both CU: 8×8; PU: 8×8 and CU: 8×8; PU: 4×4 – the most computably complex according to Table 7.2), indicates check pair of blocks: CU: 16×16; PU: 16×16 and CU: 8×8; PU: 8×8, which is less computably complex. The resulting reduction in complexity compensates for the increase resulting from considering the pair of blocks in other situations.

7.3.4 Soft-decisive algorithms as methods for control over Encoding Time vs. Compression Efficiency trade-off

In the previous subsection, it was observed that when the soft-decisive variant of decision algorithms (Subsection 7.3.2) is used with Extended Architecture, both $BD\text{-}RATE$ and the encoding time reduction (Time Savings: TS, Formula 3.3) are changing. Using a bigger value of the α or β parameter, a better $BD\text{-}RATE$ can be achieved, but the TS decreases. Likewise, when a small α or β parameter is used, then a higher TS is achieved, but at the cost of increased $BD\text{-}RATE$. Figure 7.2 presents the evaluation results of the soft-decisive algorithm on a plane $BD\text{-}RATE - TS$. Results for both Basic and Extended Architectures (Section 5.1 and 6.1) are presented for comparison. Presented results are relative to HM.

As shown in the figure, results for Basic Architecture (Section 5.1) with AlgPrb align to an almost flat line. The Encoding Time vs. Coding Efficiency (ETvsCE) trade-off improves as the β parameter increases. For the Basic Architecture with AlgIdx., the ETvsCE trade-off changes, but by a very small margin. Similar observations can be made for Extended Architecture with AlgIdx. Unfortunately, a saddle effect is observed. The most interesting are results for Extended Architecture with AlgPrb. The

$BD\text{-}RATE$ monotonically fell as the value of β increased. This means that by using just one parameter (β) it is possible to change the ETvsCE trade-off in quite a wide range: $\langle 1.77; 3.44 \rangle$ in terms of $BD\text{-}RATE$ and $\langle 62.02; 69.22 \rangle$ in terms of TS.

The soft-decisive variants of the decision algorithm allow for control over the ETvsCE trade-off. The use of the proposed mechanism allows for precise fine-tuning of the encoder complexity. Additionally, the quality practically does not change: the difference of $BD\text{-}PSNR$ is no more than 0.09 dB. What is more, the control is done by only a single parameter and does not require any changes to the decision algorithm or ANN model. As mentioned in Section 2.5, such a mechanism has a very practical use, e.g., in multiple coding scenarios (in the server that encodes numerous video streams and the number of streams is varying) or when an encoder tries to fit the restriction of frame encoding time during transmission [Hu23]. Described method together with broadcasting application use-case analysis was published in the paper [Lo24].

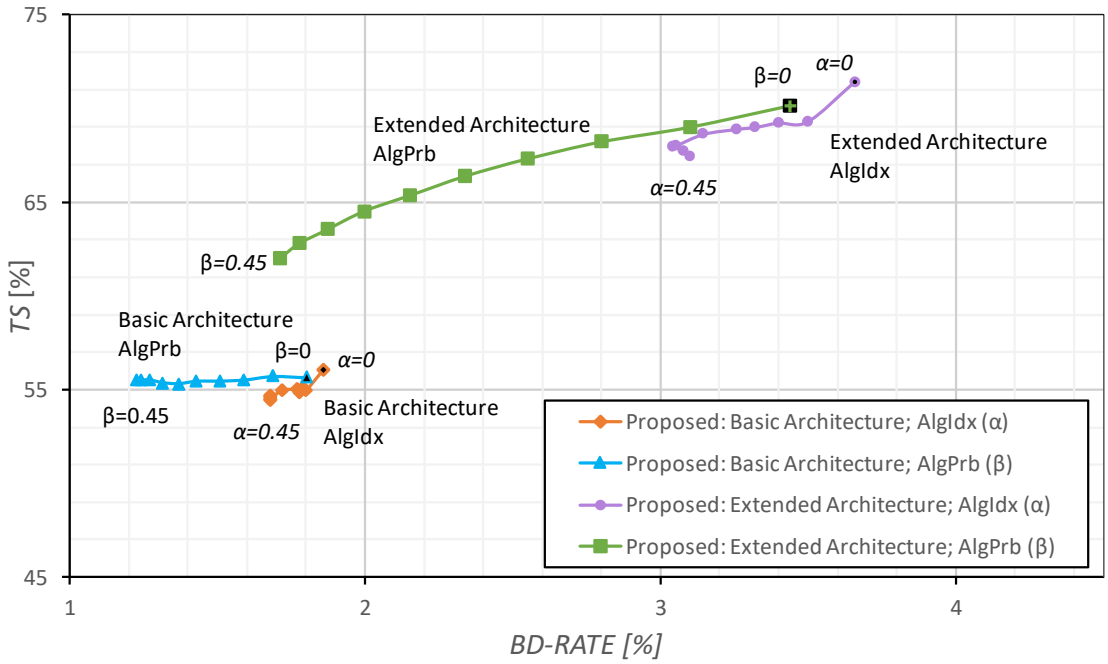

Figure 7. 2. Evaluation results of the soft-decisive algorithm on a BD-RATE vs TS plot. Presented results are relative to HM.

8 Results comparison with state of the art

8.1 Methodology of comparison with state of the art

The general methodology for evaluation of a partitioning algorithm has been presented in Chapter 2. As mentioned, this is done with the use of metrics: $BD\text{-}RATE$, $BD\text{-}PSNR$, TS and FoM, calculated for test sequences. When considering comparison of multiple state-of the-art methods, however, multiple factors have to be taken into account. The most important ones are discussed below.

The first factor is the version of the encoder that was used for the implementation and testing of given partitioning algorithm. The consecutive versions of the HM make different sets of decisions during the encoding process. This means that the distribution of the CU/PU blocks differs, so according to Subsection 7.3.1, the assessment result of the partitioning algorithm may change, especially the TS.

The second factor is the experimentation platform, both software and hardware, which impact the encoding time assessment. As discussed in Section 3.3, the time measurement is very sensitive to this factor. Along with CPU model, a major determinant is the number of CPU cores available for computation. In terms of software factors, a key factor is the implementation of the method, especially if it is ANN-based. In some cases, the ANN may be implemented directly, but in others (such as proposed in this dissertation) a dedicated framework is used.

In order to minimize discrepancies related to the above-mentioned factors and ensure proper comparison of partitioning algorithms, especially in terms of TS, the testing conditions should be as consistent as possible. As the implementation connected factors would be near impossible to neglect, at minimum the partitioning algorithms should be tested using the same hardware and operation systems. Unfortunately, the authors of most algorithms do not share their software. So, based solely on results found in the literature, the comparison with most partitioning algorithms is relatively inaccurate. Additionally, the version of the used HM software may be a significant factor. Although methods found in the literature are individually assessed versus respective HM version reference, the usage of different HM versions may shift the operating point of the encoder considerably. It can be noted that in this dissertation HM version 16.23 has been used, while the methods found in the literature employ HM versions between 15.0 and 16.20. Therefore all of these factors must be taken into account during the comparison.

In this dissertation, a comparison of proposed partitioning algorithms is divided into two parts. The first part presents a general comparison (Section 8.2) with the best-performing partitioning algorithm found in the literature. Around 90 methods were already mentioned (Section 2.4), which were found the most relevant among the state-of-the-art. Among all these methods, 7 were chosen for comparison with proposed partitioning algorithm as the most representative of the state of the art. For general comparison the following methods were chosen:

1. [Li16A] – the best performing non-ANN method, but evaluated on HM in version 15.0. The method uses the SVM classifiers to hierarchically estimate split flags of the quaternary tree. The authors of the paper do not clarify the dataset used for SVM training. Firstly is performed the complexity analysis of the whole frame, in terms of the number and direction of edges. A similar analysis is done for the currently considered CU block. The results of the analysis then are imputed to SVM to decide on a split flag. The output of the SVM may be considered uncertain, which results in considering multiple block sizes using original RD optimization. The method considers only CU blocks. The **method is chosen only for general comparison** due to the undefined training dataset and used older version of the HM (15).
2. [Zh18] – non-ANN method that uses SVM for hierarchical estimation of split flags. The decision is made using only the luma component of the CTU, by its analysis, e.g. the variance. The SVM is trained with several frames chosen from the JCT-VC dataset. The decision on PU

division is considered as an additional level of the quaternary tree. The method was implemented in HM 16.7. This **method is chosen only for general comparison** due to the use of test sequences in SVM training.

3. [Sh19] – ANN-based method for hierarchical split flag estimation including the PU division. The ANN processes luma samples corresponding to the currently considered depth level and estimates the probability of split. If the probability of split for depth level 3 is high enough, the PU division is made. The authors proposed separately trained models for each depth level and set of thresholds for PU division for different *QP* values. The ANN architecture is similar to AlexNet [Kr12], but instead of the first convolution section, three convolution sections are used with different kernel shapes. For ANN training three datasets were used: RAISE [Da15], UCID [Sh04, Sh10], and DIV2k [Ag17]. During the training process, the bit cost of the block was used as part of the loss function. The method was implemented in HM 16.9. This **method is chosen only for general comparison**, as the authors do not specify the hardware platform used for evaluation. This will be further discussed in the description of the next method (same authors).
4. [Ch20] – ANN-based method for hierarchical split flag estimation including the PU division. This method is an extension of the method presented in [Sh19] by extending the algorithm for the estimation of the PU prediction modes. However, the authors deliver evaluation results when only partitioning CTU with PU division is done. The partitioning method is enhanced by using two models with different *QP* values to better decide on split flags. Furthermore, a more advanced thresholding of ANN output (4 thresholds for each depth level) is applied. Additionally, the ANN models were slightly modified. Furthermore, the authors proposed a mechanism of adjusting threshold values to provide control over the ETvsCE tradeoff with the use of an evolution algorithm. This **method is chosen only for general comparison**, as the authors present multithread and highly optimized CPU implementation.
5. [Xu18A] – ANN-based method for hierarchical split flag estimation, only for CTU partitioning. The authors use a single ANN, that processes CTU luma samples, to estimate all split flags for the currently processed CTU block. The ANN is similar to AlexNet [Kr12] but consists of three parallelly processing convolutional parts, whose output is concatenated and processed by a single dense layer. The output of the dense layer is then processed parallelly by three fully connected parts, that estimate split flags (sigmoid) for different depth levels. The last two dense layers include the *QP* value. The partitioning is then estimated in a top-bottom fashion by thresholding the ANN output (similar to in [Li16A], threshold for certain and uncertain decisions). The ANN is trained using the RAISE dataset [Da15], with sequences in original resolution and down-sampled. The method was implemented in HM version 16.5. This **method was chosen both for general and detailed comparison**, as all decisive conditions were met.
6. [Hu21B] – a similar method to [Xu18A], single ANN used for estimation of all split flags for currently processed CTU at once. The ANN estimates split flags only for CTU partitioning. For the PU division, the authors proposed a Naïve-Bayes based method, that uses the CTU samples. The ANN is similar to [Xu18A] (AlexNet [Kr12] alike), but the convolution part consists of two parallelly processing convolutional parts (square kernels). The first one processes jointly samples from the current CTU and neighboring samples (left, top-left, and top). The second one processes only samples from the current CTU, but feature maps from the three last layers are outputted. The decision over the split flag is then made with thresholds (similar to in [Li16A], threshold for certain and uncertain decisions), in a top-bottom fashion. The ANN is trained using the same dataset as [Xu18A] (RAISE [Da15]), and cross-entropy loss. Furthermore, the authors provide control over the ETvsCE trade-off by estimation of thresholds set for target encoding time reduction. The method was implemented in HM version 16.5. This **method was chosen both for general and detailed comparison**, as all decisive conditions were met.
7. [Fe21] – ANN-based method for estimation of whole partitioning pattern, only for CU partitioning. The authors proposed a fully convolutional network that processes the luma

samples of the CTU and estimates the Division Matix of size 8×8 (same as DM_E). The network architecture includes the branches of max-pooling operation with different pooling kernel sizes, which are then interpolated to the same tensor size and further proceed. Additionally, the authors proposed an L1 norm-based loss function which was used during the training process. The output of the ANN is corrected using an algorithm similar to hard-decisive variant of AlgIdx (Section 7.2.1). The ANN is trained using the DIV2k dataset [Ag17] (only 1920x1280 resolution) and video sequences dataset CDVL [Pi13] (every 40th frame). The method was implemented in HM version 16.20. This **method was chosen both for general and detailed comparison**, as all decisive conditions were met.

The general comparison is done graphically in Figures 8.1 and 8.2. Additionally, the ANN-based methods will be compared in terms of the number of weights and MAC operation count in Tables 8.1 and 8.2.

The second part presents a detailed comparison (Section 8.3) of the proposed partitioning algorithms with solutions found in the literature. For this comparison were chosen methods which implementation and evaluation conditions were the most similar to those of proposed partitioning algorithms. Such selection will ensure the least impact of result-affecting factors, discussed earlier in this chapter. This comparison can be considered sufficiently accurate. The criterions were as follows:

- Implementation of the partitioning algorithm in HM, version at least 16 (newest major version of the software).
- Partitioning algorithm and its implementations do not impact other decisions made by the encoder.
- The implementation of the ANN uses only a single CPU core/thread for computation and computation is performed as a step of the CTU encoding.
- Test sequences are not used for training any part of the partitioning algorithm.

Three methods described earlier in this section: 5, 6 and 7, meet the above conditions and are chosen for detailed comparison. The detailed comparison is done by means of $BD\text{-}RATE$, $BD\text{-}PSNR$, TS and FoM (Tables 8.3 – 8.6 consecutively). One of the methods [Hu21B] chosen for detailed comparison offers control over the ETvsCE trade-off. Thus the comparison of proposed partitioning algorithms and this method in terms of the proposed $\Delta BD\text{-}RATE|_{TS}$ and $\Delta TS|_{BD\text{-}RATE}$ will be presented in Table 8.7.

In graphical comparisons, are presented the evaluation results for AlgIdx and AlgPrb paired with Basic and Extended Architecture (Section 5.1 and 6.1), for α or β parameters in the range $\langle 0.05; 0.45 \rangle$, with 0.05 step. For the comparison by the means of $BD\text{-}RATE$, $BD\text{-}PSNR$, TS and FoM, following cases are selected:

- Basic Architecture with AlgIdx ($\alpha = 0$);
- Basic Architecture with AlgPrb ($\beta = 0.45$);
- Extended Architecture with AlgPrb ($\beta = 0.0$);
- Extended Architecture with AlgPrb ($\beta = 0.45$).

8.2 General comparison to selected state-of-the-art methods

Sequences chosen for general comparison are the best partitioning algorithms found in the literature. As discussed in previous sections, the difference in evaluation and implementation of the partitioning algorithms have to be taken into consideration during the comparison. This applies especially to methods [Li16A, Zh18, Sh19, Ch20] and interpretation of TS results. Figure 8.1 presents the graphical comparison of the methods chosen for general comparison, in terms of $BD\text{-}RATE$ and TS. Results from papers have been unified to $BD\text{-}RATE$ vs. TS format, and are relative to HM. The operating points corresponding to the hard-decisive variants of the algorithms ($\alpha = 0$, $\beta = 0$) have been marked with black dots.

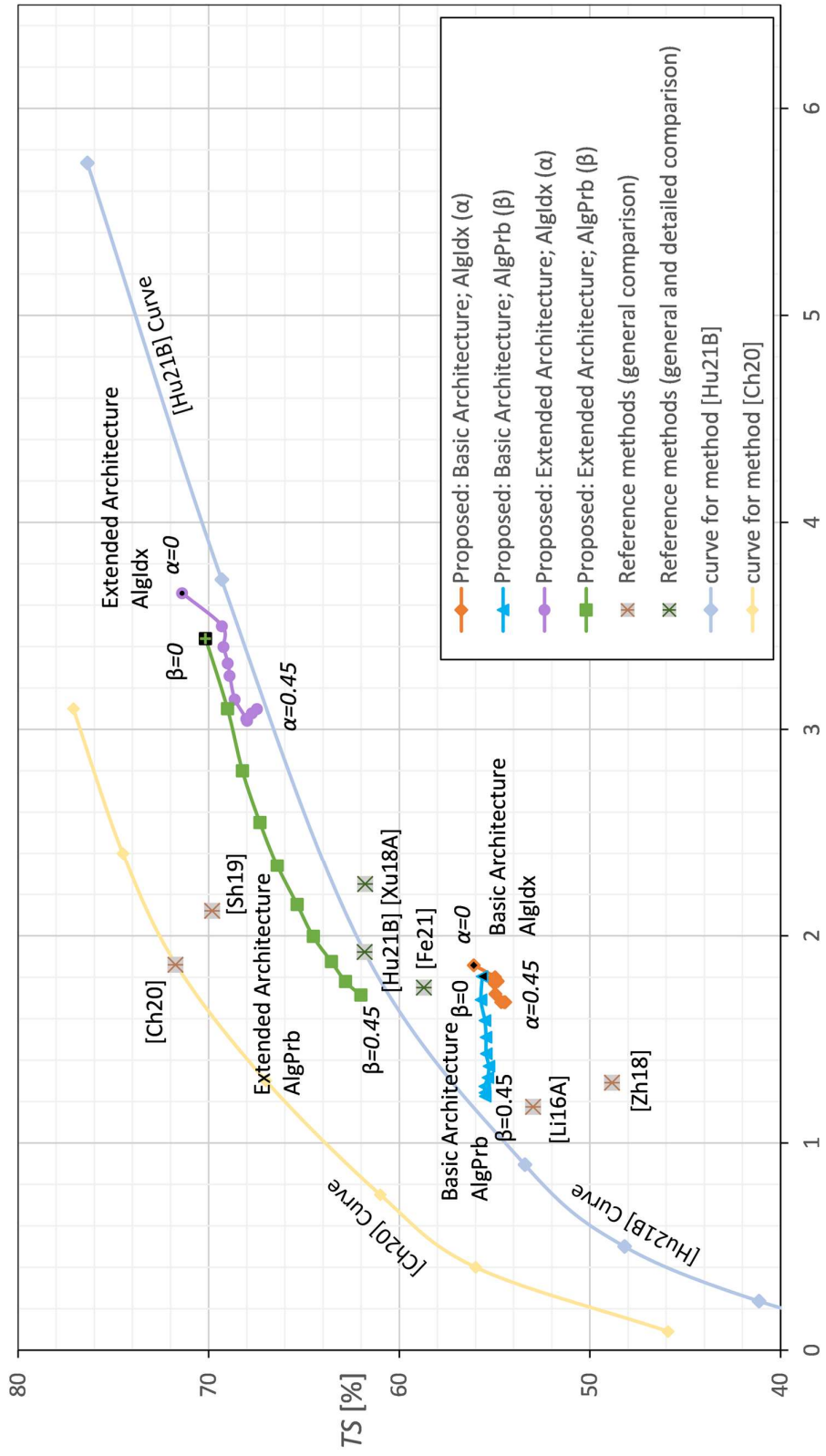

Figure 8. 1. General comparison of the proposed partitioning algorithms with state-of-the-art-solutions. Presented results are relative to HM.

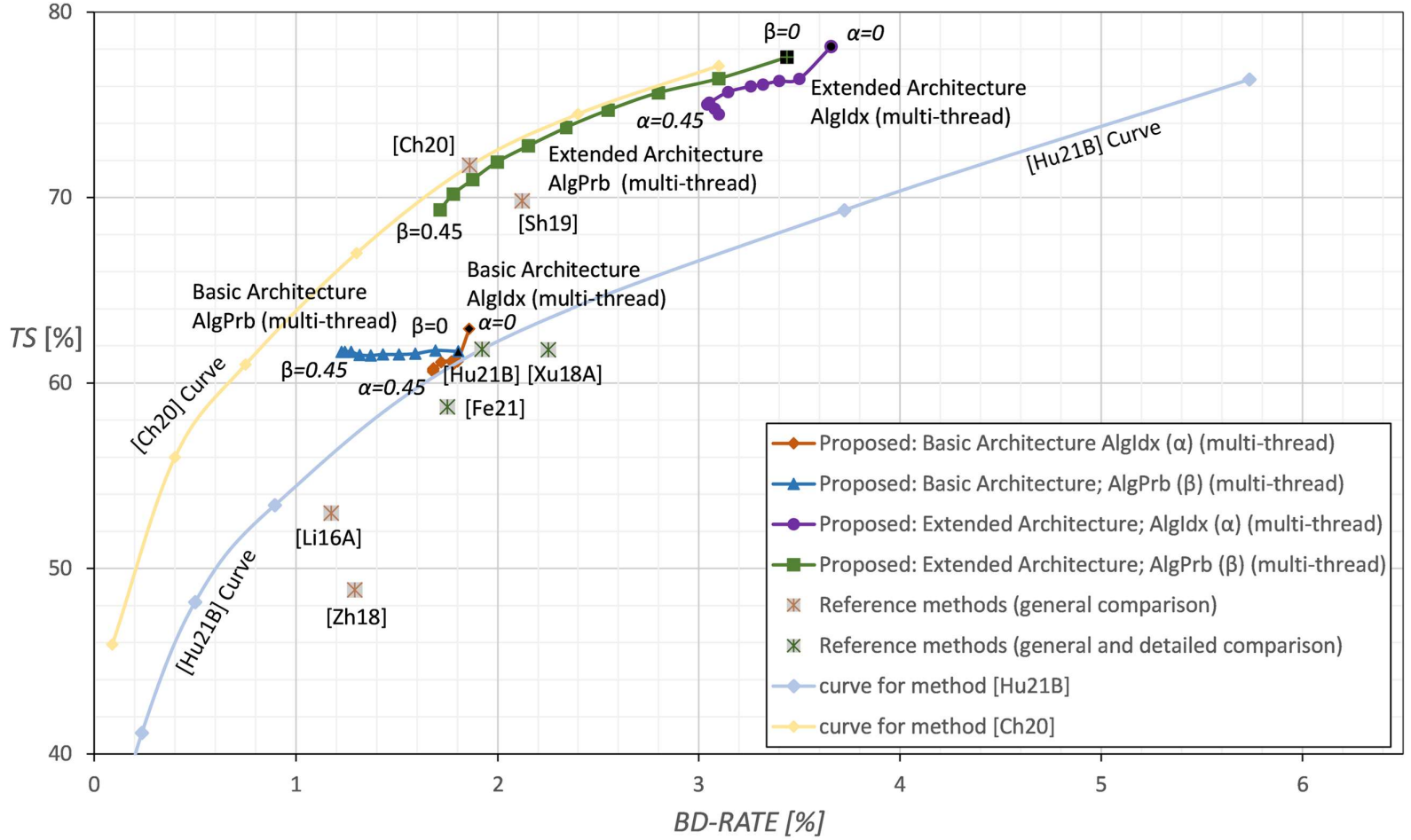

Figure 8. 2. General comparison of the proposed partitioning algorithms (simulated multi-thread implementation) with state-of-the-art solutions. Presented results are relative to HM.

The results for the Basic Architecture (Section 5.1) are competitive with the state-of-the-art solutions, where only CU partitioning is considered [Li16A, Zh18, Xu18A, Fe21]. The results for hard-decisive variants of decision algorithms (AlgIdx α=0, AlgPrb β=0) are slightly worse compared to methods for considered scope of decisions. Still, the difference to the closest method [Fe21], with similar $BD\text{-}RATE$, is ~5 p.p. in terms of encoding time reduction (Time Savings: TS, Formula 3.3). This changes when the soft-decisive variant of AlgPrb (Subsection 7.3.2.2) is considered as $BD\text{-}RATE$ is improving. Points for consecutive β values form an almost flat line, so improvement in efficiency does not change the TS. The Basic Architecture with the soft-decisive AlgPrb ($\beta = 0.45$) offers similar $BD\text{-}RATE$ as method [Li16A], but ~2 p.p. better TS. The change in trade-off is be evaluated in the next section. Soft-decisiveness in AlgIdx (Subsection 7.3.2.2) does not significantly change the evaluation results.

Considering the methods for both CU and PU [Sh19, Ch20, Hu21B], proposed algorithms with Extended Architecture (Section 6.1) are not competitive to methods [Sh19, Ch20]. These methods offer the same $BD\text{-}RATE$, but with 5-10 p.p. better TS. One should recall that these methods were not chosen for detailed comparison due to the hardware not being specified [Sh19] or multi-threaded implementation [Ch20]. As discussed in Section 4.7, the Modified HM used for evaluation is a single-threaded implementation with the sequential approach for partitioning estimation. This means that the partitioning pattern is estimated as the CTU being processed. Such an approach is the most popular one [Li16A, Zh18, Xu18A, Hu21B, Fe21] as it allows evaluation of the method in the most comparable way. The multi-threaded implementation of the methods [Sh19] and [Ch20] is the reason for the superiority in terms of the TS. This is further discussed later in this section.

Compared to the method [Hu21B], the proposed partitioning algorithms with Extended Architecture perform better, as they offer same $BD\text{-}RATE$ with better TS. The results for both AlgIdx and AlgPrb are above the curve for the method [Hu21B]. In the results for AlgxIdx, a pivot point is observed. The results for AlgPrb are ~2 p.p. better in terms of TS compared to the method [Hu21B]. However, a wider range of control over the ETvsCE trade-off is possible in method [Hu21B]. A more comprehensive comparison with the method [Hu21B] is presented in the next section.

Proposed partitioning algorithms can be easily implemented in the way that makes the processing time of the partitioning algorithm almost negligible. This implementation assumes all computations of the partitioning algorithm in one thread and the rest of the encoder computations in another thread. This means that only two threads are needed. As presented in Sections 5.3 and 6.3, the computation time of the ANN can be considered constant and is ~2,4 ms for both proposed architectures, on the machine used to conduct the evaluation. To estimate the minimal processing time of CTU without a partitioning algorithm, one can use the results of experiments with constant CTU partitioning patterns, presented in Subsection 7.3.1. The smallest encoding time was observed when only block size CU: 16×16 PU: 16×16 is considered, for QP=37. For this case, the CTU processing time was estimated as ~2.7 ms. The partitioning algorithm marginally impacts the encoding time because the ANN computation time is smaller than the fastest CTU block processing in the discussed double-threaded implementation. The partitioning algorithm impacts the encoding time only by the time of computations for the first CTU, as for the rest of the CTUs, the partitioning patterns are estimated during the CTU processing.

Simulated results for the discussed double-threaded implementation are shown in Figure 8.2. The presented results are relative to HM. The encoding time results were obtained by subtracting the accumulated ANN processing time from the encoding time and adding the time of single ANN processing. One can observe that the TS results have improved by ~5 p.p. for Basic Architecture. This makes the Basic Architecture with AlgPb ($\beta = 0.45$) the best among all algorithms that only CU is considered. Results for the Extended Architecture are improved by ~10 p.p. in terms of TS compared to results for single-threaded implementation (Figure 8.1). Results for the Extended Architecture with a soft-decisive variant of AlgPrb are very close to the curve for the method [Ch20]. The differences in TS

for the same $BD\text{-}RATE$ are ~0.5 p.p., which is a negligible difference considering that methods were not evaluated in the same HM software version and on different machines.

For the ANN-based partitioning algorithms, key factors are the size and complexity of the models. The number of weights and MAC operation count are presented in Tables 8.1 and 8.2 consecutively, for models used in chosen methods from literature and proposed Basic and Extended Architectures. The tables contain the numbers declared by the authors in the papers; unfortunately, not all are available. Therefore, all numbers were estimated using PyTorch [Pa19] implementations of models created according to the description in the papers. For fairness, models for all chosen methods were implemented, as the estimation of size and complexity may differ depending on implementations.

Table 8. 1. Comparison of the ANN model size (number of weights) used in proposed partitioning algorithms with models used in literature. Declared numbers were found in corresponding papers. Estimated numbers come from analysis of the implementation of models in the PyTorch framework.

	Estimation of the whole partitioning pattern at once					**Sequential split flag estimation**		
	Division Matrix			**Split flags**				
	Proposed Basic	**Proposed Extended**	**[Fe21]**	**[Xu18A]**	**[Hu21B]**	**Depth level**	**[Sh19]**	**[Ch20]**
Number of model weights declared by authors.	91 600	91 617	-	1 287 189	-	0	166 866	43 986
						1	43 602	43 602
						2	43 346	43 346
						3	43 346	43 346
Estimated number of model weights	91 600	91 617	278 705	1 288 210	609 978	0	166 866	43 986
						1	43 602	43 602
						2	43 346	43 346
						3	43 346	43 346

Table 8. 2. Comparison of the ANN model complexity (MAC operation count) used in proposed partitioning algorithms with models used in literature. Declared numbers were found in corresponding papers. Estimated numbers come from analysis of the implementation of models in the PyTorch framework.

	Estimation of the whole partitioning pattern at once					**Sequential split flag estimation**		
	Division Matrix			**Split flags**				
	Proposed Basic	**Proposed Extended**	**[Fe21]**	**[Xu18A]**	**[Hu21B]**	**Depth level**	**[Sh19]**	**[Ch20]**
Number in millions of operations declared by authors	6.76	8.54	-	1.55	-	0	-	0.64
						1	-	-
						2	-	-
						3	-	0.01
Estimated number in millions of operations	6.76	8.54	337.02	1.56	1.31	0	2.58	0.61
						1	0.51	0.51
						2	0.44	0.44
						3	0.42	0.42
						The best case	2.58	1.22
						The worst case	18.38	32.82

* Best case refers to a single run of the ANN for depth level 0

** Worst case refers to all ANN processing from depth level 0 to depth level 3 in quaternary tree

In terms of the model size, the proposed Basic and Extended Architectures are the smallest. The only smaller models are the ones used for sequential split flag estimation, but one should underline that methods [Sh19] and [Ch20] need four models instead of just one. The proposed models are at least three times smaller than other models for the estimation of the whole partitioning pattern at once. The

estimated sizes of the models are identical in most cases, and a small difference is spotted for the method [Xu18A].

Considering the model complexity, the proposed architectures are much less complex than the other approaches for the estimation of Division Matrix [Fe21]. Despite a much bigger MAC operation count (~50 times), the method [Fe21] achieved ~5 p.p. better results in terms of TS (Figure 8.1) compared to proposed partitioning algorithms with the Basic Architecture. Both these methods estimate only CU partitioning. The authors of the method [Fe21] reported a prediction accuracy of ~68% for all depth level values. It was observed that Basic Architecture (Section 5.2) have better prediction results for bigger depth level values. Thus, the difference in TS may be explained by better prediction of the bigger blocks, as using too small blocks leads to an increase in the encoding complexity, as shown in Subsection 7.3.1. Similar reasoning is applied for the comparison of the proposed Basic Architecture-based algorithm with the method [Xu18].

The opposite situation is observed for the Extended Architecture and method [Hu21B]. Despite the higher complexity of the Extended Architecture, the results of TS are better by ~2 p.p. (Figure 8.1). The applied approach may explain this. In method [Hu21B] all split flags are estimated at once. Authors reported that the applied models achieve ~90% prediction accuracy for depth level values 0-3. Such results are much better than $Accuracy$ (Formula 4.8) presented for Extended Architecture (~60%, Section 6.2). Then, the PU division is decided with the Naive Bayes based algorithm. In Subsection 7.3.1, it was discussed that wrong decisions for the smallest blocks (CU 8×8 and PU 4×4) significantly impact the $BD\text{-}RATE$ and TS negatively. As the proposed approach with Extended Architecture performs better in terms of TS, despite a ~6.5 times more complex model, thus the proposed approach is superior in the estimation of PU division, compared to the method [Ch20].

Comparison with the chosen methods of sequential split flag estimation ([Sh19, Ch20]) is quite a tricky issue. The first problem is that the authors published accuracy and rations of split flags without the distribution of decisions. This means that complexity may be estimated only for the best-case scenario – the algorithm stops after the first split flag – and the worst-case scenario – all possible split flags were considered. Compared to the complexity of the proposed Extended Architecture, the complexity of methods [Sh19, Ch20] is 3.3 [Sh19] or 7 [Ch20] times smaller for blocks with simple content but 2.2 [Sh19] or 3.8 [Ch20] times bigger for blocks with demanding content. Compared to simulated results for the proposed multi-threaded implementation of the partitioning algorithm with Extended Architecture and AlgPrb, the ANNs used in methods presented in [Ch20] perform very similarly.

8.3 Detailed comparison to selected state-of-the-art methods

The detailed comparison is made for averaged results for classes in test sequences classes and averaged results for all test sequences. Two of the methods chosen for detailed comparison do not provide results for two sequences from class A (“NebutaFestival” and “SteamLocomotiveTrain” – two of the most complex sequences in class according to Section 5.3 and 6,3). So, the results for class A and the mean over all test sequences are presented in two scenarios: with and without mentioned sequences. Results for $BD\text{-}RATE$ are presented in Table 8.2, for $BD\text{-}PSNR$ in Table 8.3, for TS in Table 8.4 and FoM in Table 8.5. The presented results are relative to HM. In the description of the method [Hu21B], results for multiple operating points are reported. In this comparison, results referred to by authors as “Ours3” are used. The $\alpha = 0$ means hard-decisive variant of the AlgIdx (Subsection 7.2.1). The $\beta = 0$ means hard-decisive variant of the AlgPrb (Subsection 7.2.2). The $\beta = 0.45$ means soft-decisive variant of the AlgPrb (Subsection 7.3.2.2) with highest value of β parameter (most probable soft-decisive decision).

In most cases, the same relations between averaged evaluation results in classes are observed between proposed and chosen methods from the literature. Considering the $BD\text{-}RATE$ and $BD\text{-}PSNR$,

the best results are achieved for the smallest resolutions (Class D), and the worst results are observed for class E. Disregarding class E and two sequences from class A ("NebutaFestival" and "SteamLocomotiveTrain"), the smaller the resolution, the better the results. Considering TS, the best results are achieved for class E or A, and the worst for class D. Generally, the smaller the resolution, the worse TS. Considering FoM values, a better ETvsCE trade-off is observed for smaller resolutions. Again, the worst results (the highest values) are observed for class E. Despite differences in the training datasets, all tested methods perform the worst for class E sequences, which contain the talking heads content. These observations confirm that the proposed Basic and Extended Architectures (Section 5.1 and 6.1) were trained comparably well.

Table 8. 3. Mean bitrate change (BD-RATE [%]) of the proposed methods, compared with the state-of-the-art methods chosen for the detailed comparison, averaged over the classes of test sequences. Presented results are relative to HM.

Sequence class	***BD-RATE* [%]**						
	CU partitioning only				**CU blocks and PU division**		
	Basic Architecture		**[Fe21]**	**[Xu18A]**	**Extended Architecture**		**[Hu21B]**
	AlgIdx: α = 0	**AlgPrb: β = 0.45**			**AlgPrb: β = 0**	**AlgPrb: β = 0.45**	
A* (2560×1600)	2.22	1.51	2.36	2.46	4.05	1.96	1.82
A (2560×1600)	1.99	1.19	2.06		2.95	1.40	
B (1920×1080)	2.09	1.30	1.90	2.58	3.44	1.64	1.51
C (832×480)	1.57	1.15	1.52	1.90	3.67	1.74	2.22
D (416×240)	0.93	0.69	0.68	1.17	2.68	1.24	1.82
E (1280×720)	2.93	1.95	2.85	3.46	4.81	2.86	2.26
A*,B,C,D,E **	**1.87**	**1.26**	**1.75**	**2.25**	**3.62**	**1.81**	**1.90**
A ,B,C,D,E **	**1.86**	**1.23**	**1.76**		**3.44**	**1.71**	

Table 8. 4. Mean PSNR change (BD-PSNR [dB]) of the proposed methods, compared with the state-of-the-art methods chosen for the detailed comparison, averaged over the classes of test sequences. Presented results are relative to HM.

Sequence class	***BD-PSNR* [dB]**						
	CU partitioning only				**CU blocks and PU division**		
	Basic Architecture		**[Fe21]**	**[Xu18A]**	**Extended Architecture**		**[Hu21B]**
	AlgIdx: α = 0	**AlgPrb: β = 0.45**			**AlgPrb: β = 0**	**AlgPrb: β = 0.45**	
A* (2560×1600)	-0.123	-0.084		-0.126	-0.224	-0.109	
A (2560×1600)	-0.102	-0.062			-0.155	-0.073	
B (1920×1080)	-0.079	-0.051		-0.090	-0.137	-0.064	
C (832×480)	-0.086	-0.063		-0.099	-0.204	-0.097	
D (416×240)	-0.061	-0.045		-0.072	-0.176	-0.083	
E (1280×720)	-0.144	-0.097		-0.164	-0.238	-0.142	
A*,B,C,D,E **	**-0.092**	**-0.064**		**-0.104**	**-0.187**	**-0.094**	
A ,B,C,D,E **	**-0.091**	**-0.061**			**-0.177**	**-0.088**	

* – only part of class A: PeopleOnStreet and Traffic video test sequences

** – mean over all sequences in enlisted classes

– results not provided by the authors

The Basic Architecture with AlgIdx achieve similar $BD\text{-}RATE$ and $BD\text{-}PSNR$ compared to the best methods with the same approach (CU blocks only) found in the literature [Xu18A, Fe21]. The proposed algorithm outperforms [Xu18A] by ~0.4 p.p., considering the average for all test sequences, and outperforms this solution in every resolution class. However, the method [Xu18A] is better by ~9 p.p. in terms of TS. The method [Fe21] achieves better results by ~0.1 p.p. for the mean over all test sequences. Results are very close for each sequence class, but the proposed partitioning algorithm is better only for class A. The difference is small despite the 1.5 times bigger training dataset used in

[Fe21]. The authors of the method [Fe21] do not provide results of $BD\text{-}PSNR$, so this comparison was not possible. Still, the difference in TS is ~5 p.p. in favor of [Fe21].

As observed in the previous section, authors of the method [Fe21] use a much more complex model, so the difference in encoding time (TS) is caused by decisions made by the algorithm. It was noticed that method [Fe21] better predicts bigger blocks, which is confirmed by results for class A. The TS is almost the same, regardless of whether the two most complex sequences in the class were considered or all sequences. The proposed partitioning algorithm achieves better $BD\text{-}RATE$ and $BD\text{-}PSNR$ for this class and better predicts small block sizes (sections 5.2 and 5.3). Considering the impact of block size on encoding time (Subsection 7.3.1), to achieve such results in class A, the method [Fe21] have to indicate mostly blocks bigger than 16×16. This explains the superiority of the method [Fe21] over the Basic Architecture with hard-decisive AlgIdx.

Table 8. 5. Mean encoding time reduction (TS [%]) of the proposed methods, compared with the state-of-the-art methods chosen for the detailed comparison, averaged over the classes of test sequences Presented results are relative to HM.

Sequence class	TS [%]						
	CU partitioning only				CU blocks and PU division		
	Basic Architecture		[Fe21]	[Xu18A]	Extended Architecture		[Hu21B]
	AlgIdx: α = 0	AlgPrb: β = 0.45			AlgPrb: β = 0	AlgPrb: β = 0.45	
A* (2560×1600)	56.69	57.46	74.25	65.9	72.23	63.86	62.20
A (2560×1600)	65.94	63.72	74.60		74.92	67.26	
B (1920×1080)	61.38	59.90	66.79	70.61	72.70	64.08	64.34
C (832×480)	47.85	48.85	51.24	53.25	66.93	58.30	61.13
D (416×240)	42.99	43.50	39.53	49.63	63.40	55.55	56.13
E (1280×720)	62.54	62.05	70.50	72.28	72.92	65.19	65.40
A*,B,C,D,E **	**53.96**	**53.89**	**58.72**	**61.85**	**69.34**	**61.06**	**61.74**
A ,B,C,D,E **	**56.08**	**55.5**	**60.35**		**70.16**	**62.02**	

Table 8. 6. Figure of Merit (FoM) metric for the proposed methods, as compared with state-of-the-art methods chosen for the detailed comparison.

Sequence class	FoM						
	CU partitionig only				CU blocks and PU division		
	Basic Architecture		[Fe21]	[Xu18A]	Extended Architecture		[Hu21B]
	AlgIdx: α = 0	AlgPrb: β = 0.45			AlgPrb: β = 0	AlgPrb: β = 0.45	
A* (2560×1600)	3.92	2.63	3.18	3.73	5.61	3.07	2.93
A (2560×1600)	3.02	1.87	2.76		3.94	2.08	
B (1920×1080)	3.41	2.17	2.84	3.65	4.73	2.56	2.35
C (832×480)	3.28	2.35	2.97	3.57	5.48	2.98	3.63
D (416×240)	2.16	1.59	1.72	2.36	4.23	2.23	3.24
E (1280×720)	4.69	3.14	4.04	4.79	6.60	4.39	3.46
A*,B,C,D,E **	3.47	**2.34**	2.98	3.64	5.22	**2.96**	3.08
A ,B,C,D,E **	3.32	**2.22**	2.92		4.90	**2.76**	

* – only part of class A: PeopleOnStreet and Traffic video test sequences

** – mean over all sequences in enlisted classes

– results not provided by the authors

The conclusions will be the same for Basic Architecture with hard-decisive variant AlgPrb ($\beta = 0$), as the results were very similar to AlgIdx. However, this changes when the soft-decisive variant of AlgPrb ($\beta = 0.45$) is considered. As observed in Subsection 7.3.3, the soft-decisive variant of the decision algorithm in the Basic Approach does not impact the TS. Therefore, the $BD\text{-}RATE$ increases with the control parameter. For AlgPrb with $\beta = 0.45$, the average $BD\text{-}RATE$ changes to 1.23%, which

is better by 0.5 p.p. than method [Fe21]. The Basic Architecture with AlgPrb ($\beta = 0.45$) is better for almost all resolution classes.

The improvement in $BD\text{-}RATE$ changes the ETvsCE trade-off. The Basic Architecture with AlgIdx offer a better ETvsCE trade-off than the method [Xu18A] (0.17) but a worse ETvsCE trade-off than [Fe21] (by 0.49) in terms of FoM. The change of the decision algorithm to AlgPrb with $\beta = 0.45$ offers a 0.64 better trade-off than the method [Fe21]. This means that the proposed method outperforms solutions found in the literature in terms of the ETvsCE trade-off.

Considering methods that jointly estimate CU block sizes and PU division, Extended Architecture with the hard-decisive variant of AlgPrb offer almost 8 p.p. faster encoding than other methods in this comparison [Xu18A, Hu21B], but with almost two times worse $BD\text{-}RATE$ and $BD\text{-}PSNR$. The FoM values are much bigger, but as discussed in Section 3.4, the relatively big difference in both $BD\text{-}RATE$ and TS makes comparison using this metric unsuitable in this case.

For Extended Architecture with the soft-decisive variant of AlgPrb ($\beta = 0.45$), the TS averaged over all test sequences is nearly the same as results reported for methods from the literature. Compared to methods [Xu18A, Hu21B], the proposed partitioning algorithm achieves at least 0.1 p.p. $BD\text{-}RATE$ and 0.01 dB $BD\text{-}PSNR$ better results. The proposed algorithm is superior for resolution classes D and C. As discussed in Section 5.3 and 6.3, smaller CU blocks are used more frequently for smaller resolutions. The conclusion is that the proposed method is better in indicating smaller blocks. Observations for TS and FoM for resolution classes are very similar.

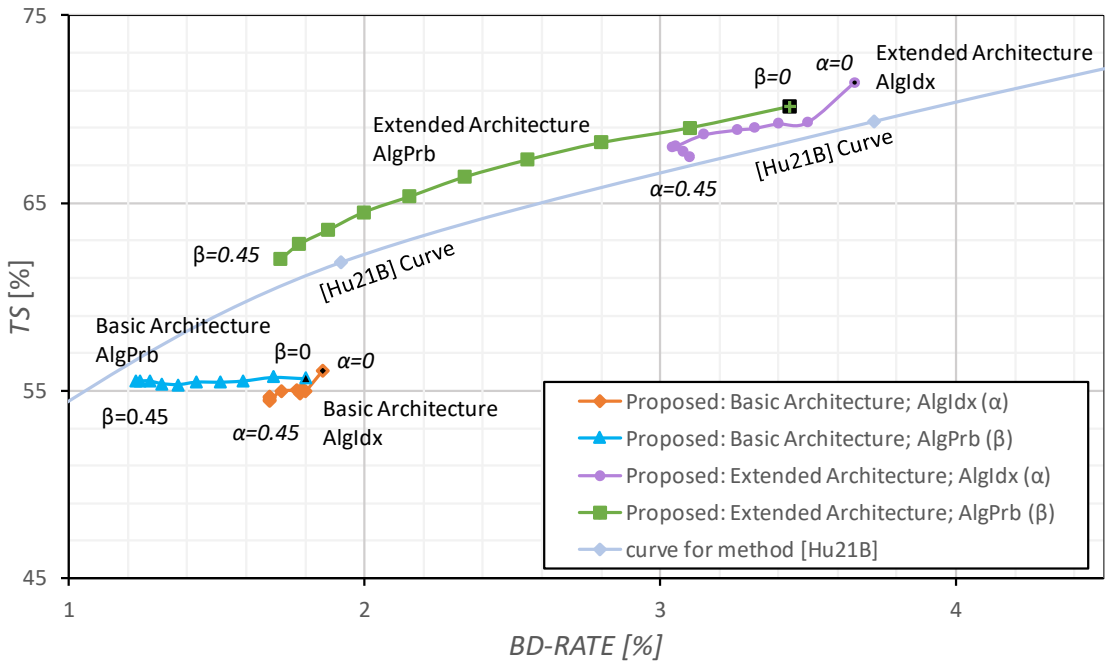

Figure 8. 3. Results for soft-decisive variants of the proposed decision algorithm and curve for method [Hu21B] used as the reference for calculation of proposed metrics $\Delta BD\text{-}RATE|_{TS}$ and $\Delta TS|_{BD\text{-}RATE}$. Presented results are relative to HM.

The FoM value averaged over all sequences is 0.1 better for the proposed method compared to [Hu21B]. However, both methods offer control over the ETvsCE trade-off. To properly compare these methods, the comparison should be taken with the operating point of the same $BD\text{-}RATE$ or TS. Still, the FoM value will significantly depend on the chosen operating points and may be

misleading. Therefore, the author's metrics $\Delta BD\text{-}RATE|_{TS}$ and $\Delta TS|_{BD\text{-}RATE}$, described in Section 3.4, were used to compare the proposed method with [Hu21B]. To demonstrate the proposed metrics, all proposed soft-decisive methods were evaluated. For metrics calculation, points were always selected in such a way as to match the ranges of $BD\text{-}RATE$. The results are presented in Table 8.7. Additionally, results from Tables 8.3. and 8.5 for proposed methods and method [Hu21B] are represented graphically in Figure 8.3. Presented results are relative to HM.

As shown in Figure 8.3, both algorithms that use Basic Architecture are under the curve for method [Hu21B]. The results of $\Delta BD\text{-}RATE|_{TS}$ and $\Delta TS|_{BD\text{-}RATE}$ reflects well the observations of the chart. For the Basic Architecture and AlgIdx, the metrics indicate that the method is worse by ~1 p.p. in terms of $BD\text{-}RATE$ and ~6 p.p. in terms of TS, which corresponds to the figure. The almost flat line for the Basic Architecture with AlgPrb is closer to the [Hu21B] curve, and the $\Delta BD\text{-}RATE|_{TS}$ and $\Delta TS|_{BD\text{-}RATE}$ are smaller.

Both curves for the Extended Architecture are above the curve for [Hu21B], and the proposed metrics values are negative. According to proposed metrics, the proposed algorithm with AlgPrb provides ~0.5 p.p. better $BD\text{-}RATE$ for the same TS and ~2 p.p. better TS for the same $BD\text{-}RATE$. The curve for AlgIdx is closer to the [H21B] curve, the results are accordingly worse.

Table 8. 7. Results of the proposed methods reported through application of the proposed metrics of the proposed metrics: $\Delta BD\text{-}RATE|_{TS}$ and $\Delta TS|_{BD\text{-}RATE}$ as compared to the reference method [Hu21B]

ANN Architecture	Decisive algorithm variant	$\Delta BD\text{-}RATE\|_{TS}$ [p.p.]	$\Delta TS\|_{BD\text{-}RATE}$ [p.p.]
Basic Architecture	Alg-Idx	1.02	5.99
	Alg-Prb	0.36	3.65
Extended Architecture	Alg-Idx	-0.30	-1.42
	Alg-Prb	-0.53	-1.95

From this comparison, two conclusions can be drawn:

- **The proposed metrics for assessment methods with control over the ETvsCE trade-off are viable tools for comparison of methods.** As demonstrated above, the values of the metrics reliably reflect the relation between compared methods.
- **The analysis above shows the superiority of the proposed methods (Extended Architecture with AlgPrb) over the best-performing method found in the literature [Hu21B]** for the same $BD\text{-}RATE$. It should be noted that the proposed metric offers a narrower range of control. Therefore, **the control is much simpler, with just a single parameter**, compared to the estimation of multiple thresholds with a heuristic model [Hu21B].

9 Exploration experiments

9.1 The goal of the exploration experiments

In the preceding part of the dissertation, the focus was on the main achievements: the Basic (Chapter 5) and Extended (Chapter 6) Architectures, and decision algorithms (Chapter 7): AlgPrb and AlgIdx. These achievements are the result of extensive research and experiments. For the sake of clarity, only the most successful research paths, which directly led to these main achievements, have been presented (Section 5.4 and 6.4).

In this chapter the remaining body of research is presented, which consist of multiple exploration experiments. Some of them were not that successful, e.g. in term of increasing the performance of proposed methods, but they still hold scientific value. Among others, in this chapter are explored: the impact of contextual data on the encoding, the use of contextuality in the partitioning algorithm.

The chapter ends by addressing considerations related to global optimization of partitioning patterns with the usage of Hidden Markov Model (HMM) [Ba66] (Section 9.8).

9.2 Broadened performance evaluation of the proposed partitioning algorithms

The Basic and Extended Architectures (Section 5.1 and 6.1) are designed for Intra mode. The respective models were evaluated on the default test content (test sequences – Section 3.5) defined by the MPEG group for HEVC video encoding in so-called Common Test Conditions (CTC) [CTCHEVC]. Notably, there is a growing demand for the transmission of 4k resolution video and content captured from computer screens (Screen Content Coding) [For1]. These trends were reflected in the CTC for subsequent standards, which were developed in parallel with the doctoral dissertation. Therefore, such types of content are considered in this section.

The selected additional content are sequences in 4k resolution (3840×2160), indicated by CTC for VVC [CTCVVC] (classes A1 and A2), and class F from CTC for HEVC [CTCHEVC] (for evaluation of Screen Content Coding profile). Details of these sequences are presented in Section 3.5 (Table 3.2).

The evaluation was performed according to description in Section 3.1, and for algorithm configurations: Basic Architecture with AlgPrb (β = 0), Basic Architecture with AlgPrb (β = 0.45), Extended Architecture with AlgPrb (β = 0.0) and Extended Architecture with AlgPrb (β = 0.45). The β = 0 means hard-decisive variant of the AlgPrb (Subsection 7.2.2). The β = 0.45 means soft-decisive variant of the AlgPrb (Subsection 7.3.2.2) with highest value of β parameter (most probable soft-decisive decision).

9.2.1 Evaluation on 4k resolution sequences

The results of the evaluation are presented in Table 9.1. The presented results are relative to HM. For comparison, averaged results for class A [CTCHEVC] (the closest resolutions) and averaged results over all test sequences (Section 3.5) are presented.

For both the Basic and Extended Architecture the results for $BD\text{-}RATE$ are similar. For hard-decisive variant of the AlgPrb (β = 0), the ~1 p.p. worse results are observed compared to class A. The use of soft-decisive variant (β = 0.45) significantly improves the results for classes A1 and A2 (to 2.05 % of $BD\text{-}RATE$). However, the results are 0.5 p.p. worse than for class A.

Considering the encoding Time reduction (Time Savings: TS, Formula 3.3), the results for Extended are slightly worse (~2.5 p.p. compared to class A). For the soft-decisive variant of AlgPrb, as the β parameter increases, the TS is decreasing similarly as for class A. This is not the case for the Basic Architecture. Firstly, for the hard-decisive variant of AlgPrb the TS is significantly better for A1 and A2

– up to 5 p.p.. Secondly, for the soft-decisive variant, the TS drops by at least 1.5 p.p. This means that the phenomenon observed for soft-decisive AlgPrb (Section 7.3.3), where $BD\text{-}RATE$ improves as the TS remains the same, does not occur.

Table 9. 1. Evaluation of proposed partitioning algorithms for additional video content (classes A1 and A2). The presented results are relative to HM. The All class refers to the mean over results for all test sequences.

Sequence class	$BD\text{-}RATE$ [%]				TS [%]			
	Basic Architecture		Extended Architecture		Basic Architecture		Extended Architecture	
	AlgPrb $\beta = 0$	AlgPrb $\beta = 0.45$	AlgPrb $\beta = 0$	AlgPrb $\beta = 0.45$	AlgPrb $\beta = 0$	AlgPrb $\beta = 0.45$	AlgPrb $\beta = 0$	AlgPrb $\beta = 0.45$
A1 (3840×2160)	2.95	1.74	3.77	2.05	68.76	65.31	72.41	65.62
A2 (3840×2160)	2.71	1.81	3.83	2.05	65.60	64.16	72.09	64.92
A (2560×1600)	1.94	1.19	2.95	1.40	61.45	63.72	74.92	67.26
All	1.80	1.23	3.44	1.71	54.20	55.5	69.22	62.02

It should be recalled that according to the training assessment of the models in Sections 5.2 and 6.2, Basic Architecture was assessed as trained better. The reason for this is the fit of the Basic Architecture to default content. As the type of content is different, the model prediction is less certain. The amount of computations for soft decisions is too big to be compensated by the reduction of computations that comes from the avoidance of the smallest block, described in Subsection 7.2.2.

9.2.2 Evaluation on Screen Content Coding sequences

The results of the evaluation are presented in Table 9.2. The presented results are relative to HM. For comparison, averaged results for classes C, E [CTCHEVC] (closest resolutions) and averaged results over all test sequences (Section 3.5) are presented.

Table 9. 2. Evaluation of proposed partitioning algorithms for additional video content (class HEVC F). The presented results are relative to HM. The All class refers to the mean over results for all test sequences.

Sequence class	$BD\text{-}RATE$ [%]				TS [%]			
	Basic Architecture		Extended Architecture		Basic Architecture		Extended Architecture	
	AlgPrb $\beta = 0$	AlgPrb $\beta = 0.45$	AlgPrb $\beta = 0$	AlgPrb $\beta = 0.45$	AlgPrb $\beta = 0$	AlgPrb $\beta = 0.45$	AlgPrb $\beta = 0$	AlgPrb $\beta = 0.45$
HEVC F	1.57	1.62	4.08	3.65	47.12	48.68	60.62	56.58
C	1.55	1.15	3.67	1.74	46.17	48.85	66.93	58.30
E	2.77	1.95	4.81	2.86	62.06	62.05	72.92	65.19
All	1.80	1.23	3.44	1.71	54.20	55.5	69.22	62.02

For the hard-decisive variant of AlgPrb, the results for class F are similar to those for class C. Given the Basic Architecture, the differences are negligible for both $BD\text{-}RATE$ and encoding time reduction (Time Savings: TS, Formula 3.3). Surprisingly, for $\beta = 0.45$, both $BD\text{-}RATE$ and TS increase, so the saddle effect is observed. For Extended Architecture with AlgPrb ($\beta = 0$), a difference in $BD\text{-}RATE$ between classes C and F is 0.4 p.p.. The soft-decisive variant does not improve the $BD\text{-}RATE$ by much, while the decrease in TS is observed.

The abovementioned observations can be explained by the fact that **the content of the HEVC F class is significantly different than the default content**. The video capture of the computer screens often contains plain regions and occasionally sharp edges. Therefore, the partitioning algorithm must be much more accurate in indicating block size. This explains the results observed for Extended Architecture. The simultaneous increase of $BD\text{-}RATE$ and TS for Basic Architecture with AlgPrb (when

increasing β) indicates unfamiliarity of the model with sequence content. Thus, the ANN output would be very uncertain (multiple depth levels with almost the same probability). When the small block size should be chosen, the soft-decisive algorithm may indicate a bigger, and less computably complex one. As shown in Subsection 7.3.1, in such a situation, it is more beneficial, in terms of TS, to check two bigger block sizes instead of just considering the smallest block sizes. This explains the observed phenomenon.

9.2.3 Conclusions for broadened performance evaluation

Results for additional content are similar to results for classes with corresponding (or similar) resolutions in test sequences. A small deterioration of the results is observed. This is because models in both Basic and Extended Architectures are highly tailored to the dataset content. The test sequences were similar to images in the training dataset, unlike the sequences in additional content. This means that the proposed models are highly sensitive to input data. For practical use, the model should be trained with a dataset carefully tailored to the problem, or a dataset should be appropriately generalized for broad content. Nevertheless, the performance deterioration is no more than 1 p.p. in terms of $BD\text{-}RATE$ and 3 p.p. in terms of TS.

9.3 Architecture modification – serialization of the feature map processing

The ANN architecture for Basic Approach (Figure 5.1, Section 5.1) is low complex. The Basic Architecture (Section 5.1) proved to be in pair with state-of-the-art in terms of $BD\text{-}RATE$, as shown in Chapter 8. As presented in Section 5.4, adjusting the hyperparameters did not result in a model that performs better with similar and smaller complexity. In this section is described another approach to reduce then computational complexity of the ANN architecture.

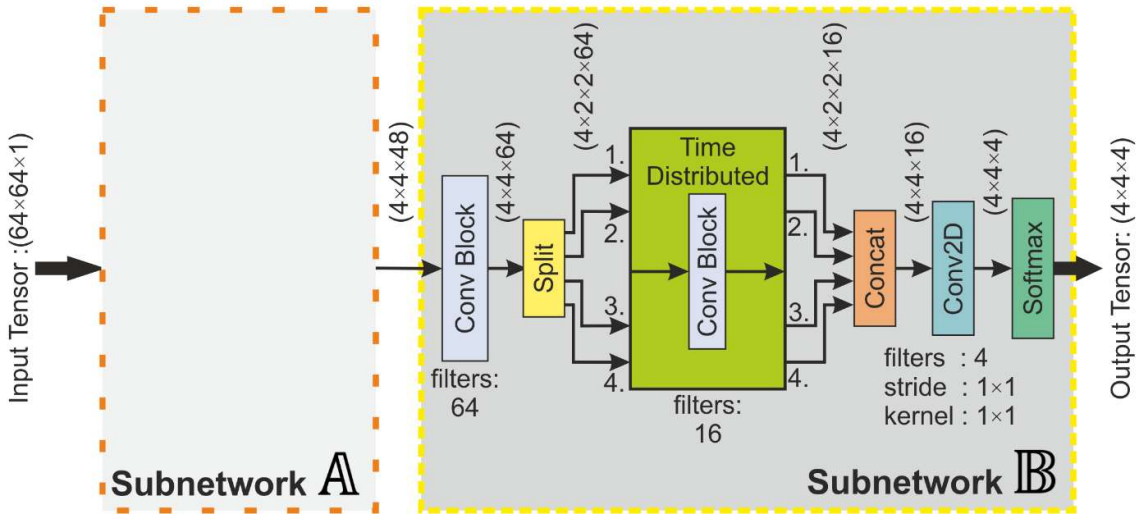

Figure 9. 1. The ANN architecture for Basic Approach architecture in Serial variant. The "Conv Block" refers to block of layers, presented in Figure 5.2 (Section 5.1).

As presented in Section 5.1, the ANN architecture for Basic Approach is composed of two subnetworks (Fig 5.1). The second one is referred to as Subnetwork 𝔹. This subnetwork recalls the quaternary tree to mimic the quaternary tree structure used in HEVC. The characteristic feature is that after the first Conv Block (Figure 5.2, Section 5.1), the feature maps are split into four parts and processed by separate Conv Blocks. As described in Section 5.1, those Conv Blocks do not share weights, as the split decision may be taken differently in subareas of the CTU. However, such an approach reduces gradient flow in the backpropagation phase of training. So, the weights of the

convolution filters in these Conv Blocks are updated using only a quarter of the gradient from output due to the configuration of further layers.

The gradient flow in the ANN architecture for Basic Approach may restrict the model training. Sharing weight among all convolutions will use the whole gradient information in training, resulting in more generalized features extracted by convolution filters. Additionally, a better gradient flow may positively impact feature extraction in other layers of the model. Such modification may improve the evaluation results of a model or make it possible to train similarly performing but less complex models.

To train such a network, the Subnetwork $\mathbb{B}$ is modified by substituting four parallel Conv Block by just one, which processes feature maps one by one. The "Time distributed" module from the TensorFlow framework [TENSORFLOW] was used for implementation. Thus, the quaternary-tree-inspired feature map processing is preserved. Further in this section, the architecture that uses the "Time distributed" module is referred to as the **Serial variant**, while the architecture with four separate Conv Blocks is referred to as the **Parallel variant**. The ANN architecture for Basic Approach in the Serial variant is presented in Figure 9.1.

To check the impact of the proposed modification, Serial and Parallel variants of ANN architecture for Basic Approach were trained and evaluated in four configurations of the model hyperparameters:

S1. No change in the number of filters (Parallel variant is the Basic Architecture).
S2. The number of filters in the second layer of Subnetwork $\mathbb{B}$ reduced by half.
S3. The number of filters in the first and second layers of Subnetwork $\mathbb{B}$ reduced by half.
S4. The number of filters in the first and second layers of Subnetwork $\mathbb{B}$ and the number of filters in all layers in Subnetwork $\mathbb{A}$ reduced by half.

For each hyperparameter configuration a set of models for *QP* values {22, 27, 32, 37} was trained, according to the description in Section 4.6. Evaluation results for the above model configurations are presented in Table 9.3. The evaluation of the models was performed according to description in Section 3.1. The presented results are relative to HM. The hard-decisive variant of AlgPrb (Subsection 7.2.2) was used to evaluate models. It should be mentioned that the training process was repeated several times to confirm the convergence of the models.

Table 9. 3. Evaluation results for Serial and Parallel variants of ANN architecture for Basic Approach in proposed configurations of hyperparameters. The presented results are relative to HM.

Index	Number of filters in layers: Subnetwork $\mathbb{A}$	Number of filters in layers: Subnetwork $\mathbb{B}$	Subnetwork $\mathbb{B}$ variant	*BD-RATE* [%]	*BD-PSNR* [dB]
S1	12 ; 24 ; 36 ; 48	64-16-4	Parallel	1.80	-0.090
			Serial	**1.80**	**-0.089**
S2	12 ; 24 ; 36 ; 48	64-8-4	Parallel	1.85	-0.091
			Serial	**1.83**	**-0.090**
S3	12 ; 24 ; 36 ; 48	32-8-4	Parallel	1.83	-0.091
			Serial	**1.81**	**-0.089**
S4	6 ; 12 ; 28 ; 24	32-8-4	Parallel	2.02	-0.099
			Serial	**1.98**	**-0.098**

For all sets of hyperparameters, the Serial variants of the architecture achieved better or the same evaluation results. Reduction of filter number in Subnetwork $\mathbb{A}$ significantly worsens evaluation results of the model, up to 0.2 p.p.. However, the change in the number of filters in Subnetwork $\mathbb{B}$ increases the *BD-RATE* and *BD-PSNR* only by a small margin. When the number of filters in the first and second layers of the Subnetwork $\mathbb{B}$ reduced by half (S3), the Serial variant of the architecture achieves almost the same evaluation results as the Basic Architecture. Set of ANN models with hyperparameter configuration S3 and trained for *QP* values {22, 27, 32, 37} is referred to as the **Modified Basic Architecture**. Therefore, the proposed serialization allows training a less complex model without

significant evaluation results change. The complexity comparison, in terms of Multiply and Accumulate (MAC) operation count, for the Modified Basic Architecture and Basic Architecture is presented in Table 9.4.

It can be noticed that the number of weights is almost halved in the Modified Basic Architecture. However, the MAC operation count is reduced only by ~340k operations. That is because the Subnetwork $\mathbb{A}$ is not modified, and most computations are performed there due to the biggest dimensions of feature maps. One should mention that the Modified Basic Architecture adds a delay in the processing of feature maps due to the serialization of processing. Nonetheless, the weights from the Serial variant of the architecture can be easily transferred to the matching Parallel variant. In this case, convolutional filters in parallel Conv Blocks will have the same weights, but the processing could be parallelized.

Table 9. 4. Complexity comparison of the Modified Basic Architecture and Basic Architecture.

	Modified Basic Architecture (Serial)	**Basic Architecture (Parallel)**
Weights number	42 832	91 600
MAC operation count [M]	6.43	6.76

A detailed comparison in terms of $BD\text{-}RATE$ and encoding time reduction (Time Savings: TS, Formula 3.3), of Modified Basic Architecture and Basic Architecture, is presented in Table 9.5. The evaluation of the models was performed according to description in Section 3.1. The presented results are relative to HM. The table presents the averaged results for resolution classes and the mean over all test sequences. Results of TS are extended with the Modified Basic Architecture in Parallel variant (the $BD\text{-}RATE$ results for the Parallel variant are the same as for the Serial variant).

Table 9. 5. Detailed results of the evaluation of Modified Basic Architecture and Basic Architecture in Modified HM. The presented results are relative to HM. The All class refers to the mean over results for all test sequences.

JCT-VC class	***BD-RATE* [%]**		***TS* [%]**		
	Modified Basic Architecture (Serial)	**Basic Architecture (Parallel)**	**Modified Basic Architecture (Serial)**	**Modified Basic Architecture (Parallel)**	**Basic Architecture (Parallel)**
A	1.95	1.94	60.62	61.87	61.45
B	2.01	2.01	59.27	60.46	60.38
C	1.56	1.55	45.44	46.04	46.17
D	0.92	0.93	39.30	39.68	41.34
E	2.84	2.78	60.80	62.19	62.06
All	**1.81**	**1.80**	**53.01**	**53.96**	**54.20**

In terms of $BD\text{-}RATE$, the performance of the Modified Basic Architecture is in pair with the Basic Architecture. For most resolution classes, the results are marginally different (up to 0.01 p.p.), with the biggest difference for class E - 0.06 p.p.. However, in terms of TS the Modified Basic Architecture in the Serial variant of the architecture performs ~1.2 p.p. worse than the Basic Architecture. However, when weights are transferred to the matching Parallel variant of the architecture, the difference shrinks to 0.24 p.p., which is within the margin of error. The significant difference in TS, compared to Basic Architecture, is observed only in class D.

The smaller MAC operation count did not measurably impact the encoding times. Additionally, it was observed that the serialization in the model observably decreased TS. Therefore, the proposed serialization allowed the training of smaller models – in terms of the number of weights and complexity – without significant degradation in performance. Transferring the weights from the Serial to the matching Parallel variant of the architecture solves the issue with TS decrease. Therefore, the proposed

Modified Basic Architecture is less complex and much smaller, which is beneficial, e.g., in hardware implementation. The results presented in this section were published in the paper [Lo23].

9.4 Impact of CTUs spanning beyond the boundaries of the picture

In section 2.2.3, it was shown that HEVC defines specific partitioning restrictions when the resolution of the sequence is not divisible by CTU size. The missing samples are filled in a very defined way and for the CTU spanning beyond the boundaries of the image, the partitioning forces putting these samples in separate blocks. As discussed in Section 4.2, this was why images in the training dataset were purposely cropped to avoid such blocks. Such CTUs may negatively impact the model training. However, encoding sequences with resolutions non-divisible by CTU size is quite common – such a case is included in test sequences, e.g., in the resolution class B (1920×1080), where image height is not divisible by 64.

To check the impact of CTUs spanning beyond the image boundaries on the model training, the following additional training datasets were created using preprocessed data:

- Images from DIV2k are first cropped to resolutions divisible by 8.
- Images from DIV2k are first cropped to resolutions divisible by 2.

The unavailable samples were filled the same way as in HEVC (Subsection 2.2.3). Datasets for ANN training were created according to the description in Subsection 3.6.2. Then, ANN architecture for Basic Approach (Section 5.1) was trained using these additional training datasets. For each created dataset a set of models for *QP* values {22, 27, 32, 37} was trained, according to the description in Section 4.6. The evaluation of the models was performed according to description in Section 3.1. The evaluation results in Modified HM for these models are presented in Table 9.6. The results of Basic Architecture (Section 5.3) are included for comparison. The presented results are relative to HM and represent the mean over the results for all test sequences.

Table 9. 6. Evaluation results for ANN architecture for Basic Approach (Section 5.1) trained using datasets created from differently cropped images (DIV2k). The presented results are relative to HM and represent the mean over the results for all test sequences.

Cropping of DIV2k images used for training dataset	***BD-RATE* [%]**	***BD-PSNR* [dB]**
Cropping to resolutions divisible by 8	1.85	-0.091
Cropping to resolutions divisible by 2	1.82	-0.089
Cropping to resolutions divisible by CTU size (Basic Architecture)	**1.80**	**-0.089**

The difference between the used datasets is very small, at a maximum of 0.05 in terms of $BD\text{-}RATE$ and 0.02 dB in terms $BD\text{-}PSNR$. Nonetheless, the best evaluation results are observed for a dataset created with images cropped to resolutions divisible by CTU size, which was used to train the Basic Architecture. The conclusion is that the impact of CTUs spanning the picture boundaries is negative, but the differences in the achieved $Accuracy$ (Formula 4.8) of the trained models are modest, yet present.

9.5 Contextuality in the RD Optimization

As described in Section 2.3.2, the partitioning decisions in HM [HM] are made locally for currently processed samples. Consecutive CTUs are processed sequentially. The partitioning algorithm in HM finds suboptimal decisions locally for a given CTU. However, these decisions are influenced by multiple context information, e.g.:

- Samples from neighboring blocks that are used for prediction.
- Encoder internal tools like Most Portable Mode [HM] (reduces the pool of prediction modes to check in the current block based on prediction modes chosen in previous blocks).

- The context of the entropic encoder (CABAC) used in HEVC.

Multiple sets of decisions are considered for currently processed CTU, where differences in compression efficiency between two such sets could be minimal. This means that even a small change in video sequence or image may impact the decisions, including the partitioning. The idea is to consider the contextuality of the partition process within the partitioning algorithm.

The goal of the experiment presented in this section is to examine the impact of small changes in the image on partitioning decisions. The small change in the image is defined as a minor change in a subset of image samples. Such small changes do not impact the visual impression of the image. Such small changes are, e.g., different realization of the noise in the image or minor changes in single samples.

The experiment is set up as follows. Firstly, a small modification is applied to images from the DIV2k dataset. Next, images from the modified DIV2k dataset are encoded with HM. Then, the partitioning patterns are extracted. Lastly, the partitioning patterns for modified and non-modified images are compared in corresponding CTUs. Cases of different partitioning patterns are counted. Then, the percentage of non-identical partitioning patterns is calculated for the whole dataset.

The above experiment procedure was performed for the following image modifications:

- Adding random noise to the image, with Gaussian distribution quantized to integer sample values. The noise for each sample is calculated independently. The intensity of this random noise is described by Root Mean Square (RMS), defined with formula 9.1. In the described experiment, the following values of noise RMS were tested: 0.01, 0.05, 0.10, 0.50, 1.00. Those noise RMS values were chosen to ensure the number of modified samples is relatively small (effectively several modified samples).
- Change the value of one, precisely indicated image sample by adding 1. Three different locations of modified samples were (Figure 9.2) :
    - Top-left sample of the image (referred to as Top).
    - Middle-left sample of the image (referred to as Middle).
    - Bottom-right sample of the image (referred to as Bottom).

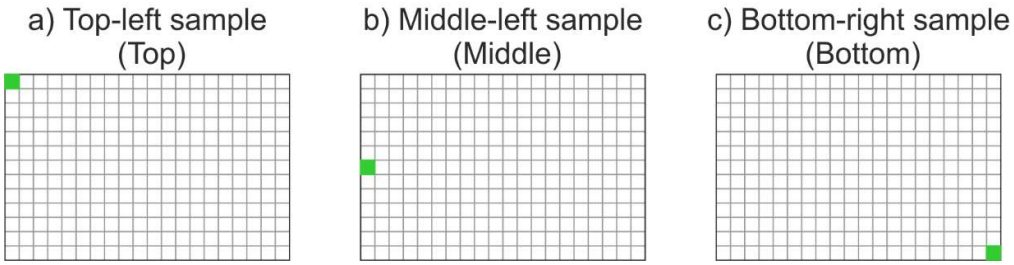

Figure 9. 2. Location of the modified image sample (green): a) Top, b) Middle, c) Bottom.

The results of the experiment are presented in Table 9.7. It should be noted that the applied modifications of images did not change noticeably the bitrate or quality of the encoded image. For the images with added random noise, it can be observed that even the smallest RMS of the noise significantly impacts the partitioning decisions in HM RD Optimization. For the noise RMS of 0.01 (effectively several modified samples in the image) 63.43% of partitioning patterns are changed. As expected, the percentage is rising with the increase of noise RMS, although the relation is not monotonic. For example, a higher percentage was observed for noise RMS 0.05 compared to noise RMS 0.10. This means that the result highly depends on the exact noise realization applied to the image.

The dependency of noise realization on changing the partitioning pattern is confirmed by the results for changing the value of precisely defined image points. The small change in sample value of the first

processed CTU changes almost 11% of all CTUs. Such a small change caused a snowball effect on the decisions of RD Optimization. The percentage of changed CTU patterns for the Middle case is over half that of the Top case. Thus, sudden, even small change, can significantly impact the context in the encoder. The result for the Bottom case is that a sample modification, even in the last encoded CTU block, may change the partitioning.

Table 9. 7. Percentage of changed partitioning patterns estimated by HM, for modified images from DIV2k.

Modification type	Modification Description	Percentage of CTUs with changed partitioning patterns
Random pixels ±1 with set RMS of the noise	0.01	63.43%
	0.05	71.73%
	0.10	71.11%
	0.50	72.53%
	1.00	73.23%
Single pixel ±1 change	Top	10.80%
	Middle	6.70%
	Bottom	0.01%

The results of the abovementioned experiment demonstrate that RD Optimization is a very sensitive process to even small changes. The partitioning algorithm should consider this effect as noise is present in natural content sequences. **As the encoding context changed by modifying samples, the same effect would have a choice of different partitioning patterns.** Thus, only the ideally reproduced sequence of consecutive partitioning patterns results in the same bitstream. If the partitioning algorithm does not consider this effect, the encoded image bitrate or quality may deteriorate.

The above observations are significant for partitioning algorithms that use ANN. As the model may be overfit to certain noise types, the less efficient partitioning patterns will be chosen if the other type of noise is present in the image. Effectively, the encoding performance will decrease. Thus, the context of the encoder should be considered within a partitioning algorithm.

9.6 Architecture modification – the usage of contextual information

The importance of the encoding context for the partitioning pattern was discussed in Section 9.5. The context is not utilized in ANN architecture for either Basic or Extended Approaches (Sections 5.1. and 6.1). In this section, modifications of the ANN architecture for Basic Approach (Figure 5.1, Section 5.1) are explored to include contextual data in ANN processing. Thus, the impact of contextual data on ANN model performance is examined, concerning training accuracy and results of the evaluation in the encoder.

The following contextual information was selected for processing by ANN:

- Adjacent samples from neighboring blocks. In the HM these samples may be used for prediction in currently processed CTU.
- Adjacent Division Matrices from neighboring blocks. These Division Matrices deliver hints about the shape of objects on currently processed CTU.
- Division Matrices from *P* previous blocks. This data delivers information on the current context of CABAC connected to partitioning patterns.

Figure 9.3 presents the arrangement of selected contextual data concerning the current CTU block.

In this section, the ANN architecture for Basic Approach (Section 5.1) was used as the base for modifications. Therefore, additional contextual data are adjusted accordingly before being processed. The ANN is designed to process data in the 3D tensor (Section 4.4), where the values are normalized to

the range $\langle 0; 1 \rangle$. Further description of data processing takes into account the use of the ANN architecture for Basic Approach.

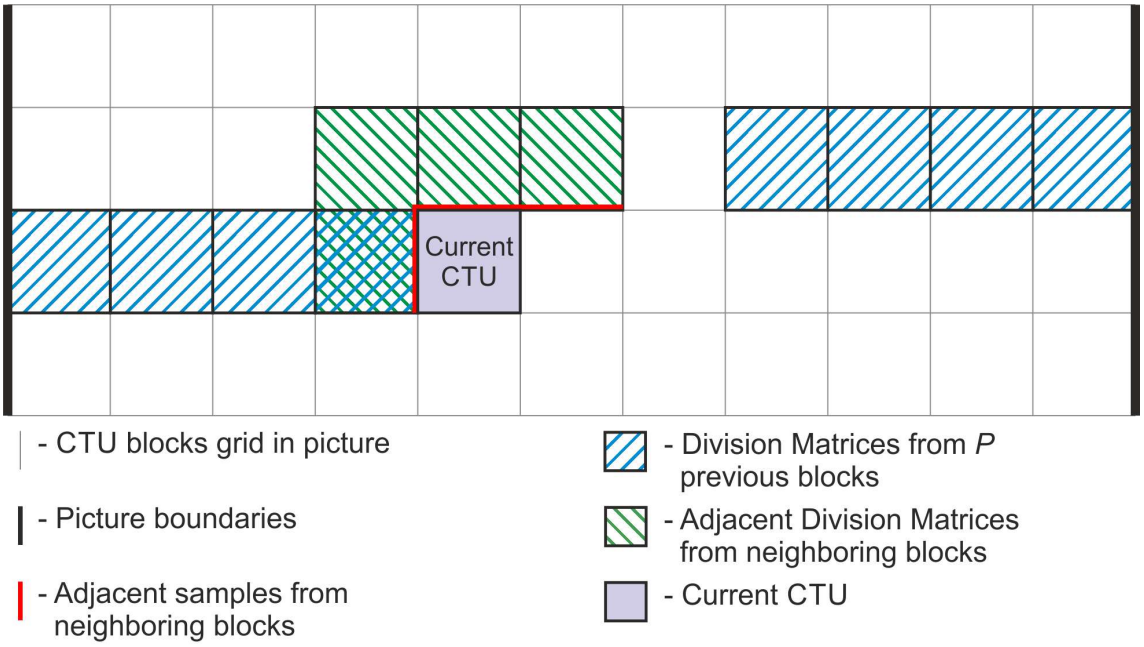

Figure 9. 3. The arrangement of selected context data concerning the currently processed CTU block.

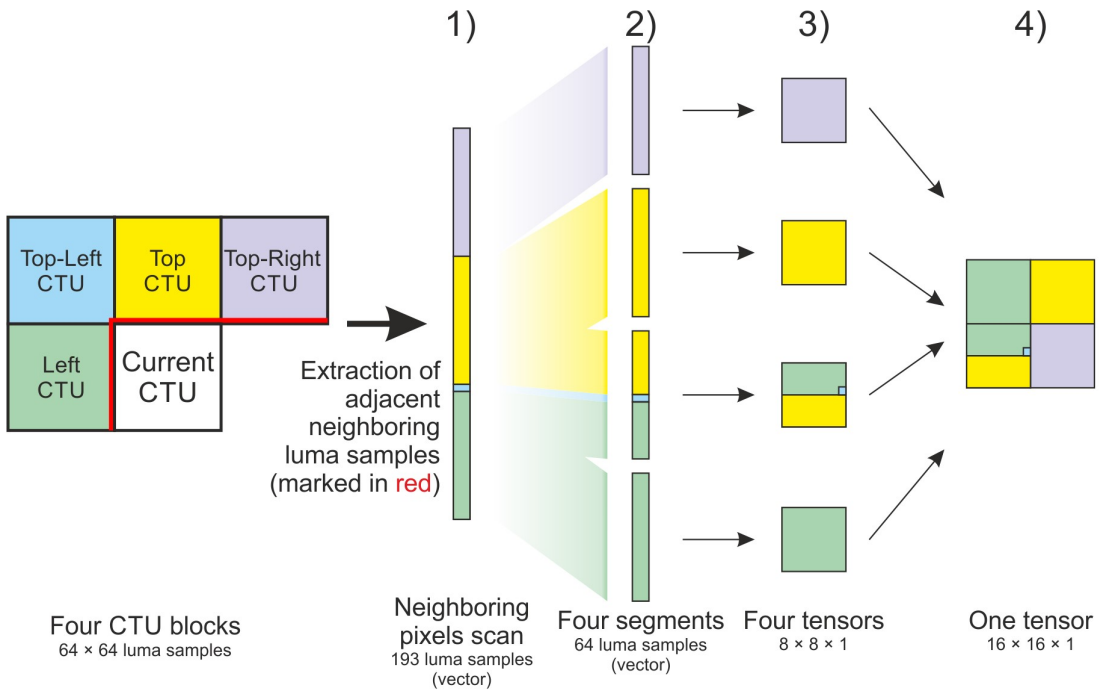

Figure 9. 4. Procedure for creating tensor with adjacent neighboring luma samples. 1) Luma samples are organized in vector of size 193. 2) From the vector of samples, four vectors of size 64 samples are extracted. 3) Samples from extracted vectors are converted into 2D matrices of size 8×8. These matrices are further concatenated into a single tensor of size 16×16×1.

In the case of proposed contextual data, two of these are Division Matrices (DMs). The DMs are first converted to DM_B format. Then, all the DMs associated with certain types of contextual data are concatenated along the channel dimension. This results in a tensor of size $4\times4\times C$, where C corresponds to the number of DMs associated with the contextual data type. Lastly, the values in the tensor are normalized to the range $\langle 0; 1\rangle$.

The last proposed contextual data is the neighboring samples. In HEVC, prediction can use samples from a single line, either from a neighboring row or a column [HEVC]. All of these samples, potentially used for prediction, can be structured into a vector of 193 samples: 64 (left) + 1 (top-left) + 64 (top) + 64 (top-right). The 3D tensor, which can be processed by ANN, is created using the procedure presented in Figure 9.4. This resulting tensor has a size of 16×16×1. Lastly, the samples in tensor are normalized to the range $\langle 0; 1\rangle$.

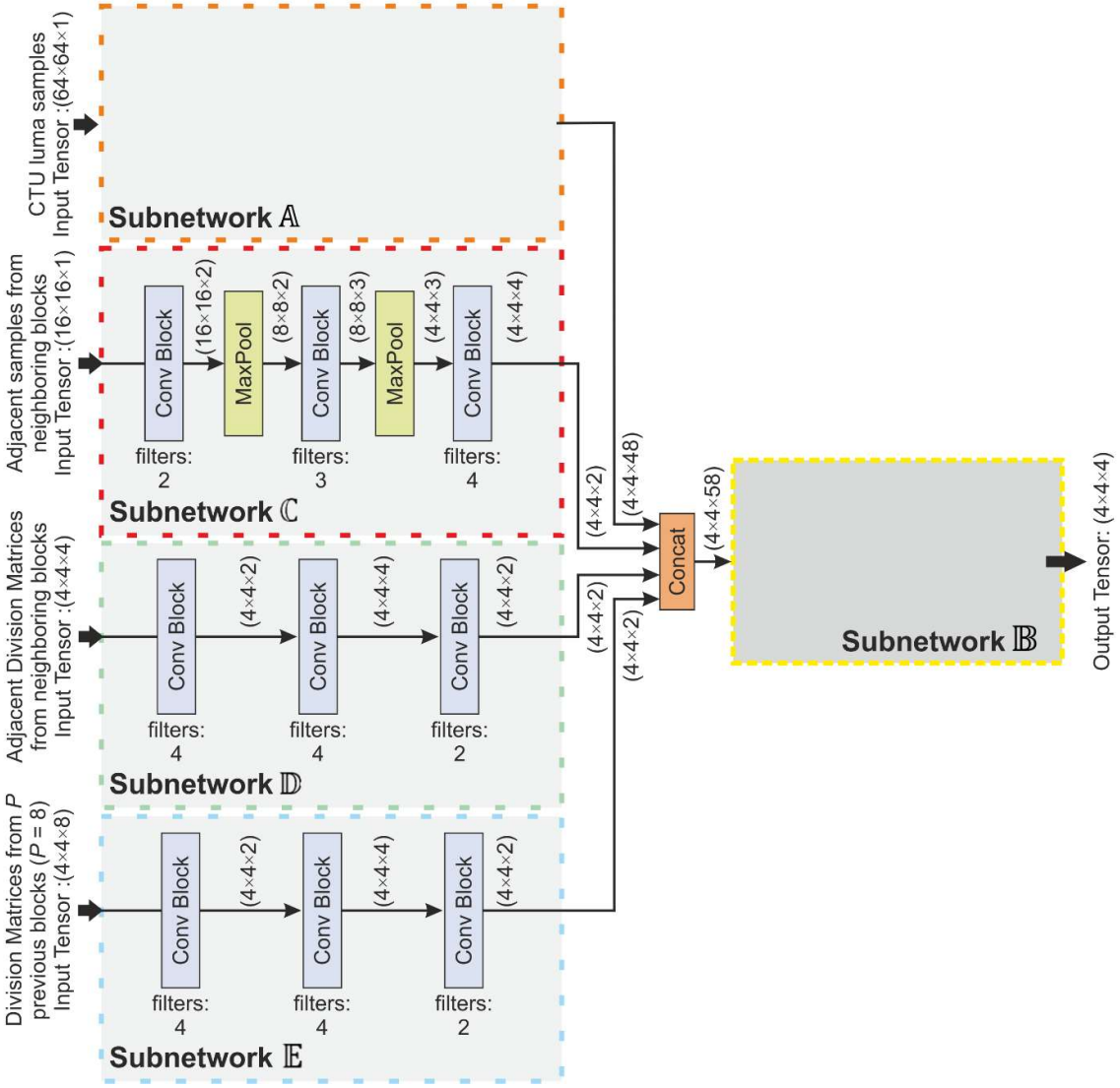

Figure 9. 5. The ANN architecture for Basic Approach (Section 5.1, Figure 5.1) with contextual data modification (used in Context-Aware Architecture). The "Conv Block" refers to block of layers, presented in Figure 5.2 (Section 5.1).

To process the additional contextual data, the ANN architecture for Basic Approach was extended by three additional subnetworks (Figure 9.5):

- Subnetwork $\mathbb{C}$ – dedicated to the processing of adjacent luma samples for neighboring blocks. The subnetwork is composed of three Conv Blocks (Figure 5.2, Section 5.1), with the number of filters: 2, 3, and 4 in consecutive Conv Blocks. Additionally, MaxPool layers are added after the first two Conv Blocks.
- Subnetwork $\mathbb{D}$ - dedicated to the processing of adjacent Division Matrices from neighboring blocks. The subnetwork is composed of three Conv Blocks (Figure 5.2), with the number of filters: 4, 4, and 2 in consecutive Conv Blocks.
- Subnetwork $\mathbb{E}$ - dedicated to the processing of Division Matrices from *P* previous blocks. The number of previous DMs is 8, as this value performed the best in experiments. The subnetwork is composed of three Conv Blocks (Figure 5.2), with the number of filters: 4, 4, and 2 in consecutive Conv Blocks.

The feature maps outputted from subnetworks $\mathbb{A}, \mathbb{C}, \mathbb{D}$, and $\mathbb{E}$ are then concatenated along the channel dimension. The concatenated tensor is then processed by the Subnetwork $\mathbb{B}$. Subnetworks $\mathbb{A}$ and $\mathbb{B}$ remained unchanged compared to the ANN architecture for Basic Approach. The ANN architecture for Basic Approach with contextual data modification is presented in Figure 9.5.

The dataset, which includes contextual information, is required to train the ANN architecture for Basic Approach with contextual data modification. Such a dataset was created with the procedure described in Section 4.2 but including context data. Separate datasets were made for each *QP* from CTC [CTCHEVC]. Further in this dissertation, a set of ANN models with architecture presented in Figure 9.5, and trained for *QP* values {22, 27, 32, 37}, is referred to as **Context-Aware Architecture**. Training of the models was performed according to the description in Section 4.6. The training results for the Context-Aware Architecture are presented in Table 9.8. Training results for Basic Architecture (Sections 5.1 and 5.3) are included for comparison.

The training results prove that the additional data delivered to the model improved the training results. The $Accuracy$ values for Context-Aware Architecture are better by at least 0.5 p.p. for each QP compared to the Basic Architecture. This applies to both the Training and the Validation Subsets. Still, the most important are the results for the evaluation on test sequences. The Modified HM (Section 4.7) was updated to support such models. The evaluation results for both Basic Architecture and Context-Aware Architecture are presented in Table 9.9. For the evaluation, AlgPrb in hard-decisive variant (Subsection 7.2.2) was used as the decision algorithm. The evaluation of the models was performed according to description in Section 3.1. The presented results are relative to HM.

Table 9. 8. Training results of Context-Aware Architecture (Basic Architecture with the context data modification). Results for Basic Architecture are presented for comparison.

QP	**Context-Aware Architecture**		**Basic Architecture**	
	Training Subset	**Validation Subset**	**Training Subset**	**Validation Subset**
22	**74.9**	**74.9**	74.2	74.4
27	**73.9**	**73.5**	73.0	72.9
32	**73.2**	**73.0**	72.5	72.4
37	**71.3**	**70.3**	70.5	69.7

Despite better $Accuracy$ values, the evaluation results for the Context-Aware Architecture are worse in terms of $BD\text{-}RATE$ and $BD\text{-}PSNR$. The exception is the resolution class A, where the results are equal. The Basic Architecture performs slightly better for classes B, C, and D. The biggest difference in evaluation results is observed for class E, where the Basic Architecture with context data modification performs worse by 0.3 p.p. in terms of $BD\text{-}RATE$ and 0.012 dB in terms of $BD\text{-}PSNR$. The reason for this may be as follows. The partitioning patterns estimated by the Context-Aware Architecture are more

accurate to HM than those of Basic Architecture. However, when the ANN indicates a partitioning pattern different from HM, the CABAC context may be disturbed to such a degree that the overall bitrate is worse despite more accurate prediction.

Table 9. 9. Evaluation results of Context-Aware Architecture (Basic Architecture with the context data modification) in Modified HM. Results for Basic Architecture are presented for comparison. `All` refers to mean over all test sequences. The presented results are relative to HM.

JCT-VC class	Context-Aware Architecture		Basic Architecture	
	***BD-RATE* [%]**	***BD-PSNR* [dB]**	***BD-RATE* [%]**	***BD-PSNR* [dB]**
A	1.94	-0.099	1.94	-0.099
B	2.03	-0.077	2.02	-0.076
C	1.60	-0.087	1.55	-0.085
D	0.98	-0.064	0.93	-0.061
E	3.07	-0.150	2.78	-0.138
All	**1.87**	**-0.092**	**1.80**	**-0.089**

As better training results were observed for the Context-Aware Architecture, additional data surely helps to model the partitioning process properly. Therefore, the evaluation results are slightly worse than those of the Basic Architecture, which suggests overfitting. The difference in the content of sequences between the training dataset and test sequences may be significant enough that the ANN is misled. Different sets of contextual modifications were applied to the ANN architecture for Basic Approach to check the significance of the contextual data used. Thus, all possible configurations of additional subnetworks were trained (set of models for *QP* values {22, 27, 32, 37} trained according to the description in Section 4.6) and evaluated (according to description in Section 3.1). The evaluation results for these models are presented in Table 9.10. The evaluation of the models was performed according to description in Section 3.1. The presented results are relative to HM.

Table 9. 10. Evaluation of models with different configurations input subnetworks. The presented results are relative to HM.

Configuration of input subnetworks	***BD-RATE* [%]**	***BD-PSNR* [dB]**
𝔸, ℂ, 𝔻, 𝔼 **(Context-Aware Architecture)**	1.87	-0.092
𝔸, ℂ	1.87	-0.092
𝔸, ℂ, 𝔻	**1.81**	**-0.089**
𝔸, ℂ, 𝔼	1.83	-0.090
𝔸, 𝔻	1.83	-0.091
𝔸, 𝔻, 𝔼	1.88	-0.092
𝔸, 𝔼	1.84	-0.091
𝔸 (Basic Architecture)	**1.80**	**-0.089**

𝔸 – subnetwork for processing of luma samples of current CTU block.
ℂ - subnetwork for processing adjacent luma samples for neighboring blocks.
𝔻 - subnetwork for processing adjacent Division Matrices from neighboring blocks.
𝔼 - subnetwork for processing Division Matrices from N previous blocks.

For the models with additional inputs, the best evaluation results were achieved when subnetworks ℂ and 𝔻 were added to the ANN architecture for Basic Approach. Therefore, none of the configurations with additional inputs did not outperformed the Basic Architecture. Unfortunately, the results do not directly indicate the impact of specific contextual data. Considering adding only one additional subnetwork, the smallest *BD-RATE* and *BD-PSNR* were achieved for adding Subnetwork 𝔻, and the biggest for adding Subnetwork ℂ. For pairs of subnetworks, the best results are observed for ℂ and 𝔻 and the worst for 𝔻, 𝔼. Recalling the conclusions for Context-Aware Architecture, the worse evaluation result implies the overfit for the training dataset. Thus, by adding a subnetwork for processing neighboring luma samples or both subnetworks for processing Division Matrices, the model overfits the

most. For other combinations, this effect seems to be suppressed. However, the differences are too small to confirm this relation.

Unfortunately, adding the contextual data failed to improve the ANN architecture for Basic Approach. The following reasons may explain this:

- The contextual dependencies are too complicated for the ANN to model them correctly. The more sophisticated ANN architecture could fix this problem. During the experiments, an attempt was made to tune the model hyperparameters, but without success.
- Additional data delivered to a model do not provide proper contextual information.
- The training dataset is too small, or the context is not diverse enough to generalize the model.
- The applied method of delivering the contextual information to the model may not be appropriate. This refers to both ANN processing and training.

The use of context information in partitioning algorithms has the potential to improve performance and is a very interesting topic for further research. Different approaches to exploit the contextuality of decisions in HEVC are presented in the next section. The results presented in this section were published in the paper [Lo21].

9.7 Ground truth augmentation by utilization of noise

In previous sections, several observations were made according to the contextuality of the partitioning decisions. Firstly, it was shown in Section 9.5 that small changes in the picture may significantly change the decisions made by RD Optimization in HM. The decision may be made between options with a very small difference in performance. Therefore, even small disturbances in the encoding context may significantly change the sequence of partitioning decisions. Secondly, including the contextual data in ANN input, described in Section 9.6, improved the accuracy of the model prediction, considering the Training and Validation Subsets. However, these models performed slightly worse in the evaluation on test sequences. It may be interpreted as the model decisions are strongly adapted to certain decision noise. Thirdly, it was stated in Section 9.6 that the improved accuracy of the model makes the partitioning algorithm choose a partitioning pattern different than HM less frequently. This paradoxically causes bigger disturbances in the encoder context, resulting in a worse overall compression efficiency.

Concluding the above observations, the goal to ideally mimic the sequence of partitioning decisions made by RD Optimization in HM is not achievable as long as the model accuracy is not ideal. This is particularly important for ANN-based algorithms, as the negative impact of encoding context disturbance will always be observed in their case. Even a significant increase in the training dataset size does not solve the problem. In Chapter 8, the proposed ANN models were compared to state-of-the-art solutions. Most of them reported that better accuracy of the models was achieved with bigger training datasets [Xu18A, Sh19, Ch20, Hu21B, Fe21]. Therefore, the evaluation results were very similar or even worse than the proposed models. This also means that classical data augmentation methods, e.g., image flipping or color corrections, would have the same effect. Every change in the image would require an estimation of the partitioning patterns in the HM. Thus, additional training samples would be generated, similar to adding more images to datasets. Further, the unaware application of the image modification may introduce artificial dependencies taught by the model that could negatively impact the partitioning decisions. Therefore, this impact is out of the scope of this dissertation.

In the context of the above discussion, the author's method of **ground truth augmentation** is presented. The idea is to train a model by considering several possible partitioning patterns for the CTU samples. Then, the ANN may indicate a partitioning pattern that is slightly less efficient but also less impactful to the encoding context. The results may be much better than a sudden, unexpected partitioning pattern, significantly impacting the encoding context. As mentioned earlier, mapping multiple partitioning patterns to the same CTU samples can make the model less sensitive to decision

noise. One can use images with small added noise (Section 9.5) as a source of alternative partitioning patterns for a CTU. Therefore, **the model can be trained with CTU samples from non-modified images and partitioning patterns which are estimated by encoding images in multiple noised versions**. Other combinations (CTU samples from noised images with partitioning patterns from original images or CTU samples from noised images with partitioning patterns from noised images) were tested, but the results did not improve the evaluation results or make it worse.

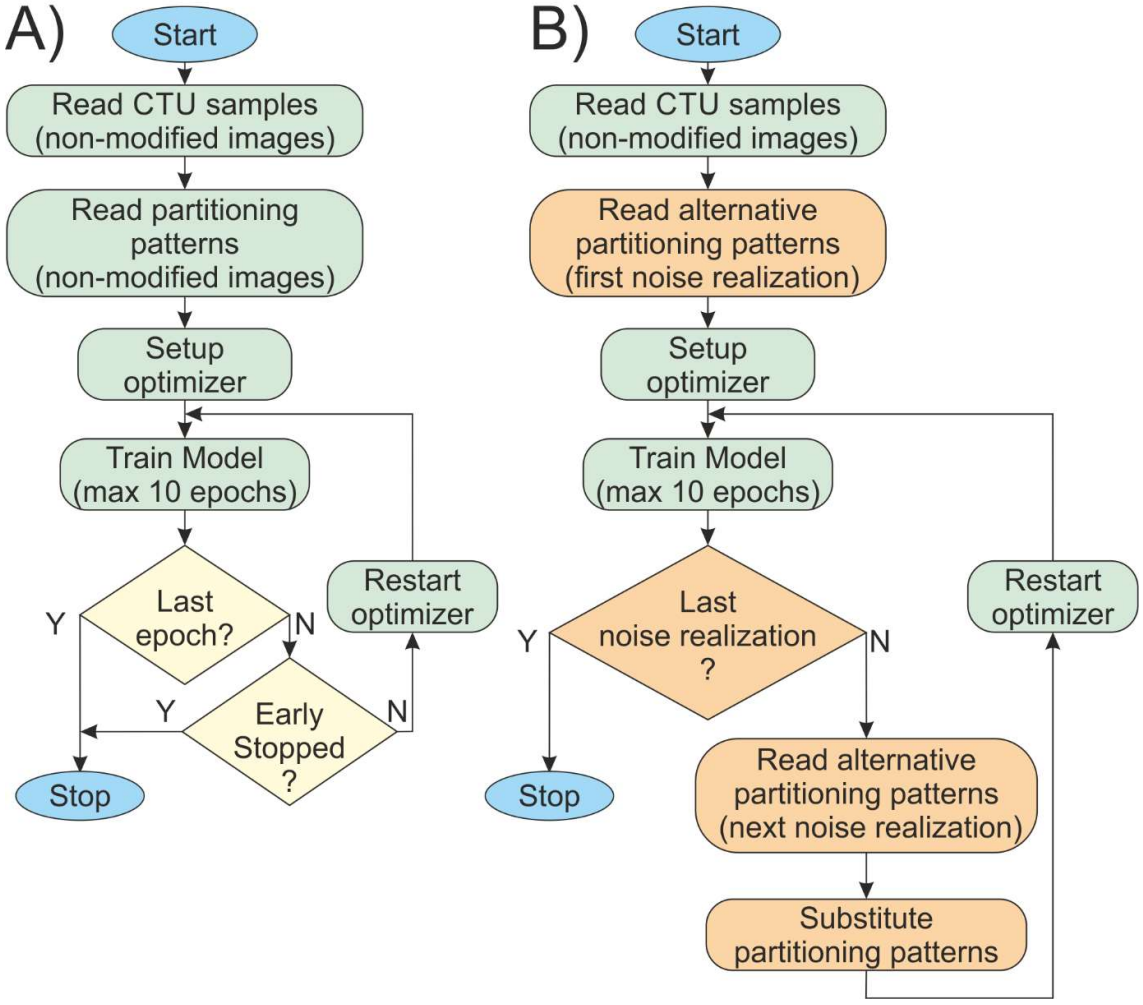

Figure 9. 6. Training procedure: A) Original, B) Modified. The modified steps of the procedure are filled with orange color.

To check the viability of the proposed ground truth augmentation, datasets to train models were prepared by adding noise to DIV2k [Ag17] images. The term "noise realization" refers to the application of noise defined in Section 9.5 (changes the value of randomly chosen samples) generated for each image in the dataset. Noise realizations differ by the initial state of the random value generator used for noise generation. Datasets for model training were created following the procedure presented in Section 4.2, using noised images generated in the following configurations:

1) The noise with RMS = 0.05 – in 10 consecutive random realizations.
2) The noise with RMS = 0.5 – in 10 consecutive random realizations.
3) The noise with RMS = 1.0 – in 10 consecutive random realizations.

4) The noise with RMS: 0.05, 0.10, 0.30, 0.50, 0.70, 1.00, 1.20, 1.50, 1.70, 2.00 in consecutive random realizations.
5) The noise with RMS: 2.00, 1.70, 1.50, 1.20, 1.00, 0.70, 0.50, 0.30, 0.10, 0.05 in consecutive random realizations.

The training procedure (Section 4.6) was slightly modified to include multiple partitioning patterns for a given CTU. For every 10^{th} training epoch, the set of partitioning patterns is substituted with the one from the next noise realization. It should be mentioned that the Early Stopping (Subsection 4.6.3) may terminate the training. In such a case, the set of partitioning patterns is substituted with the one from the next noise realization, and the training is resumed. The modification to the training procedure is presented in Figure 9.6

A set of models with ANN architecture for Basic Approach (Section 5.1) is trained for each proposed dataset with augmented ground truth data. Such set consists of 4 models, each for one *QP:* {22, 27, 32, 37}. As the models were trained using the modified training procedure described earlier, the comparison of the model $Accuracy$ (Formula 4.8) will not be representative. The models were evaluated in Modified HM (Section 3.1) with the hard-decisive variant of AlgPrb (Subsection 7.2.2) as the decision algorithm. The evaluation results are presented in Table 9.11. The evaluation of the models was performed according to description in Section 3.1. The presented results are relative to HM.

Training the ANN architecture for Basic Approach using datasets with augmented ground truth data improved the evaluation results in each case. The worst evaluation results were achieved for the noise with RMS of 1.0 in 10 consecutive random realizations (Idx 3). Slightly better results are observed for both noise with RMS rising from 0.05 to 2.00 in consecutive random realizations (Idx 4) and noise with RMS of 0.5 in 10 consecutive random realizations (Idx 2). The best results were achieved for the smallest RMS (0.05) noise in 10 realizations (Idx 1) and the falling RMS from 2.00 to 0.05 in consecutive random realizations (Idx 5). Thus, partitioning patterns estimated for small noise with small RMS should be used at least in the last training phases to get the best results.

Table 9. 11. Evaluation results for training the ANN architecture for Basic Approach using augmented ground truth datasets. Results for the Basic Architecture (Section 5.3) are included for comparison. The presented results are relative to HM.

Idx	Dataset	$BD\text{-}RATE$[%]	$BD\text{-}PSNR$[dB]
1	Noise with RMS: 0.05 – in 10 consecutive random realizations	**1.72**	**-0.084**
2	Noise with RMS: 0.5 – in 10 consecutive random realizations	1.73	-0.085
3	Noise with RMS: 1.0 – in 10 consecutive random realizations	1.76	-0.087
4	The noise with rising RMS from 0.05 to 2.00 in 10 consecutive random realizations	1.73	-0.085
5	The noise with falling RMS from 2.00 to 0.05 in consecutive random realizations	**1.71**	**-0.084**
	Basic Architecture (default training dataset)	**1.80**	**-0.089**

The proposed approach improved the evaluation results by 0.09 p.p. in terms of $BD\text{-}RATE$ and 0.005 dB in terms of $BD\text{-}PSNR$. The improvements are significant, especially as the model architecture did not change. Models trained with proposed ground truth augmentation outperform any modification of the hyperparameters discussed in Section 5.4.

To further test the proposed solution, the best-performing datasets with augmented ground truth data – indices 1 and 5 (noise with RMS 0.05 in 10 consecutive random realizations and noise of falling RMS from 2.00 to 0.05 in consecutive random realizations) were used to train ANN architecture for Extended Approach (Section 6.1). Training and evaluation are exactly the same as for models with ANN architecture for Basic Approach, described earlier in this section. The results are presented in Table 9.14. The evaluation of the models was performed according to description in Section 3.1. The presented

results are relative to HM. Similarly, as for ANN architecture for Basic Approach, the evaluation results have improved. Results for both used datasets with augmented ground truth data are very similar. The $BD\text{-}RATE$ decreased by 0.07 p.p., and $BD\text{-}PSNR$ is smaller by 0.004 dB.

Table 9. 12. Evaluation results for training ANN architecture for Extended Approach using the best performing augmented ground truth datasets. Results of the Extended Architecture trained with the default dataset are included for comparison. The presented results are relative to HM.

Dataset	$BD\text{-}RATE$ [%]	$BD\text{-}PSNR$ [dB]
The noise with RMS: 0.05 – in 10 consecutive random realizations	3.37	-0.173
The noise with falling RMS from 2.00 to 0.05 in consecutive random realizations	3.38	-0.173
Extended Architecture (Default training dataset)	**3.44**	**-0.177**

As shown in this Section, **the idea of training a model to fit several possible partitioning patterns for the CTU proved to increase the model performance** in terms of $BD\text{-}RATE$ and $BD\text{-}PSNR$. Improvements are observed for both proposed Basic and Extended architectures. However, this idea was not incorporated into the proposed partitioning algorithm and was not compared with state-of-the-art results. The reason is that the improvements are primarily observable for hard-decisive variants of the decision algorithm and small values of control parameters for soft-decisive variants. **As the Basic and Extended Architectures proved to perform similarly or even better than state-of-the-art solutions, the method proposed in this section will surely increase the superiority**. Unfortunately, as authors of other methods do not provide the model training framework, the impacts of the proposed ground truth augmentation method on other models could not be checked. Thus, **this method remains a promising topic for further research.**

9.8 Global optimization of CTU partitioning with the Viterbi algorithm

Previous sections 9.6 and 9.8 present two methods for handling the problem of the encoding contextuality. Both of these methods impact the ANN model to achieve this goal. Despite the promising results, a series of downsides was observed. The limitation of those methods is that the contextuality is dealt in the local optimization process. The proposed methods are able to at most neglect the negative impact of disturbed encoding context. Still, a single, very context-disturbing partitioning pattern can be indicated, and proposed methods cannot identify such a situation to adjust accordingly. A method for refining the previously estimated partitioning patterns can be used to address this issue. Therefore, in this section, an algorithm for global optimization of partitioning patterns is proposed.

The ANN in the Basic and Extended Approaches (Section 5.1 and 6.1) are design to estimate the partitioning pattern by processing only samples of currently processed CTU (particularly luma samples). Thus, this process is independent for each CTU. In this case, the partitioning patterns for all CTUs can be estimated first, and then global optimization may be performed. Further, the independence of ANN predictions allows the formulation of the Hidden Markov Model (HMM) [Ba66]. Therefore, the Viterbi algorithm [Vi67] can be used to find the recursive optimal solution of the state sequence estimation. Such a method may shape the encoding context by adjusting consecutive CTUs and effectively achieving better compression efficiency.

Given a single image consisting of V successive CTUs, for each CTU the ANN estimates a Division Tensor DT, as described in Section 4.5. For particular CTU with index $v \in [0; V-1]$, the Division Tensor is denoted as DT_v.

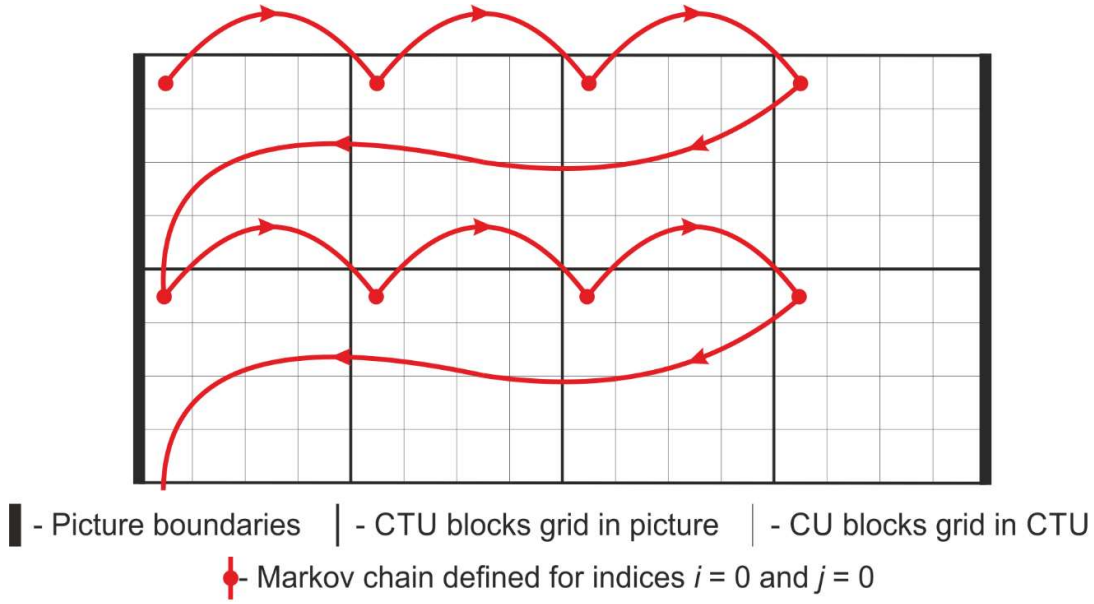

Figure 9. 7. Visual representation of Hidden Markov Model (HMM) defined for exemplary indices. The arrow specifies the connection direction in HMM.

For each pair of indices i and j, a separate HMM is constructed as progression of states in CTU order, as shown in Figure 9.7. Said HMM consists of **hidden states** [Ba66], where "hidden" corresponds to the fact that the realization of this process is unknown and is sought. For each CTU index v there are D states, each corresponding to a different depth level $d \in [0; D-1]$. The transition between the states is possible only along the *v* direction, with full connection between different depth levels. Such defined HMM (for indices i and j) can be represented with 2D lattice, as shown in Figure 9.8.

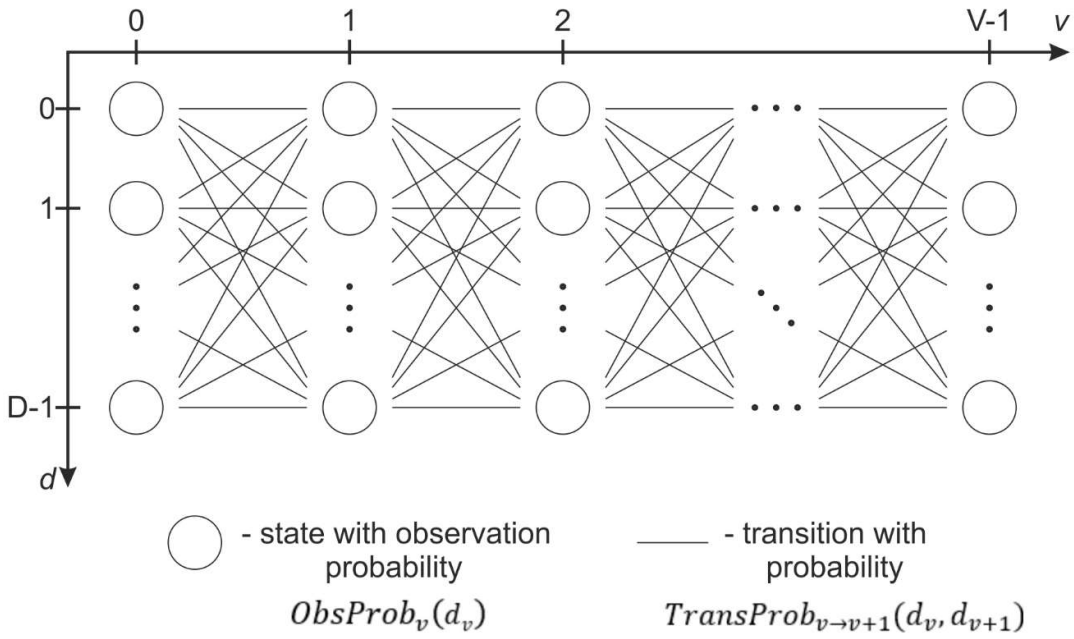

Figure 9. 8. The 2D lattice for Hidden Markov Model (HMM), defined pair of indices i and j.

For each pair of indices i and j, the goal is to find a sequence of $d_0, d_1, d_2, \dots, d_{V-1}$, where d_v is state value for v-th state in HMM, further interpreted as depth level value. A search criterion for the sequence is Maximum A Posteriori (MAP), that is maximization of probability $P(d_0, d_1, d_2, \dots, d_{V-1})$ after making observations:

$$P(d_0, d_1, d_2, \dots, d_{V-1}) = \prod_{v=0}^{V-2} TransProb_{v \to v+1}(d_v, d_{v+1}) \cdot \prod_{v=0}^{V-1} ObsProb_v(d_v) \tag{9.1}$$

where:

- $TransProb_{v \to v+1}(d_v, d_{v+1})$ is a **transition probability** from state with CTU index v (horizontal axis on Figure 9.8) and depth level d_v (vertical axis on Figure 9.8) to a state with CTU index $v+1$ with depth level d_{v+1}.
- $TransObs_v(d_v)$ is **observation probability** for d_v (vertical axis on Figure 9.8).

Based on the MAP rule, the most probable sequence $d_0, d_1, d_2, \dots, d_{V-1}$can be found:

$$d_0, d_1, d_2, \dots, d_{V-1} = ArgMax_{d_0, d_1, d_2, \dots, d_{V-1}}\big(P(d_0, d_1, d_2, \dots, d_{V-1})\big) \tag{9.2}$$

Among multiple algorithm available in literature, the Viterbi algorithm [Vi67] has been used in the dissertation. As the probabilities can be expressed in logarithmic scale, the $Goal$ function to be maximized is defined as follows:

$$\begin{aligned} Goal(d_0, d_1, d_2, \dots, d_{V-1}) &= \log\big(MAP(d_0, d_1, d_2, \dots, d_{V-1})\big) = \\ &= \sum_{v=0}^{V-2} \log\big(TransProb_{v \to v+1}(d_v, d_{v+1})\big) + \sum_{v=0}^{V-1} \log\big(ObsProb_v(d_v)\big). \end{aligned} \tag{9.3}$$

With HMM defined as above, **a novel global optimization algorithm for CTU partitioning is proposed**. Thus, the observation probabilities are the probabilities estimated with ANN ($DT_v[i, j, d]$ $d \in [0; D-1]$), so:

$$\log\big(ObsProb_v(d_v)\big) = \log(DT_v[i, j, d_v]). \tag{9.4}$$

The transition probabilities are calculated with the Potts Cost function [Ge84], defined as follows:

$$log\big(TransProb_{v \to v+1}(d_v, d_{v+1})\big) = \begin{cases} 0, & if\ |d_v - d_{v+1}| = 0 \\ \lambda, & otherwise \end{cases}. \tag{9.5}$$

Such an estimated sequence should favor choosing the same depth level value and add additional cost when the depth level value is changed. Thus, the method aims to find such a set of depth level values that will suppress the context disturbance caused by inaccurate predictions of the ANN in consecutive blocks.

Execution of Viterbi algorithm for particular HMM for indices i, j, results in a sequence of depth level $d_0, d_1, d_2, \dots, d_{V-1}$ for each CTU index v. These depth levels are then packed into Division Matrices (DM) such that: $DM_v[i, j] = d_v$. The process is repeated independently for each i, j, so that partitioning patterns for each CTU in the image $DM_v[i, j]$ are attained.

Unfortunately, these DM may be non-conformant with HEVC syntax. Therefore, each DM is processed by the correction algorithm, similar to the hard-decisive variant of AlgIdx (Subsection 7.2.1). This correction algorithm follows the same procedure as defined in Section 7.1, but the DM is processed instead of DT. The input DM is referred to as DM_{input} and output DM is referred to as DM_{output},

starting from current depth level $cdl = 0$, $N \times N$ equal the size of DT, $m = 0$, $n = 0$. The algorithm defines the steps of the procedure as follows:

Procedure: ***AlgGlobal*(*cdl*, *N*, *m*, *n*):**

1. Denote the area $\mathrm{A}_{m,n,N}$ of the current block inside of $\mathrm{DM}_{input}[i,j]$ and $\mathrm{DM}_{output}[i,j]$ as specified by the indices m, n and the size N, as in Formula 9.6 (the same as in Section 7.1, Formula 7.1).

$$\mathrm{A}_{m,n,N} = \{(i,j) \mid i \in [m; m+N-1],\ j \in [n; n+N-1]\}. \tag{9.6}$$

2. Analyze the DT values in the denoted area $\mathrm{A}_{m,n,N}$, e.g. $\mathrm{DM}_{input}[i,j]$, $(i,j) \in \mathrm{A}_{m,n,N}$.
   - If N is equal 1, set $\mathrm{DM}_{output}[m,n] \leftarrow cdl$ and terminate the algorithm.
   - Calculate the C_G value defined as follows:

$$C_G = \frac{1}{N^2} \cdot \sum_{(i,j) \in \mathrm{A}_{m,n,N}} \mathrm{Iv}\left(\mathrm{DM}_{input}\,[i,j] = cdl\right), \tag{9.7}$$

   where $\mathrm{Iv}(\cdot)$ is the Iverson function [Fo99], such that $\mathrm{Iv}(true) = 1$ and $\mathrm{Iv}(false) = 0$.
3. Make decision according to the analysis of the considered DT area $\mathrm{A}_{m,n,N}$:
   - If $C_G > 0.5$, fill the values in DM corresponding to the area of the current block with cdl (Formula 7.2), and terminate the recursion.
   - Otherwise: Recurse the procedure for next depth level $cdl + 1$. Split the block following the quaternary tree (Figure 7.1) and perform recursive call for each subdivided block as in Formula 7.3 using ***AlgGlobal*** procedure.

The proposed global optimization algorithm was implemented in Modified HM and tested with the Basic Architecture (Section 5.1). The method was evaluated for Potts cost (λ) in the range $\langle 0; 0.9 \rangle$ with the step of 0.1. The method was evaluated on images used for training (cropped DIV2k dataset, Section 4.2) and test sequences (Section 3.5). The evaluation results for the global optimization algorithm are presented in Table 9.13. For comparison, Table 9.13 includes the evaluation results for the proposed partitioning algorithm with the Basic Architecture and hard-decisive variant of both AlgIdx (Subsection 7.2.1) and AlgPrb (Subsection 7.2.2). The evaluation of the models was performed according to description in Section 3.1. The presented results are relative to HM.

Table 9. 13. Evaluation results (bitrate and quality) for the global optimization algorithm (AlgGlobal) with the use of Basic Architecture. The presented results are relative to HM.

Transition Cost value (λ) in AlgGlobal	**DIV2K Training**		**DIV2K Validation**		**JCT-VC**	
	***BD-RATE* [%]**	***BD-PSNR* [dB]**	***BD-RATE* [%]**	***BD-PSNR* [dB]**	***BD-RATE* [%]**	***BD-PSNR* [dB]**
0	**1.48**	**-0.068**	**2.07**	**-0.082**	**1.79**	**-0.088**
0.1	**1.44**	**-0.067**	**1.90**	**-0.080**	**1.78**	**-0.088**
0.2	**1.52**	**-0.070**	**1.94**	**-0.083**	**1.79**	**-0.088**
0.3	1.52	-0.071	2.13	-0.089	1.83	-0.090
0.4	1.65	-0.076	2.14	-0.091	1.88	-0.092
0.5	1.74	-0.083	2.25	-0.090	1.94	-0.095
0.6	1.81	-0.084	2.28	-0.098	2.00	-0.098
0.7	1.80	-0.083	2.41	-0.102	2.07	-0.102
0.8	1.82	-0.085	2.38	-0.099	2.14	-0.105
0.9	1.94	-0.090	2.24	-0.094	2.21	-0.109
For comparison: Local optimization solutions proposed in Section 7.2						
AlgIdx	**1.45**	**-0.069**	**1.73**	**-0.075**	**1.86**	**-0.091**
AlgPrb	1.34	-0.066	1.78	-0.075	1.80	-0.089

Due to similarities, the results of the proposed global optimization algorithm are discussed in comparison to AlgIdx. A saddle effect is observed regardless of the evaluated dataset, as the best results are observed for the Transition Cost λ value of 0.1. Besides that, the evaluation results differ for DIV2k and test sequences. Given the DIV2k dataset, the $BD\text{-}RATE$ and $BD\text{-}PSNR$ are worse for the global optimization algorithm, regardless of cost value. This observation applies to both Training and Validation subsets. The exception is Training Subset and the Transition Cost λ value of 0.1, where the results are almost the same. This means that the model is closely fitted to the content of the DIV2k images, so any change in the decisions causes a bitrate increase. However, the situation is different for test sequences. For the smallest Transition Cost λ values (0, 0.1, 0.2, and 0.3), the proposed global optimization algorithm is significantly better (up to 0.08 p.p. in $BD\text{-}RATE$ and 0.003 dB in $BD\text{-}PSNR$) compared to Basic Architecture with the hard-decisive AlgIdx. Such improvement is very close to the 0.1 p.p. evaluation result increase criterion, defined for model hyperparameter search (Subsection 4.6.3)

Since the information of ANN certainty is lost in the proposed global optimization algorithm, the comparison to AlgPrb is not fair. A version of global optimization algorithm that will preserve such information of ANN certainty is a topic for future research. Still, the results of the proposed global optimization algorithm are marginally better for test sequences (for λ =0.1: 0.02 p.p. in $BD\text{-}RATE$ and 0.001 dB in $BD\text{-}PSNR$) compared to AlgPrb.

The results of the proposed method prove that global optimization of partitioning patterns may improve the bitrate of encoded sequences. Such global optimization may be applied with any method that independently estimates the partitioning patterns for each CTU. It should be noted that the method presented in this section is one of the many approaches for global optimization tested during the research. However, it is the only one that yields improvement.

The encoding time reduction (Time Savings: TS, Formula 3.3) for the best performing Transition Cost λ values (0, 0.1, 0.2, and 0.3) is shown in Table 9.14. The evaluation of the models was performed according to description in Section 3.1. The presented results are relative to HM and represent the mean results over all test sequences (Section 3.5). The presented results are the mean over all sequences and QP values. Results for AlgIdx and AlgPrb are included for comparison.

The proposed global optimization requires the estimation of partitioning patterns for all CTUs before any of these may be used. In a single-threaded implementation, the proposed algorithm for global optimization in terms of TS performs similarly to the Basic Architecture with decision algorithms presented in Chapter 7. The difference is no more than 5.1 p.p. **Therefore, the ANN allows the global optimization of the CTU partitioning at a negligible computational complexity cost.** However, the multithreaded implementation proposed in Section 8.1 will not improve the TS, as the estimation of partitioning patterns could not be computed parallelly to the rest of the encoder computations. Still, the topic of global optimization of the CTU partitioning remains a very promising direction for further research.

Table 9. 14. Evaluation results (encoding time reduction) for the global optimization algorithm with the use of Basic Architecture. Presented results are relative to HM.

Transittion Cost value (λ)	**JCT-VC**
	***TS* [%]**
0	**51.56**
0.1	**51.00**
0.2	**51.52**
0.3	**50.98**
For comparison: Local optimization solutions proposed in Section 7.2	
AlgIdx	**56.08**
AlgPrb	54.20

10 Dissertation summary

This dissertation presents research on video encoder control algorithms, more precisely, partitioning algorithms that are used in decision optimization processes for video encoders. The main goal was the development of a partitioning algorithm that significantly reduces encoding time while maintaining compression efficiency as close to the reference encoder as possible. Such a partitioning algorithm derives the partitioning pattern using the ANN model and decision algorithm. The experimental part was performed in the context of HEVC video coding technology in Intra mode.

In Section 1.3, **two research theses were stated, and for both, scientific evidence has been provided** in this dissertation. The summary of these proofs is presented in Section 10.1. Additionally, the dissertation presents additional research achievements, recapped in Section 10.2. Furthermore, the work done during the research is overviewed in Section 10.3. Lastly, Section 10.4 presents conclusions from the dissertations and topics for future research.

10.1 Original achievements related to theses

10.1.1 The first thesis

The first thesis (T1) stated in this dissertation is "**The utilization of the Artificial Neural Network with a decision algorithm can significantly decrease the computational complexity of the video encoder as compared to HEVC reference model encoder.**"

In this dissertation, two approaches for partitioning algorithms were explored (Chapter 4): Basic Approach, which considers only partitioning CTU block into CU blocks, and Extended Approach, which additionally indicates the PU division. The proposed partitioning algorithms use ANN to estimate the whole partitioning pattern at once. **Two novel ideas were devised:**

- **ANN estimates probabilities of depth level values. These probabilities are outputted as 3D tensor.**
- **Non-trivial decision algorithms that process the ANN output.**

As the ANN model, original architecture was developed**.** The idea for the ANN was to mimic the quaternary tree in the alignment of the layers. Thus, it was used to create Basic (Chapter 5) and Extended (Chapter 6) architectures, which were used in the corresponding approaches. The output of the ANN is a 3D tensor, which consists of depth level probability vectors for specific subareas in the CTU. The proposed networks are very small regarding the number of weights (~91k) and low complex regarding the MAC operation count (~6M for Basic and ~8M for Extended). This was confirmed in comparison with models used in ANN-based state-of-the-art methods (Section 8.2). **It was emphasized that the partitioning pattern obtained directly from the ANN may be non-conformant with HEVC syntax** (e.g., it may represent non-rectangular blocks).

In addition to the mentioned two approaches for partitioning with ANNs, two decision algorithms were proposed: AlgIdx (Subsection 7.2.1) and AlgPrb (Subsection 7.2.2). **AlgIdx is a straightforward approach designed similarly to methods found in the literature. AlgPrb employs the author's idea for leveraging the certainty of the ANN by adequately interpreting the depth level probabilities**.

Initially, algorithms were considered in **hard-decisive variants**, where the decision algorithm always outputs a single partitioning pattern. For AlgPrb, **significant encoding time reduction** (Time Savings: TS, Formula 3.3) **is achieved** (~55% for the Basic Approach and ~70% for the Extended Approach) **with a slight increase in $\boldsymbol{BD\text{-}RATE}$** (1.8% for Basic Approach and 3.44% for Extended Approach). **AlgPrb outperforms AlgIdx** in terms of $BD\text{-}RATE$: 0.06 p.p. in the Basic Approach and 0.22 p.p. in the Extended Approach. However, the hard-decisive variants of the algorithms do perform worse than state-of-the-art methods.

During further research, more sophisticated, **soft-decisive variants of decision algorithms were developed** (Section 7.3), where more than one partitioning pattern can be implied, according to the output of the ANN. Algorithms are controlled with a single parameter (α for AlgIdx and β for AlgPrb). With this variant of the decision algorithms, **the best results are achieved for the author's AlgPrb algorithm**:

- Basic Approach: TS of 55.50% and $BD\text{-}RATE$ of 1.23% (for $\beta = 0.45$).
- Extended Approach: TS of 62.02% and $BD\text{-}RATE$ of 1.71% (for $\beta = 0.45$).

Compared to state-of-the-art methods in Chapter 8, the proposed partitioning algorithms are superior with respect to the trade-off between Encoding Time vs Compression Efficiency (TS vs. $BD\text{-}RATE$). The proposed partitioning algorithms are better in terms of FoM by 0.64 for the Basic Approach and 0.12 for the Extended Approach. **Therefore, the T1 is proven.** The described methods and results were published in the paper [Lo24].

10.1.2 The second thesis

The second thesis (T2) stated in this dissertation is **"The employment of Artificial Neural Network with a soft-decision algorithm enables a single parameter control over the Encoding Time vs Compression Efficiency trade-off."**

One of the main achievements of this dissertation is the development of the soft-decisive variants of the decision algorithms (Section 7.3). In **soft-decisive variants of decision algorithms** (Section 7.3), more than one partitioning pattern is implied, according to the output of the ANN. The certainty of the ANN is regulated according to estimated depth level probabilities. This regulation **is controlled with a single parameter** (α for AlgIdx and β for AlgPrb), which impacts how often a set of partitioning patterns is indicated (instead of a single partitioning pattern).

For the soft-decisive variants of the decision algorithms, the following tendencies are expected:

- Checking multiple partitioning patterns improves compression efficiency but at the cost of encoding time.
- Considering only one partitioning pattern reduces the encoding time, but the bitrate is increased.

However**, a decision algorithm that is too straightforward is disturbing these tendencies, as the certainty of the ANN model is leveraged insufficiently**. This was observed for AlgIdx algorithm (Subsection 7.3.3), where for the soft-decisive variant, the saddle effect was observed in results, along with small changes in encoding time reduction (Time Savings: TS, Formula 3.3) and $BD\text{-}RATE$. Furthermore, **the computational complexity of partitioning patterns is not identical**, as was shown in Subsection 7.3.1. As observed for Basic Architecture (Section 5.1) with soft-decisive AlgPrb: **in case of low ANN certainty, checking two less computationally complex blocks is more efficient than using a single, more computationally complex one**. In this case, the soft-decisiveness of the algorithm improved the $BD\text{-}RATE$, maintaining the TS.

The abovementioned tendencies were observed for the Extended Architecture with the soft-decisive AlgPrb. As the β increases, a monotonic increase of $BD\text{-}RATE$ is observed, with a monotonic decrease of TS (Subsection 7.3.3). Therefore, **the proposed partitioning algorithm offers control over the Encoding Time vs Compression Efficiency trade-off** (TS vs. $BD\text{-}RATE$).

The proposed soft-decisive variants of the decision algorithms were compared with the state-of-the-art solutions in Sections 8.2 and 8.3. Considering the aspect of control, **the proposed method is much easier to manage than methods found in the literature**. The proposed method employs a single control parameter, as opposed to a set of thresholds estimated with a heuristic model [Hu21B] or evolution algorithm [Ch20]. Considering the effectiveness of the algorithm, **the proposed Extended algorithm with AlgPrb in the soft-decisive variant proved to be superior compared to the best**

method found in the literature [Hu21B] by 0.53 p.p. in terms of $BD\text{-}RATE$ and 1.95 p.p. in terms of TS. Such a gain in the context of video coding is considerable. **Therefore, the T2 is proven.** The described methods and results were published in the paper [Lo24].

10.2 Additional original achievements of the dissertation

1. **Supporting results for Thesis T1**. Exploratory experiments that examine various aspects of the developed methods are presented. These research directions do not introduce new scientific theses but support the proof of thesis T1. The attained results also expand the state of knowledge with new results.
    a. **Performance of the proposed models and decision algorithms in additional content.** Proposed ANN architectures with AlgPrb were tested for content that was not considered during the development of ANN architecture and decision algorithm. It was shown in Section 9.2 that for the additional context (Section 3.5), the proposed partitioning algorithm performed ~1 p.p. worse in terms of $BD\text{-}RATE$. However, the decision algorithm in the soft-decisive variant still offered control over the ETvsCE trade-off in most cases.
    b. **Serial variants for proposed ANN architecture.** The proposed ANN architectures employ a quaternary-tree-like arrangement of layers, as shown in Section 5.1 and 6.1. In Section 9.3, the reduced gradient flow in the training phase was identified, which limits the model performance in the evaluation. A serial network variant was proposed to fix this issue. It was presented in Section 9.3 that using the Serial variant of the architecture allows training a model with reduced size (42 832 instead of 91 600) and complexity (0.3M less MAC operations) but with almost the same performance as the Basic Architecture (Section 5.1). This part of the research was published in the paper [Lo23].
2. **A novel metric for comparison of the partitioning algorithms that offer the control of the Encoding Time vs Compression Efficiency (ETvsCE) trade-off.** This metric has been developed during works in the dissertation related to the author's algorithm for partitioning patterns estimation with controllable ETvsCE trade-off (thesis T2). Preliminarily, the superiority of the proposed algorithm was demonstrated with a graphical comparison (Section 8.3). It was highlighted that in the literature, no method was found for numerical comparison of such partitioning algorithms, as discussed in Section 3.4. Therefore, the author's metrics were proposed: $\Delta\text{BD-RATE}|_{\text{TS}}$ and $\Delta\text{TS}|_{\text{BD-RATE}}$ (Section 3.4). The values of proposed metrics were calculated along with the graphical comparison of the methods with control over the ETvsCE trade-off. As discussed in Section 8.3, **the results of the proposed metrics accurately reflect the observations from the graphical comparison.** Therefore, the proposed metrics can be used interchangeably with the graphical comparison, as it only expresses what can be readily observable with the proposed metrics.
3. **Research on contextuality and chaos in the decision-making process in video encoders.**
    a. **An original experiment to determine the impact of contextuality and chaos in decision-making in a video encoder.** As discussed in this dissertation (Section 2.1, Subsection 2.3.2, Section 9.5), the decision-making process highly depends on the encoding context. Thus, this process can be considered chaotic. No study on this effect or scale of the phenomenon was found in the literature. **Therefore, in this dissertation, a simple experiment was proposed to measure the impact of contextual effects on partitioning decisions** (Section 9.5). In the proposed experiment, two scenarios are tested, namely adding random noise of small RMS, as well as modification of precisely defined sample by 1. It was shown that even for very small added noise (effectively several samples modified), as much as 65.43% of partitioning patterns were decided differently. Additionally, modifying a single sample can change 10.8% of partitioning patterns. **Thus, the scale of the contextual effects was determined using the proposed experiment.**
    b. **Use of contextual data in ANN architecture for partitioning prediction.** The impact of even negligible changes on the decision-making process was determined with the proposed

experiment (Section 9.5). Therefore, a modification for ANN was proposed (Section 9.6), which processes additional data to consider the current encoding. Such contextual data was identified, and the modification was applied to Basic Architecture**. An increase in training accuracy was observed, but the evaluation results deteriorated.** It was noted that for the model trained to mimic the decisions of HM, the partitioning pattern that is chosen differently has an increasing negative impact as the overall accuracy of prediction increases. This part of the research was published in the paper [Lo21].

c. **The method for ground truth data augmentation by utilization of noise.** Considering the methods from the literature used for comparison with the proposed method (Chapter 8), in most cases, the authors reported better accuracy of models, which was achieved with bigger training datasets. Still, the proposed partitioning algorithm was assessed as superior in the evaluation on test sequences. This indicates the same effect observed for contextual data use in the ANN model (Section 9.6)**. Thus, dataset size increase and data augmentation methods from machine learning were found to be ineffective**. It was concluded that instead of mimicking the HM partitioning decisions, **it is better to train the model with multiple, similar performing partitioning patterns**. The original **ground truth data augmentation method** was proposed (Section 9.7). The model is trained with multiple partitioning patterns for the same CTU samples. Partitioning patterns for a given CTU are determined by encoding images with slightly different noise realizations. The training procedure for the model was defined (Section 9.7). It was shown that **models trained with the proposed method perform better in evaluation** (~0.9 p.p. in terms of $BD\text{-}RATE$).

4. **Proposed global optimization algorithm for partitioning patterns.** The proposed partitioning algorithms use the ANN, which only processes samples from currently processed CTU (in particular, luma samples). It was discussed in Section 9.8 that such a method allows **independent estimation of partition patterns for each CTU** in the encoding scope. Therefore, **the independence of observations allows the creation of Hidden Markov Model** (HMM) [Ba66] for specific subareas in consecutive CTUs (Section 9.8). Then, **the globally optimized solution can be found using the Viterbi algorithm** [Vi67]. This global optimization returns the partitioning patterns, which are non-conformant with HEVC syntax. Thus, a straightforward correction algorithm is used, similar to the proposed AlgIdx in the soft-decisive variant. **The global optimization achieved better results by 0.08% in $BD\text{-}RATE$ and 0.003 dB in $BD\text{-}PSNR$, compared to the proposed AlgIdx in the hard-decisive variant.** It should be noted that ordinary global optimization of CTU partitioning is practically impossible to perform due to computational complexity (Subsection 2.3.1), even for small images consisting of few CTUs. However, **the proposed global optimization algorithm does not substantially increase the encoding time**, especially when paired with methods presented for thesis T1. Compared to hard-decisive variant of AlgIdx (Section 7.2.1), a ~5 p.p. worse encoding time reduction (Time Savings: TS, Formula 3.3) is observed.
5. **Implementation of the Modified HM.** The proposed partitioning algorithm was evaluated using the author's HM [HM] modification. **The software allows the fast implementation of the partitioning algorithm, including ANN-based, without influencing the rest of the decision-making algorithms in the HM (Subsection 2.3.2).** The description of the Modified HM is presented in Section 4.7. **This software was shared in open-access with paper [Lo24]**. The availability of this software significantly reduces the time it takes to begin research on partitioning algorithms. This is important because implementing ANN in HM is a significant entry work, blocking many researchers in this field. The amount of work needed to implement the modifications is outlined in the following section.

10.3 Overview of the work done

The research work presented in this dissertation required preparation of the following software:

- The Modified HM – the modification of the HM encoder and decoder [HM], described in Section 4.7. The HM software (~93 thousand code lines in C++ [CPP]) was analyzed, including the scripts for building the project. Implementing all necessary modifications and algorithms required adding or modifying ~3000 lines of code. The LibTorch [LIBTORCH] library has also been added to the software.
- Preparation of the original software, e.g., image conversion, training dataset preparation, training dataset analysis, model training (TenforFlow [TENSORFLOW]), model conversion (PyTorch [PyTorch]), and results analysis, which consist of ~25 000 lines of Python code.

Along with the code, a significant part of the work done is connected to dataset preparation and analysis. The DIV2k dataset [Ag17] was encoded and processed multiple times: separate datasets for each *QP* included in CTC [CTCHEVC], experiments with small noise added, and modification of single sample in the picture presented in Section 9.5, preparation of datasets for Basic Approach, Extended Approach and architecture with contextual data modification. The size of data used for training the models was almost 1 TB.

Another significant part of the work done is the training of the models. Overall, ~500 training experiments were performed according to the strategy presented in Section 3.7. This results in ~2000 trained models (4 models for each experiment). The models from each training experiment were then analyzed with metrics described in Subsection 3.6.1. Further, the trained models were evaluated on test sequences with proposed decision algorithms. This evaluation was performed for ~50 training experiments. It should be mentioned that selected models were evaluated for each decision algorithm in both hard and soft-decisive variants (20 evaluations per model). Lastly, the best-performing models were evaluated using the complete time assessment procedure presented in Section 3.3. In order to store the experiment data (models, model assessment data, encoder configurations, encoded bitstreams, encoding logs), ~8 TB of mass storage was required.

As mentioned in Section 3.7, computations were performed using a computing cluster. However, it would be imprecise to determine computation time regarding the use of this cluster (hardware changes made during the research, different specifications of the components). Therefore, the computation time was estimated referring to continuous single-threaded computations. Depending on the task, computations required CPU or both CPU and GPU. Computation time was estimated depending on the tasks:

- Preparing all training datasets required ~1200 days of continuous computations (CPU).
- Training and assessment of models required ~2000 days of continuous computations (CPU and GPU)
- Evaluation of the models and decision algorithms on test sequences required ~3000 days of continuous computing (CPU).
- Time assessment of selected models required ~600 days of continuous computing (CPU).

The above estimates sum up to ~4800 days of single-threaded continuous computations using CPU and ~2000 days of single-threaded continuous computations using CPU with GPU. Thus, performing the computations was possible only by employing a cluster of machines with a multi-threaded CPU and a GPU. Considering a cluster of six machines, each with 12 CPU threads and a single GPU, the overall computation time is estimated as ~70 days for computation that required only CPU and ~334 days for computation that required both CPU and GPU. This estimation does not include the availability of cluster, supervision of the machines, and experiment preparation.

10.4 Future research topics

The observations and conclusions made in this dissertation can be further used in developing a partitioning algorithm for more advanced video encoding methods. Most of these topics yielded very

interesting observations and promising results. Thus, these subjects will be explored in future works in the following research directions:

- Adaptation of the proposed partitioning algorithm to the Inter mode, by e.g. training models with dedicated dataset for Inter mode, adjustments of the model to the PU division in Intra model, modification of the model to include data related to temporal dependencies in video sequence. Preliminary results for this research direction were published in the paper [Lo24].
- Adaptation of the proposed algorithm to more advanced video coding techniques, such as VVC [VVC]
- Further development of the ANN architecture. The most promising directions are:
    - Application of ANN architectures as ResNet [He16], DenseNet [Hu17], Visual Transformer [Do20].
    - Analysis of the training dataset in terms of size and content, further analysis of image content on the ANN performance.
    - Development of architectures tailored to specific QP values or single architecture for all QP values.
    - Research over approach for training, different than mimicking RD Optimization, regarding the contextuality of the partitioning. Promising options are, e.g. further development of the proposed ground truth augmentation method, the use of Reinforcement learning, etc.
    - Development of an architecture that internally models the contextuality of the partitioning, e.g. RNN , LSTM [Go16, Sh20].
- Further development on decision algorithms, concerning better use of ANN output and wider control over the Encoding Time vs Coding Efficiency trade-off.
- Research on a global partitioning optimization algorithm, e.g. redefining Hidden Markov Model (HMM), applying different cost methods, and training of dedicated ANN for this algorithm.
- Optimization of the Modified HM implementation, use of efficient ANN frameworks, and development of multi-threaded implementation such as proposed double-threaded (Section 8.2).

10.5 Conclusions

It can be concluded that the dissertation explored encoding control algorithms, focusing on partitioning pattern estimation. The research aimed to design a partitioning algorithm using an ANN model and a non-trivial decision algorithm. Two approaches — Basic and Extended — were proposed with corresponding ANN architectures. Additionally, two decision algorithms, AlgIdx and AlgPrb, were introduced in soft and hard-decisive variants.

It was shown that proposed partitioning algorithms for both presented approaches are superior, in terms of $BD\text{-}RATE$ or encoding time reduction, to solutions found in the literature. The proposed AlgPrb in the soft-decisive variant allowed easy control over the Encoding Time vs Coding Efficiency trade-off. Both theses stated in this dissertation were confirmed. Therefore, it was shown that the proposed algorithm, which is composed of the ANN and non-trivial decision algorithm, is the most efficient among those found in the literature.

Along with the main achievements, several additional research directions were explored. The subject of contextual effect on encoding decisions was investigated. The ANN models that process additional contextual data were proposed and evaluated. The ground truth augmentation method was proposed and tested. A decision algorithm for global optimization of partitioning patterns was designed and evaluated.

Finally, topics for future work have been discussed. Some of the above-mentioned ideas are already being investigated by the author. The results are expected to be attained in the upcoming months and years.

11 Publications of the author

International journals:

[Lo24] M. Lorkiewicz, O. Stankiewicz, M. Domański, H. -M. Hang and W. -H. Peng, "Complexity-Efficiency Control With ANN-Based CTU Partitioning for Video Encoding," in IEEE Access, vol. 12, pp. 102536-102551, 2024, doi: 10.1109/ACCESS.2024.3433424.

International conferences

[Lo21] M. Lorkiewicz, O. Stankiewicz, M. Domanski, H. -M. Hang and W. -H. Peng, "Fast Selection of INTRA CTU Partitioning in HEVC Encoders using Artificial Neural Networks," 2021 Signal Processing Symposium (SPSympo), LODZ, Poland, 2021, pp. 177-182, doi: 10.1109/SPSympo51155.2020.9593483.

[Lo23] M. Lorkiewicz, O. Stankiewicz, M. Domański, H. -M. Hang and W. -H. Peng, "Complexity Reduction of ANN Model for CU Size Selection in HEVC," 2023 Signal Processing Symposium (SPSympo), Karpacz, Poland, 2023, pp. 111-116, doi: 10.23919/SPSympo57300.2023.10302659.

- O. Stankiewicz, T. Grajek, S. Maćkowiak, J. Stankowski, S. Różek, M. Lorkiewicz, M. Wawrzyniak, M. Domański, "Region-of-Interest-Based Video Coding for Machines," 2024 IEEE International Conference on Multimedia and Expo Workshops (ICMEW), Niagara Falls, ON, Canada, 2024, pp. 1-6, doi: 10.1109/ICMEW63481.2024.10645441.
- J. Stankowski, M. Lorkiewicz, K. Klimaszewski, “System Architecture for Real-Time Comparison of Audio Streams for Broadcast Supervision”, Image Processing and Communications Challenges 10, 10th International Conference, IP&C’2018 Bydgoszcz, Poland, November 2018, Proceedings / red. Michał Choraś, Ryszard S. Choraś - Cham, Switzerland : Springer International Publishing, 2019 - p. 245-252.
- M. Lorkiewicz, J. Stankowski and K. Klimaszewski, "Algorithm for Real-Time Comparison of Audio Streams for Broadcast Supervision," 2018 25th International Conference on Systems, Signals and Image Processing (IWSSIP), Maribor, Slovenia, 2018, pp. 1-5, doi: 10.1109/IWSSIP.2018.8439599.

ISO/IEC MPEG Documents

- O. Stankiewicz, K. Wegner, A. Dziembowski, M. Lorkiewicz, G. Lee, J. Seo, M. Domański, "Proposed test materials for 3DoF+ or Omnidirectional 6DoF",ISO/IEC JTC1/SC29/WG11 MPEG2018, M44461, Macao, China, 8-12 October 2018.
- K. Wegner, T. Grajek, A. Grzelka, M. Lorkiewicz, R. Ratajczak, O. Stankiewicz, J. Stankowski, H. Zabinski, M. Domanski, "Depth Estimation Reference Software extension for lightfield images",ISO/IEC JTC1/SC29/WG11 MPEG2019, M46126, Marrakech, Maroko, 14-18 January 2019.
- A. Grzelka, M. Lorkiewicz, A. Dziembowski, D. Mieloch, "[MPEG-I Visual] PUT proposal of PoznanFencing posetraces",ISO/IEC JTC1/SC29/WG11 MPEG2019, M50646, Genewa, Switzerland 7-11 October 2019.

- S. Różek, O. Stankiewicz, S. Maćkowiak, T. Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, D. Cywiński, J. Szekiełda, M. Domański, "[VCM] Improved RoI preprocessing and retargeting for VCM",ISO/IEC JTC1/SC29/WG04 MPEG2024, M66523, Online, 22-26 January 2024.
- S. Różek, O. Stankiewicz, S. Maćkowiak, T. Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, D. Cywiński, J. Szekiełda, M. Domański, "[VCM] CE1.2: Improved RoI retargeting for VCM",ISO/IEC JTC1/SC29/WG04 MPEG2024, M66979, Rennes, France, 22-26 April 2024.
- M. Lorkiewicz, O. Stankiewicz, S. Różek, S. Maćkowiak, T. Grajek, M. Wawrzyniak, J. Stankowski, D. Cywiński, J. Szekiełda, M. Domański, "[FCM] Anchor generation crosscheck",ISO/IEC JTC1/SC29/WG04 MPEG2024, M66980, Rennes, France, 22-26 April 2024.
- S. Różek, O. Stankiewicz, S. Maćkowiak, T. Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, D. Cywiński, J. Szekiełda, M. Domański, "[VCM] CE1-related: Improved RoI retargeting for VCM with optimized scaling factors",ISO/IEC JTC1/SC29/WG04 MPEG2024, M66988, Rennes, France, 22-26 April 2024.
- S. Różek, O. Stankiewicz, S. Maćkowiak, T. Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, D. Cywiński, J. Szekiełda, M. Domański, "[VCM] CE0-related: RoI-based retargeting for VCM without resolution change",ISO/IEC JTC1/SC29/WG04 MPEG2024, M67865, Rennes, France, 22-26 April 2024.
- S. Różek, O. Stankiewicz, S. Maćkowiak, T. Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, D. Cywiński, J. Szekiełda, M. Domański, "[VCM] Cross-check of CE1.1",ISO/IEC JTC1/SC29/WG04 MPEG2024, M67866, Rennes, France, 22-26 April 2024.
- O. Stankiewicz, S. Różek, J. Stankowski, S. Maćkowiak, T. Grajek, M. Wawrzyniak, M. Lorkiewicz, M. Domański, "[VCM] Cross-check of m67501",ISO/IEC JTC1/SC29/WG04 MPEG2024, M68029, Rennes, France, 22-26 April 2024.
- O. Stankiewicz, S. Różek, J. Stankowski, S. Maćkowiak, T. Grajek, M. Wawrzyniak, M. Lorkiewicz, M. Domański, "[VCM] Cross-check of m67502",ISO/IEC JTC1/SC29/WG04 MPEG2024, M68030, Rennes, France, 22-26 April 2024.
- S. Różek, O. Stankiewicz, S. Maćkowiak, T. Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, D. Cywiński, J. Szekiełda, M. Domański, "[VCM][CE0] Results of CE0.1",ISO/IEC JTC1/SC29/WG04 MPEG2024, M68233, Sapporo, Japan, 15-19 June 2024.
- O. Stankiewicz, S. Różek, S. Maćkowiak, T. Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, M. Domański, "[VCM] [CTCS-related] Additional encoder options for resolution control of RoI-based Retargeting",ISO/IEC JTC1/SC29/WG04 MPEG2024, M68834, Sapporo, Japan, 15-19 June 2024.
- O. Stankiewicz, S. Różek, S. Maćkowiak, T. Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, M. Domański, "[VCM] Format conversion and bilinear interpolation clean-up",ISO/IEC JTC1/SC29/WG04 MPEG2024, M68835, Sapporo, Japan, 15-19 June 2024.
- O. Stankiewicz, S. Różek, S. Maćkowiak, T. Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, D. Cywiński, J. Szekiełda, M. Domański, "[VCM][CTCS] PUT results for

Inner_RA and all E2E scenarios.",ISO/IEC JTC1/SC29/WG04 MPEG2024, M68838, Sapporo, Japan, 15-19 June 2024.

- M. Lorkiewicz, J. Stankowski, O. Stankiewicz, S. Różek, S. Maćkowiak, T. Grajek, M. Wawrzyniak, M. Domański, "[FCM] Crosscheck of m68310 – VTM version update",ISO/IEC JTC1/SC29/WG04 MPEG2024, M69338, Sapporo, Japan, 15-19 June 2024.
- S. Różek, O. Stankiewicz, S. Maćkowiak, T. Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, M. Domański, "[VCM][CE0] Results of CE0.1",ISO/IEC JTC1/SC29/WG04 MPEG2024, M69415, Kemer, Türkiye, 4-8 November 2024.
- O. Stankiewicz, S. Różek, S. Maćkowiak, T. Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, M. Domański, "[VCM] Quality control for ROI-based tools",ISO/IEC JTC1/SC29/WG04 MPEG2024, M70248, Kemer, Türkiye, 4-8 November 2024.
- O. Stankiewicz, S. Różek, M. Lorkiewicz, T. Grajek, S. Maćkowiak, M. Wawrzyniak, J. Stankowski, M. Domański, "[VCM] Neural-network-based chroma reconstruction",ISO/IEC JTC1/SC29/WG04 MPEG2024, M70249, Kemer, Türkiye, 4-8 November 2024.
- S. Różek, O. Stankiewicz, S. Maćkowiak, T. Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, M. Domański, "[VCM][CE0] Cross-check of CE0.5",ISO/IEC JTC1/SC29/WG04 MPEG2024, M70427, Kemer, Türkiye, 4-8 November 2024.
- S. Różek, O. Stankiewicz, S. Maćkowiak, T. Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, M. Domański, "[VCM] Cross-check of m69993",ISO/IEC JTC1/SC29/WG04 MPEG2024, M70429, Kemer, Türkiye, 4-8 November 2024.
- O. Stankiewicz, S. Różek, S. Maćkowiak, T. Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, M. Domański, "[VCM][CE4] Cross-check of CE4.3",ISO/IEC JTC1/SC29/WG04 MPEG2024, M70471, Kemer, Türkiye, 4-8 November 2024.
- O. Stankiewicz, S. Różek, T. Grajek, S. Maćkowiak, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, M. Domański, "[VCM] Cross-check of m70199",ISO/IEC JTC1/SC29/WG04 MPEG2024, M70484, Kemer, Türkiye, 4-8 November 2024.
- O. Stankiewicz, S. Różek, S. Maćkowiak, T. Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, M. Domański, "[VCM] On RoI-based tool configuration",ISO/IEC JTC1/SC29/WG04 MPEG2024, M70539, Kemer, Türkiye, 4-8 November 2024.
- O. Stankiewicz, S. Różek, M. Lorkiewicz, T. Grajek, S. Maćkowiak, M. Wawrzyniak, J. Stankowski, M. Domański, "[VCM] Background transmission using RoI-based retargeting",ISO/IEC JTC1/SC29/WG04 MPEG2024, M70542, Kemer, Türkiye, 4-8 November 2024.
- M. Domański, M. Lorkiewicz, Hubert Żabiński, T. Grajek, O. Stankiewicz, S. Różek, S. Maćkowiak, J. Stankowski, "[VCM] Improved neural-network-based chroma reconstruction",ISO/IEC JTC1/SC29/WG04 MPEG2025, M70724, Genewa, Switzerland, 20-24 January 2025.
- D. Mieloch, M. Lorkiewicz, J. Stankowski, "Non-EE2: Picture-level mirroring and rotation",ISO/IEC JTC1/SC29/WG04 MPEG2025, M70906, Genewa, Switzerland, 20-24 January 2025.
- D. Mieloch, M. Lorkiewicz, A. Dziembowski, J. Stankowski, D. Karwowski, "Non-EE2: Adaptive picture-level vertical mirroring",ISO/IEC JTC1/SC29/WG04 MPEG2025, M70908, Genewa, Switzerland, 20-24 January 2025.

- D. Karwowski, D. Mieloch, M. Lorkiewicz, "Non-EE2: Optimization of probability estimation in CABAC",ISO/IEC JTC1/SC29/WG04 MPEG2025, M70909, Genewa, Switzerland, 20-24 January 2025.
- S. Różek, O. Stankiewicz, S. Maćkowiak, T. Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, M. Domański, "[VCM][CE0] Results of CE0.1",ISO/IEC JTC1/SC29/WG04 MPEG2025, M71404, Genewa, Switzerland, 20-24 January 2025.
- S. Różek, O. Stankiewicz, S. Maćkowiak, T. Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, M. Domański, "[VCM][CE0] Cross-check of CE0.5",ISO/IEC JTC1/SC29/WG04 MPEG2025, M71408, Genewa, Switzerland, 20-24 January 2025.
- S. Różek, O. Stankiewicz, M. Lorkiewicz, H. Żabiński, S. Maćkowiak, T. Grajek, M Wawrzyniak, J. Stankowski, M. Domański, "[VCM][CE6.1] Automatic quality control for ROI-based tools",ISO/IEC JTC1/SC29/WG04 MPEG2025, M71410, Genewa, Switzerland, 20-24 January 2025.
- O. Stankiewicz, S. Różek, M. Lorkiewicz, T. Grajek, S. Maćkowiak, H. Żabiński, M Wawrzyniak, J. Stankowski, M. Domański, "[VCM] [CE5.1] Background compression using RoI-based retargeting",ISO/IEC JTC1/SC29/WG04 MPEG2025, M71446, Genewa, Switzerland, 20-24 January 2025.
- S. Różek, O. Stankiewicz, S. Maćkowiak, T Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, M. Domański, "[VCM][CE6] Cross-check of CE6.2",ISO/IEC JTC1/SC29/WG04 MPEG2025, M71459, Genewa, Switzerland, 20-24 January 2025.
- S. Różek, O. Stankiewicz, S. Maćkowiak, T Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, H. Żabiński, M. Domański, "[VCM] Cross-check of m71111 (SR interpolation method)",ISO/IEC JTC1/SC29/WG04 MPEG2025, M71471, Genewa, Switzerland, 20-24 January 2025.
- S. Różek, O. Stankiewicz, S. Maćkowiak, T Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, M. Domański, "[VCM][CE5] Cross-check of CE5.2",ISO/IEC JTC1/SC29/WG04 MPEG2025, M71576, Genewa, Switzerland, 20-24 January 2025.
- S. Różek, O. Stankiewicz, S. Maćkowiak, T Grajek, M. Wawrzyniak, J. Stankowski, M. Lorkiewicz, M. Domański, "[VCM] Cross-check of m71302",ISO/IEC JTC1/SC29/WG04 MPEG2025, M71596, Genewa, Switzerland, 20-24 January 2025.

Patent Application to World Intellectual Property Organization (WIPO)

- M. Domański, T. Grajek, M. Lorkiewicz, S. Maćkowiak, S. Różek, O. Stankiewicz, M. Wawrzyniak, Application: EP23461544, 31.03.2023, “Image data coding methods and systems”, also as: CN2023117570W, WO2024198240A1.

12 References

[Ab16] M. Abadi et. Al. ”TensorFlow: a system for large-scale machine learning” In Proc. 12th USENIX conference on Operating Systems Design and Implementation, Nov 2016, USA, pp. 265–283.

[Ab22] B. Abdallah, S. Ben Jdidia, F. Belghith, M. Ali Ben Ayed and N. Masmoudi, "A CNN-based QTMT partitioning decision for the VVC standard," 2022 IEEE International Conference on Design & Test of Integrated Micro & Nano-Systems (DTS), Cairo, Egypt, 2022, pp. 1-5, doi: 10.1109/DTS55284.2022.9809888.

[Ag17] E. Agustsson and R. Timofte, "NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study," IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, 2017, pp. 1122-1131, doi: 10.1109/CVPRW.2017.150.

[Al17] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey, 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

[Am18] H. Amer, A. Rashwan and E. -h. Yang, "Fully Connected Network for HEVC CU Split Decision equipped with Laplacian Transparent Composite Model," 2018 Picture Coding Symposium (PCS), San Francisco, CA, USA, 2018, pp. 189-193, doi: 10.1109/PCS.2018.8456290.

[Am20] M. Amna, W. Imen and S. F. Ezahra, "LeNet5-Based approach for fast intra coding," 2020 10th International Symposium on Signal, Image, Video and Communications (ISIVC), Saint-Etienne, France, 2021, pp. 1-4, doi: 10.1109/ISIVC49222.2021.9487529.

[Am21] M. Amna, W. Imen and S. F. Ezahra, "Deep Learning For Intra Frame Coding," 2021 International Conference on Engineering and Emerging Technologies (ICEET), Istanbul, Turkey, 2021, pp. 1-4, doi: 10.1109/ICEET53442.2021.9659742.

[Ap24] “Supported media formats in Motion”, Apple, [Online] Available: https://support.apple.com/en-au/guide/motion/motn1252ada3/mac.

[AV1] Y. Chen et al., “An Overview of Core Coding Tools in the AV1 Video Codec,” in 2018 Picture Coding Symposium PCS 2018 - Proceedings, pp. 41–45, 2018. doi: 10.1109/PCS.2018.8456249

[AVC] Generic Coding of Audio-Visual Objects, Part10: Advanced Video Coding, Standard ISO/IEC 14496-10, Mar. 2006

[Ba22A] S. Bakkouri and A. Elyousfi, "Early Termination of CU Partition Based on Boosting Neural Network for 3D-HEVC Inter-Coding," in IEEE Access, vol. 10, pp. 13870-13883, 2022, doi: 10.1109/ACCESS.2022.3147502.

[Ba66] L. E. Baum, T. Petrie, „Statistical Inference for Probabilistic Functions of Finite State Markov Chains". The Annals of Mathematical Statistics 37, Vol. 6, pages 1554–1563., doi:10.1214/aoms/1177699147, 1966.

[Ba97] R. Bartels, J. Beatty, B. Barsky " An Introduction to Splines for Use in Computer Graphics and Geometric Modeling", Morgan Kaufmann, Los Altos, CA, 1987.

[Bi06] Ch. M. Bishop "Pattern Recognition and Machine Learning", Springer, 2006, ISBN 0-387-31073-8.

[Bj01] G. Bjøntegaard, "Calculation of average PSNR differences between RD curves", ITU-T SG16 / Q6, Doc. VCEG-M33, 2001.

[Bo12] F. Bossen, B. Bross, K. Suhring and D. Flynn, "HEVC Complexity and Implementation Analysis," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1685-1696, Dec. 2012, doi: 10.1109/TCSVT.2012.2221255.

[Bo68] L. Boltzmann, "Studien über das Gleichgewicht der lebendigen Kraft zwischen bewegten materiellen Punkten", Wiener Berichte. 58: 517–560, 1868

[Bo98] L. Bottou, "Online algorithms and stochastic approximations", In D. Saad, editor, "Online Learning in Neural Networks",Cambridge University Press, Cambridge, UK, 1998

[Br21A] B. Bross, J. Chen, J. -R. Ohm, G. J. Sullivan and Y. -K. Wang, "Developments in International Video Coding Standardization After AVC, With an Overview of Versatile Video Coding (VVC)," in Proceedings of the IEEE, vol. 109, no. 9, pp. 1463-1493, Sept. 2021, doi: 10.1109/JPROC.2020.3043399

[Br21B] B. Bross et al., "Overview of the Versatile Video Coding (VVC) Standard and its Applications," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 31, no. 10, pp. 3736-3764, Oct. 2021, doi: 10.1109/TCSVT.2021.3101953.

[Br89] J. S. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters". In Proceedings of the 2nd International Conference on Neural Information, Processing Systems (NIPS'89), MIT Press, Cambridge, MA, USA, 211–217, 1989.

[Br90] J.S. Bridle "Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition", In: Soulié, F.F., Hérault, J. (eds) Neurocomputing. NATO ASI Series, vol 68. Springer, Berlin, Heidelberg, 1990 https://doi.org/10.1007/978-3-642-76153-9_28

[BT709] "BT.709 : Parameter values for the HDTV standards for production and international programme exchange", ITU-R Rec. 709-06. June 7, 2015.

[Ca20] F. Cao and Q. Bao, "A Survey On Image Semantic Segmentation Methods With Convolutional Neural Network," 2020 International Conference on Communications, Information System and Computer Engineering (CISCE), Kuala Lumpur, Malaysia, 2020, pp. 458-462, doi: 10.1109/CISCE50729.2020.00103.

[Ce15] Y. -F. Cen, W. -L. Wang, X. -W. Yao, "A fast CU depth decision mechanism for HEVC", Information Processing Letters, Volume 115, Issue 9, 2015, Pages 719-724, ISSN 0020-0190,https://doi.org/10.1016/j.ipl.2015.04.001.

[Ce21] E Çetinkaya, H. Amirpour, M. Ghanbari, C. Timmerer, “CTU depth decision algorithms for HEVC: A survey”. Signal Processing: Image Communication, 2021. 99. 116442. 10.1016/j.image.2021.116442.

[Ch13] G. Chen, Z. Pei, L. Sun, Z. Liu and T. Ikenaga, "Fast intra prediction for HEVC based on pixel gradient statistics and mode refinement," 2013 IEEE China Summit and International Conference on Signal and Information Processing, Beijing, China, 2013, pp. 514-517, doi: 10.1109/ChinaSIP.2013.6625393.

[Ch15] F. Chollet, et. all "Keras", 2015, Available: https://github.com/fchollet/keras

[Ch18] K. Chen, X. Zeng and Y. Fan, "CNN Oriented Fast CU Partition Decision and PU Mode Decision for HEVC Intra Encoding," 2018 14th IEEE International Conference on Solid-

State and Integrated Circuit Technology (ICSICT), Qingdao, China, 2018, pp. 1-3, doi: 10.1109/ICSICT.2018.8564981.

[Ch19A] Y. Chen, L. Yu, T. Li, H. Wang and S. Wang, "Fast CU Size Decision Based on AQ-CNN for Depth Intra Coding in 3D-HEVC," 2019 Data Compression Conference (DCC), Snowbird, UT, USA, 2019, pp. 561-561, doi: 10.1109/DCC.2019.00073.

[Ch19B] D. Choi, Ch. Shallue, Z. Nado, J. Lee, Ch. Maddison, G. Dahl, "On Empirical Comparisons of Optimizers for Deep Learning", 2019, 10.48550/arXiv.1910.05446.

[Ch20] Z. Chen, J. Shi and W. Li, "Learned Fast HEVC Intra Coding," in IEEE Transactions on Image Processing, vol. 29, pp. 5431-5446, 2020, doi: 10.1109/TIP.2020.2982832.

[Ci12] D. Ciregan, U. Meier and J. Schmidhuber, "Multi-column deep neural networks for image classification," 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 2012, pp. 3642-3649, doi: 10.1109/CVPR.2012.6248110.

[Ci22] Y. -S. Ciou, M. -J. Chen, C. -H. Yeh, C. -R. Lu, M. -C. Hsieh and C. Lo, "Fast Multi-Type Tree Partition for H.266/VVC Inter Coding," 2022 IEEE International Conference on Consumer Electronics - Taiwan, Taipei, Taiwan, 2022, pp. 471-472, doi: 10.1109/ICCE-Taiwan55306.2022.9869026.

[CMAKE] CMake documentation, Available: https://cmake.org/documentation/

[Co06] T. Cover, J. Thomas, "Elements of Information Theory, 2nd Edition", Wiley-Interscience, 2006.

[Co12] G. Correa, P. Assuncao, L. Agostini and L. A. da Silva Cruz, "Performance and Computational Complexity Assessment of High-Efficiency Video Encoders," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1899-1909, Dec. 2012, doi: 10.1109/TCSVT.2012.2223411

[Co15] G. Correa, P. A. Assuncao, L. V. Agostini and L. A. da Silva Cruz, "Fast HEVC Encoding Decisions Using Data Mining," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 25, no. 4, pp. 660-673, April 2015, doi: 10.1109/TCSVT.2014.2363753

[Co16] G. Corrêa, P. A. Assunção, L. V. Agostini and L. A. da Silva Cruz, "Pareto-Based Method for High Efficiency Video Coding With Limited Encoding Time," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 26, no. 9, pp. 1734-1745, Sept. 2016, doi: 10.1109/TCSVT.2015.2469533.

[Co23] M. Coban, F. Le Léannec, K. Naser, J. Ström and L. Zhang, "Algorithm description of Enhanced Compression Model 10 (ECM 10)", JVET-AE2025, July 2023.

[CPP] Programming Language C++, Standars ISO/IEC 14882:2014, Geneva, Switzerland: International Organization for Standardization (ISO), 2014.

[Cr16] O. C. Cristina, U. R. Mihnea and P. Ionut, "HEVC intra partitioning and mode decision using histograms of oriented gradients," 2016 12th IEEE International Symposium on Electronics and Telecommunications (ISETC), Timisoara, Romania, 2016, pp. 277-280, doi: 10.1109/ISETC.2016.7781111.

[CTCHEVC] "Common test conditions and software reference configurations", Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 12th Meeting, Document: JCTVC-L1100, WG11 m28412, Geneva, Switzerland, 2013.

[CTCVVC] F Bossen, J Boyce, X Li, V Seregin, and K Shring, “Vtm common test conditions and software reference configurations for sdr video,” Joint Video Exploration Team (JVET), JVET-T2010, October, 2020.

[Da15] D-T Dang-Nguyen, C. Pasquini, V. Conotter, G. Boato, "RAISE: a raw images dataset for digital image forensics" In Proceedings of the 6th ACM Multimedia Systems Conference (MMSys '15). Association for Computing Machinery, New York, NY, USA, 219–224., 2015, https://doi.org/10.1145/2713168.2713194.

[De09] J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

[De16] X. Deng, M. Xu and C. Li, "Hierarchical Complexity Control of HEVC for Live Video Encoding," in IEEE Access, vol. 4, pp. 7014-7027, 2016, doi: 10.1109/ACCESS.2016.2612691.

[Do10] M. Domański, Obraz cyfrowy. Wydawnictwa Komunikacji i Łączności, 2010

[Do16] T. Dozat,“Incorporating Nesterov Momentum into Adam,” in ICLR workshop, Caribe Hilton, San Juan, Puerto Rico, May 2016

[Do20] A. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, 2020, ArXiv, abs/2010.11929.

[Du15] B. Du, W. -C. Siu and X. Yang, "Fast CU partition strategy for HEVC intra-frame coding using learning approach via random forests," 2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), Hong Kong, China, 2015, pp. 1085-1090, doi: 10.1109/APSIPA.2015.7415439.

[Du18] K. Duan, P. Liu, K. Jia and Z. Feng, "An Adaptive Quad-Tree Depth Range Prediction Mechanism for HEVC," in IEEE Access, vol. 6, pp. 54195-54206, 2018, doi: 10.1109/ACCESS.2018.2871558.

[Dz22] A. Dziembowski, D. Mieloch, J. Stankowski and A. Grzelka, "IV-PSNR—The Objective Quality Metric for Immersive Video Applications," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 32, no. 11, pp. 7575-7591, Nov. 2022, doi: 10.1109/TCSVT.2022.3179575.

[Er24] Ericsson “Ericsson Mobility Report 2024”, June 2024, [Online], Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/reports/june-2024

[Fa24] R. Farahani, Z. Azimi, C. Timmerer,R. Prodan, “Towards AI-Assisted Sustainable Adaptive Video Streaming Systems: Tutorial and Survey”, 2024.

[Fe18A] Z. Feng, P. Liu, K. Jia and K. Duan, "A deep convolutional neural network approach for complexity reduction on intra-mode HEVC," 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), Chongqing, China, 2018, pp. 551-555, doi: 10.1109/ICIVC.2018.8492898.

[Fe18B] Z. Feng, P. Liu, K. Jia and K. Duan, "Fast Intra CTU Depth Decision for HEVC," in IEEE Access, vol. 6, pp. 45262-45269, 2018, doi: 10.1109/ACCESS.2018.2864881.

[Fe21] A. Feng, C. Gao, L. Li, D. Liu and F. Wu, "Cnn-Based Depth Map Prediction for Fast Block Partitioning in HEVC Intra Coding," 2021 IEEE International Conference on Multimedia and Expo (ICME), Shenzhen, China, 2021, pp. 1-6, doi: 10.1109/ICME51207.2021.9428069.

[Fe22] G. Fengwei, C. Yong and X. Shuai, "Fast Algorithm Design of HEVC Intra Prediction," 2022 International Conference on Innovations and Development of Information

Technologies and Robotics (IDITR), Chengdu, China, 2022, pp. 38-42, doi: 10.1109/IDITR54676.2022.9796501.

[FFMPEG] ffmpeg software, Available: https://ffmpeg.org/

[Fo99] Folland, G.B. (1999). Real Analysis: Modern Techniques and Their Applications (Second ed.). John Wiley & Sons, Inc. ISBN 978-0-471-31716-6.

[For1] The Brainy Insights " Video Streaming Market Size by Type (Live Video Streaming and Non-Linear Video Streaming), Deployment (Cloud, On-Premises and Hybrid), Solution (Internet Protocol TV, Over-the-Top (OTT) and Pay-TV), Service (Consulting, Managed Services and Training & Support), Platform (Gaming Consoles, Laptops & Desktops, Smartphones & Tablets, and Smart TV), Revenue Model (Advertising, Rental and Subscription), End User (Enterprise and Consumer), Regions, Global Industry Analysis, Share, Growth, Trends, and Forecast 2024 to 2033", Report Id: TBI-13729, August-2024, [Online] Available: https://www.thebrainyinsights.com/report/video-streaming-market-13729

[For2] KBV Research " Global Machine Learning Market Size, Share & Industry Trends Analysis Report By Enterprise Size (Large Enterprises, and SMEs), By Component (Services, Software, and Hardware), By End-use, By Regional Outlook and Forecast, 2023 - 2030", Report Id: KBV-16536, July-2023, Online, Available: https://www.kbvresearch.com/machine-learning-market/

[For3] Kings Research “Machine Learning Market Size, Share, Growth & Industry Analysis, By Enterprise Type (Small & Medium Sized Enterprises, Large Enterprises), By Deployment (Cloud, On Premise), By End User (BFSI, IT & Telecommunication, Healthcare, Retail, Advertising & Media) and Regional Analysis, 2023-2030”, Report ID: KR172, November-2023, Online, Available: https://www.kingsresearch.com/machine-learning-market-172

[For4] Allied Market Research “Neural Processor Market Size, Share, Competitive Landscape and Trend Analysis Report, by Application, by End User : Global Opportunity Analysis and Industry Forecast, 2021-2031”, Report Code: A13155, January 2023, Online, Available: https://www.alliedmarketresearch.com/neural-processor-market-A13155

[Fu69] K. Fukushima, "Visual Feature Extraction by a Multilayered Network of Analog Threshold Elements," in IEEE Transactions on Systems Science and Cybernetics, vol. 5, no. 4, pp. 322-333, Oct. 1969, doi: 10.1109/TSSC.1969.300225.

[Fu82] K. Fukushima S. Miyake "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition". Competition and Cooperation in Neural Nets. Lecture Notes in Biomathematics. Vol. 45. Springer. pp. 267–285, 1982

[Ga16] Y. Gao, P. Liu, Y. Wu and K. Jia, "Quadtree Degeneration for HEVC," in IEEE Transactions on Multimedia, vol. 18, no. 12, pp. 2321-2330, Dec. 2016, doi: 10.1109/TMM.2016.2598481.

[Ga17] B. Gao, L. Pavel "On the Properties of the Softmax Function with Application in Game Theory and Reinforcement Learning",2017, 10.48550/arXiv.1704.00805.

[Ga22] V. Gallego, D. r. Insua, Current Advances in Neural Networks” Annual Review of Statistics and Its Application, Vol 9., 2022, doi: https://doi.org/10.1146/annurev-statistics-040220-112019.

[Ge84] S. Geman, G. Geman “Stochastic Relaxation, Gibbs Distribuition and the Bayesian Restoration of Images”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 6, pages 721-741, 1984.

[Gi02] J. W. Gibbs, "Elementary Principles in Statistical Mechanics", 1902.

[Gl10A] X. Glorot, A. Bordes, Y. Bengio "Deep Sparse Rectifier Neural Networks" Journal of Machine Learning Research, vol 15, 2010 .

[Gl10B] X. Glorot, Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks", in Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, Chia Laguna Resort, Sardinia, Italy. Volume 9 of JMLR: W&CP 9.

[Go16] I. Goodfellow, Y. Bengio, A. Courville, ‘Deep Learning’, MIT Press, 2026.

[Ha08] S. Haykin, ‘Neural Networks and Learning Machines’, Pearson, 2008.

[Ha16] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[Ha23] E. Hassan,M.Y. Shams,N.A. Hikal, et al. "The effect of choosing optimizer algorithms to improve computer vision tasks: a comparative study" Multimed Tools Appl 82, 16591–16633, 2023, https://doi.org/10.1007/s11042-022-13820-0

[He15] K. He, X. Zhang, S. Ren and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 1026-1034, doi: 10.1109/ICCV.2015.123.

[He20] B. Heidari, M. Ramezanpour, "Reduction of intra-coding time for HEVC based on temporary direction map" J Real-Time Image Proc 17, 567–579, 2020, https://doi.org/10.1007/s11554-018-0815-7

[He21] Q. He, W. Wu, L. Luo, C. Zhu and H. Guo, "Random Forest Based Fast CU Partition for VVC Intra Coding," 2021 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Chengdu, China, 2021, pp. 1-4, doi: 10.1109/BMSB53066.2021.9547117.

[HEVC] Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 2: High Efficiency Video Coding, ISO/IEC IS 23008-2, also ITU-T Rec. H.265, Geneva, Switzerland, 2013.

[HM] 2D HEVC reference codec available online: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.23.

[Hu17] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 2261-2269, doi: 10.1109/CVPR.2017.243.

[Hu21A] B. Huang, Z. Chen, K. Su, J. Chen and N. Ling, "Low-Complexity Rate-Distortion Optimization for HEVC Encoders," in IEEE Transactions on Broadcasting, vol. 67, no. 3, pp. 721-735, Sept. 2021, doi: 10.1109/TBC.2021.3077771.

[Hu21B] Y. Huang, L. Song, R. Xie, E. Izquierdo and W. Zhang, "Modeling Acceleration Properties for Flexible INTRA HEVC Complexity Control," in IEEE Transactions on

Circuits and Systems for Video Technology, vol. 31, no. 11, pp. 4454-4469, Nov. 2021, doi: 10.1109/TCSVT.2021.3053635.

[Hu23] Y. Huang, J. Xu, C. Zhu, L. Song and W. Zhang, "Precise Encoding Complexity Control for Versatile Video Coding," in IEEE Transactions on Broadcasting, vol. 69, no. 1, pp. 33-48, March 2023, doi: 10.1109/TBC.2022.3187813.

[In24] Intel Corporation, “Quick overview of Intel’s Neural Processing Unit (NPU)” Available: https://intel.github.io/intel-npu-acceleration-library/npu.html

[Io15] S. Ioffe, Ch. Szegedy "Batch normalization: accelerating deep network training by reducing internal covariate shift." In Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37 (ICML'15). JMLR.org, 448–456. 2015.

[Is21] M. Z. Islam and B. Ahmed, "Visualization of CTU Partitioning And Time Complexity Analysis in HEVC and VVC," 2021 3rd International Conference on Electrical & Electronic Engineering (ICEEE), Rajshahi, Bangladesh, 2021, pp. 169-172, doi: 10.1109/ICEEE54059.2021.9718772.

[Ja07] V. Jain et al., "Supervised Learning of Image Restoration with Convolutional Networks," 2007 IEEE 11th International Conference on Computer Vision, Rio de Janeiro, Brazil, 2007, pp. 1-8, doi: 10.1109/ICCV.2007.4408909.

[Ja19] M. Jamali and S. Coulombe, "Fast HEVC Intra Mode Decision Based on RDO Cost Prediction," in IEEE Transactions on Broadcasting, vol. 65, no. 1, pp. 109-122, March 2019, doi: 10.1109/TBC.2018.2847464

[Ji14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell "Caffe: Convolutional Architecture for Fast Feature Embedding", In Proceedings of the 22nd ACM international conference on Multimedia (MM '14). Association for Computing Machinery, New York, NY, USA, 675–678, 2014 https://doi.org/10.1145/2647868.2654889

[JPEG] “Information technology – Digital compression and coding of continuous-tone still images: JPEG File Interchange Format (JFIF)”, Recommendation ITU-T T.871, International Telecommunication Union – Standardization Sector (ITU-T), 2011

[Jy24] H. Jyothi, M. Komala and S. Mallikarjunaswamy, "A Comprehensive Survey on Technologies in Video-based Event Detection and Recognition Using Machine Learning and Deep Learning Techniques," 2024 Second International Conference on Networks, Multimedia and Information Technology (NMITCON), Bengaluru, India, 2024, pp. 1-5, doi: 10.1109/NMITCON62075.2024.10698959.

[Ka18] T. Katayama, K. Kuroda, W. Shi, T. Song and T. Shimamoto, "Low-complexity intra coding algorithm based on convolutional neural network for HEVC," 2018 International Conference on Information and Computer Technologies (ICICT), DeKalb, IL, USA, 2018, pp. 115-118, doi: 10.1109/INFOCT.2018.8356852.

[Ka19] D. Karwowski, Zrozumieć Kompresję Obrazu. Podstawy Technik Kodowania Stratnego oraz Bezstratnego Obrazów, 2019.

[Kh13] M. U. K. Khan, M. Shafique and J. Henkel, "An adaptive complexity reduction scheme with fast prediction unit decision for HEVC intra encoding," 2013 IEEE International Conference on Image Processing, Melbourne, VIC, Australia, 2013, pp. 1578-1582, doi: 10.1109/ICIP.2013.6738325.

[Kh16] P. Khurana, A. Sharma, S. N. Singh and P. K. Singh, "A survey on object recognition and segmentation techniques," 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2016, pp. 3822-3826.

[Kh24] M. Khadir, M. F. Hashmi, D. M. Kotambkar and A. Gupta, "Innovative Insights: A Review of Deep Learning Methods for Enhanced Video Compression," in IEEE Access, doi: 10.1109/ACCESS.2024.3450814

[Ki13] J. Kim, Y. Choe and Y. -G. Kim, "Fast Coding Unit size decision algorithm for intra coding in HEVC," 2013 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2013, pp. 637-638, doi: 10.1109/ICCE.2013.6487050.

[Ki14] D. Kingma, J. Ba, " Adam: A Method for Stochastic Optimization", International Conference on Learning Representations, 2014.

[Ki16A] N. Kim, S. Jeon, H. J. Shim, B. Jeon, S. -C. Lim and H. Ko, "Adaptive keypoint-based CU depth decision for HEVC intra coding," 2016 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Nara, Japan, 2016, pp. 1-3, doi: 10.1109/BMSB.2016.7521923.

[Ki16B] H. -S. Kim and R. -H. Park, "Fast CU Partitioning Algorithm for HEVC Using an Online-Learning-Based Bayesian Decision Rule," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 26, no. 1, pp. 130-138, Jan. 2016, doi: 10.1109/TCSVT.2015.2444672.

[Ki19] K. Kim and W. W. Ro, "Fast CU Depth Decision for HEVC Using Neural Networks," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 29, no. 5, pp. 1462-1473, May 2019, doi: 10.1109/TCSVT.2018.2839113

[Kr12] A. Krizhevsky, I. Sutskever, G. E. Hinton "ImageNet classification with deep convolutional neural networks", In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12), 2012, Curran Associates Inc., Red Hook, NY, USA, 1097–1105.

[Ku20] Y. -T. Kuo, P. -Y. Chen and H. -C. Lin, "A Spatiotemporal Content-Based CU Size Decision Algorithm for HEVC," in IEEE Transactions on Broadcasting, vol. 66, no. 1, pp. 100-112, March 2020, doi: 10.1109/TBC.2019.2960938

[Ku20A] W. Kuang, Y. -L. Chan, S. -H. Tsang and W. -C. Siu, "DeepSCC: Deep Learning-Based Fast Prediction Network for Screen Content Coding," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 7, pp. 1917-1932, July 2020, doi: 10.1109/TCSVT.2019.2929317.

[La20] J. Laitinen, A. Lemmetti and J. Vanne, "Real-Time Implementation Of Scalable Hevc Encoder," 2020 IEEE International Conference on Image Processing (ICIP), Abu Dhabi, United Arab Emirates, 2020, pp. 1166-1170, doi: 10.1109/ICIP40778.2020.9191135.

[Le10] Y. LeCun, K. Kavukcuoglu and C. Farabet, "Convolutional networks and applications in vision," Proceedings of 2010 IEEE International Symposium on Circuits and Systems, Paris, France, 2010, pp. 253-256, doi: 10.1109/ISCAS.2010.5537907.

[Le15] J. Lee, S. Kim, K. Lim and S. Lee, "A Fast CU Size Decision Algorithm for HEVC," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 25, no. 3, pp. 411-421, March 2015, doi: 10.1109/TCSVT.2014.2339612.

[Le18] C. -Y. Lee, P. Gallagher and Z. Tu, "Generalizing Pooling Functions in CNNs: Mixed, Gated, and Tree," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 4, pp. 863-875, 1 April 2018, doi: 10.1109/TPAMI.2017.2703082.

[Le20] S. Leijnen, F. van Veen, “The Neural Network Zoo”. proceedings, MDPI, Vol 47, 12 May 2020, doi: https://doi.org/10.3390/proceedings2020047009.

[Le98] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based learning applied to document recognition". Proceedings of the IEEE, 86(11), 2278-2324, 1998

[Li14B] M. Li, T. Zhang, Y. Chen, A. Smola "Efficient mini-batch training for stochastic optimization." Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, doi:10.1145/2623330.2623612.

[Li15] T.-Y. Lin et al. "Microsoft COCO: Common Objects in Context", arxiv, 2015, avaliable at: https://arxiv.org/abs/1405.0312

[Li16A] D. Liu, X. Liu and Y. Li, "A Fast Coding Unit Size Decision Algorithm for HEVC Intra Coding Based on Image Complexity," 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Chengdu, China, 2016, pp. 874-877, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2016.180.

[Li16B] Z. Liu, X. Yu, Y. Gao, S. Chen, X. Ji and D. Wang, "CU Partition Mode Decision for HEVC Hardwired Intra Encoder Using Convolution Neural Network," in IEEE Transactions on Image Processing, vol. 25, no. 11, pp. 5088-5103, Nov. 2016, doi: 10.1109/TIP.2016.2601264.

[Li16C] Z. Liu, X. Yu, S. Chen and D. Wang, "CNN oriented fast HEVC intra CU mode decision," 2016 IEEE International Symposium on Circuits and Systems (ISCAS), Montreal, QC, Canada, 2016, pp. 2270-2273, doi: 10.1109/ISCAS.2016.7539036.

[Li17A] Y. Li, G. Yang, Y. Zhu, X. Ding and X. Sun, "Adaptive Inter CU Depth Decision for HEVC Using Optimal Selection Model and Encoding Parameters," in IEEE Transactions on Broadcasting, vol. 63, no. 3, pp. 535-546, Sept. 2017, doi: 10.1109/TBC.2017.2704423.

[Li17B] T. Li, M. Xu and X. Deng, "A deep convolutional neural network approach for complexity reduction on intra-mode HEVC," 2017 IEEE International Conference on Multimedia and Expo (ICME), Hong Kong, China, 2017, pp. 1255-1260, doi: 10.1109/ICME.2017.8019316.

[Li18] Y. Li, Z. Liu, X. Ji and D. Wang, "CNN Based CU Partition Mode Decision Algorithm for HEVC Inter Coding," 2018 25th IEEE International Conference on Image Processing (ICIP), Athens, Greece, 2018, pp. 993-997, doi: 10.1109/ICIP.2018.8451290.

[Li20A] Y. Liu and A. Wei, "A CU Fast Division Decision Algorithm with Low Complexity for HEVC," 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chongqing, China, 2020, pp. 1028-1032, doi: 10.1109/ITNEC48623.2020.9084705.

[Li20B] T. Li, M. Xu, X. Deng and L. Shen, "Accelerate CTU Partition to Real Time for HEVC Encoding With Complexity Control," in IEEE Transactions on Image Processing, vol. 29, pp. 7482-7496, 2020, doi: 10.1109/TIP.2020.3003730.

[Li20C] J. Liu, Y. Sun, C. Han, Z. Dou and W. Li, "Deep Representation Learning on Long-Tailed Data: A Learnable Embedding Augmentation Perspective," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 2967-2976, doi: 10.1109/CVPR42600.2020.00304.

[Li21A] T. -L. Lin et al., "Efficient Quadtree Search for HEVC Coding Units for V-PCC," in IEEE Access, vol. 9, pp. 139109-139121, 2021, doi: 10.1109/ACCESS.2021.3118806.

[Li21B] Y. Li, L. Li, Z. Zhuang, Y. Fang and Y. Yang, "ResNet Approach for Coding Unit Fast Splitting Decision of HEVC Intra Coding," 2021 IEEE Sixth International Conference on Data Science in Cyberspace (DSC), Shenzhen, China, 2021, pp. 130-135, doi: 10.1109/DSC53577.2021.00025.

[Li22A] J. Li, S. Zhang and F. Yang, "Random Forest Accelerated CU Partition for Inter Prediction in H.266/VVC," 2022 IEEE International Conference on Multimedia and Expo (ICME), Taipei, Taiwan, 2022, pp. 01-06, doi: 10.1109/ICME52920.2022.9859664.

[Li22B] Y. Li et al., "Optimal Stopping Theory-Enabled VVC Intra Prediction with Texture," 2022 7th International Conference on Communication, Image and Signal Processing (CCISP), Chengdu, China, 2022, pp. 530-535, doi: 10.1109/CCISP55629.2022.9974416.

[Li22C] Y. Liu, M. Abdoli, T. Guionnet, C. Guillemot and A. Roumy, "Light-Weight CNN-Based VVC Inter Partitioning Acceleration," 2022 IEEE 14th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP), Nafplio, Greece, 2022, pp. 1-5, doi: 10.1109/IVMSP54334.2022.9816276.

[Li22D] Y. Li, L. Zhang and J. Xu, "CNN-based Partitioning Structure Prediction for VVC Intra Speedup: Bottom-Up-based and Top-Down-based," 2022 IEEE International Symposium on Circuits and Systems (ISCAS), Austin, TX, USA, 2022, pp. 1953-1957, doi: 10.1109/ISCAS48785.2022.9937673.

[LIBTORCH] "PyTorch C++ API" Available: https://pytorch.org/cppdocs/index.html

[Lo21] M. Lorkiewicz, O. Stankiewicz, M. Domanski, H. -M. Hang and W. -H. Peng, "Fast Selection of INTRA CTU Partitioning in HEVC Encoders using Artificial Neural Networks" 2021 Signal Processing Symposium (SPSympo), LODZ, Poland, 2021, pp. 177-182, doi: 10.1109/SPSympo51155.2020.9593483.

[Lo23] M. Lorkiewicz, O. Stankiewicz, M. Domański, H. -M. Hang and W. -H. Peng, "Complexity Reduction of ANN Model for CU Size Selection in HEVC," 2023 Signal Processing Symposium (SPSympo), Karpacz, Poland, 2023, pp. 111-116, doi: 10.23919/SPSympo57300.2023.10302659.

[Lo24] M. Lorkiewicz, O. Stankiewicz, M. Domański, H. -M. Hang and W. -H. Peng, "Complexity-Efficiency Control With ANN-Based CTU Partitioning for Video Encoding," in IEEE Access, vol. 12, pp. 102536-102551, 2024, doi: 10.1109/ACCESS.2024.3433424.

[Lo99] D. G. Lowe, "Object recognition from local scale-invariant features", Proceedings of the International Conference on Computer Vision. Vol. 2. pp. 1150–1157, 1999, doi:10.1109/ICCV.1999.790410.

[Lu13] J. Lu, F. Liang, L. Xie and Y. Luo, "A fast block partition algorithm for HEVC," 2013 9th International Conference on Information, Communications & Signal Processing, Tainan, Taiwan, 2013, pp. 1-5, doi: 10.1109/ICICS.2013.6782918.

[Lu19] J. Lu and Y. Li, "Fast Algorithm for CU Partitioning and Mode Selection in HEVC Intra Prediction," 2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Suzhou, China, 2019, pp. 1-5, doi: 10.1109/CISP-BMEI48845.2019.8966035.

[Lu20] Y. Lu, X. Huang, H. Liu, Y. Zhou, H. Yin and L. Shen, "Hierarchical Classification for Complexity Reduction in HEVC Inter Coding," in IEEE Access, vol. 8, pp. 41690-41704, 2020, doi: 10.1109/ACCESS.2020.2977422.

[Ma03] D. MacKay, "Information Theory, Inference and Learning Algorithms", Cambridge University Press, 2003.

[Ma23A] E. Mahurin, "Qualocmm® Hexagon™ NPU," 2023 IEEE Hot Chips 35 Symposium (HCS), Palo Alto, CA, USA, 2023, pp. 1-19, doi: 10.1109/HCS59251.2023.10254715.

[Ma23B] A. Mao, M. Mohri, Y. Zhong, "Cross-entropy loss functions: theoretical analysis and applications", In Proceedings of the 40th International Conference on Machine Learning (ICML'23), Vol. 202. JMLR.org, Article 992, 23803–23828, 2023

[Mc14] K. McCann, C. Rosewarne, B. Bross, M. Naccari, K. Sharman, G. Sullivan, "High Efficiency Video Coding (HEVC) Test Model 16 (HM 16) Encoder Description, "Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Document:

[Me21A] A. Mercat, A. Mäkinen, J. Sainio, A. Lemmetti, M. Viitanen and J. Vanne, "Comparative Rate-Distortion-Complexity Analysis of VVC and HEVC Video Codecs," in IEEE Access, vol. 9, pp. 67813-67828, 2021, doi: 10.1109/ACCESS.2021.3077116.

[Me21B] V. V. Menon, H. Amirpour, C. Timmerer and M. Ghanbari, "INCEPT: Intra CU Depth Prediction for HEVC," 2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP), Tampere, Finland, 2021, pp. 1-6, doi: 10.1109/MMSP53017.2021.9733517.

[Mi13] K. Miyazawa et al., "Real-time hardware implementation of HEVC video encoder for 1080p HD video," 2013 Picture Coding Symposium (PCS), San Jose, CA, USA, 2013, pp. 225-228, doi: 10.1109/PCS.2013.6737724.

[Mi16] D. Mishkin, J. Matas, "All you need is a good init", In Proc. Of International Conference on Learning Representations (ICLR), 2016.

[Mo15] S. Momcilovic, N. Roma, L. Sousa and I. Milentijevic, "Run-Time Machine Learning for HEVC/H.265 Fast Partitioning Decision," 2015 IEEE International Symposium on Multimedia (ISM), Miami, FL, USA, 2015, pp. 347-350, doi: 10.1109/ISM.2015.70.

[Mo24] B. B. Moser, A. S. Shanbhag, F. Raue, S. Frolov, S. Palacio and A. Dengel, "Diffusion Models, Image Super-Resolution, and Everything: A Survey," in IEEE Transactions on Neural Networks and Learning Systems, 2024, doi: 10.1109/TNNLS.2024.3476671

[MPEG2] Generic Coding of Moving Pictures and Associated Audio Information – Part 2: Video. (MPEG-2), ISO/IEC 13818-2 and ITU-T Rec. H.262, November 1994.

[Na20] N. Najafabadi, M. Ramezanpour "Mass center direction-based decision method for intraprediction in HEVC standard", J. Real-Time Image Process. 17, 5 , 1153–1168, Oct 2020 , https://doi.org/10.1007/s11554-019-00864-z.

[Na24] P. S. Nair and M. S. Nair, "KSVM-Based Fast Intra Mode Prediction in HEVC Using Statistical Features and Sparse Autoencoder," in IEEE Access, vol. 12, pp. 48846-48852, 2024, doi: 10.1109/ACCESS.2024.3382570.

[Ni22] C. -T. Ni, S. -H. Lin, P. -Y. Chen and Y. -T. Chu, "High Efficiency Intra CU Partition and Mode Decision Method for VVC," in IEEE Access, vol. 10, pp. 77759-77771, 2022, doi: 10.1109/ACCESS.2022.3193401.

[Nv23] "Convolutional Layers User's Guide", Nvidia Deep Learning Performance documentaion, Available: https://docs.nvidia.com/deeplearning/performance/dl-performance-convolutional/index.html

[On21] ONNX Runtime developers, "ONNX Runtime", 2021, Available: https://onnxruntime.ai/

[Or22] A. Orhon, Y. Wadhwa, Kim, F. Rossi, Vignesh Jagadeesh, et al., "Deploying Transformers on the Apple Neural Engine", Apple Machine Learning Reseach, Computer Vision, research area Speech and Natural Language Processing, Highlight June 2022.

[Pa14] Z. Pan, S. Kwong, M. -T. Sun and J. Lei, "Early MERGE Mode Decision Based on Motion Estimation and Hierarchical Depth Correlation for HEVC," in IEEE Transactions on Broadcasting, vol. 60, no. 2, pp. 405-412, June 2014, doi: 10.1109/TBC.2014.2321682.

[Pa19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In Proc: 33rd International Conference on Neural Information Processing Systems, Curran Associates Inc., Vancouver BC Canada, 2019, Article 721, pp. 8026–8037. [Online] Available: https://dl.acm.org/doi/10.5555/3454287.3455008.

[Pa20] S. Paul, A. Norkin and A. C. Bovik, "Speeding Up VP9 Intra Encoder With Hierarchical Deep Learning-Based Partition Prediction," in IEEE Transactions on Image Processing, vol. 29, pp. 8134-8148, 2020, doi: 10.1109/TIP.2020.3011270.

[Pi13] M. H. Pinson, "The Consumer Digital Video Library [Best of the Web]," IEEE Signal Processing Magazine, vol. 30, no. 4, pp. 172,174, July 2013 doi: 10.1109/MSP.2013.2258265.

[Pi15] P. O. Pinheiro and R. Collobert, "From image-level to pixel-level labeling with Convolutional Networks," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015, pp. 1713-1721, doi: 10.1109/CVPR.2015.7298780.

[Pr23] A. Prakash and S. Chauhan, "A Comprehensive Survey of Trending Tools and Techniques in Deep Learning," 2023 International Conference on Disruptive Technologies (ICDT), Greater Noida, India, 2023, pp. 289-292, doi: 10.1109/ICDT57929.2023.10151083.

[Qi16] A. Qing, W. Zhou, H. Wei, X. Zhou, G. Zhang and J. Yang, "A fast CU partitioning algorithm in HEVC inter prediction for HD/UHD video," 2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), Jeju, Korea (South), 2016, pp. 1-5, doi: 10.1109/APSIPA.2016.7820718.

[Re19] W. Ren, J. Su, C. Sun and Z. Shi, "An IBP-CNN Based Fast Block Partition For Intra Prediction," 2019 Picture Coding Symposium (PCS), Ningbo, China, 2019, pp. 1-5, doi: 10.1109/PCS48520.2019.8954522.

[Rh12] C.E. Rhee, K. Lee, T. S. Kim, and H. -J. Lee, "A survey of fast mode decision algorithms for inter-prediction and their applications to high efficiency video coding," in IEEE Transactions on Consumer Electronics, vol. 58, no. 4, pp. 1375-1383, November 2012,

[Ri03A] I. Richardson, "The H 264 advanced video compression standard," by John Wiley & Sons, Inc., 2nd edition, 2003.

[Ri03B] I. Richardson, "H.264 and MPEG-4 video compression video coding for nextgeneration multimedia," by John Wiley & Sons, Inc., 2003.

[Ri04] D. Ringach, R. Shapley,"Reverse correlation in neurophysiology", Cognitive Science, Volume 28, Issue 2, 2004, Pages 147-166, ISSN 0364-0213, https://doi.org/10.1016/j.cogsci.2003.11.003.

[Ri24] A. Rico et al., "AMD XDNA™ NPU in Ryzen™ AI Processors," in IEEE Micro, doi: 10.1109/MM.2024.3423692.

[Ro09] G. Van Rossum,F.L. Drake, "Python 3 Reference Manual", Scotts Valley, CA: CreateSpace, 2009.

[Ro51] H. Robbins, S. Monro, "A Stochastic Approximation Method. The Annals of Mathematical Statistics", 22(3), 1951.

[Ry18] S. Ryu and J. Kang, "Machine Learning-Based Fast Angular Prediction Mode Decision Technique in Video Coding," in IEEE Transactions on Image Processing, vol. 27, no. 11, pp. 5525-5538, Nov. 2018, doi: 10.1109/TIP.2018.2857404.

[Sa14] A. M. Saxe, J. McClelland, S. Ganguli "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks", In Proc. Of International Conference on Learning Representations (ICLR), 2014.

[Sa20] M. Saldanha, G. Sanchez, C. Marcon, and L. Agostini, "Complexity Analysis of VVC Intra Coding," 2020 IEEE International Conference on Image Processing (ICIP), Abu Dhabi, United Arab Emirates, 2020, pp. 3119-3123, doi: 10.1109/ICIP40778.2020.9190970.

[Sa23] K. Saravanan, A. Z. Kouzani, “Advancements in On-Device Deep Neural Networks”. information, MDPI Vol 14, 21 August 2023, doi: https://doi.org/10.3390/info14080470

[Sc20] R. Schmidt, F. Schneider, P. Hennigm, "Descending through a Crowded Valley -- Benchmarking Deep Learning Optimizers", 2020, 10.48550/arXiv.2007.01547.

[Sh12] X. Shen, L. Yu and J. Chen, "Fast coding unit size selection for HEVC based on Bayesian decision rule," 2012 Picture Coding Symposium, Krakow, Poland, 2012, pp. 453-456, doi: 10.1109/PCS.2012.6213252.

[Sh13] L. Shen, Z. Zhang and P. An, "Fast CU size decision and mode decision algorithm for HEVC intra coding," in IEEE Transactions on Consumer Electronics, vol. 59, no. 1, pp. 207-213, February 2013, doi: 10.1109/TCE.2013.6490261.

[Sh19] J. Shi, C. Gao and Z. Chen, "Asymmetric-Kernel CNN Based Fast CTU Partition for HEVC Intra Coding," 2019 IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, 2019, pp. 1-5, doi: 10.1109/ISCAS.2019.8702494.

[Sh20] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network,", Physica D: Nonlinear Phenomena, Volume 404, 2020, 132306, ISSN 0167-2789,https://doi.org/10.1016/j.physd.2019.132306.

[Si20] Í. Siqueira, G. Correa and M. Grellert, "Rate-Distortion and Complexity Comparison of HEVC and VVC Video Encoders," 2020 IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS), San Jose, Costa Rica, 2020, pp. 1-4, doi: 10.1109/LASCAS45839.2020.9069036.

[Si20B] K. P. Silva, L. F. Arcaro and R. S. de Oliveira, "Methods for Comparing Execution Times of Different Input Data when Real-Time Tasks Run on Complex Computer Architectures," 2020 X Brazilian Symposium on Computing Systems Engineering (SBESC), Florianopolis, Brazil, 2020, pp. 1-8, doi: 10.1109/SBESC51047.2020.9277839

[So17] N. Song, Z. Liu, X. Ji and D. Wang, "CNN oriented fast PU mode decision for HEVC hardwired intra encoder," 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Montreal, QC, Canada, 2017, pp. 239-243, doi: 10.1109/GlobalSIP.2017.8308640.

[St06] D. B. Stewart "Measuring Execution Time and Real-Time Performance", Embedded Systems Conference, Boston, September 2006.

[St16] J. Stankowski, D. Karwowski, K. Klimaszewski, K. Wegner, O. Stankiewicz and T. Grajek, "Analysis of the complexity of the HEVC motion estimation," 2016 International Conference on Systems, Signals and Image Processing (IWSSIP), Bratislava, Slovakia, 2016, pp. 1-4, doi: 10.1109/IWSSIP.2016.7502731.

[Su12] G. J. Sullivan, J. -R. Ohm, W. -J. Han and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1649-1668, Dec. 2012, doi: 10.1109/TCSVT.2012.2221191.

[Su19] Sh. Sun, Z. Cao, H. Zhu, J. Zhao "A Survey of Optimization Methods from a Machine Learning Perspective", 2019, 10.48550/arXiv.1906.06821.

[Su98] G. J. Sullivan and T. Wiegand, "Rate-Distortion Optimization for Video Compression," IEEE Signal Processing Magazine, Nov. 1998.

[Sw10] K. V. S. Swaroop and K. R. Rao, "Performance analysis and comparison of JM 15.1 and Intel IPP H.264 encoder and decoder," 2010 42nd Southeastern Symposium on System Theory (SSST), Tyler, TX, USA, 2010, pp. 371-375, doi: 10.1109/SSST.2010.5442807.

[Ta16] H. L. Tan, C. C. Ko and S. Rahardja, "Fast Coding Quad-Tree Decisions Using Prediction Residuals Statistics for High Efficiency Video Coding (HEVC)," in IEEE Transactions on Broadcasting, vol. 62, no. 1, pp. 128-133, March 2016, doi: 10.1109/TBC.2015.2505406.

[Ta17] K. -H. Tai, M. -Y. Hsieh, M. -J. Chen, C. -Y. Chen and C. -H. Yeh, "A Fast HEVC Encoding Method Using Depth Information of Collocated CUs and RD Cost Characteristics of PU Modes," in IEEE Transactions on Broadcasting, vol. 63, no. 4, pp. 680-692, Dec. 2017, doi: 10.1109/TBC.2017.2722239.

[Ta19] N. Tang et al., "Fast CTU Partition Decision Algorithm for VVC Intra and Inter Coding," 2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), Bangkok, Thailand, 2019, pp. 361-364, doi: 10.1109/APCCAS47518.2019.8953076.

[Ta22] J. Tang and S. Sun, "Optimization of CU Partition Based on Texture Degree in H.266/VVC," 2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Chiang Mai, Thailand, 2022, pp. 402-408, doi: 10.23919/APSIPAASC55919.2022.9979850.

[TENSORFLOW] Tensorflow API documentation, Available: https://www.tensorflow.org/api_docs.

[To19] P. Topiwala, M. Krishnan, W. Dai, “Performance comparison of VVC, AV1 and EVC,” Proc. SPIE, Applications of Digital Image Proc. XLII, 1113715, 6 Sep. 2019

[Ts22] Y. -H. Tsai, C. -R. Lu, M. -C. Hsieh, M. -J. Chen, C. -M. Yang and C. -H. Yeh, "Fast Intra Coding Algorithm Based on Visual Perception Analysis for H.266/VVC," 2022 IET International Conference on Engineering Technologies and Applications (IET-ICETA), Changhua, Taiwan, 2022, pp. 1-2, doi: 10.1109/IET-ICETA56553.2022.9971682.

[Tu19] E. E. Tun, S. Aramvith and Y. Miyanaga, "Fast Coding Unit Encoding Scheme for HEVC Using Genetic Algorithm," in IEEE Access, vol. 7, pp. 68010-68021, 2019, doi: 10.1109/ACCESS.2019.2918508.

[Ur23] N. Usha Bhanu and C. Saravanakumar, "Investigations of Machine Learning Algorithms for High Efficiency Video Coding (HEVC)," 2023 International Conference on Signal Processing, Computation, Electronics, Power and Telecommunication (IConSCEPT), Karaikal, India.

[Va20] M. Vashisht and B. Kumar, "A Survey Paper on Object Detection Methods in Image Processing," 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA), Gunupur, India, 2020, pp. 1-4, doi: 10.1109/ICCSEA49143.2020.9132871.

[Vi12] M. Viitanen, J. Vanne, T. D. Hämäläinen, M. Gabbouj and J. Lainema, "Complexity analysis of next-generation HEVC decoder," 2012 IEEE International Symposium on Circuits and Systems (ISCAS), Seoul, Korea (South), 2012, pp. 882-885, doi: 10.1109/ISCAS.2012.6272182.

[Vi22] M. Viitanen, J. Sainio, A. Mercat, A. Lemmetti and J. Vanne, "From HEVC to VVC: The First Development Steps of a Practical Intra Video Encoder," in IEEE Transactions on Consumer Electronics, vol. 68, no. 2, pp. 139-148, May 2022, doi: 10.1109/TCE.2022.314601.

[Vi67] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," in IEEE Transactions on Information Theory, vol. 13, no. 2, pp. 260-269, April 1967, doi: 10.1109/TIT.1967.1054010.

[VP9] D. Mukherjee et al., “A Technical Overview of VP9—The Latest Open-Source Video Codec,” in SMPTE Motion Imaging Journal, vol.124, no. 1, pp. 44-54, Jan. 2015. doi: 10.5594/j18499.

[VVC] "Information technology — Coded representation of immersive media — Part 3: Versatile video coding". International Organization for Standardization (2nd ed.). September 2022. ISO/IEC 23090-3:2022. Retrieved 16 February 2021.

[Wa17] Y.-X. Wang, D. Ramanan, M.l Hebert, "Learning to model the tail", In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 7032–7042, 2017.

[Wa18B] Z. Wang, S. Wang, X. Zhang, S. Wang and S. Ma, "Fast QTBT Partitioning Decision for Interframe Coding with Convolution Neural Network," 2018 25th IEEE International Conference on Image Processing (ICIP), Athens, Greece, 2018, pp. 2550-2554, doi: 10.1109/ICIP.2018.8451258.

[Wa20] Y. Wang and Z. Su, "An Efficient Intra Prediction Algorithm for HEVC Intra-coding," 2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Chengdu, China, 2020, pp. 407-412, doi: 10.1109/CISP-BMEI51763.2020.9263529.

[Wa21] T. Wang, F. Li, X. Qiao and P. C. Cosman, "Low-Complexity Error Resilient HEVC Video Coding: A Deep Learning Approach," in IEEE Transactions on Image Processing, vol. 30, pp. 1245-1260, 2021, doi: 10.1109/TIP.2020.3043124.

[We22] I. Werda, A. Maraoui, F. E. Sayadi and N. Masmoudi, "Fast CU partition and intra mode prediction method for HEVC," 2022 IEEE 9th International Conference on Sciences of

Electronics, Technologies of Information and Telecommunications (SETIT), Hammamet, Tunisia, 2022, pp. 562-566, doi: 10.1109/SETIT54465.2022.9875798.

[Wi03A] T. Wiegand, G. J. Sullivan, G. Bjonntegaard, A. Luthra, "Overview of the H.264/AVC Video Coding Standard", in IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560-576, July 2003.

[Wi03B] D.R. Wilson, T. R. Martinez, "The general inefficiency of batch training for gradient descent learning", Neural Networks, Volume 16, Issue 10, 2003, Pages 1429-1451, ISSN 0893-6080, https://doi.org/10.1016/S0893-6080(03)00138-2.

[X265] “x265: H.265 / HEVC video encoder application library,” Available: https://www.videolan.org/developers/x265.html,

[Xu18A] M. Xu, T. Li, Z. Wang, X. Deng, R. Yang and Z. Guan, "Reducing Complexity of HEVC: A Deep Learning Approach," in IEEE Transactions on Image Processing, vol. 27, no. 10, pp. 5044-5059, Oct. 2018, doi: 10.1109/TIP.2018.2847035.

[Xu18B] C. Xu, P. Liu, Y. Wu, K. Jia and W. Dong, "A Fast CTU Depth Selection Algorithm for H.265/HEVC Based on Machine Learning," 2018 IEEE 3rd International Conference on Signal and Image Processing (ICSIP), Shenzhen, China, 2018, pp. 154-161, doi: 10.1109/SIPROCESS.2018.8600458.

[Xu19] Y. Xu and X. Huang, "Hardware-Oriented Fast CU Size and Prediction Mode Decision Algorithm for HEVC Intra Prediction," 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), Bombay, India, 2019, pp. 1-5, doi: 10.1109/I2CT45611.2019.9033606.

[Ya07] Y. Yao, L. Rosasco, A. Caponnetto, 'On Early Stopping in Gradient Descent Learning"+B50. Constr Approx 26, 289–315, 2007, https://doi.org/10.1007/s00365-006-0663-2.

[Ya12] S. Yan, L. Hong, W. He and Q. Wang, "Group-Based Fast Mode Decision Algorithm for Intra Prediction in HEVC," 2012 Eighth International Conference on Signal Image Technology and Internet Based Systems, Sorrento, Italy, 2012, pp. 225-229, doi: 10.1109/SITIS.2012.41.

[Ya20A] H. Yang, L. Shen, X. Dong, Q. Ding, P. An and G. Jiang, "Low-Complexity CTU Partition Structure Decision and Fast Intra Mode Decision for Versatile Video Coding," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 6, pp. 1668-1682, June 2020, doi: 10.1109/TCSVT.2019.2904198.

[Ya20B] J. -W. Yang, P. -R. Lai, P. -C. Fu and J. -S. Wang, "Tradeoff between Parallel Efficiency and Coding Efficiency of HEVC: A Load Balancing Solution with Convolutional Neural Networks," 2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), Genova, Italy, 2020, pp. 248-250, doi: 10.1109/AICAS48895.2020.9073933.

[Ya90] K. Yamaguchi, K. Sakamoto, T. Akabane, Y. Fujimoto, "A Neural Network for Speaker-Independent Isolated Word Recognition" First International Conference on Spoken Language Processing (ICSLP 90). Kobe, Japan, November 1990 .

[Yu15] X. Yu, Z. Liu, J. Liu, Y. Gao and D. Wang, "VLSI friendly fast CU/PU mode decision for HEVC intra encoding: Leveraging convolution neural network," 2015 IEEE International Conference on Image Processing (ICIP), Quebec City, QC, Canada, 2015, pp. 1285-1289, doi: 10.1109/ICIP.2015.7351007.

[Za22] A. Zafar, M. Aamir, N. Mohd Nawi, A. Arshad, S. Riaz, A. Alruban, A.K. Dutta, S.A. Almotairi, "Comparison of Pooling Methods for Convolutional Neural Networks", Appl. Sci. 2022, 12, 8643. https://doi.org/10.3390/app12178643.

[Zh04] T. Zhang, B. Yu, "Boosting with Early Stopping: Convergence and Consistency" Ann. Statist.. 33, 2004, 10.1214/009053605000000255.

[Zh14] J. Zhu, Z. Liu, D. Wang, Q. Han and Y. Song, "HDTV1080p HEVC Intra encoder with source texture based CU/PU mode pre-decision," 2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC), Singapore, 2014, pp. 367-372, doi: 10.1109/ASPDAC.2014.6742917.

[Zh18] Y. Zhang, Z. Pan, N. Li, X. Wang, G. Jiang, and S. Kwong, "Effective Data Driven Coding Unit Size Decision Approaches for HEVC INTRA Coding," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 28, no. 11, pp. 3208-3222, Nov. 2018, doi: 10.1109/TCSVT.2017.2747659.

[Zh20A] Y. Zhang, S. Kwong, S. Wang, "Machine learning based video coding optimizations: A survey" Information Sciences, Volume 506, 2020, Pages 395-423, ISSN 0020-0255,https://doi.org/10.1016/j.ins.2019.07.096.

[Zh20B] Q. Zhang, Y. Wang, L. Huang and B. Jiang, "Fast CU Partition and Intra Mode Decision Method for H.266/VVC," in IEEE Access, vol. 8, pp. 117539-117550, 2020, doi: 10.1109/ACCESS.2020.3004580.

[Zh20C] Q. Zhang, Y. Zhao, B. Jiang, L. Huang and T. Wei, "Fast CU Partition Decision Method Based on Texture Characteristics for H.266/VVC," in IEEE Access, vol. 8, pp. 203516-203524, 2020, doi: 10.1109/ACCESS.2020.3036858

[Zh21A] J. Zhao, T. Cui and Q. Zhang, "Fast CU Partition Decision Strategy Based on Human Visual System Perceptual Quality," in IEEE Access, vol. 9, pp. 123635-123647, 2021, doi: 10.1109/ACCESS.2021.3110292.

[Zh21B] Q. Zhang, R. Guo, B. Jiang and R. Su, "Fast CU Decision-Making Algorithm Based on DenseNet Network for VVC," in IEEE Access, vol. 9, pp. 119289-119297, 2021, doi: 10.1109/ACCESS.2021.3108238.

[Zh22] J. Zhao, A. Wu, B. Jiang and Q. Zhang, "ResNet-Based Fast CU Partition Decision Algorithm for VVC," in IEEE Access, vol. 10, pp. 100337-100347, 2022, doi: 10.1109/ACCESS.2022.3208135.

[Zh23A] T. Zhao, Y. Huang, W. Feng, Y. Xu and S. Kwong, "Efficient VVC Intra Prediction Based on Deep Feature Fusion and Probability Estimation," in IEEE Transactions on Multimedia, vol. 25, pp. 6411-6421, 2023, doi: 10.1109/TMM.2022.3208516.

[Zh23B] Y. Zhang, B. Kang, B. Hooi, S. Yan and J. Feng, "Deep Long-Tailed Learning: A Survey" in IEEE Transactions on Pattern Analysis & Machine Intelligence, vol. 45, no. 09, pp. 10795-10816, Sept. 2023, doi: 10.1109/TPAMI.2023.3268118.

[Zo04] W. Zouch, A. Samet, M. A. Ben Ayed, F. Kossentini and N. Masmoudi, "Complexity analysis of intra prediction in H.264/AVC," Proceedings. The 16th International Conference on Microelectronics, 2004. ICM 2004., Tunis, Tunisia, 2004, pp. 713-717, doi: 10.1109/ICM.2004.1434766.