CTU Depth Decision Algorithms for HEVC: A Survey

Ekrem Çetinkaya[a,c], Hadi Amirpour[a,c], Mohammad Ghanbari[a,b] and Christian Timmerer[a]

[a]*Christian Doppler Laboratory ATHENA, Alpen-Adria-Universität, Klagenfurt, Austria*
[b]*School of Computer Science and Electronic Engineering, University of Essex, Colchester, UK*
[c]*These authors contributed equally to this work.*

ARTICLE INFO

Keywords:
HEVC
Coding tree unit
Complexity
CTU partitioning
Statistics
Machine learning

ABSTRACT

High-Efficiency Video Coding (HEVC) surpasses its predecessors in encoding efficiency by introducing new coding tools at the cost of an increased encoding time-complexity. The Coding Tree Unit (CTU) is the main building block used in HEVC. In the HEVC standard, frames are divided into CTUs with the predetermined size of up to 64×64 pixels. Each CTU is then divided recursively into a number of equally sized square areas, known as Coding Units (CUs). Although this diversity of frame partitioning increases encoding efficiency, it also causes an increase in the time complexity due to the increased number of ways to find the optimal partitioning. To address this complexity, numerous algorithms have been proposed to eliminate unnecessary searches during partitioning CTUs by exploiting the correlation in the video. In this paper, existing CTU depth decision algorithms for HEVC are surveyed. These algorithms are categorized into two groups, namely statistics and machine learning approaches. Statistics approaches are further subdivided into neighboring and inherent approaches. Neighboring approaches exploit the similarity between adjacent CTUs to limit the depth range of the current CTU, while inherent approaches use only the available information within the current CTU. Machine learning approaches try to extract and exploit similarities implicitly. Traditional methods like support vector machines or random forests use manually selected features, while recently proposed deep learning methods extract features during training. Finally, this paper discusses extending these methods to more recent video coding formats such as *Versatile Video Coding* (VVC) and *AOMedia Video 1* (AV1).

1. Introduction

Video streaming has become an essential part of today's Internet traffic. The majority of the global Internet traffic consists of video (75% in 2017), and its share is expected to grow in the future (82% by 2022) [1]. This steep increase in video traffic has created challenges in several blocks of the video streaming solutions that need to be addressed. The building blocks of the video streaming can be expressed as content provisioning, content delivery, and content consumption. In this survey, we will focus on the content provisioning part (*i.e.*, video coding) of the video streaming scheme, while for content delivery and consumption parts, we refer to Bentaleb *et al.* [2].

A video is a sequence of images with redundant information. Spatial and temporal redundancy in videos can be exploited to reduce their size. This process requires efficient coding tools, since finding similarities is the key for exploiting redundancy, and the vast amount of information in videos makes this process difficult. Also, as the framerate, resolution, and bit depth of video content increases, the amount of information available in the video increases significantly, making the reduction in the amount of redundancy more and more important. This led to the need to develop more advanced video encoders beyond the existing *Advanced Video Coding* (AVC) standard [3]. *High Efficiency Video Coding* (HEVC) is the successor of AVC that has been developed by the Joint Collaborative Team on Video Coding (JCT-VC) which has improved the existing encoding tools and introduced new ones to increase encoding efficiency [4]. Compared to AVC, HEVC can reduce the bitrate of video by 50% [4].

Modern video encoders adopt a block-based structure for motion compensation to improve encoding efficiency. In the block-based structure, frames are divided into several smaller blocks that vary in size depending on the complexity of the content. These blocks are later used in the motion compensation part of the encoder, in which the encoder tries to predict the block using the best-matched block in the current frame (Intra) or in the previously encoded frames (Inter) based on the motion information. After a block is predicted, the residual error information is transformed into transform blocks, and they are entropy encoded.

Each video codec uses (slightly) different block structures. Older video coding standards like MPEG-2 [5] uses a fixed block size of 16×16 pixels, while its transform blocks have a size of 8×8 samples. The more recent AVC standard introduced a more flexible block structure.

In AVC, frames are divided into macroblocks of varying sizes up to 16×16 pixels, and each macroblock can be further partitioned into variable sizes. For intra prediction, sub-macroblock sizes of 16×16, 8×8, and 4×4 pixels are allowed. For inter prediction, the sub-macroblock sizes of 16×16, 16×8, 8×16, and 8×8pixels are searched and a motion vector is assigned to each sub-macroblock. When 8×8 is an optimal sub-macroblock size candidate, 8×4, 4×8, and 4×4 sub-macroblock sizes are also checked for

ORCID(s): 0000-0002-6084-6249 (E. Çetinkaya); 0000-0001-9853-1720 (H. Amirpour); 0000-0002-0031-5243 (C. Timmerer)

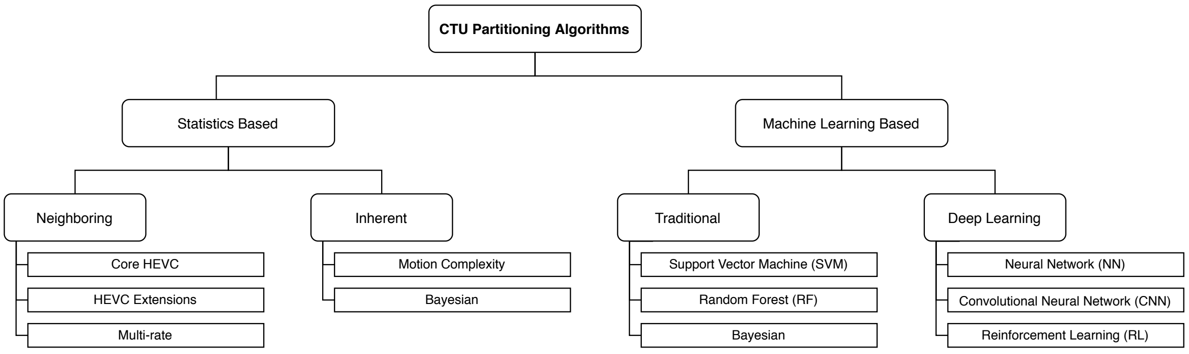

Figure 1: Classification of CTU partitioning algortihms.

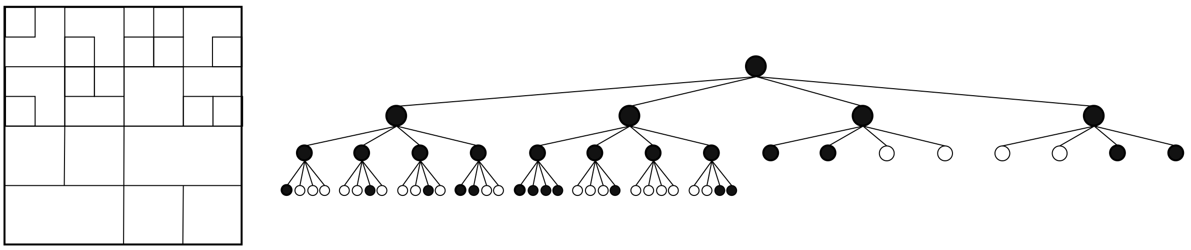

Figure 2: Variable block size partitioning of a frame using quad-tree in [9].

each 8×8 sub-macroblock. Depending on the size selected, one of two 8×8 or 4×4 transform blocks will eventually be selected [6].

Using larger blocks to exploit more spatial redundancy can increase the efficiency and flexibility of the encoder, especially for higher resolution videos [6, 7]. HEVC [8] introduces a new block partitioning structure called Coding Tree Unit (CTU), which can vary in size from 8×8 to 64×64 pixels. Despite the increased efficiency and flexibility of the HEVC, using the CTU as a building block also leads to significant complexity in encoding time, making the use of HEVC a challenging task for several applications, *e.g.*, live streaming. To cope with this increased complexity, many algorithms have been proposed to reduce the process of rate-distortion optimization (RDO) by eliminating unnecessary searches for optimal CTU partitioning using several available sources of information.

This paper provides a comprehensive study of CTU partitioning algorithms. We give detailed information about the CTU structure of HEVC in Section 2. We then classify the existing methods and mainly focus on the CU depth decision algorithms for HEVC. We categorize existing approaches into statistics based and machine learning (ML) based methods. The former benefits from the statistical correlation in the video, while the latter uses machine learning methods to extract correlation. We present these approaches in Section 3 and Section 4, respectively. Fig. 1 shows the broad categorization for CU depth decision algorithms used in this paper. In Section 5, we discuss the overall findings and possible future directions including emerging coding formats such as *Versatile Video Coding* (VVC) and *AOMedia Video 1* (AV1). Finally, we conclude the paper with Section 6.

2. Overview of HEVC CTU partitioning

In HEVC, frames are divided into tiles or slices, which are further divided into non-overlapped CTUs. Each CTU can then be split into several square regions of equal sizes, known as coding units (CUs), using a quad-tree structure.

2.1. Coding Tree Unit (CTU)

CTUs use a quad-tree structure, that was already used in video compression [10, 9, 11]. In [9], this structure is used at the frame level, where the partitioning of a frame is done by split and merging processes. First, a frame is divided into sub-blocks, and if sub-blocks have the same motion vectors, these sub-blocks are merged. By merging the regions that optimize rate-distortion cost, a variable block size motion compensation is allowed where blocks can be L-shaped, rectangular-shaped, or square-shaped regions. The block structure is illustrated in Fig. 2.

HEVC uses a *quad-tree* structure for partitioning a CTU. The entire block is represented by a *root node*, which has a depth value of 0. Each node in the tree can have either four child nodes or zero child-nodes (*i.e.*, *leaf node*). The depth level of the nodes increases by 1 when traversed towards the bottom of the tree. If we consider the block size in the root node as $l_{max} \times l_{max}$ and the depth as 0, then each sub-block at depth d has the size $(l_{max} \times l_{max})/2^d$.

In HEVC, CTUs have a predetermined size of up to 64×64 pixels, where $l_{max} = 64$. Each CTU can then be split recursively into square sized CUs. Each division increases the depth by 1, *e.g.*, 64×64 is *depth* 0 and 8×8 is *depth* 3. An example of a CTU quad-tree is shown in Fig. 3. The difference in block partitioning between AVC and HEVC is shown in Fig. 4. We can clearly see the effect of different

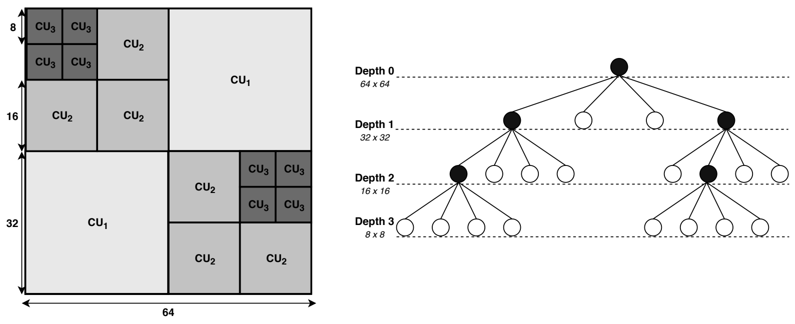

Figure 3: CTU partitioning of HEVC.

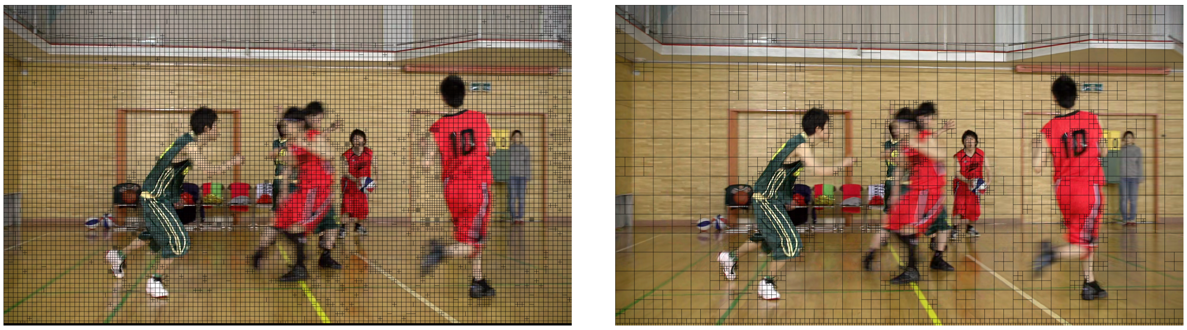

Figure 4: Comparison of block partitioning between AVC and HEVC for the 100th frame of the BasketballDrive sequence. Both frames have been encoded at the same bitrate.

maximum block sizes between two codecs (16×16 for AVC and 64×64 for HEVC). Both codecs use smaller block sizes, *i.e.*, greater depths, for areas containing more texture or motion complexity *e.g.*, center of the frame, where the players move. It can also be seen that HEVC takes larger CUs for the areas with less texture or motion information, which results in a saving of more bitrates.

Furthermore, coding unit (CU), prediction unit (PU), and transform unit (TU) concepts have been introduced in HEVC which, are associated with a CTU.

2.2. Coding Unit (CU)

Each leaf node of a quad-tree, representing a square region inside the CTU is called CU that can be from 8×8 to 64×64 pixels. Fig. 3 exemplifies a CTU partitioning that contains 16 CUs or leaf nodes with sizes from 8×8 to 32×32 pixels. For each CU, three Coding Blocks (CBs) are associated in the video frame buffer, one for luma (Y) sample, and two for chroma (Cb, Cr) samples.

2.3. Prediction Unit (PU)

A decision on whether to perform inter- or intra-picture prediction on a block is made at the CU level. Each CU is split into PUs according to the prediction type. As with AVC, three prediction types are available for each CU: *(i)* inter-coded CU, *(ii)* skipped CU, and *(iii)* intra coded CU. Various PU modes are illustrated in Fig. 6.

2.3.1. Inter Coded CUs

There are eight different modes for the inter-picture prediction type, *i.e.*, four square- or rectangular-shaped and four asymmetric modes. A CU with a size of $2N \times 2N$ can be split into one of the following modes: single PU with size ($PART\_2N \times 2N$), four PUs with sizes ($PART\_N \times N$), two PUs with sizes ($PART\_2N \times N$) or ($PART\_N \times 2N$) for square- or rectangular-shaped modes. A CU can be further split into two PUs with sizes ($PART\_2N \times nU$), ($PART\_2N \times nD$), ($PART\_nL \times 2N$) or ($PART\_nD \times 2N$) in the asymmetric mode. It should be noted that ($PART\_N\times N$) is checked only when a CU has its minimum size ($2N \times 2N$ = 8×8 for main profile). For other CU sizes, ($PART\_N\times N$) is similar to splitting the CU into four smaller sub-CUs. Also, asymmetric modes are disabled when CU size is equal to 8×8 to reduce the complexity.

2.3.2. Skipped CUs

Skipped CU mode is a special inter-coded CU mode where both motion vector and residual energy are zero. For each $2N \times 2N$ CU, only one $2N \times 2N$ PU is considered for the skipped mode.

2.3.3. Intra Coded CUs

Two PU modes are available for intra-picture coding of a CU, ($PART\_2N \times 2N$) and ($PART\_N \times N$). Similar to CU, three prediction blocks (PBs) are considered in the video frame buffer for each color component.

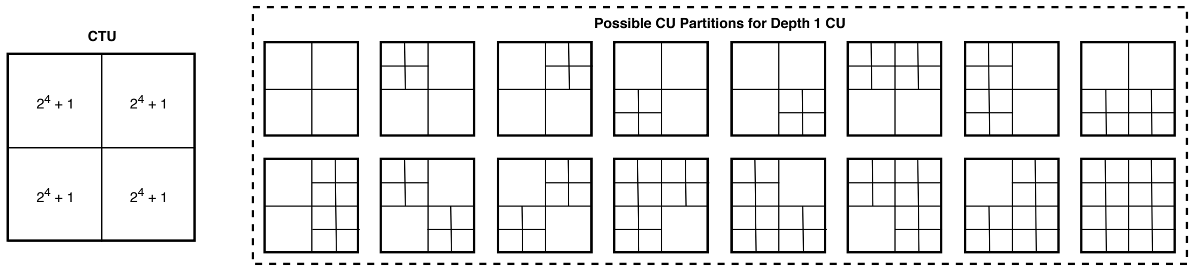

Figure 5: Illustration of possible CU partitions.

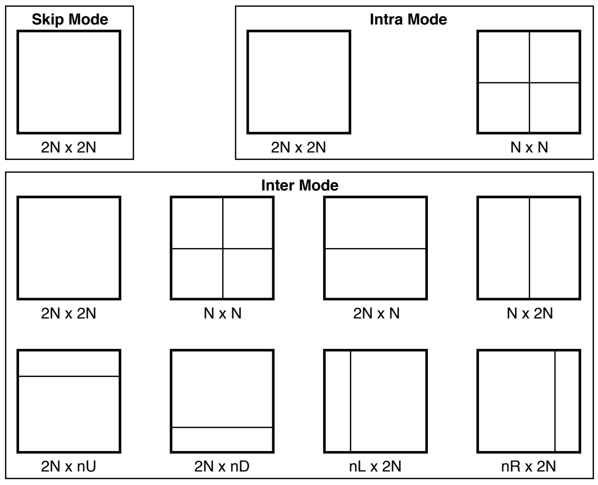

Figure 6: PU splitting modes.

2.4. Transform Unit (TU)

When the optimal prediction mode is selected for each leaf CU, residual errors are transformed into a TU, and a residual quad-tree (RQT) structure is used to determine the best TU partitioning for each leaf CU. Square-shaped TUs with sizes between 4×4 to 32×32 samples are supported in HEVC [12].

2.5. Overview of HEVC CTU Partitioning

There are numerous possible patterns for the division of a single CTU. It is easy to understand that from a bottom-up approach. Let us assume a CU with a depth 2. For this particular CU, there are two options, split or not-split. Now, if we go up one level to depth 1, we have four possible sub-CUs with depth 2 and two possible partitions for each sub-CU. Thus, for depth 1, there are 2^4 possible sub-CU partitions and also one more option that is non-split, which gives a total of $2^4 + 1$ options in total for a single depth 1 CU. Following the same approach, if we go up one more level to depth 0, there will be $(2^4 + 1)^4$ possible partitions for depth 0. Again, we also have the option to not split, so there is a total of $(2^4+1)^4+1 = 83522$ possible partitions for a single CTU. To find an optimal CTU partitioning from the 83522 possible partitions, HEVC searches 85 CUs with different sizes for each CTU. These 85 CUs are: one 64×64, four 32×32, sixteen 16×16, and sixty four 8×8 pixels blocks.

Table 1
Encoding Time-complexity when depth 0 (No D0) or depth 3 (No D3) is skipped.

Class	All Intra		Random Access		Low Delay B	
	No D0	No D3	No D0	No D3	No D0	No D3
Class A	88 %	58 %	84 %	72 %	83 %	72 %
Class B	89 %	57 %	86 %	72 %	85 %	73 %
Class C	90 %	59 %	86 %	73 %	85 %	75 %
Class D	89 %	61 %	87 %	71 %	88 %	71 %
Class E	89 %	57 %	86 %	72 %	85 %	73 %
Average	89 %	58 %	86 %	72 %	85 %	73 %

Table 2
Decoding Time-complexity when depth 0 (No D0) or depth 3 (No D3) is skipped.

Class	All Intra		Random Access		Low Delay B	
	No D0	No D3	No D0	No D3	No D0	No D3
Class A	103 %	94 %	118 %	98 %	113 %	96 %
Class B	110 %	99 %	113 %	99 %	112 %	99 %
Class C	106 %	97 %	115 %	98 %	113 %	99 %
Class D	101 %	96 %	103 %	97 %	103 %	95 %
Class E	102 %	97 %	124 %	99 %	116 %	99 %
Average	104 %	96 %	114 %	98 %	111 %	97 %

In addition to finding the correct CU depth structure, the PU modes and the TU partitioning for each possible CU must also be correctly determined. Thus, the search for the optimal CTU structure using a brute force approach to determine the one with the minimum rate-distortion (RD) cost using a Lagrangian multiplier, takes the largest amount of time in the encoding process [13]. To show how eliminating one CTU depth search affects encoding efficiency and time-complexity, we encoded several sequences where the depth 0 is eliminated by setting the maximum CTU size to 32×32 pixels. Moreover, we also eliminated depth 3 by setting the minimum CU size to 16×16 pixels. The results have been summarized in Table 1, Table 2, and Table 3, respectively. In these tables, the classes represent the category of videos based on the video resolution [14]. *All Intra*, *Random Access*, and *Low Delay B* are the HEVC configuration files used during the encoding [14]. All reported results are obtained by comparing the encoding results with HEVC reference software (HM 16.20) using a modified configuration, which limits the minimum or maximum CU sizes, and the unmodified configure. It is clear that limiting the depth of CTU partitioning can reduce the time complexity at the cost of bitrate increase.

Table 3
BD-PSNR and BD-Rate when depth 0 (No D0) or depth 3 (No D3) is skipped.

Class	All Intra				Random Access				Low Delay B			
	No D0		No D3		No D0		No D3		No D0		No D3	
	BD PSNR	BD Rate	BD PSNR	BD Rate	BD PSNR	BD Rate	BD PSNR	BD Rate	BD PSNR	BD Rate	BD PSNR	BD Rate
Class A	-0.40 %	0.81	-1.02 %	2.11	-0.73 %	1.91	-1.36 %	3.64	-0.72 %	2.09	-1.13 %	3.3
Class B	-0.16 %	0.48	-0.03 %	0.09	-0.67 %	2.93	-0.03 %	0.09	-0.45 %	1.83	-0.03 %	0.02
Class C	-0.20 %	0.87	-0.35 %	1.43	-0.68 %	2.26	-0.29 %	1.4	-0.58 %	2.71	-0.26 %	1.22
Class D	-0.26 %	0.39	-5.78 %	6.74	-0.05 %	0.25	-4.41 %	8.77	-0.29 %	0.61	-4.03 %	8.91
Class E	-0.58 %	1.46	-1.32 %	3.33	-1.13 %	3.87	-1.09 %	3.88	-1.3 %	4.79	-1.0 %	3.83
Average	-0.32 %	0.79	-2.02 %	3.25	-0.65 %	2.16	-1.69 %	3.92	-0.68 %	2.42	-1.49 %	3.81

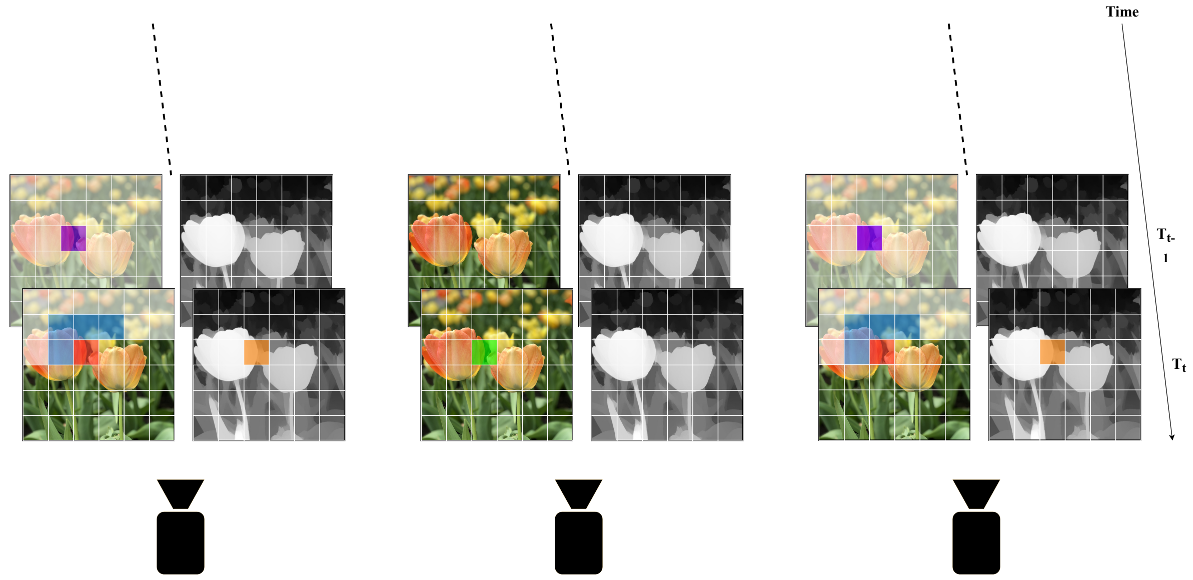

Figure 7: CTUs are shown in different colors. ***Red***: Current CTU, ***Blue***: Four spatially neighboring CTUs, ***Purple***: Temporally neighboring CTU, ***Green***: Co-located interview CTU in the base view for MV-HEVC and 3D-HEVC, and ***Orange***: Co-located CTU in the depth frame for 3D-HEVC.

To overcome the high complexity raised from the CTU partitioning in HEVC, many algorithms have been proposed which try to use the similarity between the partitioning of spatial and/or temporal neighboring CTUs or inherent features of each CTU, to skip the search for unnecessary CUs. For each non-border CTU, there are four spatially neighboring CTUs in the same frame and one temporally co-located CTU in the reference frame. Moreover, additional neighboring CTUs can be found in the HEVC extensions, which are designed for specific scenarios. We briefly introduce them in the following section.

2.6. Overview of HEVC Extensions

MV-HEVC [15] is an extension of HEVC, which allows efficient encoding of multiple camera views by enabling the use of interview references in motion-compensated prediction. Views are divided into base and dependent views. Base

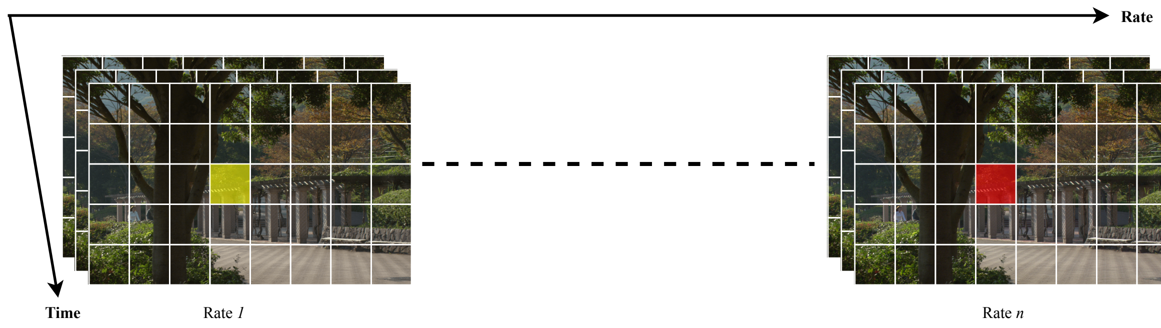

Figure 8: Multirate encoding. Red block represents current CTU while yellow block represents co-located CTU in the previously encoded representation.

views are encoded using HEVC simulcast, *i.e.*, each view is encoded independently.

Dependent views exploit dependencies between the views and use reconstructed base view frames as additional reference frames for motion compensation. Therefore, in addition to the available spatial and temporal CTUs in HEVC, interview CTU, *i.e.*, co-located CTU in the base view, can be used for prediction. The interview co-located CTU is illustrated in Fig. 7.

The 3D extension of HEVC (3D-HEVC) [15] is an extension of HEVC that supports the encoding of multiple views and their associated depth information. Similar to MV-HEVC, texture frames can utilize information from four spatially neighboring CTUs, temporally co-located CTU(s), and interview CTU(s). In addition to the above-mentioned CTUs, information of co-located CTU in the corresponding depth frame can also be used if the depth frame has been previously encoded. Otherwise, the co-located CTU in the associated texture frame can be used to predict the depth level of the depth frame. Fig. 7 shows the association of the texture and depth frames.

HEVC Screen Content Coding (SCC) [16] has been developed to provide improved encoding efficiency for videos that contain a significant amount of screen-captured content, as characteristics of these videos differ from those of camera-captured contents. To achieve this, several tools have been added to the basic HEVC that are specifically designed for screen content. One of these tools is Intra-Block Copy (IBC), which is another CU mode that is added along with the existing conventional Intra (Cintra) and Inter modes. It can be considered as motion estimation inside a frame at the PU level. When a CU is encoded in IBC, PUs of this CU are searched for similar reconstructed blocks within the current frame. Another tool is Palette Mode (PLT), which focuses on color information since screen content videos usually contain a a small number of different colors. PLT first enumerates each distinct color in the block, and these indexes are used for each sample rather than actual values to define color information. Also, Adaptive Color Transform (ACT) is proposed for color-coding since the majority of screen content videos use the RGB color space and not YCbCr. In HEVC-SCC, an image block can be encoded directly in the RGB space or can be converted to the YCoCg space during encoding, depending on the content characteristic of the block. Finally, Adaptive Motion Vector Resolution (AMVR) is added to deal with discrete motion in the screen content video, which can be represented by integer motion vectors. This is because, in screen-captured videos, movement is precisely aligned with pixels in general. AMVR allows the motion vectors to switch between integer and fractional values.

2.7. Multirate Encoding

Adaptive HTTP streaming [17, 18] provides multiple representations of the same content in different qualities and resolutions. This allows clients to request segments in a dynamic and adaptive way depending on the network conditions. When a representation is encoded, the CTU depth information from that representation can be used by the other representations. Therefore, in addition to the four spatially and one temporally co-located CTUs, a co-located CTU in the other representation can also be used to increase the accuracy of the depth prediction for the current CTU. Fig. 8 shows co-located CTU in the previously encoded representation.

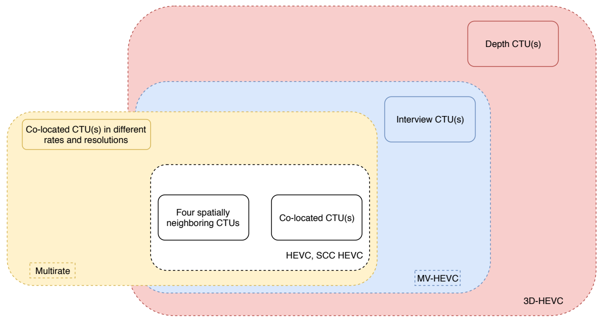
Depth CTU(s)
Interview CTU(s)
Co-located CTU(s) in different rates and resolutions
Four spatially neighboring CTUs
Co-located CTU(s)
HEVC, SCC HEVC
Multirate
MV-HEVC
3D-HEVC

Figure 9: Available CTUs for HEVC and its extensions

2.8. Summary

Fig. 9 summarizes the available CTUs that can be used by the current CTU. Four spatially neighboring CTUs and one co-located CTU in the temporally neighboring frame are available for HEVC and all its extensions. Interview CTU is added to the list of available reference CTUs for MV-HEVC, and the depth CTU is also available for 3D-HEVC together with all the CTUs mentioned above. Finally, co-located CTUs from previously encoded representations are also available for multirate encoding.

In addition, since complex CTUs have a larger depth than homogeneous CTUs [19], some features are extracted to measure the complexity of individual CTUs and then to make a decision on their depth levels. Numerous approaches have been proposed to improve CTU partitioning decision of HEVC to eliminate unnecessary searches and improve overall encoding pipeline. In the following section, we provide an overview of existing statistics based approaches in the literature.

3. Statistics Based Approaches

Statistics based approaches exploit the statistical similarity in the video to reduce the time needed for the CU depth search. Since video is composed of successive frames, these approaches can achieve a significant reduction in time. These methods exploit previously known correlations in the video by manually defining a set of features, thresholds, rules, *etc.* The main advantage of these approaches is that they are faster and less complex than approaches based on machine learning, with the disadvantage of being less general and less improvement in the encoding efficiency.

In this section, statistics based approaches are categorized into *two groups* based on the main feature used by the respective approach: *(i) Neighboring* CTUs use information that is available from spatially and temporally neighboring CTUs and *(ii) inherent* approaches only use information that

Table 4
Threshold values and their corresponding depth ranges used in [20].

Threshold Value	Current CTU depth range
$Threshold\ Value = 0$	0 to 1
$1 \leqslant Threshold\ Value \leqslant 3$	0 to 2
$4 \leqslant Threshold\ Value \leqslant 6$	0 to 2
$7 \leqslant Threshold\ Value \leqslant 12$	1 to 3

is available in the current CTU.

3.1. Neighboring CTUs Approaches

The *neighboring CTUs* approaches can be further divided into approaches defined as part of the *(a)* core *HEVC* specification, *(b) HEVC extensions*, and *(c) multirate* HEVC techniques.

3.1.1. Core HEVC specification

There are several methods in the literature that exploit only temporally or spatially closed CU information [20, 21, 22, 23].

Kim *et al.* [20] propose to determine a search range for the current CTU using a threshold value estimated from the spatially neighboring CTUs which is calculated as Eq. 1.

$$Threshold\ Value = \sum_{i=0}^{3} Depth_i \tag{1}$$

Threshold values and their corresponding CU depth ranges are listed in Table 4.

Similar to [20], Shen *et al.* [21] use information from four spatially neighboring CTUs, but to categorize CTUs into four texture classes ranging from homogeneous to rich texture, using the formula in Eq. 2.

$$Depth_{pre} = \sum_{i=0}^{3} \alpha_i Depth_i \tag{2}$$

where α_i is a weight factor assigned to each neighboring CTU. Based on $Depth_{pre}$, texture type of the current CTU is determined as given in Table 5. Based on the $Depth_{pre}$, maximum or minimum depth level is changed for the current CTU.

Hou *et al.* [22] propose three criteria for the CU split decision, *i.e.*, *(i)* co-located CTU in the previous frame has larger depth, *(ii)* all neighboring CTUs have larger depths, and *(iii)* the current frame is not an I-frame. On top of that, another three criteria are used for early termination, *i.e.*, *(i)* co-located CTU has smaller depth, *(ii)* at least three neighbours have smaller depths, and *(iii)* the current frame is not an I-frame. Under the other conditions, the unmodified HEVC search is used.

Another approach proposed by Cen *et al.* [24] follows the average depth value of the four spatially neighboring CTUs. This average value is then compared to a predefined threshold value to determine if the depth range of the current CTU is [0 2] or [1 3].

Table 5
CTU texture categorization criteria used in [21].

$Depth_{pre}$	Current CTU texture type
$0.5 \leqslant Depth_{pre}$	type I
$0.5 < Depth_{pre} \leqslant 1.5$	type II
$0.5 < Depth_{pre} \leqslant 2.5$	type III
$Depth_{pre} > 2.5$	type IV

Li *et al.* [25] use spatially neighboring CTUs and an RD cost threshold together. The RD cost threshold is updated online using training frames. Left and upper CTUs are used, and the depth search range for the current CU is obtained by limiting it between the minimum and maximum depths found in these two CTUs. If the depth of the current CU is smaller than $depth_{min}$, it is split, and if it is equal to or larger than $depth_{max}$, CU is not split. Also, an early termination algorithm is proposed based on RD cost as an alternative method for all intra configuration only, which exploits the temporal correlation between adjacent frames. For this approach, frames are periodically categorized into training and test frames. The first frame of each period is a training frame which is used to extract statistical information about CU depth levels and RD costs. This information is then used to determine a threshold for early termination in test frames. These two approaches are combined for all intra configuration.

A two-level depth decision algorithm is proposed based on previously encoded frames by Huade *et al.* [26]. At the frame level, it is supposed that the depth level of all CUs within a frame are concentrated on two depths. These two depth levels are the most common CU depth levels found in the latest encoded frame of the same type. In CU level, the depth of each CU is limited to the minimum and maximum depth values found in the temporally co-located CU.

Li *et al.* [27] use the maximum depth of the temporally co-located CU (D_{pre0}) to predict the depth of the current CU (D_{cur}) using Eq. 3.

$$D_{cur} = D_{pre0} + \Delta D \tag{3}$$

where ΔD is variation between D_{cur} and D_{pre0} which is determined based on encoding parameters as shown in Eq. 4.

$$\Delta D = |D_{pre0} - D_{pre1}| + |QP_{cur} - QP_{pre0}| - |QP_{pre0} - QP_{pre1}| \tag{4}$$

Amirpour *et al.* [28, 29] use co-located CTUs to limit the depth search range for the current CTU. Four co-located CTUs in four reference frames are replaced with four spatially neighboring CTUs to determine the depth range for the current CTU. Their minimum and maximum depth values are used to limit the depth range for the current CTU.

Pan *et al.* [30] use co-located CTU in addition to spatially neighboring CTUs to make a decision on the depth level of the current CTU. CTU partitioning is terminated in depth 0 if the following condition in Eq. 5 is satisfied:

$$W = \sum_{n=0}^{4} \lambda_n C_n \geqslant \alpha \tag{5}$$

where λ_n is a weight factor and C_n represents the weight of the reference CTUs, which is 1 if the reference CTU is encoded as depth 0, otherwise, C_n is equal to 0; Depth 3 searches are skipped based on the correlation between PU mode decisions.

CTUs are classified into *simple* and *complex* CTUs in the method proposed by Zhou *et al.* [31]. If the maximum depth of a CTU is 0 or 1, it is considered as a simple CTU, otherwise, complex. A depth decision algorithm has been proposed based on the complexity of the left, upper, and co-located CTUs. If all above-mentioned neighboring CTUs are complex, depth 0 is skipped from searching, and if they are all simple, depth 3 is skipped from the search process.

Leng *et al.* [32] predict the depth of CTUs both at frame and CTU levels. Rarely used depths in the reference frame are skipped in the current frame. In the CTU level, partitioning of the current CU is terminated if *(i)* the current depth is equal to depth of the co-located CU, or *(ii)* if the current depth is equal to the depth of two or more spatially neighboring CTUs.

Shen *et al.* [33] propose to limit CU depth search range using the information in four spatially neighboring CTUs and co-located CTU. Information from these sources is weighted according to its importance, CTUs in the horizontal and vertical directions are given more weight, and are used for depth prediction. Depending on the result of prediction, the current CU is classified into one of five types ranging from homogeneous to fast motion regions, and the depth is limited depending on the classification result.

Correa *et al.* [34] categorize frames into two groups, *unconstrained frames* (F_u) and *constrained frames* (F_c). Unconstrained frames are encoded using the full RDO process and the remaining frames are encoded using the maximum depth of the co-located CTU in previous F_u to limit CTU partitioning in the current CTU. The number of F_c between F_us is controlled by target complexity. Similar to [34], [35], they not only use co-located CTU in the unconstrained frame F_u to define maximum depth of the current CTU, but also take into account the information of the spatially neighboring CTUs as well as co-located CTU in the previous constrained frame F_c.

Bae and Sunwoo [36] store CU depth information from CTUs in the previous five or six frames and use this information to decide early CU and PU terminations. A weighted structure is used to give more importance to the closer frames. If all the depths are equal, then the same depth is selected as the final depth for the current CU and PU search is conducted. Otherwise, statistical properties of CTUs are calculated and used in the decision process.

The depth distribution and RD cost of the co-located CTU are used to reduce the search range for the current CTU in method proposed by Park [37]. The search range is limited by the maximum and minimum CU depths that are used in the co-located CTU. To avoid error propagation in this approach, the search range is reset at every one-second interval, and those frames are referred to as reset frames for which the search range limitation is not applied. A predefined threshold value is used for early termination, which is calculated based on the RD cost and CU depth distribution of co-located CTU. If the RD cost of the current depth is higher than the threshold, the CU is further split. The threshold is adjusted based on the CU depth distribution in the co-located CTU in a way that if the co-located CTU contains more CUs at maximum depth, then the current CTU is less likely to be early terminated. Again, early termination is not applied for the reset frames.

Zhi *et al.* [38] do partitioning of CTUs in two steps, *i.e.*, *rough* and *accurate determination*. The rough determination step is used to predict the complexity of CU by using the depth values and prediction modes of neighboring CTUs. Instead of using information from the entire CTU, information in the edges of CTUs is used. For the left neighboring CTU, the rightmost 4×64 pixels area, and for the top neighboring CTU, the bottom-most 64×4 pixels areas are used. If the current CTU has a depth of 0 and if there is a depth of 3 in any of these areas, the RDO search is skipped, and the current CTU is split. Moreover, if the current CTU has a depth of 2 and if there is a depth of 0 in any of these areas, the CTU split will stop. In the accurate determination step, pixel values in the edges of the current CTU are used to make a decision. If the pixel values vary over a wide range, then this is accepted as an indicator of a complex CTU. On top of that, the entropy of pixel values and pixel differences in the top, left, right, and bottom edges are also used to calculate complexity. Each of these calculations is used to determine a threshold value for the complexity of the CTU. If the CTU is determined to be complex in any of these steps, it is split further; otherwise, CU splitting is stopped. The algorithm uses the decision of rough determination step unless the decision is uncertain. The accurate determination step is used only when the decision of the algorithm is uncertain.

Zhao *et al.* [39] propose a two-step depth decision algorithm. First, a depth range $[D^C_{min}\ D^C_{max}]$ is determined for the current CU (C) based on the depth values of left (L) and upper (U) CUs as in Eq. 6.

$$\begin{aligned} D^C_{min} &= min(D^L, D^U) - 1 \\ D^C_{max} &= max(D^L, D^U) + 1 \end{aligned} \tag{6}$$

In the second step, if the RD costs of already searched for child CUs are larger than that of the parent CU, the search process is terminated.

3.1.2. MV-HEVC and 3D-HEVC

Chi *et al.* [40] use the maximum depth of the co-located CU in the base view as a threshold to limit splitting the current CU in the dependent views.

Khan and Khattak [41] use HEVC simulcast, *i.e.*, views are encoded independently with HEVC, for base views. For dependent views, the maximum value of the depths of the co-located CTU in the base view and its eight neighboring CTUs are used to limit the maximum depth value of the current CU.

Wang *et al.* [42] propose an early termination depth splitting algorithm for MV-HEVC. For a base view, three spatially neighboring CUs (Left (L), Upper (U), and Upper right (UR)) as well as co-located CTU (T) are used to determine the maximum depth value for the current CU. For dependent views, co-located CTU in the base view, called interview CU (I) is used in addition to the above-mentioned neighboring CUs and D_{pre}, which will be used in the final decision, is determined as in Eq. 7.

$$D_{pre} = \begin{cases} max\{D_L, D_U, D_{UR}, D_T\} & \text{Base View} \\ max\{D_L, D_U, D_{UR}, D_T, D_I\} & \text{Dependent View} \end{cases} \tag{7}$$

To avoid wrong decisions, another condition is used to determine the maximum depth value of the current CU (C) which uses motion vector information of three spatially neighboring CUs as shown in Eq. 8.

$$D_{max} = \begin{cases} D_{pre} & MV_c = MV_L || MV_c = MV_U || MV_c = MV_{UR} \\ 3 & \text{others} \end{cases} \tag{8}$$

Zhang *et al.* [43] use a two-step strategy for 3D-HEVC to speed up the search for the optimal depth level for CUs for dependent views. First, an early merge mode decision is made to avoid unnecessary searching of intra and inter modes. If interview CTU and its four immediate neighboring CTUs in the base view are encoded as a merge mode and the RD cost of the skip mode is less than $2N \times 2N$ merge mode for the current CU, only the merge mode for the current CU is searched. Second, CU splitting is terminated if the following two conditions are satisfied: *a)* depth of the current CU is equal to or larger than the maximum depth of the interview CTU and its four immediate neighboring CTUs in the base view, and *b)* skip mode is selected as the best prediction mode for the current CU after checking all the possible prediction modes.

Wang *et al.* [44] propose a depth range selection algorithm for dependent texture views in 3D-HEVC. First, split complexity (SC) of left (L) and upper (U) spatially neighboring CTUs, temporally co-located CTU (C) and co-located interview CTU (I) is calculated using Eq. 9.

$$SC_i = \begin{cases} \frac{1}{256}\sum_{j=0}^{256} d_j & \text{if } depth_{max} = 0,1,2 \\ \frac{1}{128}\sum_{j=0}^{256} d_j & \text{if } depth_{max} = 3 \end{cases} \tag{9}$$

Where d_j represents depth level of 4×4 CUs inside a CTU and $depth_{max}$ is the maximum depth of the CTU. Second, SC of the current CTU (SC_{pre}) is predicted using the Eq. 10.

$$SC_{pre} = w_C SC_C + w_I SC_I + w_L SC_L + w_U SC_U \tag{10}$$

where w_i is a weighted value. Then a pre-determined threshold is used to find the depth range for the current CU.

Silva *et al.* [45] propose a CTU partitioning algorithm for the texture CTUs based on the interview correlation for 3D-HEVC. Independent views are encoded by simulcast HEVC, while dependent views are encoded using the depth information of the corresponding independent view. For the current CU, the corresponding CU in the independent view is determined by a disparity vector $d(Y)$ [46] which is calculated as shown in Eq. 11.

$$d(Y) = \frac{f \times l}{255}\left(\frac{1}{Z_{near}} - \frac{1}{Z_{far}}\right)Y + \frac{f \times l}{Z_{far}} \tag{11}$$

where l is the distance between two adjacent cameras, f is the focal length, Y is the depth view value, and Z values define the depth range of the scene. Positions of two CUs are mapped using Eq. 12.

$$x_{Ind} = x_{Dep} + d(Y) \tag{12}$$

where x_{Ind} is position of the current CU in dependent view and x_{Dep} is the position of the same CU in the independent view. Finally, CU depth map is refined.

Mora *et al.* [47] exploit the correlation between CU depth splitting in texture and depth for 3D-HEVC. For the cases that depth is encoded before the texture, the texture quadtree starts from the coded depth quadtree. Otherwise, the depth quadtree is limited to the maximum and minimum depth values of the texture quadtree.

Khan *et al.* [48] use the maximum depth level of interview co-located CTU and its eight spatially neighboring CTUs as a threshold to stop splitting of the current CTU for 3D-HEVC.

Fu *et al.* [49] utilize Depth Intra Skip (DIS) for depth map coding, which directly uses reconstructed value of spatial neighboring CUs to represent the current CU for 3D-HEVC.

3.1.3. Multirate

Schroeder *et al.* [19] first encode the video sequence at the highest bitrate and then use encoded representation as the reference for encoding dependent representations. As CTUs tend to take larger depths at higher bitrate/quality representation than the lower bitrate/quality representations, the maximum depth value of a CTU in the reference representation is used to limit splitting co-located CTUs in the depended representations.

The same idea has been extended in [50] to be used by dependent low-resolution representations. As CTUs do not contain the same area in different resolutions, the corresponding area (A) for the current CTU in a dependent low-resolution representation is found in the reference representation. Thereafter, the percentage of CTUs (p_i) encoded at depth (less than or equal to) $i \in \{0, 1, 2\}$ is measured. If p_i (starts from $i = 0$) is greater than or equal to a threshold θ, the current CU is not split, and the process moves on to the next CTU.

Ideas in [19] and [50] has been combined in [51] to introduce an algorithm that is used for both dependent low-quality and low-resolution representations.

Amirpour *et al.* [52] propose using both the highest and the lowest quality representation to limit CU depth search range. First, the highest quality representation is encoded using standard HEVC, then CTU information obtained in this encoding is used to encode the lowest quality representation similar to [19]. Finally, these two representations are used to put both a lower and upper bound for the CU depth search range while encoding the immediate representations. Maximum depth value in the co-located CTU in the highest representation is used as the upper bound, and minimum depth value in the co-located CTU in the lowest representation is used as the lower bound for the CU depth search range.

In another study, Amirpour *et al.* [53] focuses on improving the parallel encoding performance for multi-rate encoding. Different quality representations are chosen as the reference representation to evaluate the performance of parallel encoding. The middle-quality representation is used as the reference representation based on these experiments.

Çetinkaya *et al.* [54] proposes a fast multi-rate encoding method using machine learning (FaMe-ML) with a specific focus on parallel encoding. The lowest quality representation is chosen as the reference. The encoding information from the reference representation and the Y, U, and V information from the raw video is fed into a convolutional neural network (CNN) to obtain a split decision for a given CTU in the given quality level. The decision from the CNN is used to speed up the parallel encoding.

3.2. Inherent Approaches

Inherent approaches use information available within the CTU to give the final decision about CTU partitioning.

Kim *et al.* [23] decide CU early termination based on the RD cost of a CU. If it is below a threshold, then current CU is selected as the best one and search process is stopped. A pre-calculated threshold for each CU depth level is considered which is calculated for each CU with size $N \times N$ as $TH_{N\times N} = A_{N\times N}\, e^{(W_{N\times N} QP)}$.

In the remainder of this section, we discuss about rest of the inherent approaches by categorizing them into Bayesian and texture complexity based approaches.

3.2.1. Bayesian Approaches

Bayesian rule based CTU partitioning is an another approach applied in the literature. Bayesian rule is shown in Eq. 13.

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \tag{13}$$

where $p(x|y)$ is *posterior*, $p(y|x)$ is *likelihood*, $p(x)$ is *prior* and $p(y)$ is *evidence*. In the context of CTU depth decision, CU split decision is seen as the *posterior* and different Bayesian approaches are used to estimate it.

Shen *et al.* [55] propose selecting certain features, *e.g.*, RD cost and inter-mode prediction error, and then use them in minimizing the Bayesian risk which is further included in the CU split decision. CU split decision is defined as a two-class classification problem where split class is w_s and not-split is w_n. Bayesian Risk R of splitting a CU is defined as shown in Eq. 14.

$$\begin{aligned} R(w_s|F) &= C_{s,n}P(w_n|F) \\ R(w_n|F) &= C_{n,s}P(w_s|F) \end{aligned} \tag{14}$$

where $C_{s,n}$ is RD cost of splitting CU when the ground-truth is not-splitting and $C_{n,s}$ is vice versa. $P(w_s|F)$ is the conditional probability of splitting a CU given the feature vector F which can be re-written using the Bayesian rule in Eq. 15.

$$P(w_s|F) = \frac{P(F|w_s)P(w_s)}{P(F)} \tag{15}$$

where $P(w_s)$ is the prior probability of splitting a CU. $P(F)$ is constant so it can be ignored and if we replace $P(w_s|F)$ in 14 with $P(F|w_s)P(w_s)$, final Bayesian threshold can be written as shown in Eq. 16.

$$\frac{P(F|w_s)}{P(F|w_n)} < \frac{C_{s,n}P(w_n)}{C_{n,s}P(w_s)} \tag{16}$$

$P(F|w_i), i\epsilon\{n, s\}$ is calculated offline for each QP and resolution settings. Overall, for each CU depth decision, feature vector F is extracted and the offline calculated threshold is used for the decision. CU is not-split when this condition is met, otherwise it is split.

Lee *et al.* [56] propose a similar approach that benefits from Bayesian rule and models the CU decision problem as a two-class classification. For both CU depth skip decision and early termination, thresholds are used and calculated using the Bayesian rule. Statistical probabilities for defining thresholds are updated at predefined intervals to keep the approach relatively content-sensitive.

Xu *et al.* [57] use the motion of co-located CTUs for early CU depth search termination along with a Bayesian risk-based discriminant function for detecting skip modes. CU depths in the co-located CTUs are directly used for early split decision of depth 0 and depth 1. Also, average and predicted motion vectors are used to calculate the motion diversity of the co-located CTU, and it is used for early termination. If the co-located CTU is split and motion diversity is high, then the current CTU is early split, and the mode search is skipped. Finally, they model skip mode detection as a two-class classification and use Bayesian risk to approximate it similar to [55, 56]. Skip mode detection is also used in the early CU depth search termination. It is modeled as a two-class classification problem again, and Bayesian risk-based skip mode detection is used in the decision process.

Table 6
Statistics Approaches.

Method	Approach	Features	Mode
[20]	Neighbour CTU	Depth Values	Intra
[21]	Neighbour CTU	RD Cost, Prediction mode	Intra
[22]	Neighbour CTU, Co-located CTU	Depth Values, Frame Type	Inter/Intra
[24]	Neighbour CTU	Depth Values, RD Cost	Inter/Intra
[25]	Neighbour CTU, Co-located CTU	Depth Values, RD Cost	Intra
[26]	Co-located CTU	Depth Values	Inter/Intra
[27]	Neighbour CTU, Co-located CTU	Depth Values	Inter
[28, 29]	Co-located CTU, Neighbour CTU	Depth Values	Inter/Intra
[30]	Co-located CTU	Depth Values, PU Mode	Inter/Intra
[31]	Co-located CTU, Neighbour CTU	Depth Values	Inter/Intra
[32]	Co-located CTU	Depth Values	Inter
[33]	Co-located CTU	Depth Values, Motion Vectors, RD Cost, skip Mode	Inter
[34]	Co-located CTU	Depth Values	Inter/Intra
[35]	Co-located CTU, Neighbour CTU	Depth Values, Motion Vectors	Intra
[36]	Co-located CTU, Frame History	Depth Values	Inter
[41]	Co-located CTU, Neighbour CTU	Depth Values	Inter
[42]	Co-located CTU, Neighbour CTU	Depth Values, Motion Vectors	Inter
[45]	Neighbour CTU, Co-located CTU	Depth Values, Camera Properties	Inter
[47, 49]	Neighbour CTU	Depth Values	Inter
[19, 50, 51]	Multirate	Depth Values, Motion Vectors, Encoding Parameters	Inter
[52, 53]	Multirate	Depth Values, Reference Frames	Inter
[54]	Multirate	Depth Values, Reference Frames, Prediction Mode, RD Cost, Variance	Inter
[55]	Bayesian, Sub-CU	Variance	Inter/Intra
[56]	Bayesian, Frame History	RD Cost	Inter
[57]	Bayesian, Neighbour CTU	Motion Vectors, Depth Values, RD Cost	Inter
[58]	Bayesian	Depth Values, RD Cost	Inter/Intra
[59]	Bayesian, Neighbour CTU, Sub-CU	Variance, Depth Values, RD Cost	Intra
[60]	Texture Complexity	Motion Vectors	Inter
[61]	Texture Complexity, Sub-CU	Motion Vectors	Inter
[62]	Texture Complexity	Variance, Depth Values	Intra
[63]	Texture Complexity	Depth Values, RD Cost	Intra
[64]	Texture Complexity	Edge Complexity	Intra
[65]	Texture Complexity	Variance	Intra
[66]	Texture Complexity, Sub-CU	Variance, Edge Complexity	Intra
[67]	Texture Complexity, Co-located CTU, Neighbour CTU	Depth Values, DCT Coefficients	Intra
[68]	Texture Complexity, Sub-CU	RD Cost, SAD	Inter
[69]	Texture Complexity, Neighbour CTU	Depth Values, RD Cost	Intra
[70]	Texture Complexity, Neighbour CTU	Depth Values, Variance, Texture Complexity, DCT Coefficients	Inter
[71]	Texture Complexity	Variance, SAD, QP	Intra
[72]	Texture Complexity, Neighbour CTU	Edge Complexity, Depth Values	Intra
[73]	Texture Complexity	Motion Vectors, RD Cost	Inter
[74]	Texture Complexity, Frame History	Motion Vectors	Inter
[75]	Texture Complexity	Variance	Intra
[76]	Texture Complexity, Neighbour CTU	Variance	Intra
[77]	Texture Complexity, Frame History, Neighbour CTU	Depth Values, Motion Vectors	Inter
[78]	Texture Complexity, Neighbour CTU, Co-located CTU	Depth Values, Luminance Values, Encoding Mode	Intra

All the statistical parameters in Bayesian approaches are updated periodically to preserve the content sensitivity of the approach.

Jiménez-Moreno *et al.* [58] propose using Likelihood ratio test (LRT) in Eq. 17 for every CU depth level .

$$\frac{P(x|depth^* > d)}{P(x|depth^* = d)} \gtrless \frac{C_{s,n} - C_{n,n}}{C_{n,s} - C_{s,s}} \tag{17}$$

where x is the input feature, d is the current depth, $depth^*$ is the optimal depth, $C_{s,n}$ is cost of splitting CU when the correct decision is not splitting, $C_{n,s}$ is vice versa, $C_{s,s}$ is cost of splitting the RD cost where the correct decision is also splitting and $C_{n,n}$ is vice versa. If the ratio is lower, then CU is not split otherwise it is split. Here RD cost of splitting CU is used as the the feature, x. Statistical properties that are used in the LRT are first extracted by analyzing number of sequences offline and they are then updated online to better adapt to the changes in the content.

Lee and Jeong [59] use the pixel variance difference between CUs and sub-CUs for measuring local complexity. Also, a predicted depth value is obtained using weighted information of neighboring CTUs. These two attributes are then combined to be used in early split decision. Finally, a Bayesian decision rule based Quadratic Discriminant Analysis (QDA) is used to classify early termination for CUs. All the thresholds in this method are updated by an online learning approach using statistical properties that are extracted during encoding.

3.2.2. *Texture Complexity*

Texture complexity of the CTUs are also exploited commonly in the literature using different approaches to detect motion.

Jamali and Coulombe [79] propose an intra coding method based on the global and directional gradients. Based on the accuracy of the prediction in the current depth, CUs are classified into two categories: *split* and *non-split*. The classification is solved by using global gradient and the mean of gradient amplitudes (MGA) is calculated using Eq. 18.

$$MGA = \frac{1}{n}\sum_{i}\sum_{j}|G_X(i,j)| + |G_Y(i,j)| \tag{18}$$

where G_X and G_Y are horizontal and vertical Sobel gradient components with 3×3 convolution masks applied at each pixel of the current CU. As the CUs with larger MGA have more details, they tend to split while splitting CUs with smaller MGA are stopped at the current depth level. To apply the impact of QP and CU size, $f1$ is calculated using Eq. 19 and it is used to determine CU class.

$$f1 = \frac{MGA}{\alpha} - QP \tag{19}$$

where α is related to the CU size. For $f1$ values less than a threshold splitting process is terminated.

Moreover, the directional gradient is used for intra mode decision. Each CU is classified into a non-split class if the CU intra mode is predicted with higher accuracy at the current level, or a split class, if there is no intra mode, to effectively predict the CU partitioning. By early termination of the splitting process for the non-split class, the encoder saves a considerable amount of time and computations since further splitting would require many RDO computations to find the optimum splitting pattern.

Jian *et al.* [60] indicate that there is a strong correlation between RD cost of a CTU and the corresponding variances of pixel motion vectors. It is shown that when the motion is strong, blocks tend to take smaller CUs, and when the motion is weak, blocks tend to take larger CUs. Based on these facts, a pyramidal motion divergence (PMD) method is proposed where frames are downsampled to $1/16$, and then their estimated optical flows are used to calculate PMD features. Thereafter, k nearest neighboring-like method is used to predict CU sizes.

This approach is further improved in [61] since calculating PMD was a time-consuming process. Instead, this time variance of absolute difference (VAD) is used as a metric. MVs of neighboring CTUs are used for calculating VAD and Pyramid VAD (PVAD), which is calculated using the downsampled versions. PVAD is used as a feature in the CU split decision problem. Furthermore, the CU split decision is modeled as a Markov Random Field (MRF), and the graph cut method is used to find a decision. The encoded frame is represented as a graph, and CUs are nodes of the graph. Two terminal nodes source S and sink T represent split and unsplit decisions. SVMs are also used in the energy function of the graph to determine the unary term, which is then used for calculating the likelihood of splitting a given CU. Then CU split decision is given based on the minimum cut in the corresponding graph.

Chiang *et al.* [62] calculate variance of pixels inside the CU (Var_{cu}) and compare it with a pre-defined threshold to determine CU decision for 3D-HEVC. CU is split if Var_{cu} is larger than a threshold, otherwise depth of the CU is compared with the depth of co-located texture CU and depth search is terminated if it is larger or equal.

Li *et al.* [63] propose another method for 3D-HEVC and use pre-determined thresholds as well as RD-cost of CU at depth 0 ($J0$) to determine the maximum depth value (d_{max}) for a depth map CTU. The maximum depth value is determined as shown in Eq. 20.

$$d_{max} = \begin{cases} 0 & \text{if } J_0 \leqslant Th0 \\ 1 & \text{if } Th0 \leqslant J_0 < Th1 \\ 2 & \text{if } Th1 \leqslant J_0 < Th2 \\ 3 & \text{if } J_0 > Th2 \end{cases} \tag{20}$$

A key-point detector that finds high-frequency areas in the image is used to decide on CU depth search range by Kim *et al.* [64]. The primary motivation behind this approach is that, in HEVC, high-frequency areas are given higher CU depths. Thus, in the proposed method, an adaptive key-point threshold is decided first, and if there are not enough key points in the current CU depth level, further depths are not searched.

Nishikori *et al.* [65] use the variance of the image to determine the characteristic of the region. If the variance is below a predefined threshold, that region is regarded as flat, and the current CU depth is used; otherwise, the CU depth value is increased, and the variance is rechecked. Min and Cheung [66] propose to use edge complexity to find CU depth sizes. The edge complexity is calculated in four directions as the luminance difference of the two halves in the corresponding direction. The edge complexity is calculated for all sub-CUs, and if all edge complexity values are smaller than a predefined threshold, the corresponding CU is not split.

Huang *et al.* [67] use CU texture complexity along with spatially neighboring CTU depth information. It is calculated by quantizing the variance of the CU into five category levels. CU split decision is given based on the categorization of the CU and information from neighboring CTUs.

Xiong *et al.* [68] propose an approach that designs a new motion estimation (ME) method which can obtain the sum of absolute differences (SAD) costs of both the current CU and sub-CUs. It also defines an exponential model used to calculate the relationship between motion compensation RD cost and SAD cost. This model is used to calculate a threshold used in CU depth decisions by comparing it with the SAD cost difference.

Song *et al.* [69] use the discretization total variation (DVT) to calculate CU complexity that is further used in CU depth selection. If the DVT is high, it means that the current CU is a complex CU that should have higher depth values.

A Discrete cosine transformation (DCT) based approach is proposed by Liu *et al.* [70]. DCT coefficients are checked for early termination. If all DCT coefficients are zero, then the search range is set to 0 and 1 only; otherwise, neighboring depth information is checked. If there is not enough correlation between current CTU and neighboring CTUs, then edge gradient using Sobel edge detector is found and used as the main feature in CU depth decision.

Ramezanpour and Zargari [71] define a smoothness parameter for each CU. If this parameter is lower than a threshold, PU modes are computed for the current CU, and further division is skipped. The smoothness parameter is based on SAD values calculated for horizontal, vertical, right, and left diagonal directions. The variance of these four SADs is defined as the smoothness parameter.

Shang *et al.* [72] define edge pixel density ρ_{edge} as in Eq. 21 to represent CU texture complexity.

$$\rho_{edge} = \sum_{edge} / N^2 \tag{21}$$

where $\sum_{edge}$ is number of edge pixels produced by Canny operator and N is the width of the CU. If ρ is not equal to zero, CU size 64×64 is skipped.

CU complexity is decided by checking the cost of encoding MVs of the current CU and early search termination is done based on a threshold in the method proposed by Shan *et al.* [73].

Fernández *et al.* [74] apply motion estimation on input images and homogeneity analysis is done that is further used to give CU split decision. Input frames are analyzed before encoder starts encoding and the process is done on the GPU; thus it does not introduce any overhead for the CPU. After this process, the mean absolute deviation of the motion vectors is obtained for the CU, and the split decision is made based on that. If it is below a certain threshold, then further splits are stopped.

CTU partitioning decision is made based on the complexity of the CU both in macro and micro levels by Zhang *et al.* [75]. The first frame of the video is analyzed, and the video is categorized into three sizes based on the number of CTU blocks in the frame. If the number is below a threshold, then the number of pixel types is used to determine a rapid decision about CU split. Otherwise, a final decision is given based on the statistical properties, *e.g.*, entropy, and texture complexity of CU, and thresholds are set adaptively.

Hou *et al.* [76] calculate the complexity of the CU as $complexity = log(E)$ where E represents the variance of the pixel values in the CU. Complexity of the current CU is compared with the complexity of the left and top CTUs, and if the complexity of the current CU is smaller than the left and top CTUs, the algorithm stops splitting the CU.

Cebrián-Márquez *et al.* [77] use a pre-analysis stage called the look-ahead stage. In the look-ahead stage, the motion information of the sequence is estimated before starting encoding, and this information is later used to guide the CU depth decisions during the encoding process. In this stage, the motion estimation is carried out for each block size and on every reference frame, and the resulting RD costs are stored. Here the same ME algorithm is used as the one in the inter prediction module of standard HEVC to obtain consistent results. Instead of making a full-motion vector prediction, MVs of spatially neighboring CTUs are used as predictors to speed up the process. The predicted motion information is used for determining a cost function for the given CU, and this cost function is used in determining the final partition. This estimated cost function is calculated using the predicted MV in the look-ahead stage and the distortion rate of the predicted MV compared to the original one. This cost is used as a threshold for a split decision in a bottom-up manner. If the splitting cost is higher than this threshold, then the CU depth is decreased by one until this cost becomes smaller.

Texture similarity between temporally and spatially neighboring CTUs is used to early terminate CU depth search by Lu *et al.* [78] for HEVC-SCC extension. The density of the luminance disparity (DLD) is calculated using Eq. 22, and it is used as the measure.

$$DLD = \frac{\sum_{x=1}^{W} \sum_{y=1}^{H} |I_{cur}(x, y) - I_{col}(x, y)}{W \times H} \tag{22}$$

where W and H are width and height of the current CTU, $I_{cur}(x, y)$ and $I_{col}(x, y)$ are luminance intensities of pixel location (x, y) in the current CTU and co-located CTU. If the DLD is smaller than 1, then the current CTU is categorized as *type 1* otherwise it is categorized as *type 2*. *Type 1* means there is a small variation in luminance compared to the co-located CTU, and the current CTU is expected to have the same CU depth as the co-located CTU. For *Type 1* CU: *(i)* if the depth is smaller than that of the co-located CTU, the CU is further split; *(ii)* if the depth is larger than or equal to that of the co-located CTU and the prediction mode for the co-located CTU is not PLT, then the CU search is early terminated; *(iii)* otherwise, a full-depth search is performed. For *Type 2* CU, spatially neighboring CTUs are also included in the decision process. The maximum and minimum depth among neighboring CTUs are obtained and form a bound for the CU depth search range as D_{max} and D_{min}. If the depth of the current CU is smaller than D_{min}, the CU is split. If the depth of current CU is larger than D_{max}, the CU search is terminated. Otherwise, a full-depth search is performed.

Sun *et al.* [80] use directional variance to measure texture complexity that is further used in CU depth decision. Directional variance for image X in a given direction r can be written as shown in Eq. 23.

$$DirVar(X, r) = \frac{1}{N} \sum_{i=1}^{n} \sum_{j=1}^{k_i} \left| X_j - X_{L(r,i)} \right| \tag{23}$$

where N is the number of pixels in X, $X_{L(r,i)}$ is the average luminance value along a line with slope r and offset i, X_j is each pixel in the same line, n is total number of lines, and k_i is pixel location in line i.

This approach calculates the sum of variances along a line with a given slope which makes it sensitive to directions, *e.g.*, if there are edges along a certain direction, then the directional variance will be larger in that direction. This allows capturing texture direction information in the image. The set of slopes are determined based on the mode directionality of HEVC and four slopes are selected, *i.e.*, $0°, 45°, 90°$, and $135°$. Based on the directional variances in these slopes and pre-determined thresholds, the CU is categorized into one of the three groups, *i.e.*, homogeneous, complex, and undetermined. Homogeneous CUs are not split, complex CUs are split, and full RDO search is applied for undetermined CUs. Different threshold values are used for different QP values.

Overall, statistics based methods and common features used in them are summarized in Table 6.

3.3. Summary

Neighboring CTU information is commonly used in the statistics-based approaches since the correlation is vital for such CTUs. The most common approach here is to define a threshold and give the decision based on depth values of neighboring CTUs [20, 21, 22, 24, 25]. Some approaches directly use minimum and maximum depth values found in the neighboring CTUs to limit the search range [26, 27, 28, 29, 30, 33, 37, 38]. Additionally, categorizing frames using this CTU information and giving the decision based on the classification is another common approach [34, 35].

HEVC extensions and multirate approaches provide additional neighboring CTUs that can be used in the process. For MV-HEVC and 3D-HEVC, there is one extra co-located CTU in the interview frame, and for 3D-HEVC, there is also one extra CTU in the depth frame [15]. Numerous approaches are proposed that specifically aim to exploit those extra neighboring CTUs in these HEVC extensions [40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 19, 51, 52]. These methods also follow similar approaches as methods proposed for standard HEVC.

Inherent approaches, on the other hand, exploit the information available within the current CTU.

The bayesian rule is exploited in the context of CTU partitioning. Some approaches focus on minimizing the Bayesian risk of splitting CU by using different feature sets [55, 56, 57]. Here, RD cost is an essential feature since it can be correlated with the Bayesian risk. Also, Bayesian decision rules are used to determine CU split decision again RD cost being the main feature here as well [58, 59].

Calculating texture complexity and using it to determine CTU partitioning is also used commonly in the inherent approaches. This is useful since more complex CTUs tend to have larger depths to achieve better motion compensation. Numerous information sources are exploited here from directional gradients [79], pyramidal motion divergence [60, 61], RD cost of encoding CU [23], *etc.*However, the most common approach is to determine the texture complexity of the CTU using the variance of pixels since variance is strongly correlated with the texture complexity of the CTU [62, 63, 65, 67, 76, 80]. Moreover, motion vectors are key factors in determining texture complexity as well, and they are also exploited commonly in these methods [61, 73, 74, 77].

4. Machine Learning Based Approaches

Following the recent success of machine learning methods in numerous fields, ML based approaches have also been proposed for CTU partitioning. In this section, the approaches are categorized into two main groups, namely *(i)* traditional and *(ii)* deep learning methods. Traditional methods such as support vector machine and random forests use manually selected features for training, while deep learning methods extract useful features during the training phase from the data.

4.1. Traditional Methods

In traditional machine learning methods, the first and most important step is feature extraction. These methods typically use handcrafted features. One advantage of traditional ML methods over deep learning counterparts is their low complexity. Additionally, it is easier to interpret the results of traditional methods. Another advantage over deep learning methods can be seen when the data size is small since these methods are not dependent on the vast amount of data contrary to deep learning methods. In this section, we categorize traditional methods into three groups based on the main method used in the respective approach: *(i)* support vector machines (SVM), *(ii)* Random forests (RF), and *(iii)* bayesian learning.

4.1.1. Support Vector Machines

The main idea behind Support Vector Machine (SVM) [81] is to construct a high dimensional hyperplane that classifies data points and maps the given input to the hyperplane for classification. The main goal here is to output a hyperplane that gives minimum classification error. SVM is a supervised learning method meaning that correct labels are available for the entire training dataset. SVMs are usually used for classification tasks, but they can also be used for regression. SVMs are commonly used for CTU partitioning [82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92].

Specifically, Xue *et al.* [82] model CU splitting as a binary classification problem and use SVM to solve it. Three different SVMs are used, one for each depth level, and they are trained offline using the mean squared error (MSE) and the number of encoded bits (NEB), which are extracted from the CUs with different depth levels.

Shen and Lu [83] assign weights to the training samples based on the RD loss increase caused by the misclassification of the sample to reduce the effect of outliers and misclassification. Based on this approach, the CU depth prediction process is performed before checking depth levels to reduce time-complexity by Zhang *et al.* [84]. An initially weighted SVM is used to predict CU depth level distribution for the CTU, and the search process is shaped based on the result.

Liu *et al.* [85] utilize separate weighted SVMs for each depth level and use features related to texture complexity for training SVMs. Texture complexity of the CUs is measured using Sobel filter on luminance values of pixels and used as a feature along with QP. CU splitting is modeled as three-class (*complex CU, homogeneous CU, uncertain CU*) classification problem. If the CU is classified as a complex CU, it is further split. If it is classified as a homogeneous CU, splitting is stopped, and for uncertain CUs normal encoding process is applied.

Two linear SVMs are used for CU split and early CU termination decisions by Zhang *et al.* [86]. Depth difference and Hadamard transform-based (HAD) cost ratio between current and neighboring CTUs are used for early CU split decision. RD cost is also used along with these two features for early CU termination.

Liu *et al.* [87] use an artificial neural network (ANN) to determine weights for outputs of different SVMs. SVMs trained with variance information are given the highest weights, and SVMs trained with neighboring CTUs depths, and pixel differences are given the equal weights based on the ANN results.

Zhu *et al.* [88] use fuzzy SVM to directly predict CU depths and models CTU partitioning as a multi-class classification. Three SVM-based classifiers are used that are trained with manually selected features and updated periodically. The SVM can also predict uncertainty for those examples that are not confident, meaning they are in the risk area, and in these cases, the standard HEVC search process is used. The risk area is determined by RD cost optimization. During the training phase, each sample is weighted by fuzzy SVM so that the outliers do not reduce the overall accuracy. After the classifiers are trained, the CU structure of the remaining frames in the GOP is directly predicted. If the sample is in the risk area, then standard HEVC search is applied.

Two SVMs are used for CU decisions by Zhang *et al.* [89]. The SVM is trained offline and outputs three decisions, *i.e.*, split, non-split, or full search. If the output of the first classifier is full search, the second classifier is used, which is trained online using the data from previously encoded frames to further predict the CU size.

Erabadda *et al.* [90] use weighted SVMs that are trained online during encoding cycle in a two layered structure to determine CU depth early termination. In the the first level, two SVMs are used for each CU depth, and training data is collected using default RD optimization of HEVC. Features are selected as texture complexity, RD cost of the current depth, and context information. The weights for these SVMs are calculated using precision scores of split decisions. CUs that are not categorized by the first layer SVMs are passed to the second layer. In the second layer, there is only one SVM per CU depth level, and training data collection is half of the amount compared to the first layer. Weight calculation this time is done by F-score of CU split decisions. In both levels, SVMs are re-constructed after some period to induce some context adaptivity to the SVMs. Additionally, they use complexity control parameters to control the number of CUs that have reached the second level to be decided by exhaustive RD optimization.

A two stage algorithm for CTU partitioning is proposed by Erabadda *et al.* [91]. In the first stage, offline trained SVMs are used to make the CU splitting decision, and the second stage is used to apply the decision of the SVMs. If the decision is split, then the current level CU calculations are skipped. SVMs are not used for depth 2 since their usage decreased performance significantly during the experiments. Depth information from neighboring CTUs and co-located CTU, QP value, RD-Cost, and texture information are used as features for SVMs. A different set of features is used for depth 0 and depth 1 decisions based on their F-scores.

Xue *et al.* [92] model the CTU depth decision for HEVC-SCC as a binary classification task and train separate classifiers for each depth level. In particular, L1-loss linear SVM is used as a classifier. Due to imbalance in the training dataset for SCC, *e.g.*, split decisions are much larger than non-split, an ensemble learning approach is adopted. Random subsets are generated from the training set, which are then used for training classifiers, and outputs of classifiers are given weights for the voting function. The mean absolute deviation of luminance values for CU is used to measure texture complexity. However, since text regions are common in screen content, the range of luminance values in each CU is quite narrow. To overcome this problem, the number of luminance components and the range of luminance values in the current block are also used as additional features. Bitrate, RD cost, and QP of planar mode are also used along with the depth values of left and up CTU. For CU depth decision, the first planar mode is checked, and a feature vector is obtained. After that, a classifier is called, and the decision is made according to the output. If the output is a certain split or a certain not-split, these decisions are applied; otherwise, the full search is conducted.

4.1.2. Random Forests

Random forests (RF) [93] are also commonly used in the literature. Random forests consist of many decision trees. In decision trees, the prediction is made by gradually limiting the search range of the problem. At the root, a general elimination is done based on a general condition, and once it is moved deeper in the tree, the conditions become more and more specific, thus limiting the search range. A decision tree can have multiple output branches, and the goal is to find the output that minimizes the error. In RF, multiple decision trees that are trained with random training samples and use random feature sets are combined to produce the final output, which is obtained by combining the weighted output of the decision trees.

Duanmu *et al.* [94] propose a method for HEVC-SCC which uses decision trees for CU decision. In the beginning, statistical properties are processed to extract useful features for the decision tree, then a decision tree is trained to classify block type for screen content. Following that, another decision tree is used to decide whether or not to split a CU further. Features that capture texture complexity are used to train decision trees.

Ruiz-Coll *et al.* [95] use a decision tree to determine early termination for CU depth search. Low complexity features are used to train trees, and they are only used for CU depths 0 and 1.

CU splitting for HEVC-SCC is modeled as a two-class classification problem by Yang *et al.* [96], and decision trees are used to solve it. The following features are used: variance of luminance of the current CU, maximum gradient magnitude ofthe current CU obtained by using Sobel operator, CU depth level, information mode of spatially neighboring CUs, and RD cost of CU. A separate classifier is trained for each depth level, and each classifier is trained offline using frames from several different sequences.

Du *et al.* [97] use random Forest to predict CU depth level. CU depth range is determined using CU depths of neighboring blocks, and each block is given different weights based on their location, *i.e.*, left and top blocks have higher weights than top-left and top-right blocks. If the depth range is 0 or 1, a random forest classifier is used to give the final decision. For the remaining depth levels (*i.e.*, 2 and 3), a standard HEVC RDO process is applied.

Tahir *et al.* [98] use three different random forest classifiers for skip mode, CU split, and TU split decisions. Several features are rank-ordered using a filtering-based approach to find the optimal number of features for each of the three classifiers. This results in 10 features for skip mode, 9 features for CU split, and four features for TU split decisions. Skip mode classification followed by CU split decision and TU split decision is made for each CU level.

The multirate algorithm proposed by De Praeter *et al.* [99] uses information of blocks in the fully encoded representation that is in the same location as the current block to predict the block structure of the current block. Then a random forest is used to predict which information will be used for encoding the remaining representations. Features are chosen as the variance of the transform coefficients, motion vector variance, and information about the block structure.

Bubolz *et al.* [100] extract features for training the random forest classifier of high bitrate video then train a random forest classifier for low bitrate representations. The classifier gives the decision of whether to stick with the same depth level as high bitrate representation or further split the CU for current representation for each depth level. This method mainly focuses on transrating processes.

4.1.3. Bayesian Learning

The bayesian rule is also used within machine learning based methods. Different from the Bayesian methods in the statistics based approaches, these methods decide parameters for Bayesian rule with online or offline training.

Kim and Park [101] choose training pictures based on scene change to update statistical parameters which are used for a Bayesian classifier that decides whether to search further depths for the given CU. An offline-learning-based loss matrix calculation is also used to improve the decision process further. After obtaining the thresholds using these two approaches, early CU termination is done by calculating posterior cost and comparing it with thresholds.

Another online learning based Bayesian Decision based Block Partitioning (BDBP) method is proposed by Yao *et al.* [102]. The frames in the video are divided into two groups, *i.e.*, online learning and fast prediction, based on scene change. A Bayesian risk threshold is defined using a Gaussian Mixture Model (GMM), and it is used for the split decision. The scene change is detected using the gray level difference between the frames. At each scene change, an online learning period of six frames is started because the threshold is no longer valid. During the online learning period, the CU decision is done by standard HEVC and statistical data is collected. When the online learning period is over, the fast prediction period is restarted, and the threshold is decided in the first frame of the fast prediction period using Expectation-Maximization (EM) algorithm, and EM is initialized with the K-Means method to avoid falling into a local minimum.

Chen and Lu [103] use an online progressive three-class Bayesian classifier along with a general Bayesian classifier. The first classification is done by a three-class classifier using features from the current CU and the neighboring CTUs. If the first classifier gives an indistinct decision for the CU, a second general Bayesian classifier is used to give the final decision. Both classifiers are trained online, and parameters are updated periodically.

CTU partition is modeled as binary classification problem by Kuang *et al.* [104] for HEVC-SCC. RD cost of the optimal mode for the current CU depth level, J, is used as a feature which is defined as $J = D_{SSE} + \lambda \times R$ where D_{SSE} is sum of squared errors between the current CU and the predicted CU, R is the bit cost, and λ is the Lagrange multiplier which is defined in Eq. 24.

$$\lambda = C \times \left(\frac{QP - 12}{3.0} \right)^2 \tag{24}$$

where QP is quantization parameter value and C is a parameter determined by the picture type and coding structure. They calculate J directly from the SCC encoder, thus no overhead is introduced. For the CU depth decision, CUs are categorized into three groups based on their optimal modes, *i.e.*, Cintra, IBC, and PLT. Bayesian rule in Eq. 25 is used for determining early CU termination:

$$P_{group_n,d}(w_j|J) = \frac{P_{group_n,d}(J|w_j)P_{group_n,d}(w_j)}{P_{group_n,d}(J)} \tag{25}$$

where $j\epsilon\{split, notsplit\}$, $group$ is determined by the optimal encoding mode, $P_{group_n,d}(w_j)$ is the prior of w_j, and $P_{group_n,d}(J)$ is the probability density of J. Both of these probabilities are obtained using statistical information during the online-learning phase. Probability density of J is obtained using the formula in Eq. 26.

$$P_{group_n,d}(w_j|J) = \sum_{w_j} P_{group_n,d}(J|w_j)P_{group_n,d}(w_j) \tag{26}$$

CUs with smaller J are more likely to belong to $w_{nonsplit}$ class and to be early terminated but there are still exceptions observed during experiments. Thus, this early termination method is only applied for Cintra and IBC modes. Training frames for online-learning phase are chosen based on the scene-change detection which is done by dividing the frame into blocks and using distinct color number in each block. Those frames are then used for updating statistical parameters.

4.2. Deep Learning Methods

Following the recent success of deep learning methods in various tasks, several deep learning based methods are also proposed for CTU depth decisions algorithms. Deep

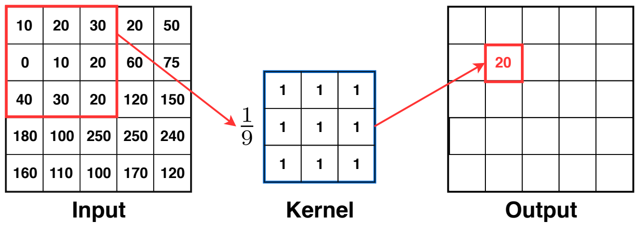

Figure 10: Sample Convolution Operation.

learning methods can benefit from an excess amount of data thanks to their ability to learn complex non-linear functions of given inputs [105]. Moreover, in deep learning methods, features are extracted implicitly by the model during training which eliminates the need for feature engineering. Deep learning methods are more complex compared to traditional methods, hence their training time is longer, and they require more data to work. However, if the requirements are met, deep learning methods generally provide better performance compared to traditional machine learning methods for a variety of tasks.

4.2.1. *Convolutional Neural Networks*

Convolutional neural networks (CNNs) are commonly used for the CTU partitioning algorithms since they are good with 2D data such as images. CNNs are a special type of neural networks (NNs) in which mainly convolution operations is used. Convolution can be written as shown in Eq. 27.

$$Y[i,j] = \sum_{k,l} f[k,l]X[i-k,j-l] \tag{27}$$

where Y is *output*, f is *kernel*, and X is *input*. An example of convolution operation can be seen in Fig. 10. In CNNs, convolution operations are applied consecutively to the given input, thus extracting different features in different layers which makes them extremely useful for capturing important features in the given image.

Kim and Ro [106] use a CNN for the CTU partitioning. CTU partition is searched as a top-down approach from depth 0 to 2, and in each level, the CNN decides on whether to split into more depth or to stop. The Network is trained using luminance values in the frame, and manually chosen features are also used as an additional vector to improve the prediction accuracy.

Xu *et al.* [107] propose to use a unique representation for the CU prediction, *i.e.*, a hierarchical CU partition map (HCPM) to efficiently represent the CU partition of the CTU. Using HCPM and CNN, it can predict the depth decision for the entire CTU in a single inference for intra frame, rather than running prediction for each depth level as in [106]. It also uses long short term memory (LSTM) to reduce the inter-frame CU complexity.

Li *et al.* [108] use a deep CNN at each depth level to predict the CU split decision. The texture of the CU is used as the only feature, and CNNs are trained offline using a raw image database. A new dataset for Intra-mode HEVC is constructed to train the network. 2000 images are encoded with the HEVC reference software, and binary labels are obtained for all CUs. The dataset is then grouped by the QP value and resolution. A separate CNN is trained for each depth level which gives a binary decision for each CU whether to split or not to split.

An asymmetric CNN structure that takes the luminance value of the CU as an input is used to predict both CU and PU partitioning decisions by Shi *et al.* [109]. The network is divided into three branches in which two of them have asymmetric kernels in convolutional layers to capture horizontal and vertical features. Subsequently, outputs of sub-networks are concatenated in depth level, which is followed by two convolutional layers that extract weight information of features. Different networks are used at each depth level, and PU prediction is modeled as a special case of CU prediction. For PU prediction, softmax activated output of the network is used as confidence level, and PU prediction is made if it is above certain thresholds, which are determined differently for each QP level.

Zhang *et al.* [110] use both statistics and machine learning based approaches. A texture complexity threshold is defined based on QP and depth values. If the CU is determined to be homogeneous by the threshold, further searches are stopped; otherwise, the CU is sent to a CNN for classification. For CNN, kernel sizes are designed dependent on the CU depth, and also neighboring CTU information is included in the fully connected layers. Furthermore, the QP is included in the loss function. These changes provide an adaptive structure for CNN. There are different CNNs for each depth level.

Chen *et al.* [111] use two CNNs for the CU and PU predictions. CNNs are trained using frames and QP levels as inputs, and the final decision is given based on the predefined threshold. The CNN is only used for the split decision when the depth is 3, which decides if the CU needs to be split further for the PU mode. The CNN also considers the edge strength of the CU and QP level when giving the decision. This is achieved by training the CNN with QP level along with input frames. For other depth levels, the standard HEVC encoding process is applied.

A deep CNN, which is designed to classify texture information and propose object locations in the frame, is used to determine CTU partitioning by Kuanar *et al.* [112]. It first produces the ROI for possible objects, then object shape detection is done, and finally, the CTU is classified into one of the four classes (*depth* $0,1,2,3$) based on the combination of object and texture features. For the ROI proposal, predetermined anchor boxes are used, and labeling is done based on a predefined threshold for intersection over union (IoU) scores. For texture features, Fisher vectors are used to aggregate local features that are obtained by the convolutions, and the Principle Component Analysis (PCA) is applied to reduce the dimensionality.

Bouaafia *et al.* [113] use both SVM and CNN for the CTU partitioning. The CTU partitioning decision is modeled as a three-level binary classification, each level represents a depth level. Several frames are encoded using the original encoder, and these are used for the online training of the SVM. The SVM is then used for consecutive frames for CU split prediction. This training period is refreshed periodically to make the SVM content-adaptive. Moreover, the CNN is used for the CU split decision to make a comparison. Residual of the CTU, which is obtained by precoding the frame using the standard HEVC, is fed into the CNN. Pre-convolution layers in the CNN take the residual and transform it into three depth levels, *i.e.*, 64×64, 32×32, and 16×16. After this operation, convolution layers extract features in all levels and then features are concatenated and passed through fully connected layers. Finally, the CU split decision is predicted at each depth level. The CNN is trained offline with data generated using the HEVC encoder for several videos in different QPs and resolutions. The CNN is preferred over SVM since it gave a better performance during the experiments.

Several approaches focus on hardware implementations of the encoder using CNNs. Liu *et al.* [114] use a CNN to reduce CU depth search complexity in the hardware implementation. A subsampling operation to CUs is applied, and they are given to a CNN as 8×8 inputs. The CNN then predicts the best CU/PU candidate pairs but only CU texture information is used as the feature.

Another CNN-based approach that aims to reduce complexity in the hardware implementation is proposed by Li *et al.* [115]. As the starting point, homogeneity detection for a CU is done using edge strength (Sobel), QP, and motion vectors. If the CU is determined as homogeneous, then the search is stopped, otherwise the final decision is given by the CNN. Three sub-networks are used for the CNN, and their outputs are combined to provide the final decision. The first network is trained with the luminance of CTU, which analyzes texture complexity. The second network is trained with Integer Motion Estimation (IME) that explores residual features of the IME. Finally, a smaller network that is trained with motion vectors is used to analyze motion vectors. After that, the outputs are concatenated and passed through multi-layer perceptron (MLP) layers that give a binary output that defines whether or not to split further for the given CU.

4.2.2. Reinforcement Learning

There have been some attempts to use Reinforcement Learning (RL) approaches in the literature as well. In RL, the main goal for the agent is to maximize reward by optimizing the actions that are taken in given situations. The environment in RL problems is modeled as Markov Decision Process (MDP). In MDP, the environment is assumed to satisfy Markov Property, meaning that the future state is only dependent on the present state and it is independent of past states.

Chung *et al.* [116] use a deep RL approach for CTU partitioning. A deep Q-learning method that uses a CNN is used to predict CU split decision. The state for the algorithm is the luminance value and QP parameter, the action is split decision, and the reward is minimizing RD-cost distortion. Different models are used for each depth level, and training is done sequentially from depth 2 towards depth 0.

Li *et al.* [117] model the CU decision problem as an MDP and use an actor-critic neural network to determine early CU termination. The CU trajectory is used as a feature, and also neighboring CTU information is incorporated. A single model is used for different depth levels that take the action as the split decision for early termination.

4.3. Others

Several other machine learning approaches are proposed in the literature that does not fit well into the aforementioned categories and, thus, those methods are presented here.

Huang *et al.* [118] achieve efficient CU and PU detection by using a neural network (NN) and gray-level co-occurrence matrix (GLCM). This approach mainly focuses on HEVC-SCC. A simple NN is used for each CU depth level to classify the CUs into screen content CU (SCCU) or camera-captured content CU (CCCU). Texture complexity based on luminance values, corner point ratio, and distribution of luminance values is used as features. After classifying the CU, efficient PU mode is assigned based on statistical properties. Finally, efficient CU size is decided using information from neighboring CTUs and GLCM. GLCM is used to evaluate the texture complexity of CUs and early terminate CU search. Angular second moment (ASM) of GLCM, which represents uniformity of gray-level distribution, is used to measure the texture complexity of CU. If ASMs for both CUs are similar, they are assumed to have the same CU depth, and the search is stopped. Each neighboring CTU contributes differently to the calculation, and the weights are determined by Pearson's correlation coefficient of ASMs.

A neural network is used for the CTU partitioning for HEVC-SCC by Duanmu *et al.* [119]. A new feature, *i.e.*, sub-CU major directional inconsistency (MDI), which measures the consistency of features in sub-CUs, is defined. Separate NN classifiers are trained offline for different QP values and CU depths using the following features: *(i)* CU variance, *(ii)* CU distinct color number, *(iii)* CU color and gradient histogram kurtosis, *(iv)* CU edge pixel percentage, and *(v)* MDI of these features. The NN outputs a number between $0-1$ rather than making a binary classification. Thus, the output value is also used as a confidence value. If it is above 0.7, then the CU is split, if it is below 0.3, the CU is not split, and if it is between 0.3 and 0.7, full RDO is applied.

Tun *et al.* [120] model the CTU partitioning as an optimization problem and use the Genetic Algorithm (GA) to decide on the CTU partitioning. RD-cost based fitness function is used for the GA and the optimization is done based on that. To reduce the complexity of calculating the RD-cost, the spatial correlation between consecutive frames are used. The keyframes are chosen periodically, and the CTU partitioning of keyframe CTUs are shared among three consecutive frames and are used in the CTU partitioning prediction. The optimization is stopped if the best population is not changed two times.

Table 7
Machine Learning Based Approaches

Method	Approach	Features	Mode
[82]	SVM	MSE, Number of Encoded Bits (NEB)	Inter
[83]	SVM	Depth Values, RD Cost, Texture Complexity	Inter
[84]	SVM	Depth Values, skip Mode, Motion Vectors, QP	Intra
[85]	SVM	Variance, Texture Complexity	Intra
[86]	SVM, Neighbour CTU	Depth Values, Texture Complexity, Hadamard Cost	Intra
[87]	SVM	Depth Values, Variance	Intra
[88]	SVM, Neighbour CTU	Depth Values, SAD, RD Cost, Motion Vectors, QP	Inter
[89]	SVM, Neighbour CTU	Depth Values, Motion Vectors, Texture Complexity, RD Cost	Intra
[90]	SVM	Depth Values, Texture Complexity, RD Cost	Intra
[91]	SVM, Neighbour CTU, Co-located CTU	Depth Values, Texture Complexity, RD Cost, QP	Inter
[92]	SVM, Neighbour CTU	Depth Values, Luminance Values of CTU, Bitrate, RD Cost, QP, SCC Mode	Intra
[94]	Decision Tree	Sub-CU Difference, Variance, Edge Complexity, Pixel Values	Intra
[95]	Decision Tree	DCT Coefficients, Variance	Intra
[96]	Decision Tree, Neighbour CTU	Varinace, Luminance Values of CTU, Edge Complexity, SCC Mode, RD Cost	Intra
[97]	RF, Neighbour CTU	Luminance Values of CTU	Intra
[98]	RF	Depth Values, Variance, Pixel Values, Motion Vectors, RD Cost, Edge Complexity, QP	Inter/Intra
[99]	RF, Multirate	Depth Values, Variance, Motion Vectors	Intra
[100]	RF, Multirate	Depth Values, QP	Inter
[101]	Bayesian	Depth Values, Scene Change	Inter/Intra
[102]	Bayesian	Depth Values, Scene Change, Luminance Values	Inter/Intra
[103]	Bayesian, Neighbour CTU	Depth Values, Variance, Texture Complexity	Intra
[104]	Bayesian	RD Cost, QP, SCC Mode	Intra
[106]	Neural Network	Depth Values, Motion Vectors	Inter/Intra
[107]	CNN, LSTM, Co-located CTU	Depth Values, Pixel Values	Inter/Intra
[108, 109]	CNN	Luminance Values of CTU	Intra
[110]	CNN	Depth Values, QP, Texture Complexity, Luminance Values of CTU	Intra
[111, 114]	CNN	Luminance Values of CTU, QP	Intra
[112]	CNN	Luminance Values of Frame	Intra
[113]	CNN, SVM	Luminance Values of Frame	Inter
[115]	CNN	Luminance Values of CTU, Edge Complexity, QP, Motion Vectors	Inter
[116]	RL, CNN	Luminance Values of CTU, QP	Intra
[117]	RL, Neural Network, Neighbour CTU	Depth Values	Inter
[118]	Neural Network, Neighbour CTU	Texture Complexity, Luminance Values of CTU	Intra
[119]	Neural Network	Variance, CTU Color Information, Edge Complexity	Intra
[120]	Genetic Algorithm, Frame History	Depth Values	Inter
[121]	KNN Classifier	Texture Complexity, Edge Complexity	Intra

Statistical analysis of image content is used as a feature and a k-nearest neighbors (k-NN) classifier is used in [121]. Image complexity measure called as edge magnitude (EM) is calculated using Sobel operator with the formula shown in Eq. 28.

$$G_{f,c} = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \sqrt{G_h(x,y)^2 + G_v(x,y)^2} \quad (28)$$

where N is the width of the CU, f is the frame number, c is the CU address, G_h and G_v are the horizontal and vertical gradients computed by the Sobel operator at pixel location (x, y). The EM is then used for an early split decision. Moreover, CU splitting is modeled as a two-class classification problem and Fischer's linear discriminant analysis (FLDA) is used to transform statistical data into a more separable format. FLDA is used for reducing the dimensionality in the data in order to become easier for the classification. The k-NN classifier is then used to directly estimate the depth of the given CU. If the RD cost and the EM are high, the CU is classified into split class. Moreover, statistical parameters are updated online when there is a scene change or fast movement using training frames which are determined by the edge magnitude ratio (ER) between frames.

All machine learning based approaches and commonly used features are summarized in Table 7.

4.4. Summary

Different machine learning approaches have been proposed to tackle the CTU depth decision problem. The main trade-off for ML based approaches is the trade-off between increased accuracy and time complexity. SVMs and RFs are still the main traditional approaches due to their lower complexity and simplicity. There still exists recent works that benefit from them [88, 89, 90, 92, 96, 98, 100]. Recently, multiple SVMs and RFs have been proposed more commonly [85, 86, 87, 88, 89, 90, 92, 96, 98]. This allows each SVM or RF to focus on a specific feature which results in a better prediction and generalization. The most common features used for SVM and RF-based methods are *(i)* CU depth values, *(ii)* texture complexity, and *(iii)* RD cost. In Bayesian learning methods, features to update statistical properties are also chosen manually, and depth values are utilized in all of the Bayesian learning approaches since they have importance. Additionally, scene change detection is key in deciding when to update the statistics so that different approaches are chosen to decide on the correct update period, *i.e.*, online learning. Features that are used by ML methods are given in Table 7.

One downside of the traditional ML methods is that the features need to be selected manually. This requires expert-level domain knowledge since the performance of the technique heavily depends on the feature set. This is not wanted in many cases since manually crafting features is prone to errors, and it can also introduce human bias to the ML method. The main advantage of traditional ML methods is that they do not introduce significant time complexity. However, their performance is not on par with their deep learning based counterparts. Giving all the available data to the ML method and allowing it to extract useful features from the data usually results in better performance, and deep learning methods are an excellent example of that. In almost all applications, deep learning methods give better performance compared to the traditional ML methods [122, 123, 124]. Also, deep learning methods are better at utilizing the excess amount of available data, which is nowadays accessible for almost all applications.

Deep learning methods, on the other hand, introduce a significant time complexity to the encoder, which is usually not desired. CNNs are used commonly due to the nature of the videos, which consist of consecutive 2D data. We can see the majority of approaches use the luminance value of the CTU as the main data for feeding the CNN and let the network extract useful features from these values. Some approaches also append additional features along with luminance values to feed the network [106, 111]. Furthermore, we can see that some methods try to solve the CTU partitioning in a top-down approach and run a CNN for each depth level [106, 108, 109, 110] while others try to predict the whole CTU partitioning in one step to minimize the time-complexity [107]. Since CTU partitioning can be modeled as MDP, several RL-based approaches are also proposed [116, 117]. We can still observe CNNs are the critical factors in RL-based methods [116].

In addition to *(i)* traditional, *(ii)* deep learning, and *(iii)* RL based methods, several different ML approaches, *i.e.*, kNN classifier, neural network, and genetic algorithm, are also used [118, 119, 120, 121]. We can categorize these methods in the traditional ML approaches, but we believe they are in between traditional and deep learning based methods and, thus, we decided to categorize them separately. There have also been some attempts that tried to combine both statistic based and ML based approaches to benefit from them at the same time [87, 110, 117].

5. Discussion and Future Work

There has been an increase in machine learning based approaches in the last years. However, statistics based approaches are still applied in the literature. The main problem with ML based approaches is the complexity introduced by the ML method resulting in simple network structures being used so far. However, recent developments in deep learning suggest that using deeper convolutional networks significantly improves the performance, specifically while working with image data since deeper networks are better at capturing hierarchical information, which is present in many situations [125, 126, 127]. Moreover, recent advancements in the deep learning field made it possible to use very complex models with significantly reduced inference time even for challenging tasks [128, 129]. Moreover, to overcome substantial data requirements of deep learning methods, few-shot or one-shot learning methods have been proposed, and their performance had improved substantially [130, 131, 132]. Furthermore, these methods are combined and used in video-related tasks, which is challenging for deep learning techniques, and resulted in a good performance despite using little data [133]. We believe a few-shot learning approach can be helpful in the CTU partitioning decisions since each video has different characteristics. The better approach will be tailoring the model for each video, which might be possible with this improved training scheme. Traditional ML methods are still helpful in cases where the task is evident. Thus, it is easy to design features, and the available data is not sufficient enough to fully utilize a deep learning method. However, we believe the main direction to follow is to preserve the performance of deep learning methods and decrease their complexity.

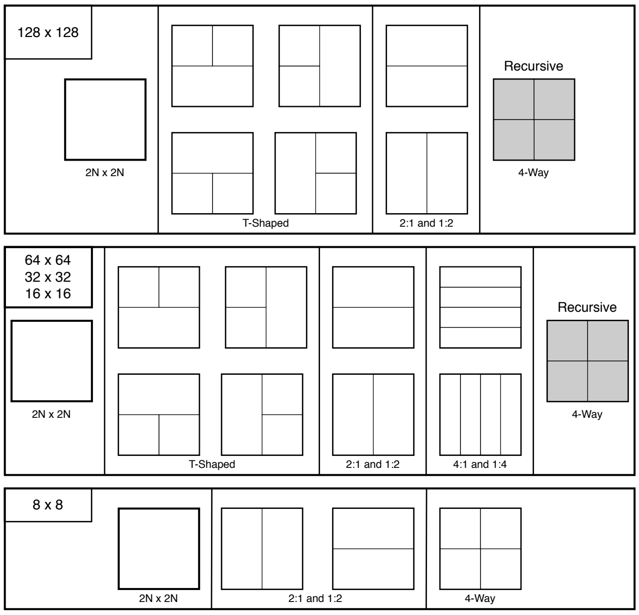

Figure 11: AV1 superblock partitioning.

On the other hand, for statistics based approaches, unexplored statistical correlations can still be found, and existing correlations can be benefited better. The main advantage of statistical approaches is that they do not introduce any noticeable time-complexity in the encoding process. One possible approach might be to use ML methods to extract statistical information beforehand to have a better understanding and move the complexity of ML out of the encoding framework.

Emerging codecs like AV1 [134] and Versatile Video Coding (VVC) [135] introduce more bitrate reduction compared to HEVC [4], while their encoding time complexity increases which makes their usage costly and challenging, especially for online applications.

AV1 and VVC introduce different block partitioning structures than HEVC. AV1 uses superblock partitioning that starts from 128×128 sized blocks. A 10-way partition-tree structure is used to sub-partition each block. For each $2N \times 2N$ block there exist *a)* one $2N \times 2N$, *b)* four "T" shaped, *c)* two 2:1 and 1:2, *d)* two 4:1 and 1:4, and *e)* one 4-way split patterns. Among aforementioned split patterns, 4-way split is the only pattern that can be recursively partitioned until it reaches the lowest 4×4 size. Moreover, there are some special cases for AV1 superblock partitioning: *a)* Two 4:1 and 1:4 splits are not available for 128×128 and 8×8 blocks, and *b)* "T" shaped split is not available for 8×8 blocks. The AV1 superblock partitioning is summarized in Fig.11.

VVC utilizes Quaternary Tree plus (Multi) binary-ternary Tree (QTMT) coding block structure with the maximum CTU size of 128×128 pixels. Both binary and ternary splits are enabled for each leaf node of the quad-tree which makes the partitioning more flexible and complex compared to HEVC. Leaf nodes of multi-type trees are called Coding Units (CUs), which are not units only for coding but for prediction and transform as well.

In VVC, each block can be split into five patterns: *a)* one Quad-Tree (QT) structure, *b)* two Binary Tree (BT) structures including horizontal binary tree (BH) and vertical binary tree (BV), and *c)* two Ternary Tree (TT) structures including horizontal ternary tree (TH) and vertical ternary tree (TV). These five structures are shown in Fig .12. It should be noted that the QT structure is not allowed for non-QT splits.

Although VVC and AV1 achieve efficient complexity reduction, there is only a tiny number of improved CTU partitioning methods available in the literature due to their recent developments. However, since both of these codecs follow a block structure architecture, algorithms introduced in this survey can be implemented in these codecs either directly or with minor modifications.

Gu and Wen [136] propose a mid-depth based block structure determination for AV1 which is similar to [60] and [137] combined.

Inspired by [71, 72, 94, 86], Chen *et al.* [138] propose an algorithm for VVC that consists of three steps. First, the variance of pixels for 32×32 CUs is calculated, and if it is lower than a pre-determined threshold, the CU is classified to be homogeneous, and further splitting is stopped. Second, absolute gradients of each pixel in horizontal and vertical directions are computed using the Sobel operator (D_x, D_y). If conditions in Eq. 29 are met for 32×32 CU, it is partitioned by QT structure and other options are skipped:

$$((1 < D_x/D_y < TH_2)\ or\ (1 < D_y/D_x < TH_2))\\ and\ ((D_x > TH_3)\ and\ (D_y > TH_3)) \tag{29}$$

Finally, if neither of the aforementioned conditions are not met, the variance of each sub-block is computed for all five types of splitting and then partitioning is continued only for the type that has the maximum variance.

6. Conclusion

In this paper, existing CTU partitioning approaches for HEVC are surveyed. The methods are categorized into two

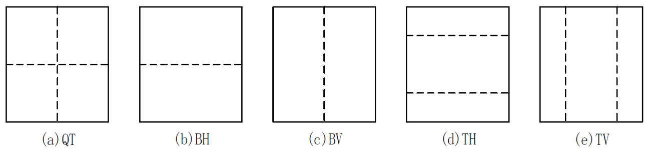

Figure 12: VVC partitioning.

major groups: *statistics* based and *machine learning* based. In statistic based approaches, the proposed methods exploit statistical similarity in the video by manually defining a set of features, thresholds, rules, *etc.* Neighboring approaches use the available information in the spatially and temporally neighboring CTUs. In contrast, inherent approaches focus on information available within the current CTU. Similar to the statistics based approaches, machine learning based approaches exploit similarity in the video but benefit from machine learning methods during the process. Traditional ML methods such as SVM and RF are used for which the features need to be chosen manually. Deep learning based methods, on the other hand, extract useful features implicitly. Recent emerging codecs, AV1 and VVC, provide improved encoding structure using new tools and improving existing ones. However, due to their recent development, there are not enough studies conducted for block partitioning of these codecs.

Acknowledgment

The financial support of the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development, and the Christian Doppler Research Association, is gratefully acknowledged. Christian Doppler Laboratory ATHENA: `https://athena.itec.aau.at/`.

References

[1] Cisco, Cisco Visual Networking Index: Forecast and Methodology, 2017–2022 (White Paper) (2019).

[2] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, R. Zimmermann, A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP, IEEE Communications Surveys & Tutorials 21 (2018) 562–585.

[3] T. Wiegand, G. J. Sullivan, G. Bjontegaard, A. Luthra, Overview of the H. 264/AVC video coding standard, IEEE Transactions on circuits and systems for video technology 13 (2003) 560–576.

[4] G. J. Sullivan, J.-R. Ohm, W.-J. Han, T. Wiegand, Overview of the high efficiency video coding (HEVC) standard, IEEE Transactions on circuits and systems for video technology 22 (2012) 1649–1668.

[5] I. ITU-T, I. JTC, Generic Coding of Moving Pictures and Associated Audio Information-part 2: Video, 1995.

[6] I. Kim, J. Min, T. Lee, W. Han, J. Park, Block Partitioning Structure in the HEVC Standard, IEEE Transactions on Circuits and Systems for Video Technology 22 (2012) 1697–1706.

[7] S. Ma, C.-C. J. Kuo, High-Definition Video Coding with Super-Macroblocks, in: VCIP, 2007.

[8] M. Wien, High Efficiency Video Coding: Coding Tools and Specification, Springer Publishing Company, Incorporated, 2014.

[9] M. H. Chan, Y. B. Yu, A. G. Constantinides, Variable Size Block Matching Motion Compensation with Applications to Video Coding,

IEE Proceedings I - Communications, Speech and Vision 137 (1990) 205–212.
[10] Jiajun Zhang, M. Omair Ahmad, M. N. S. Swamy, Quadtree structured region-wise motion compensation for video compression, IEEE Transactions on Circuits and Systems for Video Technology 9 (1999) 808–822.
[11] G. J. Sullivan, R. L. Baker, Efficient quadtree coding of images and video, IEEE Transactions on Image Processing 3 (1994) 327–331.
[12] M. Winken, P. Helle, D. Marpe, H. Schwarz, T. Wiegand, Transform codinginthe HEVC Test Model, 2011 18th IEEE International Conference on Image Processing (2011) 3693–3696.
[13] Z. Feng, P. Liu, K. Jia, K. Duan, Fast Intra CTU Depth Decision for HEVC, IEEE Access 6 (2018) 45262–45269.
[14] F. Bossen, et al., Common test conditions and software reference configurations, JCTVC-L1100 12 (2013) 7.
[15] G. Tech, Y. Chen, K. Müller, J. Ohm, A. Vetro, Y. Wang, Overview of the Multiview and 3D Extensions of High Efficiency Video Coding, IEEE Transactions on Circuits and Systems for Video Technology 26 (2016) 35–49.
[16] J. Xu, R. Joshi, R. A. Cohen, Overview of the Emerging HEVC Screen Content Coding Extension, IEEE Transactions on Circuits and Systems for Video Technology 26 (2016) 50–62.
[17] C. Müller, C. Timmerer, A Test-bed for the Dynamic Adaptive Streaming over HTTP Featuring Session Mobility, in: Proceedings of the Second Annual ACM Conference on Multimedia Systems, MMSys '11, ACM, New York, NY, USA, 2011, pp. 271–276. doi:10.1145/1943552.1943588.
[18] I. Sodagar, The MPEG-DASH Standard for Multimedia Streaming Over the Internet, IEEE MultiMedia 18 (2011) 62–67.
[19] D. Schroeder, P. Rehm, E. Steinbach, Block structure reuse for multi-rate high efficiency video coding, in: 2015 IEEE International Conference on Image Processing (ICIP), 2015, pp. 3972–3976. doi:10.1109/ICIP.2015.7351551.
[20] D. Kim, Y. Kim, W. Park, Selective CU depth range decision algorithm for HEVC encoder, in: The 18th IEEE International Symposium on Consumer Electronics (ISCE 2014), 2014, pp. 1–2. doi:10.1109/ISCE.2014.6884344.
[21] L. Shen, Z. Zhang, P. An, Fast CU size decision and mode decision algorithm for HEVC intra coding, IEEE Transactions on Consumer Electronics 59 (2013) 207–213.
[22] W. Hsu, H. Hang, Fast coding unit decision algorithm for HEVC, in: 2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, 2013, pp. 1–5. doi:10.1109/APSIPA.2013.6694353.
[23] J. Kim, Y. Choe, Y. Kim, Fast Coding Unit size decision algorithm for intra coding in HEVC, in: 2013 IEEE International Conference on Consumer Electronics (ICCE), 2013, pp. 637–638. doi:10.1109/ICCE.2013.6487050.
[24] Y.-F. Cen, W.-L. Wang, X.-W. Yao, A fast CU depth decision mechanism for HEVC, Information Processing Letters 115 (2015) 719 – 724.
[25] Z. Li, Y. Zhao, Z. Dai, K. Rogeany, Y. Cen, Z. Xiao, W. Yang, A fast CU partition method based on CU depth spatial correlation and RD cost characteristics for HEVC intra coding, Signal Processing: Image Communication 75 (2019) 141 – 146.
[26] S. Huade, L. Fan, C. Huanbang, A fast CU size decision algorithm based on adaptive depth selection for HEVC encoder, in: 2014 International Conference on Audio, Language and Image Processing, 2014, pp. 143–146. doi:10.1109/ICALIP.2014.7009774.
[27] Y. Li, G. Yang, Y. Zhu, X. Ding, X. Sun, Adaptive Inter CU Depth Decision for HEVC Using Optimal Selection Model and Encoding Parameters, IEEE Transactions on Broadcasting 63 (2017) 535–546.
[28] H. Amirpour, A. Pinheiro, M. Pereira, M. Ghanbari, Fast Depth Decision in Light Field Compression, in: 2019 Data Compression Conference (DCC), 2019, pp. 552–552. doi:10.1109/DCC.2019.00064.
[29] H. Amirpour, A. M. G. Pinheiro, M. Pereira, M. Ghanbari, Fast and Efficient Lenslet Image Compression, CoRR abs/1901.11396 (2019).
[30] Z. Pan, S. Kwong, Y. Zhang, J. Lei, H. Yuan, Fast Coding Tree Unit depth decision for high efficiency video coding, in: 2014 IEEE International Conference on Image Processing (ICIP), 2014, pp. 3214–3218. doi:10.1109/ICIP.2014.7025650.
[31] C. Zhou, F. Zhou, Y. Chen, Spatio-temporal correlation-based fast coding unit depth decision for high efficiency video coding, Journal of Electronic Imaging 22 (2013) 1 – 14.
[32] J. Leng, L. Sun, T. Ikenaga, S. Sakaida, Content based hierarchical fast coding unit decision algorithm for HEVC, in: 2011 International Conference on Multimedia and Signal Processing, volume 1, IEEE, 2011, pp. 56–59.
[33] L. Shen, Z. Liu, X. Zhang, W. Zhao, Z. Zhang, An Effective CU Size Decision Method for HEVC Encoders, IEEE Transactions on Multimedia 15 (2013) 465–470.
[34] G. Correa, P. Assuncao, L. Agostini, L. A. da Silva Cruz, Complexity control of high efficiency video encoders for power-constrained devices, IEEE Transactions on Consumer Electronics 57 (2011) 1866–1874.
[35] G. Correa, P. Assuncao, L. Agostini, L. A. D. S. Cruz, Coding Tree Depth Estimation for Complexity Reduction of HEVC, in: 2013 Data Compression Conference, 2013, pp. 43–52. doi:10.1109/DCC.2013.12.
[36] J. H. Bae, M. H. Sunwoo, Adaptive Early Termination Algorithm Using Coding Unit Depth History in HEVC, Journal of Signal Processing Systems 91 (2019) 863–873.
[37] S. J. Park, CU encoding depth prediction, early CU splitting termination and fast mode decision for fast HEVC intra-coding , Signal Processing: Image Communication 42 (2016) 79–89.
[38] Z. Liu, M. Zhang, D. Lai, C. An, A Novel Fast Partition Algorithm Based on EDGE Information in HEVC, Automatic Control and Computer Sciences 53 (2019) 471–479.
[39] L. Zhao, X. Fan, S. Ma, D. Zhao, Fast intra-encoding algorithm for high efficiency video coding, Signal Processing: Image Communication 29 (2014) 935–944.
[40] T. Silva, L. da Silva Cruz, L. Agostini, Inter-view prediction of coding tree depth for HEVC-based multiview video coding, 2013, pp. 165–168. doi:10.1109/ICECS.2013.6815380.
[41] S. N. Khan, S. Khattak, Early decision of CU splitting, using base view information, for low complexity MV-HEVC, in: 2017 International Multi-topic Conference (INMIC), 2017, pp. 1–6. doi:10.1109/INMIC.2017.8289476.
[42] P. Wang, X. Liu, B. Shao, A Fast CU Decision Algorithm for MV-HEVC, in: 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), 2015, pp. 217–221. doi:10.1109/SmartCity.2015.74.
[43] N. Zhang, D. Zhao, Y.-W. Chen, J.-L. Lin, W. Gao, Fast encoder decision for texture coding in 3D-HEVC, Signal Processing: Image Communication 29 (2014) 951 – 961.
[44] Y. Wang, Y. Wang, Y. Shi, A fast CU Size Decision Algorithm for 3D-HEVC, MATEC Web of Conferences 61 (2016) 07006.
[45] Guangsheng Chi, Xin Jin, Qionghai Dai, A quad-tree and statistics based fast CU depth decision algorithm for 3D-HEVC, in: 2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW), 2014, pp. 1–5. doi:10.1109/ICMEW.2014.6890543.
[46] D. Tian, P.-L. Lai, P. Lopez, C. Gomila, View synthesis techniques for 3D video, in: A. G. Tescher (Ed.), Applications of Digital Image Processing XXXII, volume 7443, International Society for Optics and Photonics, SPIE, 2009, pp. 233 – 243. URL: https://doi.org/10.1117/12.829372. doi:10.1117/12.829372.
[47] E. G. Mora, J. Jung, M. Cagnazzo, B. Pesquet-Popescu, Initialization, Limitation, and Predictive Coding of the Depth and Texture Quadtree in 3D-HEVC, IEEE Transactions on Circuits and Systems for Video Technology 24 (2014) 1554–1565.
[48] S. N. Khan, N. Muhammad, S. Farwa, T. Saba, S. Khattak, Z. Mahmood, Early CU Depth Decision and Reference Picture Selection for Low Complexity MV-HEVC, Symmetry 11 (2019).
[49] C.-H. Fu, H. Chen, Y.-L. Chan, S.-H. Tsang, X. Zhu, Early termination for fast intra mode decision in depth map coding using DIS-inheritance, Signal Processing: Image Communication 80 (2020) 115644.

[50] D. Schroeder, A. Ilangovan, E. Steinbach, Multi-rate encoding for HEVC-based adaptive HTTP streaming with multiple resolutions, in: 2015 IEEE 17th International Workshop on Multimedia Signal Processing (MMSP), 2015, pp. 1–6. doi:10.1109/MMSP.2015.7340822.

[51] D. Schroeder, A. Ilangovan, M. Reisslein, E. Steinbach, Efficient Multi-Rate Video Encoding for HEVC-Based Adaptive HTTP Streaming, IEEE Transactions on Circuits and Systems for Video Technology 28 (2018) 143–157.

[52] H. Amirpour, E. Çetinkaya, C. Timmerer, M. Ghanbari, Fast Multi-Rate Encoding for Adaptive HTTP Streaming, in: 2020 Data Compression Conference (DCC), IEEE, 2020, pp. 358–358.

[53] H. Amirpour, E. Çetinkaya, C. Timmerer, M. Ghanbari, Towards optimal multirate encoding for HTTP adaptive streaming, in: International Conference on Multimedia Modeling (MMM), Springer, 2021, pp. 469–480.

[54] E. Çetinkaya, H. Amirpour, C. Timmerer, M. Ghanbari, FaME-ML: Fast Multirate Encoding for HTTP Adaptive Streaming Using Machine Learning, in: 2020 IEEE International Conference on Visual Communications and Image Processing (VCIP), IEEE, 2020, pp. 87–90.

[55] X. Shen, L. Yu, J. Chen, Fast coding unit size selection for HEVC based on Bayesian decision rule, in: 2012 picture coding symposium, IEEE, 2012, pp. 453–456.

[56] J. Lee, S. Kim, K. Lim, S. Lee, A Fast CU Size Decision Algorithm for HEVC, IEEE Transactions on Circuits and Systems for Video Technology 25 (2015) 411–421.

[57] Z. Xu, B. Min, R. C. Cheung, A fast inter CU decision algorithm for HEVC, Signal Processing: Image Communication 60 (2018) 211 – 223.

[58] A. Jiménez-Moreno, E. Martínez-Enríquez, F. D. de María, Bayesian adaptive algorithm for fast coding unit decision in the High Efficiency Video Coding (HEVC) standard, Signal Processing: Image Communication 56 (2017) 1 – 11.

[59] D. Lee, J. Jeong, Fast intra coding unit decision for high efficiency video coding based on statistical information, Signal Processing: Image Communication 55 (2017) 121 – 129.

[60] J. Xiong, H. Li, Q. Wu, F. Meng, A fast HEVC inter CU selection method based on pyramid motion divergence, IEEE transactions on multimedia 16 (2013) 559–564.

[61] J. Xiong, H. Li, F. Meng, S. Zhu, Q. Wu, B. Zeng, MRF-Based Fast HEVC Inter CU Decision With the Variance of Absolute Differences, IEEE Transactions on Multimedia 16 (2014) 2141–2153.

[62] J.-C. Chiang, K.-K. Peng, C.-C. Wu, C.-Y. Deng, W.-N. Lie, Fast intra mode decision and fast CU size decision for depth video coding in 3D-HEVC, Signal Processing: Image Communication 71 (2019) 13 – 23.

[63] T. Li, H. Wang, Y. Chen, L. Yu, Fast depth intra coding based on spatial correlation and rate distortion cost in 3D-HEVC, Signal Processing: Image Communication 80 (2020) 115668.

[64] N. Kim, S. Jeon, H. J. Shim, B. Jeon, S. Lim, H. Ko, Adaptive keypoint-based CU depth decision for HEVC intra coding, in: 2016 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), 2016, pp. 1–3. doi:10.1109/BMSB.2016.7521923.

[65] T. Nishikori, T. Nakamura, T. Yoshitome, K. Mishiba, A fast CU decision using image variance in HEVC intra coding, in: 2013 IEEE Symposium on Industrial Electronics Applications, 2013, pp. 52–56. doi:10.1109/ISIEA.2013.6738966.

[66] B. Min, R. C. C. Cheung, A Fast CU Size Decision Algorithm for the HEVC Intra Encoder, IEEE Transactions on Circuits and Systems for Video Technology 25 (2015) 892–896.

[67] X. Huang, H. Jia, K. Wei, J. Liu, C. Zhu, Z. Lv, D. Xie, Fast algorithm of coding unit depth decision for HEVC intra coding, in: 2014 IEEE Visual Communications and Image Processing Conference, 2014, pp. 458–461. doi:10.1109/VCIP.2014.7051605.

[68] J. Xiong, H. Li, F. Meng, Q. Wu, K. N. Ngan, Fast HEVC Inter CU Decision Based on Latent SAD Estimation, IEEE Transactions on Multimedia 17 (2015) 2147–2159.

[69] Y. Song, Y. Zeng, X. Li, B. Cai, G. Yang, Fast CU size decision and mode decision algorithm for intra prediction in HEVC, Multimedia Tools and Applications 76 (2017) 2001–2017.

[70] J. Liu, H. Jia, G. Xiang, X. Huang, B. Cai, C. Zhu, D. Xie, An adaptive inter CU depth decision algorithm for HEVC, in: 2015 Visual Communications and Image Processing (VCIP), 2015, pp. 1–4. doi:10.1109/VCIP.2015.7457873.

[71] M. Ramezanpour, F. Zargari, Early termination algorithm for CU size decision in HEVC intra coding, in: 2015 9th Iranian Conference on Machine Vision and Image Processing (MVIP), 2015, pp. 45–48.

[72] X. Shang, G. Wang, T. Fan, Y. Li, Fast CU size decision and PU mode decision algorithm in HEVC intra coding, in: 2015 IEEE International Conference on Image Processing (ICIP), 2015, pp. 1593–1597. doi:10.1109/ICIP.2015.7351069.

[73] N. Shan, W. Zhou, Z. Duan, H. Wei, A Fast Coding Unit Depth Decision Algorithm for HEVC Inter Prediction, in: 2015 Ninth International Conference on Frontier of Computer Science and Technology, 2015, pp. 316–320. doi:10.1109/FCST.2015.32.

[74] D. G. Fernández, A. A. Del Barrio, G. Botella, C. García, Fast CU size decision based on temporal homogeneity detection, in: 2016 Conference on Design of Circuits and Integrated Systems (DCIS), 2016, pp. 1–6. doi:10.1109/DCIS.2016.7845379.

[75] M. Zhang, D. Lai, Z. Liu, C. An, A novel adaptive fast partition algorithm based on CU complexity analysis in HEVC, Multimedia Tools and Applications 78 (2019) 1035–1051.

[76] Jiangpeng Hou, Dongmei Li, Zhaohui Li, Xiuhua Jiang, Fast CU size decision based on texture complexity for HEVC intra coding, in: Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC), 2013, pp. 1096–1099. doi:10.1109/MEC.2013.6885226.

[77] G. Cebrián-Márquez, J. L. Martínez, P. Cuenca, Adaptive inter CU partitioning based on a look-ahead stage for HEVC, Signal Processing: Image Communication 76 (2019) 97 – 108.

[78] Y. Lu, H. Liu, Y. Lin, L. Shen, H. Yin, Efficient coding mode and partition decision for screen content intra coding, Signal Processing: Image Communication 68 (2018) 249–257.

[79] M. Jamali, S. Coulombe, Coding unit splitting early termination for fast HEVC intra coding based on global and directional gradients, in: 2016 IEEE 18th International Workshop on Multimedia Signal Processing (MMSP), 2016, pp. 1–5. doi:10.1109/MMSP.2016.7813356.

[80] X. Sun, X. Chen, Y. Xu, Y. Xiao, Y. Wang, D. Yu, Fast CU size and prediction mode decision algorithm for HEVC based on direction variance, Journal of Real-Time Image Processing 16 (2019) 1731–1744.

[81] M. A. Hearst, Support Vector Machines, IEEE Intelligent Systems 13 (1998) 18–28.

[82] F. Mu, L. Song, X. Yang, Z. Luo, Fast coding unit depth decision for HEVC, in: 2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW), 2014, pp. 1–6. doi:10.1109/ICMEW.2014.6890647.

[83] X. Shen, L. Yu, CU splitting early termination based on weighted SVM, EURASIP journal on image and video processing 2013 (2013) 4.

[84] Y. Zhang, S. Kwong, X. Wang, H. Yuan, Z. Pan, L. Xu, Machine Learning-Based Coding Unit Depth Decisions for Flexible Complexity Allocation in High Efficiency Video Coding, IEEE Transactions on Image Processing 24 (2015) 2225–2238.

[85] D. Liu, X. Liu, Y. Li, Fast CU Size Decisions for HEVC Intra Frame Coding Based on Support Vector Machines, in: 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), 2016, pp. 594–597. doi:10.1109/DASC-PICom-DataCom-CyberSciTec.2016.168.

[86] T. Zhang, M. Sun, D. Zhao, W. Gao, Fast Intra-Mode and CU Size Decision for HEVC, IEEE Transactions on Circuits and Systems for Video Technology 27 (2017) 1714–1726.

[87] Y. Liu, Z. Chen, J. Fang, P. Chang, SVM-Based Fast Intra CU Depth Decision for HEVC, in: 2015 Data Compression Conference, 2015, pp. 458–458. doi:10.1109/DCC.2015.32.

[88] L. Zhu, Y. Zhang, S. Kwong, X. Wang, T. Zhao, Fuzzy SVM-Based Coding Unit Decision in HEVC, IEEE Transactions on Broadcasting 64 (2018) 681–694.

[89] Y. Zhang, Z. Pan, N. Li, X. Wang, G. Jiang, S. Kwong, Effective Data Driven Coding Unit Size Decision Approaches for HEVC INTRA Coding, IEEE Transactions on Circuits and Systems for Video Technology 28 (2018) 3208–3222.

[90] B. Erabadda, T. Mallikarachchi, G. Kulupana, A. Fernando, Content Adaptive Fast CU Size Selection for HEVC Intra-Prediction, in: 2019 IEEE International Conference on Consumer Electronics (ICCE), 2019, pp. 1–2. doi:10.1109/ICCE.2019.8662119.

[91] B. Erabadda, T. Mallikarachchi, G. Kulupana, A. Fernando, Fast CU Size Decisions for HEVC Inter-Prediction Using Support Vector Machines, in: 2019 27th European Signal Processing Conference (EUSIPCO), IEEE, 2019, pp. 1–5.

[92] Y. Xue, X. Wang, L. Zhu, Z. Pan, S. Kwong, Fast Coding Unit Decision for Intra Screen Content Coding Based on Ensemble Learning, in: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 8543–8547. doi:10.1109/ICASSP.2019.8682707.

[93] L. Breiman, Random Forests, Mach. Learn. 45 (2001) 5–32.

[94] F. Duanmu, Z. Ma, Y. Wang, Fast Mode and Partition Decision Using Machine Learning for Intra-Frame Coding in HEVC Screen Content Coding Extension, IEEE Journal on Emerging and Selected Topics in Circuits and Systems 6 (2016) 517–531.

[95] D. Ruiz-Coll, V. Adzic, G. Fernández-Escribano, H. Kalva, J. L. Martínez, P. Cuenca, Fast partitioning algorithm for HEVC Intra frame coding using machine learning, in: 2014 IEEE International Conference on Image Processing (ICIP), 2014, pp. 4112–4116. doi:10.1109/ICIP.2014.7025835.

[96] H. Yang, L. Shen, P. An, Efficient screen content intra coding based on statistical learning, Signal Processing: Image Communication 62 (2018) 74 – 81.

[97] B. Du, W. Siu, X. Yang, Fast CU partition strategy for HEVC intra-frame coding using learning approach via random forests, in: 2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2015, pp. 1085–1090. doi:10.1109/APSIPA.2015.7415439.

[98] M. Tahir, I. A. Taj, P. A. Assuncao, M. Asif, Fast video encoding based on random forests, Journal of Real-Time Image Processing (2019).

[99] J. De Praeter, A. J. Díaz-Honrubia, N. Van Kets, G. Van Wallendael, J. De Cock, P. Lambert, R. Van de Walle, Fast simultaneous video encoder for adaptive streaming, in: 2015 IEEE 17th International Workshop on Multimedia Signal Processing (MMSP), 2015, pp. 1–6. doi:10.1109/MMSP.2015.7340802.

[100] T. Bubolz, M. Grellert, B. Zatt, G. Correa, Coding Tree Early Termination for Fast HEVC Transrating Based on Random Forests, in: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 1802–1806. doi:10.1109/ICASSP.2019.8683833.

[101] H. Kim, R. Park, Fast CU Partitioning Algorithm for HEVC Using an Online-Learning-Based Bayesian Decision Rule, Journal of Visual Communication and Image Representation 26 (2016) 130–138.

[102] Y. Yao, X. Yang, T. Jia, X. Jiang, W. Feng, Fast Bayesian decision based block partitioning algorithm for HEVC, Multimedia Tools and Applications 78 (2019) 9129–9147.

[103] J. Chen, L. Yu, Effective HEVC intra coding unit size decision based on online progressive Bayesian classification, in: 2016 IEEE International Conference on Multimedia and Expo (ICME), 2016, pp. 1–6. doi:10.1109/ICME.2016.7552970.

[104] W. Kuang, Y. Chan, S. Tsang, W. Siu, Online-Learning-Based Bayesian Decision Rule for Fast Intra Mode and CU Partitioning Algorithm in HEVC Screen Content Coding, IEEE Transactions on Image Processing 29 (2020) 170–185.

[105] G. F. Montufar, R. Pascanu, K. Cho, Y. Bengio, On the number of linear regions of deep neural networks, in: Advances in neural information processing systems, 2014, pp. 2924–2932.

[106] K. Kim, W. W. Ro, Fast CU Depth Decision for HEVC Using Neural Networks, IEEE Transactions on Circuits and Systems for Video Technology 29 (2018) 1462–1473.

[107] M. Xu, T. Li, Z. Wang, X. Deng, R. Yang, Z. Guan, Reducing complexity of HEVC: A deep learning approach, IEEE Transactions on Image Processing 27 (2018) 5044–5059.

[108] T. Li, M. Xu, X. Deng, A deep convolutional neural network approach for complexity reduction on intra-mode HEVC, in: 2017 IEEE International Conference on Multimedia and Expo (ICME), 2017, pp. 1255–1260. doi:10.1109/ICME.2017.8019316.

[109] J. Shi, C. Gao, Z. Chen, Asymmetric-Kernel CNN Based Fast CTU Partition for HEVC Intra Coding, in: 2019 IEEE International Symposium on Circuits and Systems (ISCAS), 2019, pp. 1–5. doi:10.1109/ISCAS.2019.8702494.

[110] Y. Zhang, G. Wang, R. Tian, M. Xu, C. C. J. Kuo, Texture-Classification Accelerated CNN Scheme for Fast Intra CU Partition in HEVC, in: 2019 Data Compression Conference (DCC), 2019, pp. 241–249. doi:10.1109/DCC.2019.00032.

[111] K. CHEN, X. ZENG, Y. FAN, CNN Oriented Fast CU Partition Decision and PU Mode Decision for HEVC Intra Encoding, in: 2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), 2018, pp. 1–3. doi:10.1109/ICSICT.2018.8564981.

[112] S. Kuanar, K. Rao, M. Bilas, J. Bredow, Adaptive CU mode selection in HEVC intra prediction: a deep learning approach, Circuits, Systems, and Signal Processing 38 (2019) 5081–5102.

[113] S. Bouaafia, R. Khemiri, F. E. Sayadi, M. Atri, Fast CU partition-based machine learning approach for reducing HEVC complexity, Journal of Real-Time Image Processing (2019) 1–12.

[114] Z. Liu, X. Yu, Y. Gao, S. Chen, X. Ji, D. Wang, CU Partition Mode Decision for HEVC Hardwired Intra Encoder Using Convolution Neural Network, IEEE Transactions on Image Processing 25 (2016) 5088–5103.

[115] Y. Li, Z. Liu, X. Ji, D. Wang, CNN Based CU Partition Mode Decision Algorithm for HEVC Inter Coding, in: 2018 25th IEEE International Conference on Image Processing (ICIP), 2018, pp. 993–997. doi:10.1109/ICIP.2018.8451290.

[116] C. Chung, W. Peng, J. Hu, HEVC/H.265 coding unit split decision using deep reinforcement learning, in: 2017 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), 2017, pp. 570–575. doi:10.1109/ISPACS.2017.8266543.

[117] N. Li, Y. Zhang, L. Zhu, W. Luo, S. Kwong, Reinforcement learning based coding unit early termination algorithm for high efficiency video coding, Journal of Visual Communication and Image Representation 60 (2019) 276 – 286.

[118] C. Huang, Z. Peng, F. Chen, Q. Jiang, G. Jiang, Q. Hu, Efficient CU and PU Decision Based on Neural Network and Gray Level Co-Occurrence Matrix for Intra Prediction of Screen Content Coding, IEEE Access 6 (2018) 46643–46655.

[119] F. Duanmu, Z. Ma, Y. Wang, Fast CU partition decision using machine learning for screen content compression, in: 2015 IEEE International Conference on Image Processing (ICIP), 2015, pp. 4972–4976. doi:10.1109/ICIP.2015.7351753.

[120] E. E. Tun, S. Aramvith, Y. Miyanaga, Fast Coding Unit Encoding Scheme for HEVC Using Genetic Algorithm, IEEE Access 7 (2019) 68010–68021.

[121] D. Lee, J. Jeong, Fast CU size decision algorithm using machine learning for HEVC intra coding, Signal Processing: Image Communication 62 (2018) 33 – 41.

[122] K. Zhang, W. Zuo, Y. Chen, D. Meng, L. Zhang, Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising, IEEE Transactions on Image Processing 26 (2017) 3142–3155.

[123] Z.-Q. Zhao, P. Zheng, S.-t. Xu, X. Wu, Object detection with deep learning: A review, IEEE transactions on neural networks and learning systems (2019).

[124] L. Wang, Y. Qiao, X. Tang, Action recognition with trajectory-pooled deep-convolutional descriptors, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 4305–4314.

[125] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

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

[127] R. Eldan, O. Shamir, The power of depth for feedforward neural networks, in: Conference on learning theory, 2016, pp. 907–940.

[128] J. Redmon, A. Farhadi, YOLO9000: better, faster, stronger, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 7263–7271.

[129] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, arXiv preprint arXiv:1804.02767 (2018).

[130] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 1126–1135.

[131] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, T. M. Hospedales, Learning to compare: Relation network for few-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1199–1208.

[132] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al., Matching networks for one shot learning, in: Advances in neural information processing systems, 2016, pp. 3630–3638.

[133] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, L. Van Gool, One-shot video object segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 221–230.

[134] Y. Chen, D. Murherjee, J. Han, A. Grange, Y. Xu, Z. Liu, S. Parker, C. Chen, H. Su, U. Joshi, C. Chiang, Y. Wang, P. Wilkins, J. Bankoski, L. Trudeau, N. Egge, J. Valin, T. Davies, S. Midtskogen, A. Norkin, P. de Rivaz, An Overview of Core Coding Tools in the AV1 Video Codec, in: 2018 Picture Coding Symposium (PCS), 2018, pp. 41–45. doi:10.1109/PCS.2018.8456249.

[135] G. Sullivan, J. Ohm, Versatile Video Coding–towards the Next Generation of Video Compression, in: 2018 Picture Coding Symposium (PCS), 2018.

[136] J. Gu, J. Wen, Mid-depth Based Block Structure Determination for AV1, in: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 1617–1621. doi:10.1109/ICASSP.2019.8682955.

[137] J. Gu, M. Tang, J. Wen, Y. Han, Adaptive Intra Candidate Selection With Early Depth Decision for Fast Intra Prediction in HEVC, IEEE Signal Processing Letters 25 (2018) 159–163.

[138] J. Chen, H. Sun, J. Katto, X. Zeng, Y. Fan, Fast QTMT Partition Decision Algorithm in VVC Intra Coding based on Variance and Gradient, in: 2019 IEEE Visual Communications and Image Processing (VCIP), 2019, pp. 1–4. doi:10.1109/VCIP47243.2019.8965674.