Article

Low-Complexity Fast CU Classification Decision Method Based on LGBM Classifier

Yanjun Wang, Yong Liu, Jinchao Zhao and Qiuwen Zhang *

College of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China
* Correspondence: 2012032@zzuli.edu.cn; Tel.: +86-371-8660-9559

Abstract: At present, the latest video coding standard is Versatile Video Coding (VVC). Although the coding efficiency of VVC is significantly improved compared to the previous generation, standard High-Efficiency Video Coding (HEVC), it also leads to a sharp increase in coding complexity. VVC significantly improves HEVC by adopting the quadtree with nested multi-type tree (QTMT) partition structure, which has been proven to be very effective. This paper proposes a low-complexity fast coding unit (CU) partition decision method based on the light gradient boosting machine (LGBM) classifier. Representative features were extracted to train a classifier matching the framework. Secondly, a new fast CU decision framework was designed for the new features of VVC, which could predict in advance whether the CU was divided, whether it was divided by quadtree (QT), and whether it was divided horizontally or vertically. To solve the multi-classification problem, the technique of creating multiple binary classification problems was used. Subsequently, a multi-threshold decision-making scheme consisting of four threshold points was proposed, which achieved a good balance between time savings and coding efficiency. According to the experimental results, our method achieved a significant reduction in encoding time, ranging from 47.93% to 54.27%, but only improved the Bjøntegaard delta bit-rate (BDBR) by 1.07%~1.57%. Our method showed good performance in terms of both encoding time reduction and efficiency.

Keywords: LGBM; intra coding; fast coding algorithm; versatile video coding

1. Introduction

With the rapid progress in the fields of communication and computing, digital video has become widely available on the internet and people are gradually inclined to pursue higher definition video, and with it, a huge amount of data is generated. This poses a huge challenge to the infrastructure of the telecom sector. With the great impact of COVID-19 [1], people have changed their traditional shopping and learning patterns, with many customers becoming accustomed to online shopping and many businesses trying to sell their goods in the form of live video streaming on internet platforms. Many school students have also been forced to start online classes. The dramatic increase in data has forced streaming providers to meet the needs of a wide range of users only in the form of reduced resolution of videos [2]. Currently, mobile devices are available due to their convenience, flexibility, and security. The market share of mobile devices is high, but they are limited by their size and their data processing capacity is very limited; the high complexity of video codecs for mobile devices remains a pressing issue that requires urgent resolution.

The standardization of HEVC [3] has resulted in a significant improvement in video compression performance, but HEVC is also helpless in the face of the recent data explosion and does not provide the desired performance and required coding efficiency for video applications and related industries. In response to such problems, international organizations have also established new video coding standards. In the latter half of 2015, the establishment of the Joint Video Exploration Team (JVET) [4] was the result of a collaboration...
between the Moving Picture Experts Group (MPEG) and the Video Coding Experts Group (VCEG), whose mission was to explore more advanced video compression techniques and develop next-generation video compression standards. As the newest video coding standard [5], Versatile Video Coding (VVC) provides significant improvements compared to HEVC, providing better compression performance with the same video quality. Compared to HEVC, VVC supports higher resolution video and utilizes an increased coding tree unit (CTU) size of 128 × 128 pixels. Furthermore, both the encoder and decoder support concurrent processing, with the added benefit of the decoder selectively decoding the required video region.

The utilization of the latest QTMT division structure [6] is one of the primary contributors to the enhanced performance observed in VVC coding, as shown in Figure 1. The division of the coding tree unit (CTU) in VVC includes five types: quadtree (QT), horizontal binary tree splitting (BH), vertical binary tree partition (BV), horizontal ternary tree splitting (TH), and vertical ternary tree splitting (TV). The flexible partitioning and use of many new coding tools have led to a dramatic increase in coding performance while also leading to a dramatic increase in complexity. VVC luminance and chrominance blocks are different from HEVC in that VVC forms a dual-tree coding structure. Within a given CTU, the chrominance block can possess a coding tree structure that is independent of the luminance samples. This allows for the use of larger coding blocks for the chrominance block than for the luminance block. The intra-frame prediction of HEVC includes two types, DC and Planar, for directional angle prediction and smooth prediction [7], and VVC extends the 33 angle prediction modes within HEVC frames to 67 in order to improve the prediction accuracy [8].

![Figure 1. The division of QTMT.](image)

VVC has added several new enhancement techniques that can improve the compression efficiency of video while maintaining video quality. The following techniques, namely quadtree with nested multi-type tree (QTMT), affine motion compensation (AMC), multiple transform selection (MTS), low-frequency non-separable transform (LFNST), and novel intra-frame prediction tools [9], have been primarily utilized. Although the application of these new techniques can allow VVC to obtain more impressive efficiency, the additional complexity is a problem we have to face. It can be seen in the literature [10] that the coding efficiency of VVC test model (VTM) was 25% higher than that of HEVC test model (HM), and the coding complexity was increased by more than 26 times.
In recent years, machine learning has played a significant role in advancing various state-of-the-art techniques in the field of image processing. Many researchers have successfully applied machine learning techniques to effectively reduce the complexity of VVC video coding. Although conventional machine learning methods such as support vector machine (SVM), decision tree (DT), and random forest (RF) [11–13] have been widely used, recent developments in deep learning have led to significant improvements in many image processing tasks, yielding satisfactory outcomes. Nevertheless, there is still significant room for improvement in terms of the balance between complexity and performance. The lightweight gradient boosting machine (LGBM) released by Microsoft is a powerful technique [14] with higher training efficiency; compared to the XGBoost algorithm, the LGBM algorithm offers advantages such as reduced memory usage, improved accuracy, support for parallelized learning, the ability to handle large-scale data, and compatibility with discrete data classes. LGBM can be highly customized to the needs of the application and is superior to the traditional machine learning algorithms presented above, based on which a solution is proposed in this paper.

Our paper introduces a low-complexity and efficient approach to CU partitioning using the LGBM classifier. The objective was to optimize coding time and minimize the impact on coding efficiency. Mainly, the three objectives were:

1. To fully explore the new features of QTMT, select more representative features for different classifiers, and extract different features for training in order to train classifiers with high accuracy and simple logic, thereby meeting the requirements of the framework.

2. To develop a novel and swift CU decision framework that is specifically tailored for the distinctive features of VVC. This methodology entailed transforming the multi-classification problem into several binary classification problems. The framework not only had high decision accuracy but also good ability to reduce the complexity.

3. To develop a decision scheme with multiple thresholds that optimizes the balance between coding complexity and efficiency while ensuring a high level of performance.

The subsequent sections of this paper are structured as follows: Section 2 presents some effective approaches to VVC complexity reduction by scholars in recent years. Section 3 presents a detailed description of the proposed algorithm implementation and introduces the solution and fast CU decision. Section 4 presents our experimental performance evaluation. Section 5 summarizes the work.

2. Background and Related Works

The domain of computer vision and pattern recognition has seen substantial advancements in recent times due to the progress in machine learning. The utilization of machine learning to tackle classification and regression problems is a powerful data-driven technique. Due to the good performance of machine learning, numerous experts and scholars are endeavoring to apply advanced machine learning techniques to video coding, aiming to decrease its complexity.

2.1. Fast CU Division Method Based on HEVC

In previous decades, numerous researchers have made notable contributions towards reducing the complexity involved in coding with HEVC. In the literature [15], a fast intra-frame pattern decision algorithm was proposed to compute the histogram of gradient patterns for each CU using the information of gradients. The best candidate patterns were selected for the rate distortion optimization (RDO) process according to the distribution pattern of the histogram. The literature [16] also presented an adaptive intra-frame pattern jumping algorithm based on pattern decision and signal processing. This algorithm utilized the statistical properties of adjacent reference samples to significantly reduce the coding complexity. The segmentation of partially homogeneous CU was accomplished in the literature [17] by utilizing the average gradient in the horizontal (AGH) and vertical (AGV) directions beforehand. Then, early execution of the termination decision was proposed.
for the remaining CUs based on two SVMs employing depth difference, the features of which included the HAD and RD cost ratios. In the cited literature [18], CTU partitioning was addressed by treating it as a multi-classification problem through the utilization of three binary classifiers that were based on the support vector machine (SVM) model of the HEVC encoder. The CU partitioning module could be predicted more efficiently.

A decision scheme for intra-frame prediction in HEVC based on the coding information of temporally adjacent frames was proposed in the literature [19]. The paper initially examined the correlation between the depth texture and non-texture cost of the present coding unit (CU), explored the direct coupling between POTCIC and CU depth, and developed a decision scheme with a POTCIC threshold. The literature [20] proposed an HM-CNN framework for predicting the depth of CUs based on CNN, where a $64 \times 64$ CTU was first used as the input to the CNN and the depth prediction of each CU depended on the $16 \times 16$ matrix representation of each $4 \times 4$ block. A fast CU partitioning method based on ResNet networks was proposed in the literature [21]. The approach employed in the paper entailed treating each depth partition as a binary classification problem, and leveraging pertinent texture information to train the ResNet network for the purpose of partitioning each depth. The method was demonstrated to yield good results.

2.2. Fast CU Division Method Based on VVC

The coding complexity reduction schemes mentioned above were developed for the HEVC standard. Since VVC uses the latest QTMT division structure and an additional 67 intra patterns, these new features lead to many algorithms of HEVC that are not directly usable. In recent years, various innovative approaches have been proposed to tackle the new characteristics of VVC. One such approach is presented in the literature [22], which introduced a fast CNN-based CU division algorithm. This approach placed an emphasis on predictive division via texture information and developed specific classifiers for various CU sizes to predict whether certain division patterns could be skipped. A more accurate loss function was designed to avoid performance degradation. A fast CU classification algorithm for VVC frames based on a support vector machine (SVM) was proposed in the literature [23], where S-NS and HS-VS classifiers were designed for different CU sizes using sequence features to reduce the complexity of SVM classifiers while improving accuracy. In the literature [24], a rapid CU classification approach was proposed based on texture characteristics. The proposed algorithm computed the texture complexity of the current coding unit (CU) to decide whether it should be partitioned into smaller CUs. Based on the correlation between the texture direction of the current coding unit (CU) and the CU splitting pattern, the algorithm selected the most suitable candidate pattern. The literature [25] proposed a fast random forest-based CU partitioning algorithm, which first classified the CU into one of three categories based on the encoding information, simple, fuzzy, or complex CU. A random forest classifier was trained for simple and complex CUs to predict the best partition, and then a random forest classifier was trained separately for fuzzy CUs to predict the best CU partitioning pattern. The literature [26] presented two proposed methods for partitioning decisions in VVC. The first method was a texture-based MTT (multi-type tree) partitioning decision method, while the second was a gradient-based intra-frame decision method, which not only reduced the redundancy in CU partitioning prediction but also helped to save processing time. Texture features were employed to forecast the intricacy and forecasting direction of the present CU. Constructing regression functions based on these features can effectively save coding time without reducing coding efficiency. The literature [27] proposed a CNN-based algorithm for rapid CU partitioning in VVC prediction. Firstly, a database was created based on various CU sizes, and the CU was then divided into multiple stages based on the division pattern. Then, a multi-stage exit CNN (MSE-CNN) was proposed, which combined conditional convolution and effective partitioning of subnets, consisting of a framework with an early exit mechanism that could effectively skip the extra redundancy checking process. Additionally, a decision scheme utilizing multiple thresholds was devised to strike
a balance between the rate-distortion (RD) performance and coding complexity. In the literature [28], a fast CU classification algorithm based on ResNet networks was proposed for efficient processing of coding units, which included three stages. Firstly, a statistical analysis of the proportion of classification patterns of CU was performed. Secondly, a ResNet-based CNN model was designed for CU prediction. A decision scheme employing dual thresholds was proposed to strike a balance between the coding complexity and rate-distortion (RD) performance. A two-stage CNN-based scheme for fast CU classification was proposed in the literature [29]. In the initial phase, a multi-branch CNN was employed to forecast the depth of the CU and the output was forwarded to the subsequent stage, and the second stage was designed to prune unnecessary computations. That is, to reduce the computational complexity of CU partitioning, a restriction was imposed on the depth range of the CU. A method based on deep learning for predicting the partitioning of the CU was proposed in the literature [30], firstly designing the hierarchical grid graph to partition the hierarchy of the VVC, then proposing a new hierarchical grid full convolutional network (HG-FCN) framework to obtain all the partitioning information of the current CU and CUs with only one inference to speed up the coding process of the CU. A novel dual-threshold decision scheme was proposed in the end to balance the trade-off between the coding complexity and performance.

2.3. Our Proposed Algorithm

The fast CU division schemes based on VVC and HEVC introduced above in recent years had had limited ability to reduce the complexity, although the coding efficiency was more desirable. Despite the improved coding efficiency of the new QTMT partitioning structure used in VVC, the cost is a sharp increase in coding time. This is because RDO makes predictions for all CU partitions to obtain the optimal partition. We can learn from the literature [31] that the QTMT partitioning structure occupies more than 95% of the coding time. This section briefly describes our low-complexity fast CU partitioning decision algorithm.

The present study proposed a low-complexity method for fast CU partitioning decisions. Our approach employed the LGBM classifier to minimize both the coding time and the impact on coding efficiency. The following are the key contributions of this study:

(1) The new features of QTMT were fully explored, more representative features for different classifiers were selected, and different features for training were extracted in order to train classifiers with high accuracy and simple logic, thereby meeting the needs of the method.

(2) The second contribution is the proposal of a new fast CU decision framework, the objective of which was to transform the multi-classification problem into several binary classification problems. The framework not only had high decision accuracy but also good ability to reduce the complexity.

(3) A multi-threshold decision scheme with a total of four threshold points was proposed, which achieved a favorable trade-off between coding efficiency and time savings.

3. Proposed Methodology

VVC inherits some of the coding features of HEVC, including the RDO process, which evaluates the partitioning and prediction patterns of multiple CUs to select the best coding scheme. $RD_{\text{cost}}$ is defined as:

$$RD_{\text{cost}} = SSE + \lambda \times Bit_{\text{mode}}$$  \hspace{1cm} (1)

where the distortion of luminance and chrominance is denoted by $SSE$, the bit cost of the prediction mode within the frame is represented by $Bit_{\text{mode}}$, and the Lagrangian multiplier is denoted by $\lambda$. Since VVC adopts a new QTMT partition structure, the partitioned structure is not only a square but also a new rectangular structure. As a result, there is a significant rise in the complexity associated with predicting the CTU partition structure. Theoretically,
VVC intra-frame CU depth and texture complexity are inextricably linked. Flat regions are more likely to be encoded with larger CU sizes, while texture-rich regions tend to be encoded by smaller CU sizes. Meanwhile, there exists a significant correlation between the depth of the coding units (CUs) and the resolution in the context of VVC. Typically, larger CU sizes are utilized for encoding high-resolution video sequences, while lower resolution video sequences tend to be encoded with smaller CU sizes. Therefore, in order to ensure high coding efficiency while being able to reduce the coding complexity, we proposed a low-complexity fast CU partitioning decision method based on the LGBM classifier. This chapter is divided into four subsections to illustrate the fast CU partitioning decision algorithm and provide a detailed discussion of the proposed algorithm. Section 3.1 describes the decision analysis, Section 3.2 describes the analysis and selection of features, Section 3.3 describes the training of the classifier, and Section 3.4 describes the threshold decision.

3.1. Analysis of Decision Making

There are 6 partitioning modes for CU division in VVC, which are NS, QT, BHT, BTV, TTH, and TTV. Figure 2 shows the partitioning order in the most primitive VTM. The process of conducting RDO to determine the optimal CU partitioning consumes a significant amount of coding time. In previous research methods, many experts and scholars have defaulted to view the partitioning decision of CU as a multi-classification problem. Since VVC uses a QTMT partitioning structure, previous methods have encountered difficulties in accurately predicting the optimal CU partitioning. For such problems, in this section, we will now discuss the characteristics of our proposed method for making low-complexity CU partitioning decisions, which was aimed at achieving a more efficient approach. Additionally, we will introduce the classifiers that were selected for our method.

![Diagram of VVC Intra-CU Partitioning Process]

Figure 2. The diagram depicts the process of intra coding in VVC.
In order to achieve a more accurate division of VVC CUs, we collected statistics for the division information of different CU sizes. We encoded all test sequences of VVC in VTM-10.0; the full internal main configuration was encoded using the default configuration file encoder_intra_main.cfg. Table 1 shows the division ratios in VTM-10.0 for different size CUs. From the table it can be concluded that no splitting (NS) was the choice for the vast majority of different CU sizes. TH and TV made up only a smaller part of the partitioning, while horizontal and vertical divisions made up a larger percentage.

Table 1. Division ratios per CU size for VTM-10.0.

<table>
<thead>
<tr>
<th>CU Size</th>
<th>NS</th>
<th>QT</th>
<th>BH</th>
<th>BV</th>
<th>TH</th>
<th>TV</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 × 64</td>
<td>0.17</td>
<td>0.83</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>32 × 32</td>
<td>0.18</td>
<td>0.35</td>
<td>0.18</td>
<td>0.17</td>
<td>0.07</td>
<td>0.06</td>
</tr>
<tr>
<td>32 × 16</td>
<td>0.41</td>
<td>0</td>
<td>0.18</td>
<td>0.23</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>32 × 8</td>
<td>0.48</td>
<td>0</td>
<td>0.18</td>
<td>0.22</td>
<td>0</td>
<td>0.12</td>
</tr>
<tr>
<td>32 × 4</td>
<td>0.77</td>
<td>0</td>
<td>0</td>
<td>0.13</td>
<td>0</td>
<td>0.10</td>
</tr>
<tr>
<td>16 × 32</td>
<td>0.41</td>
<td>0</td>
<td>0.25</td>
<td>0.17</td>
<td>0.09</td>
<td>0.07</td>
</tr>
<tr>
<td>8 × 32</td>
<td>0.47</td>
<td>0</td>
<td>0.25</td>
<td>0.14</td>
<td>0.14</td>
<td>0</td>
</tr>
<tr>
<td>4 × 32</td>
<td>0.72</td>
<td>0</td>
<td>0.15</td>
<td>0</td>
<td>0.13</td>
<td>0</td>
</tr>
<tr>
<td>16 × 16</td>
<td>0.23</td>
<td>0.11</td>
<td>0.23</td>
<td>0.24</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>16 × 8</td>
<td>0.50</td>
<td>0</td>
<td>0.18</td>
<td>0.22</td>
<td>0</td>
<td>0.09</td>
</tr>
<tr>
<td>16 × 4</td>
<td>0.61</td>
<td>0</td>
<td>0</td>
<td>0.24</td>
<td>0</td>
<td>0.15</td>
</tr>
<tr>
<td>8 × 16</td>
<td>0.49</td>
<td>0</td>
<td>0.24</td>
<td>0.18</td>
<td>0.09</td>
<td>0</td>
</tr>
<tr>
<td>4 × 16</td>
<td>0.60</td>
<td>0</td>
<td>0.24</td>
<td>0</td>
<td>0.16</td>
<td>0</td>
</tr>
<tr>
<td>8 × 8</td>
<td>0.54</td>
<td>0</td>
<td>0.23</td>
<td>0.23</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8 × 4</td>
<td>0.66</td>
<td>0</td>
<td>0</td>
<td>0.34</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4 × 8</td>
<td>0.66</td>
<td>0</td>
<td>0.34</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Our fast CU decision framework was quite different from previous work, in which the QT partition and multi-type tree (MT) partition were determined separately. However, MT partitioning is still a multi-class classification, and although the structure achieves good RD performance, it has limited ability to reduce the complexity. In VVC, QT partitioning is immediately followed by BT and TT partitioning, so we can skip BT and TT partitioning by determining QT partitioning in advance. By adopting this approach, significant coding time could be saved while reducing the coding complexity. This motivated us to propose a novel fast CU partitioning decision framework. As shown in Figure 3, in the fast CU partitioning, the initial judgment was whether to partition, and if not, the partitioning was terminated in advance, and whether the conditions were met to further determine QT partitioning, and if so, MT partitioning was automatically skipped. If QT partitioning was not determined, horizontal and vertical partitioning was judged. Then came binomial and trinomial tree partitioning. Hence, our proposed framework for fast CU partitioning decision was designed to transform the multi-classification problem into multiple binary classification problems. The framework not only had high decision accuracy but also good ability to reduce the complexity.

The first step in dealing with the binary classification problem is to find a suitable classifier. In previous work, scholars have opted for classifiers such as support vector machine (SVM), decision tree (DT), and CNN, but these classifiers have various drawbacks. We deal with a lot of data and SVM is computationally expensive, especially when dealing with large datasets. Decision trees are very easy to overfit, especially when the trees become very large or deep. This can lead to poor generalization performance of the data. CNNs require a lot of data and computational resources to be trained effectively. These are just some of the disadvantages of these classifiers, but they are not exhaustive. Depending on the strengths and weaknesses of the given problem and the specific characteristics of the data used, it is therefore important to choose the right classifier.
Figure 3. The new fast CU decision framework.

LGBM is a well-known machine learning algorithm that is frequently utilized for classification tasks. Some of the advantages of LGBM as a classifier include:

- **Speed**: LGBM is widely recognized for its speed and efficiency, which has made it a preferred option for handling large datasets.
- **Accuracy**: LGBM is known for its high accuracy and has been shown to outperform other popular machine learning algorithms, such as Random Forest and XGBoost, in some cases.
- **Flexibility**: LGBM is highly flexible and customizable, allowing users to adapt the algorithm to their specific use cases. It also supports a wide range of loss functions and evaluation metrics, making it suitable for a variety of classification problems.
- **Feature Importance**: LGBM enables the evaluation of the importance of each feature in the model through a feature importance measure. This helps with feature selection and understanding which features are the most predictive.

Overall, LGBM is a powerful and versatile algorithm that has many advantages as a classifier. Based on these advantages, we finally choose LGBM as the classifier for this paper.

### 3.2. Feature Analysis and Selection

To enhance the precision of the fast CU decision framework, certain improvements are necessary. We needed to consider selecting more representative features for the new features of QTMT division structure to avoid incurring more extra computational overhead. To find more effective decision features, we collected data from a large number of video sequences and performed many comparison experiments. Based on the results of the experiments, the proposed method utilized four primary types of features, namely global texture information, local texture information, contextual information, and encoding information.

1. **Global texture information**: In the fast coding of VVC, global texture information is widely used, and the aforementioned features are computed using the current CU’s
luminance samples. We selected five features, including the variance of the current CU (VAR); the horizontal gradient ($G_x$) and vertical gradient ($G_y$) based on the Sobel operator; $G_x$ divided by $G_y$ (ratio $G_x/G_y$); and the sum of $G_x$ and $G_y$ divided by the block area (normGradient).

Then, $G_x$ and $G_y$ can be expressed as:

$$G_x = \sum_{m=1}^{W} \sum_{n=1}^{H} A * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$  \quad (2)$$

$$G_y = \sum_{m=1}^{W} \sum_{n=1}^{H} A * \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$  \quad (3)$$

where $p(m, n)$ denotes the brightness value at position $(m, n)$, and $p(m-1, n-1)$, $p(m-1, n)$, $p(m-1, n+1)$, $p(m, n-1)$, $p(m, n+1)$, $p(m+1, n-1)$, $p(m+1, n)$, and $p(m+1, n+1)$ are the brightness values around position $(m, n)$, respectively.

(2) Local texture information: In addition to the global texture information, local texture information is a crucial aspect that we considered. The features of the local texture information include: the absolute variance of the four sub-regions (DiffVarQT); the local texture information, which is further analyzed by computing the maximum variance of the four sub-regions (MaxVarQT); the absolute variance of the upper and lower regions of the current CU (DiffVarHor); and the absolute difference of the left and right regions of the current CU (DiffVarVer).

(3) Contextual information: Since video sequences are spatially correlated, the current CUs are also partitioned with a similar partition structure to the neighboring CUs. Therefore, additionally incorporated into our method were the average QT (NeighAvgQT) and MTT (NeighAvgMTT) depth levels of adjacent CUs, as well as the number of QT (NeighHigherQT) and MTT (NeighHigherMTT) depth levels of the neighboring CUs of the parent CUs.

(4) Coding information: Since NS (No Split) was evaluated before QT and MT, we could determine the split type based on the current CU coding information. The coding information includes: QP, RD cost (CurrCost), distortion (CurrDistortion), BT RD cost (CostBT), and TT RD cost (CostTT).

Figure 4 displays the 10, 9, 9, 11, and 10 features employed in the NS, QT, HV, HBP, and VBP classifiers, respectively, which were chosen utilizing the feature selector tool. Low importance features were eliminated to effectively decrease the dataset’s dimensionality and reduce the training process’s computational cost.

3.3. Training of Classifier

We determined the feature selection of the classifier, and the key to adequately training the classifier is the tuning of the hyperparameters, as the values of NeighAvgQT and NeighAvgMTT have a significant influence on the classifier’s performance. We could obtain a more robust and general classifier by tuning the optimal choice of hyperparameters. For the choice of hyperparameters, our ultimate goal was to obtain a robust, accurate, and overfitting classifier, rather than to obtain the best accuracy or the lowest loss. Hence, to optimize the hyperparameters of the classifiers, we leveraged the Optuna framework along with the tree-structured parzen estimator (TPE) method.
The hyperparameters for LGBM-based models can generally be classified into four categories, and normally, these categories may overlap. Thus, the efficiency of improving the training speed may reduce the efficiency of improving the accuracy, and the whole process will be very troublesome if tuning is performed completely manually, so we used some automatic tuning tools to judge the general results. Optuna can automatically discover a well-balanced combination of parameters within each category by utilizing a suitable parameter grid.

The main parameters of each classifier after optimization were as follows:

When controlling the tree structure in LGBM, the two primary hyperparameters to be adjusted are max_depth and num_leaves. If you do not control the depth of the tree, it is very easy to cause overfitting. The hyperparameter max_depth can generally be set from 3 to 8. These two hyperparameters also have a mutual influence since the relationship between the two also has an influence on each other. Based on the characteristics of binary trees, it is known that the maximum value of num_leaves should be equal to max_depth 2. As a result, the range of max_depth and num_leaves cannot be separated.

The learning_rate and n_estimators hyperparameters are targeted at enhancing the accuracy of the model. In common methods to improve accuracy, multiple subtrees are generally used and the learning rate is reduced; that is, the optimization of hyperparameters involves finding the optimal values for n_estimators and learning_rate. Two crucial hyperparameters, namely, learning rate and batch size, play a significant role in enhancing the accuracy of the model. Specifically, the parameter n_estimators determines the number of decision trees employed in the algorithm, while learning_rate governs the step size of the gradient descent. To avoid overfitting in LGBM, the learning_rate hyperparameter can be adjusted to control the gradient and improve the speed of learning, usually within the range of 0.01 to 0.3. Usually, more subtrees are used and a lower learning_rate is set, and then the optimal number of iterations is found by early_stopping.

The hyperparameters to control overfitting are bagging_fraction and feature_fraction, both of which take values in the range of 0 to 1. The hyperparameter bagging_fraction refers to the percentage of training samples to train each tree. Before setting this parameter, bagging_freq needs to be set, similar to feature_fraction. The hyperparameter feature_fraction refers to the proportion of features to be randomly sampled during the training of each decision tree, and some features have high gain, which will cause the same feature to be used when splitting each subtree, so that each subtree will be very easy to homogenize. By sampling the features with lower probability and by randomly selecting a subset of features during training, we can prevent the model from repeatedly using the same features, leading to more generalized subtree features.

Figure 4. Feature importance ranking of the top 11 features of the classifiers.

The hyperparameters for LGBM-based models can generally be classified into four categories, and normally, these categories may overlap. Thus, the efficiency of improving the training speed may reduce the efficiency of improving the accuracy, and the whole process will be very troublesome if tuning is performed completely manually, so we used some automatic tuning tools to judge the general results. Optuna can automatically discover a well-balanced combination of parameters within each category by utilizing a suitable parameter grid.

The main parameters of each classifier after optimization were as follows:

When controlling the tree structure in LGBM, the two primary hyperparameters to be adjusted are max_depth and num_leaves. If you do not control the depth of the tree, it is very easy to cause overfitting. The hyperparameter max_depth can generally be set from 3 to 8. These two hyperparameters also have a mutual influence since the relationship between the two also has an influence on each other. Based on the characteristics of binary trees, it is known that the maximum value of num_leaves should be equal to max_depth 2. As a result, the range of max_depth and num_leaves cannot be separated.

The learning_rate and n_estimators hyperparameters are targeted at enhancing the accuracy of the model. In common methods to improve accuracy, multiple subtrees are generally used and the learning rate is reduced; that is, the optimization of hyperparameters involves finding the optimal values for n_estimators and learning_rate. Two crucial hyperparameters, namely, learning rate and batch size, play a significant role in enhancing the accuracy of the model. Specifically, the parameter n_estimators determines the number of decision trees employed in the algorithm, while learning_rate governs the step size of the gradient descent. To avoid overfitting in LGBM, the learning_rate hyperparameter can be adjusted to control the gradient and improve the speed of learning, usually within the range of 0.01 to 0.3. Usually, more subtrees are used and a lower learning_rate is set, and then the optimal number of iterations is found by early_stopping.

The hyperparameters to control overfitting are bagging_fraction and feature_fraction, both of which take values in the range of 0 to 1. The hyperparameter bagging_fraction refers to the percentage of training samples to train each tree. Before setting this parameter, bagging_freq needs to be set, similar to feature_fraction. The hyperparameter feature_fraction refers to the proportion of features to be randomly sampled during the training of each decision tree, and some features have high gain, which will cause the same feature to be used when splitting each subtree, so that each subtree will be very easy to homogenize. By sampling the features with lower probability and by randomly selecting a subset of features during training, we can prevent the model from repeatedly using the same features, leading to more generalized subtree features.
After the hyperparameters were optimized, we needed to consider the accuracy of the classifier for evaluation. The model’s performance could be evaluated using three metrics: the confusion matrix, the classification report, and the AUC_ROC curve. We chose the AUC_ROC curve to visualize the performance metric. The AUC_ROC curve is a performance metric that evaluates the performance of classification problems across multiple threshold settings. The ROC curve is a graphical representation of the performance of a classification model in distinguishing between different categories based on their probabilities, and the AUC measures the degree or quality of separability. The ROC curve provides valuable insight into the model’s ability to differentiate between categories.

From Figure 5 we could observe that our classifier performed better on the test set. This effectively solved the classifier overfitting problem. Based on the AUC_ROC curve, there was a strong correlation between the predicted and test values, indicating a good model fit. These results showed that the classifier could provide high-performance CU segmentation-type prediction.

**Figure 5.** AUC_ROC curve for each classifier. (a) NS_Classifier. (b) QT_Classifier. (c) HV_Classifier. (d) HBP_Classifier. (e) VBP_Classifier.

The performance of each classifier implemented in VTM is presented in Figure 6, with threshold values ranging from 0.3 to 0.7. The lower the threshold value set, the more segments were skipped and the greater the impact on coding time and BDBR; on the contrary, when the threshold value increased, the more segments were evaluated and the greater the impact on coding time reduction and the BDBR was smaller.
3.4. Threshold Decision

In our proposed decision framework, we innovatively built multi-threshold decision schemes to provide better flexibility for QT division ($TH_{QT}$), Hor/Ver ($TH_{HV}$), and BT/TT ($TH_{BT}$) division, respectively. We provided more flexibility than the traditional threshold decision scheme. The use of multiple threshold decisions made our decision framework more adaptable and configurable, as we provided more threshold point schemes that could be changed according to the user’s needs for threshold point changes. The threshold point changes were mainly obtained by changing the values of $TH_{QT}$, $TH_{HV}$, and $TH_{BT}$. The multi-threshold decision framework allowed defining different combinations of thresholds according to different values, allowing for a maximum desired trade-off between complexity and RD performance. According to the experimental results, increasing the threshold value led to a decrease in the number of skipped segmentation types. Thus, the encoder would compute more segmentation types and the coding efficiency was improved. A decrease in the threshold value increased the number of skipped segmentation types and the encoding time was reduced. We performed many experimental comparisons and finally selected the optimal four threshold points to configure the proposed framework. These threshold points were proven by extensive experimental evaluations with good performance. The high flexibility of our proposed multi-threshold decision scheme also enabled more combinations of thresholds to select the optimal threshold points according to the coding requirements. According to the multi-threshold decision scheme, we finally selected four threshold points, as shown in Table 2.

Table 2. Multi-threshold decision scheme values for the four threshold points.

<table>
<thead>
<tr>
<th>Threshold Point</th>
<th>$TH_{QT}$</th>
<th>$TH_{HV}$</th>
<th>$TH_{BT}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.65</td>
<td>0.7</td>
<td>0.65</td>
</tr>
<tr>
<td>B</td>
<td>0.6</td>
<td>0.65</td>
<td>0.6</td>
</tr>
<tr>
<td>C</td>
<td>0.55</td>
<td>0.6</td>
<td>0.55</td>
</tr>
<tr>
<td>D</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

3.5. Framing Analysis

The framework shown in Figure 7 describes the process of training and implementing the LGBM classifier in a VTM encoder. A specific collection of video sequences was selected for feature extraction and subsequent classifier training. We efficiently improved the VTM encoder to collect several statistics that contained relevant information for the CU partitioning decision. Additionally, datasets were generated for each partition type based on these collected statistics. This dataset contained relevant features extracted from
the encoded video sequences as well as encoder properties and segmentation decisions. In the preprocessing stage, the dataset was balanced and the most critical features were selected. The selected features were used as input to train the classifier, a stage that involved hyperparameter optimization and separate training of the classifier. In the final step, coding efficiency and time savings were evaluated using a modified VTM encoder that integrated an LGBM classifier to determine the QTMT partition without employing full rate-distortion optimization (RDO).

![Diagram](Figure 7. Framework for training LGBM classifier for CU partition decisions and evaluating the performance of VTM encoder.

4. Experimental Results

This chapter provides a comparative analysis of our proposed fast CU partitioning decision method in relation to other recent works that deal with similar topics. By doing so, we aimed to provide a comprehensive evaluation of the effectiveness and performance of our proposed method in comparison to existing approaches. The experimental results demonstrate the robustness of our proposed method, which was highly desirable. Section 4.1 presents the specific configuration information of the experiments, Section 4.2 shows the details of our network model training, and Section 4.3 provides a detailed analysis of the performance of our method in comparison to other methods.

4.1. Configuration and Setup

Our experimental scenarios were all performed in VVC reference software VTM10.0; the default configuration file encoder_intra_main.cfg was utilized for implementing the full internal main configuration. The four QPs were set to 22, 27, 32, and 37 for encoding, respectively. For the evaluation of the performance of the fast CU partitioning decision method, we used the same criteria as the approach described in [28,32,33]. The Bjøntegaard delta bit-rate (BDBR) and time saving (ΔT) metrics were employed to evaluate the rate-distortion (RD) performance. The video sequences used in this study, namely A1, A2, B, C, D, and E, were of varying resolutions ranging from 3840 × 2160 to 416 × 240 pixels. The experiments were all run on a computer with an Intel (R) Core (TM) i7-11800H CPU and 16 GB RAM. The NVIDIA GeForce RTX 3060 GPU was used for the graphics card...
to accelerate the training process. The rate of time saving in coding (ΔT) is determined through the following calculation:

\[ ΔT = \frac{T_{VTM} - T_{pro}}{T_{VTM}} \]  

(5)

where \( T_{VTM} \) refers to the encoding time of the original VTM10.0 encoder in this context, and \( T_{pro} \) denotes the actual encoding time of the method proposed in this paper. The experimental configuration is shown in Table 3.

### Table 3. Experimental configuration.

<table>
<thead>
<tr>
<th>Hardware</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel (R) Core (TM) i7-11800 H</td>
</tr>
<tr>
<td>RAM</td>
<td>16 GB</td>
</tr>
<tr>
<td>OS</td>
<td>Microsoft Windows 10 64 bits</td>
</tr>
<tr>
<td>GPU</td>
<td>NVIDIA GeForce RTX 3060</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference software</td>
<td>VTM 10.0</td>
</tr>
<tr>
<td>Configuration</td>
<td>All intra</td>
</tr>
<tr>
<td>QP</td>
<td>22, 27, 32, 37</td>
</tr>
</tbody>
</table>

### 4.2. Training Details

We conducted a comprehensive evaluation of our algorithm’s performance by analyzing 22 video sequences categorized from classes A1 to E. To achieve this, we utilized three distinct evaluation schemes proposed in [28,32,33]. The video sequences used in the evaluation comprised classes A1 and A2, which were newly introduced ultra-high definition (UHD) video sequences with 10-bit depth, and class B video sequences introduced by the HEVC standard with 8-bit depth. The algorithm’s performance was evaluated using the Bjøntegaard delta bit-rate (BDBR) and time saving (ΔT) metrics.

Our multi-threshold decision method consisted of four threshold points, A, B, C, and D. The combination of multiple threshold points allowed our fast CU division decision method to adapt to different application scenarios, the more prominent being the combination of threshold points C and D. The proposed approach struck a balance between the coding efficiency and computational complexity, resulting in a favorable trade-off.

### 4.3. Performance Evaluation of the Framework

For the evaluation of the performance of the fast CU partitioning decision framework, we used the same criteria as the approach described in [28,32,33]. The Bjøntegaard delta bit-rate (BDBR) and time saving (ΔT) metrics were employed to assess the rate-distortion (RD) performance. We chose the three schemes described in [28,32,33] for our performance comparison because their experimental setups were closest to our configuration and they were the most representative experimental schemes available. Our experiments also considered configurations for four threshold points, and for simplicity, threshold points B and C were finally chosen for the configuration. A detailed comparison of the results of BDBR and average complexity with the three schemes described in [28,32,33] is given in Table 4. Since some of the compared schemes did not have specific information on class B data, we eliminated class B from the classification as a reference in order to ensure a fairer comparison. Comparing our work with [28,32], we could see that the BDBR of threshold point B was much lower than that of [28,32] (1.07% < 1.27% < 2.52%), but ΔT was much higher (47.93% > 47.03% > 24.83%). The threshold point B had limited complexity reduction, but the BDBR was only 1.07% on average, which was an excellent performance. Comparing our work with [33], we could see that the BDBR of threshold point C was much lower than that of [33] (1.57% < 1.77%), but ΔT was indeed much higher (54.27% > 51.34%). The experiments demonstrated that our work with threshold points B and C achieved better
\( \Delta T \) and lower BDBR values. Especially in high resolution, our method performed very well. Compared with [33], the complexity reduction was about the same and the loss of BDBR was much lower. Therefore, our proposed fast CU division decision method could guarantee better coding efficiency while reducing the complexity.

Table 4. Comparison of the encoding performance of the proposed algorithm with references [28,32,33].

<table>
<thead>
<tr>
<th>Class</th>
<th>Sequence</th>
<th>ZHAO [28]</th>
<th>PAN [32]</th>
<th>ZHANG [33]</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BDBR</td>
<td>( \Delta T )</td>
<td>BDBR</td>
<td>( \Delta T )</td>
<td>BDBR</td>
</tr>
<tr>
<td>A1</td>
<td>Tango2</td>
<td>1.61</td>
<td>49.67</td>
<td>3.68</td>
<td>34.05</td>
<td>2.41</td>
</tr>
<tr>
<td></td>
<td>FoodMarket4</td>
<td>1.53</td>
<td>50.08</td>
<td>1.59</td>
<td>42.90</td>
<td>2.10</td>
</tr>
<tr>
<td></td>
<td>Campfire</td>
<td>1.55</td>
<td>52.32</td>
<td>2.80</td>
<td>30.08</td>
<td>2.21</td>
</tr>
<tr>
<td>A2</td>
<td>CatRobot</td>
<td>1.85</td>
<td>47.75</td>
<td>5.59</td>
<td>30.62</td>
<td>2.85</td>
</tr>
<tr>
<td></td>
<td>DaylightRoad2</td>
<td>1.45</td>
<td>51.57</td>
<td>4.43</td>
<td>29.20</td>
<td>1.93</td>
</tr>
<tr>
<td></td>
<td>ParkRunning3</td>
<td>1.42</td>
<td>47.62</td>
<td>1.61</td>
<td>21.30</td>
<td>0.85</td>
</tr>
<tr>
<td>B</td>
<td>MarketPlace</td>
<td>/</td>
<td>/</td>
<td>3.22</td>
<td>36.47</td>
<td>1.49</td>
</tr>
<tr>
<td></td>
<td>RitualDance</td>
<td>/</td>
<td>/</td>
<td>2.97</td>
<td>31.23</td>
<td>2.30</td>
</tr>
<tr>
<td></td>
<td>Cactus</td>
<td>1.28</td>
<td>44.03</td>
<td>5.20</td>
<td>25.42</td>
<td>1.93</td>
</tr>
<tr>
<td></td>
<td>BasketballDrive</td>
<td>1.58</td>
<td>43.31</td>
<td>2.96</td>
<td>32.39</td>
<td>2.07</td>
</tr>
<tr>
<td></td>
<td>BQTerrace</td>
<td>0.84</td>
<td>47.58</td>
<td>0.98</td>
<td>13.80</td>
<td>1.52</td>
</tr>
<tr>
<td>C</td>
<td>BasketballDrill</td>
<td>1.27</td>
<td>44.15</td>
<td>1.59</td>
<td>24.38</td>
<td>2.28</td>
</tr>
<tr>
<td></td>
<td>BQMall</td>
<td>1.11</td>
<td>47.37</td>
<td>2.35</td>
<td>22.41</td>
<td>1.58</td>
</tr>
<tr>
<td></td>
<td>PartyScene</td>
<td>0.78</td>
<td>46.37</td>
<td>1.84</td>
<td>14.94</td>
<td>0.76</td>
</tr>
<tr>
<td></td>
<td>RaceHorses</td>
<td>0.84</td>
<td>47.58</td>
<td>2.23</td>
<td>22.55</td>
<td>1.02</td>
</tr>
<tr>
<td>D</td>
<td>BasketballPass</td>
<td>1.34</td>
<td>38.31</td>
<td>1.56</td>
<td>21.18</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>BQSquare</td>
<td>0.82</td>
<td>46.65</td>
<td>0.84</td>
<td>9.69</td>
<td>0.41</td>
</tr>
<tr>
<td></td>
<td>BlowingBubbles</td>
<td>0.93</td>
<td>44.29</td>
<td>2.29</td>
<td>16.97</td>
<td>0.419</td>
</tr>
<tr>
<td></td>
<td>RaceHorse2</td>
<td>1.12</td>
<td>39.46</td>
<td>2.24</td>
<td>20.33</td>
<td>0.66</td>
</tr>
<tr>
<td>E</td>
<td>FourPeople</td>
<td>1.35</td>
<td>48.15</td>
<td>1.76</td>
<td>25.26</td>
<td>2.69</td>
</tr>
<tr>
<td></td>
<td>Johnny</td>
<td>1.67</td>
<td>51.60</td>
<td>1.69</td>
<td>24.92</td>
<td>3.32</td>
</tr>
<tr>
<td></td>
<td>KristenAndSara</td>
<td>1.57</td>
<td>48.48</td>
<td>2.11</td>
<td>26.21</td>
<td>2.42</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>1.27</td>
<td>47.03</td>
<td>2.52</td>
<td>24.83</td>
<td>1.77</td>
</tr>
</tbody>
</table>

The complexity and coding efficiency (BDBR) were compared above, respectively. Next, we also compared the number of CU division blocks to verify the effectiveness of the method. Figure 8 shows the comparison of the number of blocks processed. By comparing, we found that our method processed 18.86% of the blocks in the standard VTM, which showed the effectiveness of our algorithm. In high resolution, our method saved a very high number of encoding blocks as well, which showed another aspect of our better performance in high-resolution sequences.

Figure 8. Comparison of the number of blocks processed.

Figure 9 also presents the increases in \( \Delta T \) and BDBR for the different threshold points we considered in comparison to the state-of-the-art [28,32,33] solutions. The four thresh-
old points of our multi-threshold point decision scheme are shown in Figure 9 as A, B, C, and D. From Figure 8, we could clearly see that our solution was better than the solutions in [28,32,33], and our solution achieved a better balance between BDBR and $\Delta T$. Additionally, this figure proved that our solution had higher flexibility.

![Figure 9. Performance comparison of BDBR and $\Delta T$ at different threshold points.](image)

**Figure 9.** Performance comparison of BDBR and $\Delta T$ at different threshold points.

**5. Conclusions**

The primary objective of this paper was to reduce the coding complexity of VVC by presenting a fast CU division decision method that employed a LGBM classifier. Firstly, the new features of VVC were statistically analyzed to explore more representative features and different features were extracted for different classifiers for training, so that classifiers with high accuracy and simplicity could be trained. Secondly, a new fast CU decision framework was designed for the coding characteristics of VVC. Predicting in advance whether to divide the CU, whether to divide by QT, and whether to divide horizontally or vertically could reduce the huge coding complexity. To solve the multi-classification problem, it was converted into multiple binary classification tasks. The framework not only had high decision accuracy but also good ability to reduce the complexity. Subsequently, a multi-threshold decision scheme comprising four threshold points was presented, which achieved a favorable trade-off between time savings and coding efficiency. Based on the experimental results, our method effectively reduced the coding time by 47.93% to 54.27%; however, the BDBR was only improved by 1.07–1.57%. The method proposed exhibited outstanding performance in terms of both computational complexity and compression quality.

**Author Contributions:** The conceptualization of the study was performed by Y.W. and Y.L., while J.Z. contributed to the methodology. Y.L. was responsible for software development, and the validation process involved Y.W., J.Z., Q.Z. and Y.L. Y.L. conducted the formal analysis, and J.Z. was responsible for the investigation. Q.Z. provided the necessary resources for the study, data curation was performed by Q.Z., while Y.L. was responsible for the original draft preparation. Y.W. conducted the writing review and editing and also performed the visualization. Q.Z. supervised the project administration, and Y.W. was responsible for funding acquisition. 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 (grant nos. 61771432, and 61302118), the Basic Research Projects of Education Department of Henan (grant nos. 21zx003, 23A520039 and 20A880004), the Key projects Natural Science Foundation of Henan (grant no. 232300421150), the Scientific and Technological Project of Henan Province (grant no. 232102211014), and the Postgraduate Education Reform and Quality Improvement Project of Henan Province (grant nos. YJS2021KC12, YJS2023C08, and YJS2022AL034).

**Data Availability Statement:** Not applicable.
Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

- VVC: Versatile Video Coding
- HEVC: High-Efficiency Video Coding
- CU: Coding Unit
- CTU: Coding Tree Unit
- MTT: Multi-Type Tree
- QTMT: QuadTree with Nested Multi-type Tree
- RDO: Rate Distortion Optimization
- QT: QuadTree
- BTH: Horizontal Binary Tree
- BTV: Vertical Binary Tree
- TTH: Horizontal Trinomial Tree
- TTV: Vertical Trinomial Tree
- QP: Quantization Parameter
- TS: Time Saving
- BDBR: Bjøntegaard Delta Bit-Rate
- VTM: VVC Test Model
- MPEG: Moving Picture Experts Group
- VCEG: Video Coding Experts Group
- BV: Vertical Binary Tree Partition
- BH: Horizontal Binary Tree Splitting
- TV: Vertical Ternary Tree Splitting
- TH: Horizontal Ternary Tree Splitting
- AMC: Affine Motion Compensation
- MTS: Multiple Transform Selection
- LFNST: Low-Frequency Non-Separable Transform
- HM: HEVC Test Model
- NS: No Splitting

References


24. Zhang, Q.; Zhao, Y.; Jiang, B.; Huang, L.; Wei, T. Fast CU partition decision method based on texture characteristics for H. 266/VVC. *IEEE Access* 2020, 8, 203516–203524. [CrossRef]


Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.