electronics

MDPI

Article

Fast Coding Unit Partitioning Algorithm for Video Coding Standard Based on Block Segmentation and Block Connection Structure and CNN

Nana Li, Zhenyi Wang and Qiuwen Zhang *

College of Computer Science and Technology, Zhengzhou University of Light Industry, Zhengzhou 450002, China; linana@zzuli.edu.cn (N.L.); 332207050710@zzuli.edu.cn (Z.W.)
* Correspondence: 2012032@zzuli.edu.cn; Tel.: +86-371-8660-9559

Abstract: The recently introduced Video Coding Standard, VVC, presents a novel Quadtree plus Nested Multi-Type Tree (QTMTT) block structure. This structure enables a more flexible block partition and demonstrates enhanced compression performance compared to its predecessor, HEVC. However, The introduction of the new structure has led to a more complex partition search process, resulting in a considerable increase in time complexity. The QTMTT structure yields diverse Coding Unit (CU) block sizes, posing challenges for CNN model inference. In this study, we propose a representation structure termed Block Segmentation and Block Connection (BSC), rooted in texture features. This ensures that partial CU blocks are uniformly represented in size. To address different-sized CUs, various levels of CNN models are designed for prediction. Moreover, we introduce a post-processing method and a multi-thresholding scheme to further mitigate errors introduced by CNNs. This allows for flexible and adjustable acceleration, achieving a trade-off between coding time complexity and performance. Experimental results indicate that, in comparison to VTM-10.0, our "Fast" scheme reduces the average complexity by 57.14% with a 1.86% increase in BDBR. Meanwhile, the "Moderate" scheme reduces average complexity by 50.14% with only a 1.39% increase in BDBR.

Keywords: VVC; CNN; BSC; coding unit partition

Citation: Li, N.; Wang, Z.; Zhang, Q. Fast Coding Unit Partitioning Algorithm for Video Coding Standard Based on Block Segmentation and Block Connection Structure and CNN. *Electronics* **2024**, *13*, 1767. https://doi.org/10.3390/electronics13091767

Academic Editors: Miroslav Uhrina, Jaroslav Frnda and Lukas Sevcik

Received: 3 April 2024
Revised: 23 April 2024
Accepted: 28 April 2024
Published: 2 May 2024

1. Introduction

With the emergence of 4K/8K, 3D, VR/AR/MR, High Frame Rate (HFR), High Dynamic Range (HDR), and other Ultra-High-Definition (UHD) videos, there are increasingly stringent requirements for the capability of video compression, and the need for more efficient video coding standards is becoming more and more urgent. To satisfy the requirement for ultra-high definition video compression, the Joint Video Experts Team (JVET) group, jointly established by Moving Picture Expert Group (MPEG) and Video Coding Experts Group (VCEG), designed the Versatile Video Coding (VVC), which was finalized in July 2020 as the next-generation international video coding standard [1]. VVC inherits the basic framework of previous HEVC and introduces new techniques such as QTMTT [2], Intra Sub-Partitions (ISP), Multiple reference line (MRL), Local Illumination Compensation (LIC), Bi-directional Optical flow (BIO), Affine motion compensated prediction (AMC) [3], and Adaptive Loop Filtering (ALF), etc. These new techniques enable VVC to have a more powerful coding performance compared to the previous high-efficiency video coding (HEVC), but at the same time, they also bring about a huge overhead in coding time complexity. Among them, the introduction of QTMTT technology is the main part that leads to the increase in coding time complexity overhead [4,5]. In HEVC, only the quadtree (QT) partitioning structure is allowed to be used to partition each coding tree unit (CTU), while QTMTT supports the binary-tree (BT) and ternary-tree (TT) partitioning structures in addition to the QT partitioning. This flexibility allows for adaptation to different image features, making the partitioning structure more flexible and bringing more efficient coding

efficiency. However, the introduction of BT and TT leads to a more complex partition search process, resulting in significant time complexity overhead. Previous studies have shown that the time complexity overhead of partition search in VVC is above 90% [6]. Therefore, it is necessary to reduce the time complexity of VVC and decrease the time spent in the partition search process [7].

In past research, the fast partitioning of CU partitions has been widely used to reduce time complexity, extending a large variety of fast partitioning methods. In HEVC, due to the existence of only a single QT partitioning structure, numerous studies have achieved superior performance by determining the need for partitioning through manually designed features. Yet, for VVC, due to the newly introduced BT and TT partition structures, which increase the number of partitioning modes to six, including quadtree, horizontal binary-tree (BTH), vertical binary-tree (BTV), horizontal ternary-tree (TTH), vertical ternary-tree (TTV), and non-split (NS), CU partitioning has become much more flexible, making it difficult to use previous algorithms for straightforwardly reducing complexity overheads. In recent studies, attempts have been made to address this challenge through statistical analysis or machine learning methods. The fundamental idea behind these approaches is to skip modes and terminate early.

The new QTMTT structure as shown in Figure 1a, allows CUs to perform five partitions and greatly enhances the flexibility of CU blocks. Compared with HEVC, to improve the coding efficiency and adapt to higher resolution video content, VVC increases the default size of the CTU to 128 × 128, and the minimum CU size is specified to be 4 × 4. During the CU partitioning process of VVC, the CTU is always quadtree partitioned since the maximum CU is 64 × 64 by default. Furthermore, VVC specifies that only quadtree partition is allowed until the CU size is 32 × 32. Five partitions model are allowed for CUs of size 32 × 32 and its sub-CUs, which results in 15 different sizes of CU blocks as shown in Figure 1b. This partitioning structure greatly increases the flexibility of CU blocks, and the different sizes of CU blocks can be better adapted to different texture features. The process of CU partitioning is very violent, in which VVC checks the RD cost of all the partitioning modes for each CU as well as its sub-CUs, and selects the partitioning combinations with the optimal RD cost. In HEVC, up to 21 combinations need to be checked for each CU of 32 × 32 size, but for VVC, 361 combinations need to be checked. Therefore, the QTMTT structure in VVC significantly increases the overall coding complexity.

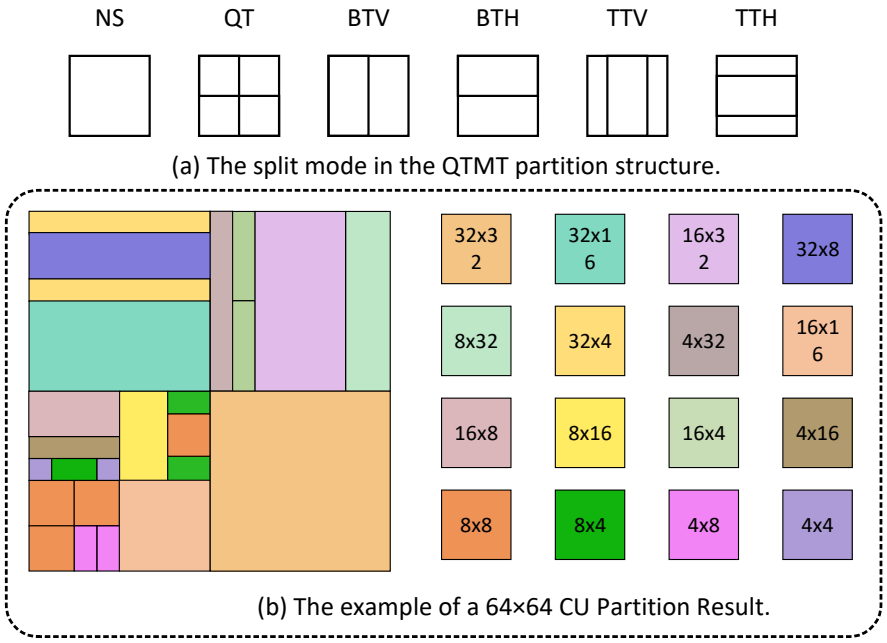

Figure 1. QTMTT structure in VVC.

Due to the complexity of the QTMTT structure, there is a wide variety of CU blocks of different sizes, which makes it difficult to perform unified inference with models. In this paper, we propose a new BSC structure that represents some of the blocks in a QTMTT as the same size and then uses different levels of CNN models to directly predict each block partitioning mode. In the CNN model, we introduced symmetric and asymmetric convolutional kernels for extracting texture features of different dimensions and added hand-crafted features in the fully connected layer to make the model more compatible with the BSC structure. After obtaining the output of the model, we use a multi-thresholding scheme to decide on the final segmentation mode, achieving tunability in terms of coding time complexity and performance.

The main contributions of this paper are as follows:

- Block segmentation and block connection structure: we design a new representation structure for partial CUs based on texture features to represent partial CU blocks as the same size.
- Different levels of CNN models: we designed different levels of CNN models to predict the partitioning mode of CU blocks, introduced asymmetric convolutional kernels for extracting different features in the model, and also introduced some external features, which proved the effectiveness.
- Multi-thresholding design: we propose a multi-thresholding scheme that sets different thresholding schemes for different levels of characteristics, realizing a trade-off between coding time complexity and coding performance.

The rest of the paper is organized as follows. Section 2 summarizes the background and related work. Section 3 explores the details of the overall algorithm. Section 4 presents the experimental results. Section 5 describes the conclusions.

2. Related Works

In previous work, a large amount of research has been spent on fast CU partitioning with the expectation of reducing the time complexity of CU partitioning. We categorize the previous methods into two groups: statistical analysis-based methods and machine learning-based methods. In this section, we summarize and review the previous research results.

2.1. Methods Based on Statistical Analysis

Methods based on statistical analysis attempt to utilize the data or features generated during the coding process to determine the final classification result. In [8], a bi-directional depth search method is proposed using previously encoded CUs and predicted mode costs. In [9], an adaptive early termination algorithm based on coding unit depth history is proposed to track CU depth history based on the CTU temporal correlation to determine the depth range of the target CUs to terminate the segmentation early. In [10], depth difference and RD loss ratio are utilized to model and perform split and early termination decisions for CUs. In [11], the correlation with the distribution of the distortion cost of neighboring block rates under different quantization parameters is analyzed, and a threshold is set to terminate the partition early based on the correlation. In [12], a methodology is introduced for Coding Unit (CU) segmentation based on the keypoint-based CU depth decision (KCD). Meanwhile, in [13], a novel fast CU segmentation decision approach is proposed, leveraging SAO edge category information as spatiotemporal encoding parameters.

In [14], a methodology is presented where partition modes are determined based on the gradient features and variance of the partitions. In [15], a methodology is introduced that utilizes the sum of the mean absolute deviation (SMAD) to quantitatively measure the vertical and horizontal texture complexity. In [16], the features of the current block and coding context are explored based on the selected intra-prediction mode. This exploration aims to skip unnecessary partitioning computations. In [17], the encoding results of BT partitions are incorporated as features in the decision process for TT partitioning. This inclusion is aimed at reducing the time complexity of TT partitioning. In [18], an efficient

algorithm is proposed for the selection of the partitioning direction of the current CU. This algorithm utilizes entropy and texture contrast as effective features to discriminate the optimal partitioning direction for the current CU. In [19], depth information from temporally and spatially adjacent blocks is extracted to predict the optimal depth of CUs. This enables the early termination of partitioning, reducing unnecessary time expenditures. In [20], explicit VVC features (EVFs) and derived VVC features (DVFs) are manually designed based on the correlation with the QTMTT structure. These features are utilized to facilitate the early termination of the nested TT block structure after QT partitioning. In [21], the distortion of CUs is obtained by calculating the difference between the original luminance pixels and the predicted luminance pixels. This feature is utilized to establish an early skip decision model for both BT and TT partitioning for each CU.

However, these algorithms heavily rely on manually designed features and heuristic rules to make decisions for video coding. Although these methods still perform well, manually designed features are usually difficult to capture complex nonlinear relationships and generalize to different situations. Moreover, summarizing features based on data analysis is not representative of all situations and only serves to show a high correlation, making it difficult for these algorithms to further improve their performance.

2.2. Methods Based on Machine Learning

In recent years, machine learning has been widely used in a large number of studies, showing excellent performance, and is rapidly being cited in the field of fast video coding. Machine learning models can automatically learn the features from large amounts of data and can represent complex nonlinear features well. In [22–24], each Coding Tree Unit (CTU) is partitioned into blocks of size 32 × 32. A Convolutional Neural Network (CNN) is employed to predict the depth range of 32 × 32 Coding Unit (CU) blocks, facilitating the premature termination of unnecessary rate-distortion optimization (RDO) computations. However, this premature termination strategy may yield counterproductive results in cases of high texture complexity, as there is no need to skip RDO calculations in such scenarios. In [25,26], a partition map is employed to represent the block partitioning structure based on QTMTT. In these works, convolutional neural network (CNN) models are constructed to predict the optimal partition map based on the original pixel values. However, the prediction of partition maps relies on intricate sub-networks, leading to suboptimal computational efficiency and posing challenges for hardware implementation. Moreover, the complexity of the partition maps may contribute to inaccuracies in the prediction process. In [27,28], a methodology is implemented wherein each 64 × 64 block is subdivided into multiple 4 × 4-sized sub-blocks. CNN is leveraged to deduce, for each 4 × 4 sub-block, a probability vector denoting the likelihood of its borders serving as partition boundaries. This information is harnessed to formulate the overall partitioning structure for the 64 × 64 block.

In [29], a methodology is introduced wherein 32 × 32 blocks are stratified based on their side lengths. The proposed Hierarchical Grid Fully Convolutional Network (HG-FCN) is employed to predict probability vectors denoting the likelihood of side lengths serving as boundaries within 32 × 32 blocks across different hierarchical levels. In [30], a partition homogeneity map (PHM) is introduced, and a Fully Convolutional Network (FCN) is employed to infer the final results. However, fundamentally, this approach aims to predict the probability of 7 × 7 blocks serving as boundaries. The inherent bottom–up predictive structure may introduce redundant computations and compromise the accuracy of predictions.

In [31], an approach is introduced, employing asymmetric convolutional kernels for the prediction of partition modes. Similarly, in [32], CNN is utilized to directly predict the partition modes of CUs. In [33,34], an adaptive pooling-variable CNN is proposed to predict the partitioning of CUs of varying sizes. However, the pooling process inevitably introduces the loss of certain features.

In the realm of current video coding research, efficient CU partitioning strategies are crucial for enhancing the encoding efficiency and reducing computational complexity. Although various deep learning-based approaches have been proposed to predict the optimal partitioning modes of CUs, these methods typically confront the challenge of utilizing models to replace partition search. Addressing this issue, this paper introduces an innovative BSC structure capable of representing CUs of varying sizes as model inputs of the same dimension. With this structure, we can directly predict a wider range of CU partitioning modes using a CNN model. Moreover, our approach not only maintains high accuracy but also incorporates a threshold scheme to flexibly adjust the acceleration efficiency, thereby achieving a superior balance between encoding efficiency and computational complexity. Our "Fast" scheme reduces the average complexity by 57.14% and increases the BDBR by 1.86%, while the "Moderate" scheme reduces the average complexity by 50.14% and increases the BDBR by only 1.39%.

3. The Proposed Method

Our proposed method uses a CNN model to replace part of the time-consuming partition search process. Figure 2 shows the flowchart of the overall algorithm. The BSC structure and CNN model are used to determine the partitioning modes for most of the CUs, and the encoder partitioning process can be guided by the model prediction results. The overall architecture of the method contains BSC mapping, segmentation mode prediction, post-processing, and CU encoding. We designed the 64 × 64 CNN Model, 32 × 32 CNN Model, and 16 × 16 CNN Model for the different sizes of CUs to predict the partitioning modes of different sizes of CUs. To enable more CUs to be predicted by the 16 × 16 CNN Model, we propose BSC structures to give more blocks of 16 × 16 mapping. Considering that the prediction against BSC structures may not be standardized, we design a post-processing algorithm to solve this problem. In addition, to increase the accuracy of the model prediction to reduce the error, we design a multi-thresholding scheme to further increase the stability of the model. After obtaining the prediction results, the encoder can directly skip this level of CU partition. Due to the multi-thresholding scheme and the multi-level network structure, the encoder can arbitrarily choose whether to use the network at this level, as well as arbitrarily adjust the threshold at a certain level, thus realizing flexible acceleration settings.

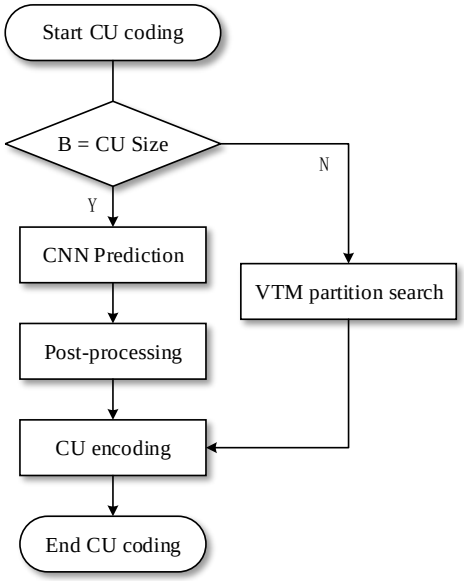

Figure 2. Flowchart of the overall algorithm (B = CU size represents one of the CU sizes: 64 × 64, 32 × 32, 16 × 16, 32 × 16, 16 × 32, 32 × 8, 8 × 32).

3.1. Block Segmentation and Block Connection Structure

VVC takes a lot of time for the partition search process. Therefore, streamlining this process is crucial for encoder efficiency. However, the size of CUs in the same layer may be different and the size of CUs in different layers may be the same, and this irregularity in size makes it difficult to make direct predictions. To make direct predictions of CUs, finding a single representation that unifies the size of CUs is the first problem that needs to be solved. As shown in Figure 3, we have quantified the blocks in the dataset that correlate with the BSC structure. It is clear that the BSC structure enables the 16 × 16 CNN model to process an additional 40% of CU blocks, which is highly valuable in terms of reducing the encoding time.

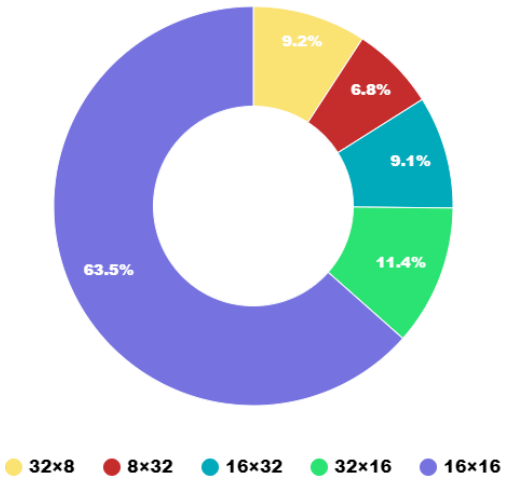

Figure 3. Block segmentation and block connection structure data statistics.

We propose a new BSC structure based on the correlation of image texture features, expecting to represent the partial CUs of the same size. As shown in Figure 4a, for CUs of 16 × 32 and 32 × 16 sizes, we segmented it into two 16 × 16 sized images from the middle to represent this CU. For the CUs of 8 × 32 and 32 × 8 sizes, we first split it into two 8 × 16 and 16 × 8 images from the center and then concatenated the left (bottom) half of the image with the right (top) half of the image to form a new 16 × 16 sized image. We try to use this BSC structure to unify four different sizes of CU blocks into the same 16 × 16 and design models to reason about them in a unified way.

Specifically, for the way it may be partitioned, we map the texture features of the original CU image as well as the partitioning mode to the newly composed image of 16 × 16 size. As shown in Figure 4b, in the block segmentation structure (taking 16 × 32 as an example), this mode of NS indicates that the original CU image texture is not complex and does not need to be partitioned, and we consider that neither of the texture features of the segmentation two images is complex, corresponding to the NS partition modes. For the BTV and TTV partition modes, there is a difference between the texture features of the left and right parts of the original image, and we believe that the two images after segmentation still retain this difference, so they still correspond to the original partitioning case. However, for the TTH partition mode, this partition mode indicates that the original CU image exhibits three different levels of texture features in the vertical direction, and when the original CU is cut from the middle, the three original different levels of features will exhibit two different levels of features in each block after the cut, corresponding to the BTH partition model. It is worth saying that, in the BSC structure, there is no design for the BTH partition case, when cut from the middle, the two newly cut images will represent the information of next-level partitioning, which cannot be extracted from the current level, which is a situation that we will solve by the original encoder in post-processing.

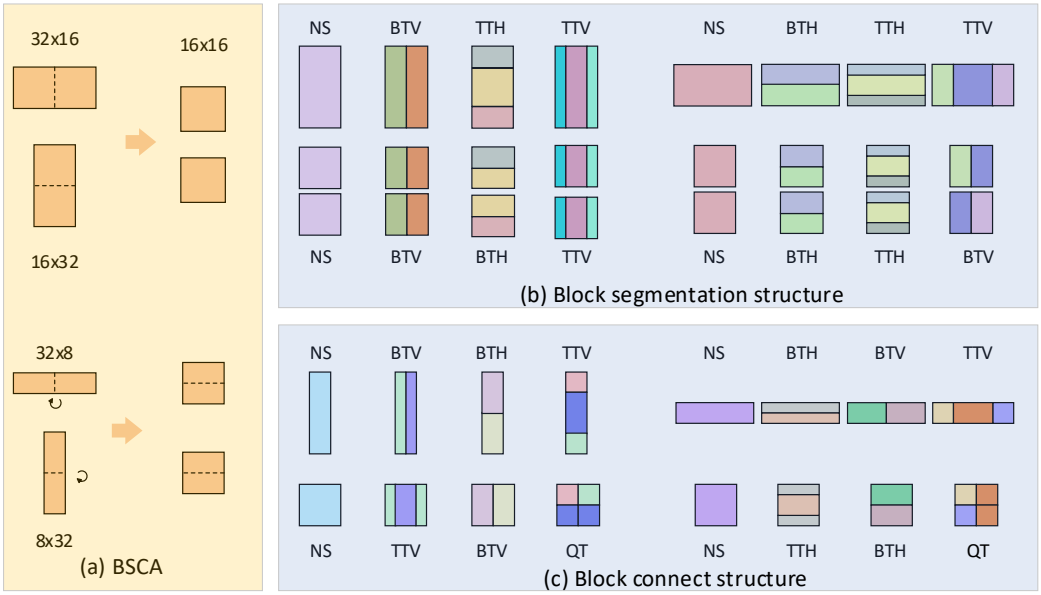

Figure 4. Block segmentation and block connection structure.

As shown in Figure 4c, in the block connection structure (taking 8 × 32 as an example), this model of NS indicates that the original CU image texture is not complex and does not need to be divided, and the spliced image corresponds to the NS partitioning model. In the model of BTV, it means that there are differences in texture information between the left and right sides of the original image, which will be represented as a three-level hierarchical architecture in the new image after the connect operation, corresponding to the TTV partitioning mode. BTH means that the upper and lower sides have different texture features, which will be transformed into different texture information between the left and right sides in the new image after the connect operation, corresponding to the BTV partitioning mode. In the TTH partitioning mode, CU exhibits three levels of texture features in the horizontal direction, and this characteristic is more consistent with the QT partitioning mode in the image after the connection. In general, BSC is a complete mapping of CU to a 16 × 16 size image; therefore, the partitioning decision of CU can be converted into a 16 × 16 size image prediction. The BSC structure is equivalent to preprocessing the original CU to facilitate the reasoning of CNN model, and will not be encoded as coding information. After CNN obtains the division mode of the CU, it will map the reasoning result to the division mode of the original CU and encode the original CU.

3.2. The Structure of CNN Models

In this paper, we use CNNs to determine the partitioning modes of CUs in the VVC internal prediction. For a block of CUs, the RD cost of all the partitioning modes for that CU and its sub-CUs is computed in VTM and the least costly partitioning case is used. We design different levels of CNNs for 64 × 64, 32 × 32, and 16 × 16 sized CUs including BSC structures to predict their partitioning modes thus replacing some of the tedious partitioning search process, which is more complex for these blocks as compared to other blocks. Our model structure is shown in Figure 5. We designed different three-model architectures for different-sized blocks.

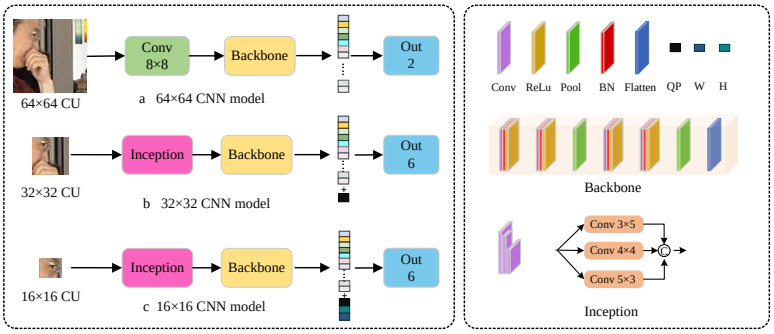

Figure 5. The structure of CNN models.

3.2.1. 64 × 64 CNN Model

The 64 × 64 CNN MODEL takes the luminance channel of the 64 × 64 CU block as input and outputs the probability of the two partition modes $\{NS, QT\}$. As shown in Figure 5a, in the context of HD video processing, we designed an 7 × 7 convolution kernel in the first layer to extract features because pixels are highly redundant in HD videos, and employing a larger convolution kernel can increase the sensory field of the next layer of convolution. This approach ensures that the extracted features are highly representative and beneficial for optimizing. To effectively reduce the complexity, the running time of the model should be as little as possible, so we did not design a deep network and complex network architecture, and only used simple convolution, pooling, and batch normalization to extract features further. The CU block of HD video does not carry much texture information, and this simple structure can extract the features of the image well without the need for a more complex model representation. After that, we use Global AvgPool and Flatten to transform the features into one-dimensional vectors, and finally, output two predicted probabilities after a fully connected layer and Softmax activation function, which represent the probabilities of the two modes of the partitioning of the 64 × 64 CU block.

3.2.2. The 32 × 32 CNN Model

Unlike the 64 × 64 CU block, the 32 × 32 CU block allows for six partitioning modes. The 32 × 32 CNN MODEL takes the luminance channel of the 32 × 32 CU block as input, but the output is the probability vector of the six partitioning modes $\{NS, QT, BTH, BTV, TTH, TTV\}$. As shown in Figure 5b, in the design of 32 × 32 CNN MODEL, instead of using the previous 7 × 7 convolution kernel as the first layer of the model, we use three different sizes of convolution kernels, 4 × 4, 5 × 3, and 3 × 5, as the three channels for extracting the features of different dimensions and connect the output with the features for input into the next layer. Due to the limitation of the pixel size of 32 × 32 CU itself, an 7 × 7 convolutional kernel is too large to easily cause the loss of some local features. We use a relatively small 4 × 4 convolutional kernel instead, and combine two asymmetric convolutional kernels, 3 × 5 and 5 × 3, to extract features in different directions for better prediction of BT, TT, and this type of segmentation. The subsequent design follows the backbone network in 64 × 64 CNN MODEL and uses Global AvgPool and Flatten to transform the features into one-dimensional vectors. Notably, in the fully connected layer of 32 × 32 CNN MODEL, we add the QPs of CUs as external features. For different QPs, the partitioning decision of the same CU may be different, which will make the CNN inference more correct. Finally, after Softmax activation function outputs six predicted probabilities, which represent the probabilities of the six partitioning modes of the 32 × 32 CU block.

3.2.3. 16 × 16 CNN Model

The 16 × 16 CNN MODEL uses the same structure as the 32 × 32 CNN MODEL. The difference is that the input of the 16 × 16 CNN MODEL does not only have the luminance channels of the original 16 × 16 CU blocks, but also contains the new 16 × 16

luminance maps formed by BSC of the 16 × 32, 32 × 16, 32 × 8, and 8 × 32 luminance channels. So, as shown in Figure 5c, as in QP, we add both the width and height information of the CUs as external features to the fully connected layer, because they are important features of this block and will have an additional impact on the partitioning decision, which will make the CNN inference more correct.

Table 1 shows the number of channels in the first convolutional layer and the number of features in the flatten layer for models at different levels. In the 64 × 64 CNN model, the first convolutional layer is designed with 16 channels using a 7 × 7 convolution, and ultimately, 64 feature maps are extracted by the backbone. In the 32 × 32 CNN model and the 16 × 16 CNN model, the first convolutional layer is designed with 16 channels using a 4 × 4 convolution, and each has eight channels using 3 × 5 and 5 × 3 convolutions, respectively. Finally, 256 and 128 feature maps are extracted by the backbone. This is to consider the impact of the number of model parameters, so not too many channels are designed to avoid an explosion in parameter quantity. Since the 32 × 32 CNN model and the 16 × 16 CNN model classify more categories, more feature maps are ultimately extracted to increase their representational ability.

Table 1. CNN models parameters.

Models	Convolutional (C×W×H)	Backbone	Flatten	Output
64 × 64 CNN	16 × 7 × 7	32 × 3 × 3 64 × 3 × 3	64 × 1	2
32 × 32 CNN	16 × 4 × 4 8 × 3 × 5 8 × 5 × 3	128 × 3 × 3 256 × 3 × 3	256 × 1	6
16 × 16 CNN	16 × 4 × 4 8 × 3 × 5 8 × 5 × 3	64 × 3 × 3 128 × 3 × 3	128 × 1	6

3.3. Dataset

The importance of datasets in training deep learning models cannot be overlooked, as they directly influence the performance and generalization capabilities of the models. We employ the Div2K [35] public image dataset for model training, which is an open dataset for super-resolution tasks, encompasses a diverse collection of images covering a broad array of scenes and content. This diversity ensures that the model performs well under various conditions. We encode the dataset using a VTM encoder in AI configuration, setting the QP values to $\{22, 27, 32, 37\}$, to extract complete CU partition information and RD costs. Based on the partition information, the partition mode and luminance channels of each CU are extracted to serve as labels and data for training. Notably, for CU blocks that conform to the BSC structure, labels and data are assigned according to the mapping depicted in Figure 4. By utilizing the Div2K dataset, we acquired over 1.78 million instances of 64 × 64 block partitions, more than 5.91 million instances of 32 × 32 block partitions, and over 12.73 million instances of 16 × 16 block partitions. Figure 6 shows the proportion of each label within the dataset, clearly indicating a significant imbalance in data distribution among different categories, regardless of the CU block size. The imbalance of data significantly affects the model's fitting. For each block size, we denote M as the number of labels with the lowest proportion. To mitigate this imbalance, we randomly select $0.8M$ from each category for the training set and $0.2M$ for the test set, thereby not only addressing the imbalance but also facilitating the identification of an optimal training set representation.

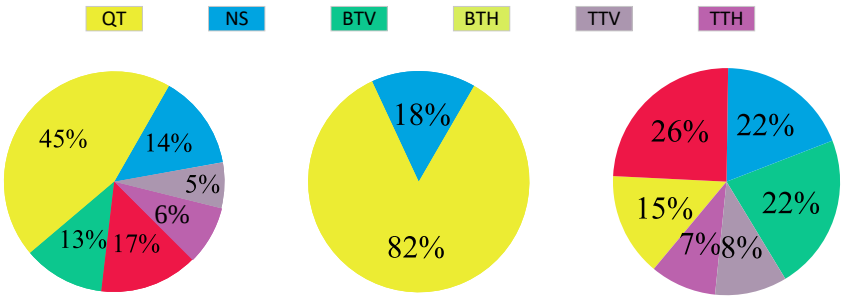

Figure 6. Distribution of datasets (32 × 32 (**left**), 64 × 64 (**center**), 16 × 16 (**right**)).

3.4. Loss

The approach in this paper treats CU mode prediction as a multistate polarization problem. In neural networks, the cross-entropy loss function is a natural choice when the task is to map the input to one of multiple categories. It is an effective measure of how the probability distribution of the model output differs from the true label. Moreover, the cross-entropy loss function is relatively simple for the computation of the gradient, which makes it easier to find the global optimal solution in the optimization process. So, we apply the basic cross entropy as the loss function with the following expression:

$$L_{CE} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) \tag{1}$$

where N is the size of the minibatch, y_i denotes the true split mode of ith CU, $\hat{y}_i$ denotes the predicted probability of ith CU.

In VVC, distinct partitioning modes result in significant variations in RD costs. Consequently, opting for different partitioning modes can entail considerably varying additional encoding expenses. This pronounced characteristic suggests that the conventional cross-entropy loss function may not perfectly capture the discrepancies between predicted and actual values. To account for this in model training, we have incorporated RD cost into the loss function to more accurately reflect the impact of different partitioning modes on encoding efficiency. The formula can be expressed as follows:

$$L_{RD} = -\frac{1}{N}\sum_{i=1}^{N} y_i\left(\frac{r_{n,m}}{r_{n,min}} - 1\right) \tag{2}$$

where $r_{n,m}$ is the RD cost of the CU at split mode m, and $r_{n,min}$ is the minimum RD cost of this CU across all possible split modes. In the above equation, $\frac{r_{n,m}}{r_{n,min}} - 1$ can be interpreted as the normalized RD cost. The term $y_i(\frac{r_{n,m}}{r_{n,min}} - 1)$ levies increased penalties on larger RD costs $r_{n,m}$, in line with the RD optimization objectives in VVC. Combining (1) and (2), the overall loss function is:

$$L = L_{RD} + \beta L_{CE} \tag{3}$$

Here, β is a positive scalar set to 1, used to adjust the relative size of the RD cost term on the cross-entropy term to ensure that both terms can be optimized efficiently. Thus, three CNN models can be correctly trained by minimizing L.

3.5. Post-Processing Operations for BSCA Structure

Our BSCA structure gives a mapping between 16 × 32, 32 × 16, 32 × 8, and 8 × 32 CU blocks to 16 × 16, and after reasoning through the model, we need to post-process the results for two main purposes:

Error reduction: In the block segmentation structure, we split a 16 × 32 and 32 × 16 block into two blocks and reason about them separately. However, the two newly segmented blocks may exhibit different texture features. In a block with a BTV partition mode, one of the segmented two blocks may exhibit a more obvious BTV partition features while

the other does not have any texture features, which are present. To avoid errors, when two blocks are predicted differently, we encode the partition modes mapped to both outcomes by adding them to the VTM. It is worth noting that we will always add the 32 × 16 BTH mode and the 16 × 32 BTV mode to the VTM, because the small blocks formed by block segmentation represent the features of the next level, and both cases cannot be mapped to the block segmentation mode.

Prescribed normalization: a trained network cannot predict every block with complete accuracy. Inevitably, there will be outliers in predicted values, which is not only a prediction error, but also violates the regularity of the partition. For example, QT partition is not allowed in the original blocks of the BSC structure. When the predicted value does not conform to the partition rules of the original block, we discard this prediction result and encode the block using VTM.

3.6. Multi-Threshold Settings

Exclusively using the prediction results of the model, deciding the partition mode of CUs can minimize the time complexity. However, the prediction results of the trained network are not completely accurate, and the presence of errors in the model can lead to the introduction of incorrect segmentation modes thus leading to a degradation of the RD performance. The error introduced by different block sizes varies, and relatively speaking, the prediction error of a block with a smaller depth brings more losses as it affects the deeper partition. Therefore, we propose a multi-thresholding scheme to realize the trade-off between coding complexity and RD performance.

3.6.1. Fixed Threshold Program

For a 64 × 64 model, errors in the prediction results may exponentially bring about a loss in RD performance compared to other, smaller CU blocks. We use a fixed threshold τ to determine the confidence level of the prediction, with τ ranging from $[0, 1]$. For prediction results, less than τ will be discarded, and the encoder performs an partition search to determine the segmentation mode, and the next layer of the prediction model is prohibited during the search process to ensure the correct decision.

3.6.2. Variable Threshold Program

For 32 × 32 and 16 × 16 models, there are multiple modes being predicted, and when using a simple fixed threshold to determine the confidence of the prediction results, setting the threshold too high many predictions will be discarded bringing limited time complexity reduction, and setting the threshold too low will check too many non-essential partition modes. So, we use a variable thresholding scheme, which is dynamically adjusted according to the probability vector of the model output. The threshold is defined as:

$$\tau = \alpha \times p_{max} \tag{4}$$

where α is a manually set fixed factor, p_{max} is the maximum value among the model outputs $p_i \in \{0, 1\} (i \in \{0, m-1\})$, and $m \in \{5, 6\}$ represents the number of different splitting modes for CUs. The threshold τ is the final value. Since the model outputs vary with different CU blocks, the threshold τ is variable. For all possible modes m of this CU, only the modes with a probability of $p_i (i \in \{0, m-1\}) \geq \alpha \times p_{max}$ are checked during the encoder's partition search, while the others are skipped.

For the most aggressive setting, $\alpha = 1$, the encoder only examines modes where the model output $p_i (i \in \{0, m-1\}) \geq p_{max}$, and the partitioning mode of the CUs is entirely determined by the model. In contrast, for $\alpha = 0$, the encoder checks modes where the model output $p_i (i \in \{0, m-1\}) \geq 0$, and the partitioning mode of the CUs is entirely determined by the original encoder's partition search. As a compromise, the parameter a is typically set between 0 and 1 in practice.

4. Experimental Results and Analyses

4.1. Experimental Configuration

CNN training configuration: We generated training datasets separately for four quantization parameters (QP) in luminance; for detailed information, please refer to reference sections. The training process was implemented using PyTorch 1.8.0 framework and Python 3.6. We optimized the parameters of CNN models using the Adam optimizer, widely adopted in deep learning models. The initial learning rate was set to 1×10^{-3}, and a smooth CosineAnnealingLR strategy was employed to adjust the learning rate, aiding the model in smoothly exploring the global optimum. Each CNN model underwent training for 50 epochs on a GeForce RTX 3060Ti. Considering the substantial dataset size and the limited information carried by individual data points, the batch size was set to 1000 to reduce sensitivity to noise and mitigate the impact of local minima.

Encoding test configuration: our approach was implemented using the VVC reference software VTM 10.0 [36]. The experiments were conducted on six classes of JVET test sequences [37], encoding under AI configurations with four quantization parameter (QP) values 22, 27, 32, 37 using the file encoder in vtm.cfg. The six classes correspond to A1 (3840 × 2160), A2 (3840 × 2160), B (1920 × 1080), C (832 × 480), D (416 × 240), and E (1280 × 720). We employed the Bjøntegaard Delta bit-rate (BD-rate/BD-BR) [38] metric for performance evaluation, measuring the RD performance difference of our method relative to the original VTM 10.0. Additionally, the time saving (TS) ratio was used to represent the percentage of time saved compared to the original VTM, providing a metric for complexity reduction. The TS is defined as:

$$TS = \frac{1}{4} \sum_{QP}^{22,27,32,37} \frac{T_{VTM}(QP) - T_{OUR}(QP)}{T_{VTM}(QP)} \tag{5}$$

4.2. Model Prediction Accuracy

This approach employs CNN to predict partition modes, and consequently, model accuracy directly influences compression performance. Figure 7 illustrates the accuracy of the models under different thresholds. For the 64 × 64 CNN model, the prediction accuracy reaches 96.8%. However, for the 32 × 32 CNN model and the 16 × 16 CNN model, their inherent prediction accuracies are only 73% and 68%, respectively. This discrepancy may be attributed to the larger number of prediction categories for the latter two models. The BSC curve in the figure shows the prediction accuracy of the 16 × 16 CNN model for the BSC structure, but its performance for the BSC structure is relatively low. This is because the BSC structure changes the original structure of the CU, which to some extent, leads to a decrease in the representation ability of the model. Setting the threshold significantly improves the model's accuracy, albeit at the cost of sacrificing some inference results. We chose two sets of different thresholds for implementing our proposed method, distinguishing them as the "Fast" scheme, which more effectively reduces the encoding complexity, and the "Moderate" scheme, which exhibits a superior RD performance. The threshold values for both schemes are presented in Table 2.

Table 2. Threshold settings for 64 × 64 CNN MODEL; α settings for 32 × 32 and 16 × 16 CNN MODEL.

Scheme	64 × 64	32 × 32	16 × 16
Fast	0.6	0.7	0.8
Moderate	0.7	0.4	0.5

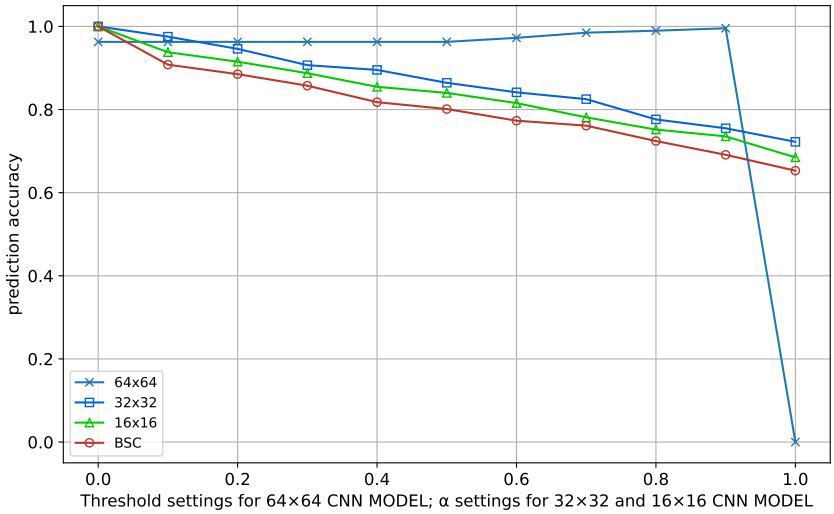

Figure 7. Prediction accuracy of different threshold models.

4.3. The Compression Performance of the Proposed Method

To validate the effectiveness of the proposed BSC structure, we conducted experiments under two threshold modes by setting whether to infer the BSC structure. Table 3 presents the results of RD performance and coding complexity reduction under these two modes, evaluated through BD-BR loss and TS. Fast (i) and Moderate (i) are the results of adding BSC structures. It can be observed that, in both the "Fast" and "Moderate" schemes, introducing the BSC structure significantly reduces time complexity. However, it also comes with more BD-BR loss, and this trend is more pronounced in the "Moderate" scheme. This is because the introduction of the BSC structure leads to more CU block partition structures determined by the model, thereby skipping more intricate partition search processes. In the "Moderate" scheme, where the model accuracy is relatively lower, more inference results are adopted, skipping more partition search processes, but the inaccuracy of the model also leads to more BD-BR loss. Additionally, the acceleration results of this method vary depending on different video sequences, with higher resolution video sequences outperforming the lower resolution ones overall, as more CU partition searches are skipped in high-resolution videos.

Utilizing models at different levels, we established three configurations (L1–L3) for the proposed fast block partitioning method. The L1 configuration indicates acceleration solely with the use of a 64 × 64 CNN model, while all remaining partitions are processed using VTM. Building upon L1, L2 incorporates a 32 × 32 CNN model to infer additional CU blocks. The L3 configuration represents an accelerated setup using three models concurrently. Figure 8 demonstrates the BD-BR loss and TS for the three configurations. It is observed that, with the introduction of deeper model layers, both BD-BR and TS increase concurrently. This is primarily due to a larger number of CU block partitioning decisions being made by the CNN models, as well as the inherent errors introduced by the models themselves. In the L1 configuration, Class A1 and A2 videos exhibit a more pronounced acceleration effect. This is attributed to the fact that, in high-resolution videos, there are more than 64 × 64 blocks that do not require partitioning, thereby skipping a significant number of partition searches. In Class D videos, the acceleration effect of L1 is almost negligible, as the 64 × 64 blocks in low-resolution videos typically need to be partitioned. Moreover, with the introduction of deeper models, the acceleration effect on encoding time is lesser. This is because the partition search time for deeper blocks is relatively short, and the value gained from replacing this process with models is correspondingly lower.

Table 3. The compression performance improvement brought by the BSC structure, with Fast and Moderate being the compression performance improved by adding the BSC structure.

	Test Sequence	Fast (i)		Fast		Moderate (i)		Moderate	
		BDBR (%)	TS (%)	BDBR (%)	TS (%)	BDBR (%)	TS (%)	BDBR (%)	TS (%)
A1	Tango2	2.35	58.15	2.05	55.89	1.96	54.62	1.77	50.32
	FoodMarket4	2.41	64.84	2.12	58.96	1.56	54.19	1.49	49.53
	Campfire	2.45	64.96	2.26	63.32	1.62	55.37	1.58	51.06
A2	DaylightToad2	1.91	63.24	1.65	59.24	1.55	49.96	1.29	46.87
	ParkRunning3	1.85	62.43	1.76	56.84	1.57	53.34	1.39	48.12
	CatRobot	1.81	59.38	1.61	57.21	1.42	50.62	1.27	46.51
B	Kimono	1.73	58.46	1.48	54.25	1.32	52.69	1.25	49.51
	ParkScene	1.62	56.14	1.58	54.62	1.48	52.94	1.35	51.16
	Cactus	1.93	57.27	1.76	54.31	1.41	49.21	1.21	45.84
	BasketballDrive	2.15	56.49	1.94	52.15	1.86	48.14	1.64	45.21
	BQTerrace	1.65	54.32	1.53	52.34	1.32	51.87	1.15	47.24
C	BasketballDrill	1.98	55.19	1.87	51.48	1.84	49.56	1.72	44.02
	PatyScene	1.23	54.55	1.13	53.14	0.83	46.51	0.58	44.23
	RaceHorsesC	1.51	57.14	1.36	51.56	1.14	50.08	0.96	47.12
	BQMall	1.62	52.13	1.38	49.65	1.02	45.17	0.87	38.75
D	BasketballPass	1.93	48.62	1.69	45.82	1.22	46.49	1.04	43.49
	BQSquare	1.39	50.54	1.21	49.75	0.96	47.82	0.75	44.12
	BlowingBubbles	1.54	55.98	1.37	53.51	0.64	48.10	0.58	45.56
	RaceHorses	1.76	55.15	1.62	50.41	1.16	49.41	0.98	44.77
E	FourPeople	2.13	56.33	1.94	55.32	1.65	47.53	1.46	45.35
	Johnny	2.37	59.32	2.13	57.35	1.63	52.91	1.56	47.21
	KristenAndSara	2.25	57.63	2.01	53.91	1.56	46.61	1.49	42.43
	Average	1.86	57.14	1.68	54.14	1.39	50.14	1.24	46.29

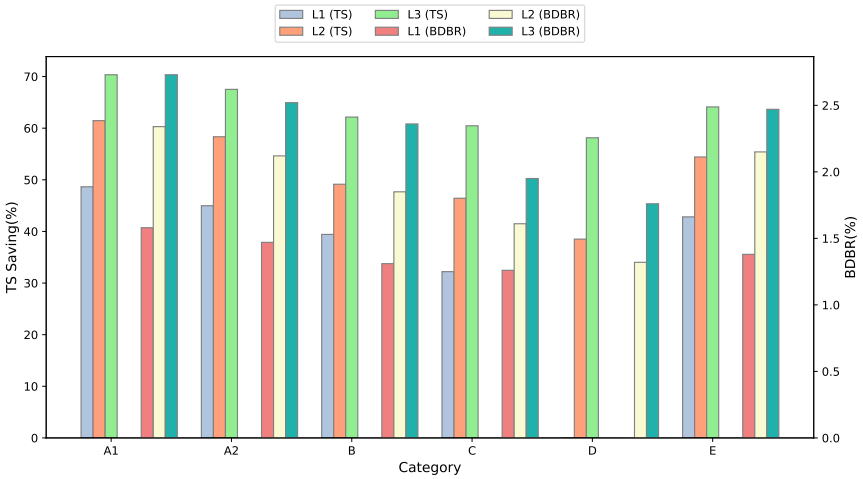

Figure 8. The compression performance of the proposed method under different configurations of L1–L3.

Table 4 and Figure 9 presents a comparison between the proposed method and other state-of-the-art approaches. The results indicate that our algorithm significantly reduces computational complexity. In our "Fast" scheme, the average complexity reduction achieved was 57.14%, with a BDBR increase of 1.86%. The "Moderate" scheme resulted in an average complexity reduction of 50.14%, with a BDBR increase of 1.39%. Comparing with algorithms proposed by [28,32,34], the "Moderate" scheme of the algorithm proposed by [32] achieved an average time saving of 47.90%, with a BDBR increase of 1.29%. The algorithm proposed by [34] resulted in an average time saving of 39.39%, with a BDBR

increase of 0.86%. In contrast, our method achieved average time savings of 2.24% and 10.75% with BDBR increases of only 0.1% and 0.53%, respectively. The algorithm proposed by [28] achieved an average time saving of 50.33%, with a BDBR increase of 1.85%. Our "Moderate" scheme resulted in a slight increase in time complexity by 0.19%, while achieving a BDBR reduction of 0.46%. The "Fast" scheme from the algorithm proposed by [32] saved an average of 55.93% in time, with a BDBR increase of 1.81%. Compared to the "Fast" schemes of algorithms [28,32], our Fast method saved an additional 6.68% and 1.21% in time complexity, respectively, with a BDBR that was essentially the same. This indicates that our algorithm demonstrates superior performance and is more efficient compared to these algorithms.

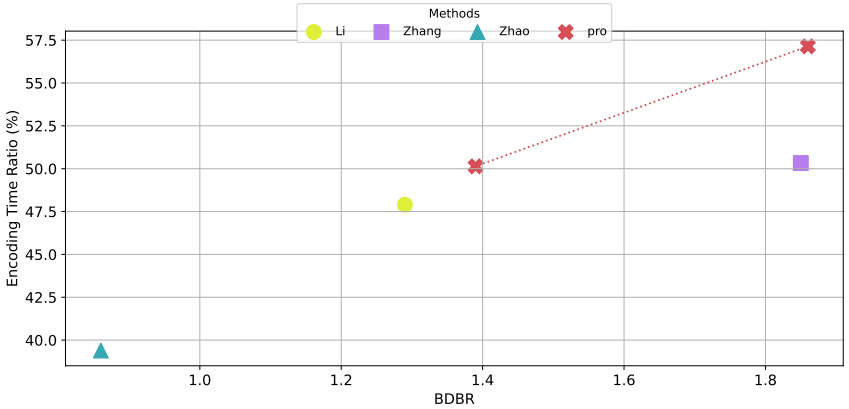

Figure 9. Comparison with other methods. The horizontal axis is the BD–BR increase and the vertical axis is the encoding time acceleration factor.

Table 4. Compare the performance of our algorithm with the performance of others' algorithms.

	Test Video	Li [32]		Zhang [28]		Zhao [34]		Fast		Moderate	
		BDBR (%)	TS (%)	BDBR (%)	TS (%)	BDBR (%)	TS (%)	BDBR (%)	TS (%)	BDBR (%)	TS (%)
A1	Tango2	1.55	50.59	2.23	76.34	0.78	38.87	2.35	58.15	1.96	54.62
	FoodMarket4	1.61	50.11	1.88	76.26	0.82	39.96	2.41	64.84	1.56	54.19
	Campfire	1.59	51.85	1.94	70.72	0.89	41.55	2.45	64.96	1.62	55.37
A2	DaylightRoad2	1.39	54.33	2.00	77.83	0.85	39.58	1.91	63.24	1.55	49.96
	CatRobot	1.77	47.92	2.71	75.08	0.92	40.14	1.85	62.43	1.57	53.34
	ParkRunning3	1.46	48.11	0.79	69.45	0.81	38.22	1.81	59.38	1.42	50.62
B	Kimono	0.98	51.04	- -	- -	0.78	37.51	1.73	58.46	1.32	52.69
	ParkScene	1.05	51.18	- -	- -	0.61	39.56	1.62	56.14	1.48	52.94
	Cactus	1.31	44.95	1.88	58.39	- -	- -	1.93	57.27	1.41	49.21
	BasketballDrive	1.49	46.16	2.03	63.74	- -	- -	2.15	56.49	1.86	48.14
	BQTerrace	0.94	48.33	1.55	52.37	0.76	41.79	1.65	54.32	1.32	51.87
C	BasketballDrill	1.18	45.17	2.32	35.49	1.25	39.21	1.98	55.19	1.84	49.56
	PatyScene	1.05	51.18	0.76	27.73	0.37	36.73	1.23	54.55	0.83	46.51
	RaceHorsesC	0.81	46.95	1.03	39.43	0.24	30.68	1.51	57.14	1.14	50.08
	BQMall	1.24	48.33	4.64	37.61	- -	- -	1.62	52.13	1.02	45.17
D	BasketballPass	1.41	40.04	1.03	25.54	- -	- -	1.93	48.62	1.22	46.49
	BQSquare	0.89	46.68	0.58	17.85	0.58	36.67	1.39	50.54	0.96	47.82
	BlowingBubbles	0.99	43.86	0.56	18.61	0.83	40.87	1.54	55.98	0.64	48.10
	RaceHorses	1.27	39.21	0.80	23.13	0.56	36.51	1.76	55.15	1.16	49.41
E	FourPeople	1.55	52.64	2.67	55.10	1.34	46.51	2.13	56.33	1.65	47.53
	Johnny	1.58	50.67	3.25	55.13	1.56	43.78	2.37	59.32	1.63	52.91
	KristenAndSara	1.63	49.82	2.43	50.87	1.57	40.85	2.25	57.63	1.56	46.61
	Average	1.29	47.90	1.85	50.33	0.86	39.39	1.86	57.14	1.39	50.14

5. Conclusions

In this paper, we propose a novel block segmentation and block connect structure (BSC) to uniformly represent the CUs of different sizes. We design CNN models tailored to different CU sizes to predict their partition modes, replacing the computationally intensive RD partition search process. Additionally, to enhance the model accuracy, we introduce a multi-threshold approach based on the characteristics of CUs of different sizes to further assess the confidence of the predictions. The experiment results show that, compared with VTM-10.0, our "Fast" scheme reduces the average complexity by 57.14% and increases the BDBR by 1.86%, while the "Moderate" scheme reduces the average complexity by 50.14% and increases the BDBR by only 1.39%.

Author Contributions: Conceptualization, N.L. and Z.W.; methodology, N.L.; software, Z.W.; validation, N.L., Q.Z. and Z.W.; formal analysis, Z.W.; investigation, Z.W.; resources, Q.Z.; data curation, Z.W.; writing—original draft preparation, N.L.; writing—review and editing, N.L.; visualization, N.L.; supervision, Q.Z.; project administration, Q.Z.; funding acquisition, Q.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China No. 61771432, and 61302118, the Basic Research Projects of Education Department of Henan No. 21zx003, and the Key projects Natural Science Foundation of Henan 232300421150, Henan Provincial Science and Technology Research Project 242102211020, the Scientic and Technological Project of Henan Province 232102211014, and the Postgraduate Education Reform and Quality Improvement Project of Henan Province YJS2023JC08. Zhongyuan Science and Technology Innovation Leadership Program 244200510026.

Data Availability Statement: The data can be shared up on request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Bross, B.; Wang, Y.K.; Ye, Y.; Liu, S.; Chen, J.; Sullivan, G.J.; Ohm, J.R. Overview of the Versatile Video Coding (VVC) Standard and Its Applications. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *31*, 3736–3764. [CrossRef]
2. Yang, H.; Shen, L.; Dong, X.; Ding, Q.; An, P.; Jiang, G. Low-Complexity CTU Partition Structure Decision and Fast Intra Mode Decision for Versatile Video Coding. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *30*, 1668–1682. [CrossRef]
3. Lin, S.; Chen, H.; Zhang, H.; Maxim, S.; Yang, H.; Zhou, J. *Affine Transform Prediction for Next Generation Video Coding*; Huawei Technologies: Geneva, Switzerland, 2015.
4. Chan, K.H.; Im, S.K. Faster Inter Prediction by NR-Frame in VVC. In Proceedings of the 2023 7th International Conference on Graphics and Signal Processing (ICGSP'23), New York, NY, USA, 23 June 2023; pp. 24–28. [CrossRef]
5. Chan, K.H.; Im, S.K. Using Four Hypothesis Probability Estimators for CABAC in Versatile Video Coding. *ACM Trans. Multimed. Comput. Commun. Appl.* **2023**, *19*, 40. [CrossRef]
6. Tissier, A.; Mercat, A.; Amestoy, T.; Hamidouche, W.; Vanne, J.; Menard, D. Complexity Reduction Opportunities in the Future VVC Intra Encoder. In Proceedings of the 2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP), Kuala Lumpur, Malaysia, 27–29 September 2019; pp. 1–6. [CrossRef]
7. Wang, Y.; Liu, Y.; Zhao, J.; Zhang, Q. Fast CU Partitioning Algorithm for VVC Based on Multi-Stage Framework and Binary Subnets. *IEEE Access* **2023**, *11*, 56812–56821. [CrossRef]
8. Gu, J.; Tang, M.; Wen, J.; Han, Y. Adaptive Intra Candidate Selection With Early Depth Decision for Fast Intra Prediction in HEVC. *IEEE Signal Process. Lett.* **2018**, *25*, 159–163. [CrossRef]
9. Bae, J.H.; Sunwoo, M.H. Adaptive Early Termination Algorithm Using Coding Unit Depth History in HEVC. *J. Signal Process. Syst.* **2019**, *91*, 863–873. [CrossRef]
10. Zhang, T.; Sun, M.T.; Zhao, D.; Gao, W. Fast Intra-Mode and CU Size Decision for HEVC. *IEEE Trans. Circuits Syst. Video Technol.* **2017**, *27*, 1714–1726. [CrossRef]
11. Menon, V.V.; Amirpour, H.; Timmerer, C.; Ghanbari, M. INCEPT: Intra CU Depth Prediction for HEVC. In Proceedings of the 2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP), Tampere, Finland, 6–8 October 2021; pp. 1–6. [CrossRef]
12. Kim, N.; Jeon, S.; Shim, H.J.; Jeon, B.; Lim, S.C.; Ko, H. Adaptive Keypoint-Based CU Depth Decision for HEVC Intra Coding. In Proceedings of the 2016 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Nara, Japan, 1–3 June 2016; pp. 1–3. [CrossRef]
13. Ahn, S.; Lee, B.; Kim, M. A Novel Fast CU Encoding Scheme Based on Spatiotemporal Encoding Parameters for HEVC Inter Coding. *IEEE Trans. Circuits Syst. Video Technol.* **2015**, *25*, 422–435. [CrossRef]

14. Chen, J.; Sun, H.; Katto, J.; Zeng, X.; Fan, Y. Fast QTMT Partition Decision Algorithm in VVC Intra Coding Based on Variance and Gradient. In Proceedings of the 2019 IEEE Visual Communications and Image Processing (VCIP), Sydney, Australia, 1–4 December 2019; pp. 1–4. [CrossRef]
15. Song, Y.; Zeng, B.; Wang, M.; Deng, Z. An Efficient Low-Complexity Block Partition Scheme for VVC Intra Coding. *J. Real-Time Image Process.* **2022**, *19*, 161–172. [CrossRef]
16. Saldanha, M.; Sanchez, G.; Marcon, C.; Agostini, L. Fast Partitioning Decision Scheme for Versatile Video Coding Intra-Frame Prediction. In Proceedings of the 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Virtual, 10–21 October 2020; pp. 1–5. [CrossRef]
17. Park, S.H.; Kang, J.W. Context-Based Ternary Tree Decision Method in Versatile Video Coding for Fast Intra Coding. *IEEE Access* **2019**, *7*, 172597–172605. [CrossRef]
18. Chen, F.; Ren, Y.; Peng, Z.; Jiang, G.; Cui, X. A Fast CU Size Decision Algorithm for VVC Intra Prediction Based on Support Vector Machine. *Multimed. Tools Appl.* **2020**, *79*, 27923–27939. [CrossRef]
19. Zhao, T.; Huang, Y.; Feng, W.; Xu, Y.; Kwong, S. Efficient VVC Intra Prediction Based on Deep Feature Fusion and Probability Estimation. *IEEE Trans. Multimed.* **2023**, *25*, 6411–6421. [CrossRef]
20. Park, S.h.; Kang, J.W. Fast Multi-Type Tree Partitioning for Versatile Video Coding Using a Lightweight Neural Network. *IEEE Trans. Multimed.* **2021**, *23*, 4388–4399. [CrossRef]
21. Li, Y.; Yang, G.; Song, Y.; Zhang, H.; Ding, X.; Zhang, D. Early Intra CU Size Decision for Versatile Video Coding Based on a Tunable Decision Model. *IEEE Trans. Broadcast.* **2021**, *67*, 710–720. [CrossRef]
22. Jin, Z.; An, P.; Shen, L.; Yang, C. CNN Oriented Fast QTBT Partition Algorithm for JVET Intra Coding. In Proceedings of the 2017 IEEE Visual Communications and Image Processing (VCIP), St. Petersburg, FL, USA, 10–13 December 2017; pp. 1–4. [CrossRef]
23. Huang, Y.H.; Chen, J.J.; Tsai, Y.H. Speed Up H.266/QTMT Intra-Coding Based on Predictions of ResNet and Random Forest Classifier. In Proceedings of the 2021 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 10–12 January 2021; pp. 1–6. [CrossRef]
24. HoangVan, X.; NguyenQuang, S.; DinhBao, M.; DoNgoc, M.; Trieu Duong, D. Fast QTMT for H.266/VVC Intra Prediction Using Early-Terminated Hierarchical CNN Model. In Proceedings of the 2021 International Conference on Advanced Technologies for Communications (ATC), Ho Chi Minh City, Vietnam, 14–16 October 2021; pp. 195–200. [CrossRef]
25. Feng, A.; Liu, K.; Liu, D.; Li, L.; Wu, F. Partition Map Prediction for Fast Block Partitioning in VVC Intra-Frame Coding. *IEEE Trans. Image Process.* **2023**, *32*, 2237–2251. [CrossRef] [PubMed]
26. Li, T.; Xu, M.; Tang, R.; Chen, Y.; Xing, Q. DeepQTMT: A Deep Learning Approach for Fast QTMT-Based CU Partition of Intra-Mode VVC. *IEEE Trans. Image Process.* **2021**, *30*, 5377–5390. [CrossRef] [PubMed]
27. Tissier, A.; Hamidouche, W.; Mdalsi, S.B.D.; Vanne, J.; Galpin, F.; Menard, D. Machine Learning Based Efficient QT-MTT Partitioning Scheme for VVC Intra Encoders. *IEEE Trans. Circuits Syst. Video Technol.* **2023**, *33*, 4279–4293. [CrossRef]
28. Zhang, Q.; Guo, R.; Jiang, B.; Su, R. Fast CU Decision-Making Algorithm Based on DenseNet Network for VVC. *IEEE Access* **2021**, *9*, 119289–119297. [CrossRef]
29. Wu, S.; Shi, J.; Chen, Z. HG-FCN: Hierarchical Grid Fully Convolutional Network for Fast VVC Intra Coding. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *32*, 5638–5649. [CrossRef]
30. Peng, Z.; Shen, L.; Ding, Q.; Dong, X.; Zheng, L. Block-Dependent Partition Decision for Fast Intra Coding of VVC. *IEEE Trans. Consum. Electron.* 2023, *early access*. [CrossRef]
31. Chen, Z.; Shi, J.; Li, W. Learned Fast HEVC Intra Coding. *IEEE Trans. Image Process.* **2020**, *29*, 5431–5446. [CrossRef]
32. Li, H.; Zhang, P.; Jin, B.; Zhang, Q. Fast CU Decision Algorithm Based on Texture Complexity and CNN for VVC. *IEEE Access* **2023**, *11*, 35808–35817. [CrossRef]
33. Tang, G.; Jing, M.; Zeng, X.; Fan, Y. Adaptive CU Split Decision with Pooling-variable CNN for VVC Intra Encoding. In Proceedings of the 2019 IEEE Visual Communications and Image Processing (VCIP), Sydney, Australia, 1–4 December 2019; pp. 1–4. [CrossRef]
34. Zhao, J.; Wang, Y.; Zhang, Q. Adaptive CU Split Decision Based on Deep Learning and Multifeature Fusion for H.266/VVC. *Sci. Program.* **2020**, *2020*, 8883214. [CrossRef]
35. Agustsson, E.; Timofte, R. NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, 21–26 July 2017; pp. 1122–1131. [CrossRef]
36. VTM-10.0 Jvet VVCSoftware_VTM GitLab. Available online: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/releases/VTM-10.0 (accessed on 25 January 2024).
37. Li, B.K.S.X.; Seregin, V. JVET Common Test Conditions and Software Reference Configurations for SDR Video. In Proceedings of the Joint Video Explor, July 2018. Available online: https://www.researchgate.net/publication/326506581_JVET-J1010_JVET_common_test_conditions_and_software_reference_configurations (accessed on 25 January 2024).
38. Bjontegaard, G. Calculation of Average PSNR Differences between RD-Curves. ITU SG16 Doc. VCEG-M33. 2001. Available online: https://api.semanticscholar.org/CorpusID:61598325 (accessed on 25 January 2024).