Article

SVM-Based Fast CU Partition Decision Algorithm for VVC Intra Coding

Jinchao Zhao, Aobo Wu and Qiuwen Zhang*

College of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China; 2004086@zzuli.edu.cn (J.Z.); 332007040541@email.zzuli.edu.cn (A.W.)
* Correspondence: 2012032@zzuli.edu.cn; Tel.: +86-371-8660-9559

Abstract: As a new coding standard, Versatile Video Coding (VVC) introduces the quad-tree plus multi-type tree (QTMT) partition structure, which significantly improves coding efficiency compared to High-Efficiency Video Coding (HEVC). The QTMT partition structure further enhances the flexibility of coding unit (CU) partitioning and improves the efficiency of VVC encoding high-resolution video, but introduces an unacceptable coding complexity at the same time. This paper proposes an SVM-based fast CU partition decision algorithm to reduce the coding complexity for VVC. First, the proportion of split modes with different CU sizes is analyzed to explore a method to effectively reduce coding complexity. Then, more reliable correlation features are selected based on the maximum ratio of the standard deviation (SD) and the edge point ratio (EPR) in sub-CUs. Finally, two SVM models are designed and trained using the selected features to provide guidance for deciding whether to divide and the direction of partition. The simulation results indicate that the proposed algorithm can save 54.05% coding time on average with 1.54% BDBR increases compared with VTM7.0.

Keywords: VVC; QTMT partition structure; SVM based; CU partition decision

1. Introduction

In recent years, ultra-high-definition (UHD) video has been widely used, and while video quality has been further improved, the large amount of video data has brought inconvenience in transmission and storage. The Joint Video Experts Team (JVET) has developed a new generation of video coding standards, Versatile Video Coding (VVC) [1], which significantly improves coding efficiency compared with High-Efficiency Video Coding (HEVC) [2]. According to established work, comparing the VVC test model (VTM) [3] with the HEVC test model (HM) [4], VTM can improve coding efficiency by approximately 40% while maintaining almost identical quality, but at the cost of 18-fold the coding complexity of HM for intra coding.

VVC introduces new coding techniques, among which the use of a quad-tree plus multi-type tree (QTMT) significantly improves coding efficiency. Different from the quad-tree (QT) partition structure in HEVC, VVC introduces a multi-type tree (MTT) split, which allows the partition shape of CU to be not only square but also rectangular. The more flexible partition structure is more effective for encoding high-resolution video. However, the QTMT structure-based coding unit (CU) partitioning process takes up more than 97% of the coding time [5], which implies a very high level of complexity.

In VVC, the default coding tree unit (CTU) size is 128 × 128, and the minimum CU size is 4 × 4. Figure 1a shows the multiple partition modes of QTMT structure, including No Split (NS), QT, vertical BT (BTV), horizontal BT (BTH), vertical TT (TTV), and horizontal TT (TTH). Moreover, some parameters in the configuration file of VTM have limitations on the CU partition. In the default configuration, MaxCUWidth and MaxCUHeight are both 64; therefore, CTU will be divided by QT. MaxBlSize and MaxTlSize are both 32 × 32; therefore, 64 × 64 CU can only be divided by QT. MaxDepth is set to 6, and MaxMttDepth is
set to 4. When the depth or the MTT depth of CU reaches the maximum value, the CU is no longer divided. In addition, the combination of different partition modes may lead to the same partition result. To limit the redundant partitions of CU, VVC prohibits BT splitting in the same direction in the middle part after TT splitting. Since MinSize is specified in the QTMT partition structure and asymmetric partitioning is allowed, CU partitioning will be limited by the CU size. TT splitting is not allowed when the width of CU is 8, and BTV split is not allowed when it is 4. Similarly, TTH split is not allowed when the height of CU is 8, and BTH split is not allowed when it is 4. Figure 1b shows a CTU divided into multiple CUs with the QTMT partition structure, and different colors represent different partition depths.

![CTU partition](image)

**Figure 1.** CTU partition with a QTMT partition structure: (a) CU split modes; (b) example of CTU partition.

Same as the method for selecting the best partition result in HEVC, VVC recursively checks the RD cost of all possible CUs by brute-force RDO search and selects the combination with the minimum RD cost as the optimal partition result. VVC introduced the MTT partition structure, resulting in an average of 5781 CUs to be checked to encode a CTU [6], far exceeding the 85 CUs ($1 + 4 + 4^2 + 4^3$) that need to be checked in HEVC. The result shows that the MTT structure has become the most time-consuming part of VVC, with an average time consumption of more than 90% [7]. Therefore, the implementation of fast CU partitioning can reduce the encoding complexity of VVC.

In this paper, we propose an SVM-based fast CU partition decision algorithm to accelerate CU partitioning. First, the standard deviation (SD) and the edge point ratio (EPR) of CU are extracted to calculate the ratio of the SD and the EPR of sub-CUs. Then, the effective features are selected by calculating the F-score value to train the SVM model offline. Finally, the SVM models are integrated into VTM7.0 to predict whether the CU is divided and the direction of partition.

The rest of this paper is summarized as follows. Section 2 reviews recent work on reducing the complexity of VVC and HEVC. Section 3 is a statistical analysis of the proportion of the partition modes of CU, then proposes a fast CU partition algorithm and introduces the features of high relevance. The experimental results of the proposed algorithm are provided and compared with state-of-the-art algorithms in Section 4. Section 5 summarizes this paper.

### 2. Related Works

Recently, many researchers are working on methods to accelerate CU partitioning in H.266/VVC and H.265/HEVC.

#### 2.1. Methods for HEVC

HEVC improves coding efficiency through a highly flexible quad-tree partition structure, but this structure takes up much encoding time. To save the coding time of HEVC,
references [8–11] used traditional methods to accelerate CU partitioning. The edge complexity of CU and sub-CUs is utilized in [8] to classify CU as divided, undivided and pending. The texture complexity of the CU and the features of neighboring CUs are used in [9] to predict whether the CU is divided. In [10], the authors proposed a method based on the Bayesian decision rule, which is including early partition and pruning. Zhang et al. [11] adopted an adaptive processing method to divide the CU reasonably by adjusting the entropy and texture contrast. Many algorithms to accelerate CU partitioning have been proposed because of the wide application of machine learning and deep learning. In [12,13], the authors utilized CU texture characteristics and adjacent CU depth information to train a decision tree model to predict CU partition. Support Vector Machine (SVM) was used to classify CU partition or non-partition in HEVC in [14–18]. Specifically, Sun et al. [14] improved the Canny edge algorithm to extract the edge points in CU and used the EPR as a feature to train the SVM classifier. In [16], the authors introduced two fast partition models of offline training SVM and online training Bayesian probability, and the feature used to train the model is the gray-level co-occurrence matrix (GLCM). Shen et al. [17] used the RD loss caused by misclassification as a weight to reduce the influence of outliers. The convolutional neural network (CNN) is widely utilized in CU splitting in HEVC. Chen et al. [19] used shallow asymmetric-kernel CNN (AK-CNN) to extract local texture features of CU. In [20], the authors proposed a CNN structure based on depth range decision for CTU classification and depth range limitation. Xu et al. [21] proposed an early termination hierarchical CNN (ETH-CNN) predict the partition result of the CTU.

2.2. Methods for VVC

VVC introduces a QTMT partition structure, which further improves flexibility, and increases encoding complexity. Many experts still use traditional methods to accelerate CU partitioning in VVC. Fan et al. [22] proposed a fast CU partition decision based on variance, gradient, and the variance of sub-CUs variance. This method only predicts the partition mode of the 32 × 32 CU, so the effect of reducing complexity when encoding low-resolution video is limited. In [23], the authors proposed an algorithm that uses the RD cost of neighboring CUs to skip TT split. Cui et al. [24] determined the CU partition mode in advance based on the gradients in the four directions of CU and sub-CU. Many approaches [7,25–29] used machine learning algorithms to accelerate CU partitioning. Thomas Amestoy et al. [25] decided on the partition mode of CU based on the random forest classifier, and the risk interval of the classifier decision is introduced to realize the tradeoff between coding complexity reduction and encoding loss. Zhang et al. [26] utilized GLCM in four directions to train the RFC model to predict CU partitioning modes. Yang et al. [27] extracted gradient information, partial texture information, and context information training decision trees. Wang et al. [28] combined two decision trees to determine CU split mode with a quadtree plus binary tree (QTBT) partition structure. Chen et al. [7] extracted entropy, texture contrast, and Haar wavelet efficiency to train SVM models online to predict the CU partition direction, since this method only determines the partition direction, it has a limited reduction in coding complexity when encoding high-resolution video. Wu et al. [29] extracted the difference in variance and gradient of sub-CUs to train two SVM models offline, and the SVM models decide whether to partition the CU and the direction of CU partitioning. CNN can also be applied to VVC. Li et al. [6] proposed a multi-stage exit CNN (MSE-CNN) model with an early exit mechanism to skip the redundancy check during CU partitioning. Zhao et al. [30] classified CU into complex and homogeneous CU and utilized an adaptive CNN structure to classify CU. CNN is utilized to analyze the texture in CU with a size of 64 × 64 in [31], and the partition probability of CU is derived by predicting the vector probability of the 4 × 4 boundaries in this CU. The lightweight neural network (LNN) model was used in [32] to predict whether to skip TT split based on partition depth and shape ratio of CU.
3. Proposed Algorithm

In this section, we make a statistical analysis of the proportion of different sizes of CU partition type at first. Then, we propose an SVM-based fast CU partition decision algorithm, which predicts whether CU is partitioned and the direction of CU partition, and extracts the SD and the EPR of CU, the ratio of the SD and the EPR of sub-CUs. Finally, we select the effective features by calculating the F-score value to train the SVM model offline.

3.1. Statistical Analysis

To reduce the complexity of CU partition, we conduct statistics on the partition modes of CU and we have counted the CU partition modes ratio of six video sequences in total. These video sequences have different resolutions and different video contents, including “FoodMarket4” with global motion in class A1, “DaylightRoad2” with translation motion in class A2, “Cactus” with complex and irregular motion in class B, and “PartyScene” with complex texture in class C, “BlowingBubbles” with smooth textures in low-resolution class D and “Johnny” with monotonous backgrounds and stationary objects in class E. All video sequences were performed on VTM7.0 and encoded using All Intra (AI) configuration under four quantization parameter (QP) value {22, 27, 32, 37}. Figure 2 displays the ratio of the partition type of CUs with sizes ranging from 64 × 64 to 8 × 8. By analyzing the proportion of CU partition modes, we obtain the following information.

1. NS accounts for more than 20% of the partition mode in all CU sizes, and in 32 × 8, 8 × 32, 16 × 8, and 8 × 16 CUs, the proportion of NS is more than 40%. In addition, as the CU size decreases, the proportion of NS gradually increases. By predicting whether CU is partitioned early, VVC can skip the RDO process of QT and MTT partition, thereby reducing complexity.

2. MTT split accounts for a relatively high proportion, especially for 32 × 16 and 16 × 32 CUs, which account for more than 60%, and the proportions of horizontal partition and vertical partition are almost the same. By predicting the direction of partition early, VVC can skip the MTT split in other directions, and reduce the cross calculation in the RDO process, thereby reducing coding complexity.

3. The proportion of QT split is relatively low, the proportion of QT split in 16 × 16 CU does not exceed 10%. Therefore, the coding complexity that can be reduced by predicting QT split is limited, and unnecessary coding performance loss increases.

According to the above analysis, this paper reduces the coding complexity by predicting whether the CU is partitioned and the partition direction.
3.2. Fast CU Partition Decision Algorithm

We model the CU partitioning process as a combined multiple binary classification problem to construct two SVM classifiers, namely S vs. NS and Hor vs. Ver. The proposed SVM classifiers are trained for different CU sizes. Specifically, $64 \times 64$ CU can only be divided by QT split or NS, the complexity of predicting its partitioning mode is limited, and the loss of coding performance may increase. In addition, CUs with a side length of 4 do not need to predict the partition direction, and the RDO process of CUs with small size does not occupy much coding time, the partition mode of such CUs is not predicted in advance. Therefore, we train the classifier models for CUs with sizes of $32 \times 32$, $32 \times 16$, $16 \times 32$, $16 \times 16$, $32 \times 8$, and $8 \times 32$, respectively.

The flowchart of the proposed algorithm is shown in Figure 3. If the CU size matches the predicted CU size, we extract the features of this CU, and use two SVM models to predict whether the CU is partitioned and the partition direction. Otherwise, the anchoring algorithm of VTM7.0 is used to divide the CU. The $32 \times 8$ CU can only BT split when partitioning in the horizontal direction, so we use the side length of the CU as the judgment condition. If the width of the CU is 8, the TT split in the vertical direction will be skipped, and if the height of the CU is 8, the TT split in the horizontal direction will be skipped. In addition, to limit the redundant partition of CU, VVC prohibits BT split in the same direction in the middle part after TT split, so the middle part after TT split is also divided by the anchor algorithm of VTM7.0.

![Figure 3. The flowchart of the proposed algorithm.](image)

In this paper, we train SVM with the offline training mode. The offline training mode can use sufficient training samples to improve the prediction accuracy and avoid the loss of
coding performance. The SVM training process is shown in Figure 4. We extract the features from the xCompressCU function in VTM, which is a recursive function that performs the CTU partitioning process from top to bottom. The labels in the training set represent whether the CU is split and the direction of the partition and these two types of labels are obtained by the code stream analysis tool (DecoderAnalyserApp) in VTM7.0.

3.3. Feature Analysis and Selection

The selection of features is crucial for training SVM classifiers, as highly correlated features can help reduce training time and improve prediction performance. By analyzing the results of CU partition, we found that CUs with complex textures are usually further split into smaller CUs for encoding, while CUs with smooth textures tend to NS. Additionally, the difference in texture complexity between adjacent blocks determines whether two blocks belong to the same CU, Therefore, we predict the partition mode of the CU by comparing the texture characteristics of the sub-CUs. Figure 5 shows the relationship between CU partition and sub-CU texture characteristics in VVC. As shown in Figure 5a, if the texture difference between two adjacent sub-CUs is large, CU tends to be divided, and vice versa. As shown in Figure 5b,c, CU tends to be horizontally divided if the difference between the texture of the two sub-CUs divided by BTH split is greater than that of the two sub-CUs divided by BTV split, and vice versa. As shown in Figure 5d, if the difference in the texture of the three sub-CUs divided by TTH split is greater than that of the three sub-CUs divided by TTV split, CU tends to be horizontally partitioned, and vice versa. According to the above analysis, we choose the following characteristics:

1. The SD reflects the texture complexity of CU by calculating the dispersion between the luminance values in the CU and the luminance mean. A large SD indicates that the CU has a complex texture and tends to be divided into smaller CUs, and a small SD indicates that the CU has a smooth texture and tends not to be divided. The SD is expressed as:

\[
SD = \sqrt{\frac{\sum_{i=0}^{H-1} \sum_{j=0}^{W-1} (f(i,j) - \bar{f})^2}{W \times H}}
\]  

Figure 4. The flowchart of training SVM offline.

\[
SD = \sqrt{\frac{\sum_{i=0}^{H-1} \sum_{j=0}^{W-1} (f(i,j) - \bar{f})^2}{W \times H}}
\]  

(1)
where \( W \) and \( H \) represent the width and height of the CU, respectively, \( f(i,j) \) is the luminance value with coordinates \((i,j)\) in CU. \( \overline{f} \) is the luminance mean of CU. \( \overline{f} \) is expressed as:

\[
\overline{f} = \frac{\sum_{i=0}^{H-1} \sum_{j=0}^{W-1} f(i,j)}{W \times H}
\]  

(2)

2. The EPR reflects the texture complexity of CU by calculating the proportion of edge points in the CU. A high EPR means that the CU has a complex texture. The most common method to extract edges is the Sobel edge detection algorithm. Firstly, the Sobel operators in the four directions are convolved with the luminance matrix of CU to obtain the gradient in each direction. The purpose of using four Sobel operators is to improve the accuracy of computing edges and anti-noise ability and to detect weak edge information accurately. The gradient is expressed as:

\[
G_0^\circ = \begin{pmatrix}
-1 & -2 & -1 \\
0 & 0 & 0 \\
1 & 2 & 1
\end{pmatrix} \ast A \\
G_{45}^\circ = \begin{pmatrix}
-2 & -1 & 0 \\
-1 & 0 & 1 \\
0 & 1 & 2
\end{pmatrix} \ast A \\
G_{90}^\circ = \begin{pmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1
\end{pmatrix} \ast A \\
G_{135}^\circ = \begin{pmatrix}
0 & -1 & -2 \\
1 & 0 & -1 \\
2 & 1 & 0
\end{pmatrix} \ast A
\]  

(3)

where \( A \) is the luminance matrix of CU. Then, the absolute values of the gradients in four directions are added to obtain the edge matrix (EM). The expression of EM is:

\[
EM = (|G_0^\circ| + |G_{45}^\circ| + |G_{90}^\circ| + |G_{135}^\circ|)
\]  

(4)

Figure 5. Example of CU partition related to CU texture: (a) example of QT split related to textures; (b) example of BTV split related to textures; (c) example of BTH split related to textures; (d) example of TT split related to textures.

If the value in EM exceeds the threshold, it is defined as an edge point. We obtain the EPR by calculating the ratio of the number of edge points to the number of elements in EM. The expression of the EPR is:

\[
EPR = \frac{ep_{num}}{EM_{num}}
\]  

(5)

where \( ep_{num} \) is the number of edge points in the CU, and \( EM_{num} \) is the number of elements in EM.

We set different edge point thresholds and select the threshold with the highest correlation with the CU partition by calculating the F-score value. By calculating the F-score for different thresholds, we set 150 and 600 as the thresholds for 8-bit and 10-bit video sequences, respectively. Figure 6 shows the result of the Sobel edge detection algorithm to detect the “BasketballPass” video sequence, and we found that CU with more edge points is divided into smaller CUs.
Figure 6. Edge detection result: (a) edge detection result of a frame in the “BasketballPass” sequence; (b) edge detection result with CU partition.

3. The gradient ratio (GR) can represent the difference between the horizontal gradient and the vertical gradient of the CU. It is expressed as:

\[ GR = \frac{G_0^\circ}{G_90^\circ} \]  \hspace{1cm} (6)

4. The \( R_q \) is the maximum difference between features of the sub-CUs divided by QT, and the features include the SD and the EPR, a large \( R_q \) indicates that CU tends to be divided. It is expressed as:

\[ R_q = \text{Max}\{f_{tl}, f_{tr}, f_{bl}, f_{br}\} - \text{Min}\{f_{tl}, f_{tr}, f_{bl}, f_{br}\} \]  \hspace{1cm} (7)

where \( f_{tl}, f_{tr}, f_{bl}, \) and \( f_{br} \) are the sub-CUs of top-left, top-right, bottom-left, and bottom-right, respectively.

5. The \( R_{dirB} \) is the ratio of the difference of features in the horizontal and vertical directions in BT split, and the features include the SD and the EPR. It is expressed as:

\[ R_{dirB} = \frac{|f_t - f_b|}{|f_l - f_r|} \]  \hspace{1cm} (8)

6. The \( R_{dirT} \) is the ratio of the difference of features in the horizontal and vertical directions in TT split, and the features include the SD and the EPR. It is expressed as:

\[ R_{dirT} = \text{Max}\{f_{tt}, f_{mbh}, f_{bt}\} - \text{Min}\{f_{tt}, f_{mbh}, f_{bt}\} \]
\[ \text{Max}\{f_{lt}, f_{mv}, f_{rt}\} - \text{Min}\{f_{lt}, f_{mv}, f_{rt}\} \]  \hspace{1cm} (9)

where \( f_{tt}, f_{mbh}, \) and \( f_{bt} \) are the feature values of the sub-CUs divided by TTH split, \( f_{lt}, f_{mv}, \) and \( f_{rt} \) are the feature values of the sub-CUs divided by TTV split. Since CU of size 32 \( \times \) 8 cannot be TTH split and CU of size 8 \( \times \) 32 cannot be TTV split, we substitute the difference of sub-CUs divided by BT.

The F-score based on the wrapper method is used to evaluate the correlation of features, and features with high correlation are used to train the SVM model. Given a sample set \( x_i, i \in [1, m] \), each sample contains \( n \) features and a label that is divided into positive and negative. The F-score of the \( j \)th feature is calculated as:

\[ F_j = \frac{1}{p-1} \sum_{i=1}^{p} (x^{(+)}_{ij} - \overline{x}^{(+)}_{j})^2 + \frac{1}{N-1} \sum_{i=1}^{N} (x^{(-)}_{ij} - \overline{x}^{(-)}_{j})^2 \]  \hspace{1cm} (10)

\[ = \frac{1}{N-1} \sum_{i=1}^{N} (x^{(+)}_{ij} - \overline{x}^{(+)}_{j})^2 + \frac{1}{N-1} \sum_{i=1}^{N} (x^{(-)}_{ij} - \overline{x}^{(-)}_{j})^2 \]
where $\bar{x}_j$ represents the mean of the $j$th feature, $\bar{x}_j^{(+)}$ and $\bar{x}_j^{(-)}$ are the mean of the $j$th feature of the samples of the two categories, respectively, $P$ and $N$ are the sample sizes of the two categories, $x_{ij}^{(+)}$ and $x_{ij}^{(-)}$ are the $j$th eigenvalues of the $x_i$.

Figure 7 shows the F-score of all features of the SVM classifiers S vs. NS and Hor vs. Ver, and the blue upper diagonal line represents the F-score of each feature of the classifier S vs. NS, and the red lower diagonal line represents the F-score of each feature of the classifier Hor vs. Ver. The F-scores of features the SD, the EPR, $R_q$(SD), and $R_q$(EPR) are higher than other features in the S vs. NS classifier, and the F-scores of features GR, $R_{dirB}$(SD), $R_{dirB}$(EPR), $R_{dirT}$(SD) and $R_{dirT}$(EPR) are higher than other features in the Hor vs. Ver classifier. We select features for the classifier according to the F-score descending order to avoid dimensional redundancy. We select the feature set of S vs. NS as the SD, the EPR, $R_q$(SD), and $R_q$(EPR). The feature set of Hor vs. Ver is GR, $R_{dirB}$(SD), $R_{dirB}$(EPR), $R_{dirT}$(SD), and $R_{dirT}$(EPR).

![Figure 7. F-score of features for SVM classifiers S vs. NS and Hor vs. Ver.](image)

### 3.4. The Principle of SVM Algorithm and Training

SVM are generally used as models to solve binary classification problems. It can process nonlinear classification with low complexity. Given a training set $\{x_i, y_i\}_{i=1}^m$, $x_i \in \mathbb{R}^n$ is the input features, $y_i \in \{-1, +1\}$ is the label. In this paper, we use the SD, the EPR, $R_q$(SD), and $R_q$(EPR) as features, CU split and not split as labels input to the S vs. NS classifier, and use GR, $R_{dirB}$(SD), $R_{dirB}$(EPR), $R_{dirT}$(SD), $R_{dirT}$(EPR) as features, horizontal partitioning and vertical partitioning as labels input to the Hor vs. Ver classifier. To solve for the hyperplane $w^T x + b = 0$, turn the problem into:

\[
\begin{align*}
\min_{w} & \frac{1}{2} \|w\|^2 \\
\text{s.t.} & \quad y_i(w^T x_i + b) \geq 1
\end{align*}
\]  

(11)

after optimization, Equation (11) is transformed into:

\[
\begin{align*}
\min_{w} & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \\
\text{s.t.} & \quad y_i(w^T \Phi(x_i) + b) \geq 1 - \xi_i, \xi_i \geq 0, i \in [1, m]
\end{align*}
\]  

(12)
where $C$ is the penalty parameter and $\xi_i$ is the relaxation variable, and $\Phi(x_i)$ represents the new vector after $x$ is mapped to a higher-dimensional space. The Lagrange function is constructed to obtain the following expression:

$$
\min \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \lambda_i \lambda_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{m} \lambda_i
$$

s.t. $\sum_{i=1}^{m} \lambda_i y_i = 0, \ C \geq \lambda_i \geq 0, \ i \in [1, m]$ (13)

where $\lambda_i$ is the Lagrange multiplier, $K(x_i, x_j)$ is the kernel function. In this paper, we use Radial Basis Function (RBF) as the kernel function, which can be applied to nonlinearly separable cases.

In order to test the generalization ability of the SVM models, instead of extracting features from the video sequences of the JVET test set [33], we extracted features from the Div2K training set [34]. This training set is developed for super-resolution and had good diversity. It contains 800 images of different content types, covering a wide range of objects, plants, and animals, as well as natural scenery including the dark environments. These images were encoded by VTM7.0 using the AI configuration and four QP values (22, 27, 32, 37). The features of CU are extracted and normalized and then combined with the labels to build a training set to train the SVM models.

We test five video sequences with different resolutions of the JVET test set to verify the accuracy of the SVM models. The video sequences are “Campfire”, “BasketballDrive”, “BQMall”, “BlowingBubbles”, and “Johnny”. Figure 8 illustrates the prediction accuracy of the two SVM models for different sizes of CU. In the S vs. NS classifier, the prediction accuracy of all sizes of CU exceeded 80%, especially the prediction accuracy of 32 $\times$ 32 CU is higher than 95%. In the Hor vs. Ver classifier, the prediction accuracy of rectangular CUs is generally higher than that of square CUs, especially for smaller size CUs, and the prediction accuracy of CU of sizes 32 $\times$ 8 and 8 $\times$ 32 is higher than 90%.

![Figure 8](image_url)  
**Figure 8.** The prediction accuracy of the proposed SVM model.

4. Experimental Results

In experiments, the proposed algorithm was implemented in VTM7.0. We encoded 22 video sequences in the JVET test set at AI configuration and the default temporary sub-sample ratio at four QP values {22, 27, 32, 37}, and evaluated the performance of the algorithm on these video sequences. These sequences are 3 from class A1, 3 from class A2, 5 from class B, 4 from class C, 4 from class D, and 3 from class E. All experiments are run on a computer with AMD Ryzen 7 5800H CPU @ 3.20 GHz, and the windows operating system. After encoding, we use the time saving rate (TS) of the proposed algorithm compared
with the original VTM to measure the complexity reduction, and use the Bjøntegaard delta bit-rate (BDBR) to evaluate the RD performance. TS is defined as:

$$TS = \frac{1}{4} \sum_{QP \in \{22, 27, 32, 37\}} \frac{T_{VTM7.0}(QP_i) - T_P(QP_i)}{T_{VTM7.0}(QP_i)} \times 100\%$$ \hspace{1cm} (14)$$

where $T_{VTM7.0}$ is the total encoding time using the anchor algorithm, and $T_P$ is the encoding time using the proposed algorithm.

Table 1 lists the experimental results of the proposed algorithm compared with VTM7.0. The improved coding efficiency of the proposed algorithm is relatively stable in all sequences, which can reduce the complexity by 54.05% on average with 1.54% RD performance loss compared with the VTM7.0. In terms of reducing the coding complexity, the proposed method performs well in different resolution test sequences, which indicates that the proposed method has a good generalization ability. For test sequences with smooth textures, such as “DaylightRoad2” and “ParkScene”, encoding time is reduced by more than 58%, which indicates that the proposed algorithm has a better effect on texture smoothing sequences.

| Class | Test Sequence | Proposed (VTM7.0) BDBR (%) | TS (%) |
|-------|---------------|-----------------------------|--------|
| A1    | Tango2        | 1.74                        | 51.43  |
|       | FoodMarket4   | 1.24                        | 47.32  |
|       | Campfire      | 1.37                        | 56.37  |
| A2    | CatRobot      | 2.04                        | 51.94  |
|       | DaylightRoad2 | 1.42                        | 58.61  |
|       | ParkRunning3  | 1.07                        | 55.93  |
| B     | Kimono        | 1.31                        | 54.68  |
|       | ParkScene     | 1.45                        | 58.73  |
|       | Cactus        | 2.07                        | 51.37  |
|       | BasketballDrive | 1.76                      | 53.34  |
|       | BQTerrace     | 1.23                        | 55.37  |
| C     | BasketballDrill | 1.60                      | 53.92  |
|       | PartyScene    | 0.87                        | 52.74  |
|       | RaceHorsesC   | 1.25                        | 51.07  |
|       | BQMall        | 1.87                        | 53.39  |
| D     | BasketballPass | 1.53                      | 53.96  |
|       | BQSquare      | 0.93                        | 54.78  |
|       | BlowingBubbles | 1.57                      | 51.33  |
|       | RaceHorses    | 1.14                        | 55.79  |
| E     | FourPeople    | 2.19                        | 55.21  |
|       | Johnny        | 2.37                        | 56.43  |
|       | KristenAndSara | 1.88                      | 55.46  |
|       | Average       | 1.54                        | 54.05  |

The performance comparison of the proposed method with other fast methods [22,29,30] is shown in Table 2, where the methods proposed in [22,30] and the proposed method experiment in VTM7.0, and the method proposed in [29] experiments in VTM10.0. Since VTM10.0 and VTM7.0 use the same method in the CU partition part, it is reasonable to compare the proposed method with [29]. It has been observed that the proposed method saves an average of 54.30% encoding time with 1.54% BDBR increase. The literature [29] predicts CU partition modes based on SVM, which effectively reduces the coding complexity. Compared with the experimental results of [29], the proposed method increases the coding complexity by 9.37% with 1.31% BDBR decrease. The methods proposed in [22,30] only
predict $32 \times 32$ CU partition modes, which can reduce the coding complexity to a certain extent. Compared with the experimental results of [30], the proposed method reduces the coding complexity by 15.08% with 0.67% BDBR increase. Compared with the experimental results of [22], the proposed method reduces the coding complexity by 5.15%. By analyzing the comparison results, the proposed method can effectively reduce the coding complexity while ensuring the video quality.

Table 2. Encoding performance of proposed algorithm compares with previous works.

| Class | Test Sequence | Wu [29] (VTM10.0) | Zhao [30] (VTM7.0) | Fan [22] (VTM7.0) | Proposed (VTM7.0) |
|-------|---------------|-------------------|-------------------|-------------------|------------------|
|       |               | BDBR (%) | TS (%) | BDBR (%) | TS (%) | BDBR (%) | TS (%) | BDBR (%) | TS (%) |
| B     | BasketballDrive | 2.38     | 67.81 | -       | -       | 3.28     | 59.35 | 1.76     | 53.34 |
|       | Cactus         | 2.78     | 66.61 | -       | -       | 1.84     | 52.44 | 2.07     | 51.37 |
|       | Kimono         | -        | -     | 0.78    | 37.51   | 1.93     | 59.51 | 1.31     | 54.68 |
|       | ParkScene      | -        | -     | 0.61    | 39.56   | 1.26     | 51.84 | 1.45     | 58.73 |
|       | BQTerrace      | 2.43     | 64.25 | 0.76    | 41.79   | 1.08     | 45.30 | 1.23     | 55.37 |
| C     | BasketballDrill| 5.39     | 65.29 | 1.25    | 39.21   | 1.82     | 48.48 | 1.60     | 53.92 |
|       | PartyScene     | 1.40     | 58.77 | 0.37    | 36.73   | 0.26     | 38.62 | 0.87     | 52.74 |
|       | RaceHorsesC    | 2.00     | 62.10 | 0.24    | 30.68   | 0.88     | 49.05 | 1.25     | 51.07 |
| D     | BQSquares      | 1.68     | 59.98 | 0.58    | 36.67   | 0.19     | 31.95 | 0.93     | 54.78 |
|       | BlowingBubbles | 2.24     | 59.94 | 0.83    | 40.87   | 0.47     | 40.35 | 1.57     | 51.33 |
|       | RaceHorses     | 1.69     | 58.98 | 0.56    | 36.51   | 0.54     | 41.69 | 1.14     | 55.79 |
| E     | FourPeople     | 4.36     | 67.14 | 1.34    | 46.51   | 2.70     | 57.57 | 2.19     | 55.21 |
|       | Johnny         | 4.34     | 67.01 | 1.56    | 43.78   | 3.22     | 56.88 | 2.37     | 56.43 |
|       | KristenAndSara | 3.56     | 66.21 | 1.57    | 40.85   | 2.78     | 55.11 | 1.88     | 55.46 |
|       | Average        | 2.85     | 63.67 | 0.87    | 39.22   | 1.59     | 49.15 | 1.54     | 54.30 |

To reduce coding complexity, the proposed method should take less time. We count the time consumption of extracting features, predicting the CU partition mode and the rest of the encoding process. The proportion of time consumption of different resolution sequences is shown in Figure 9. The time consumed by feature extraction is 0.4% to 3.9%, and prediction takes 4.3–8.6% of the time. In addition, high-resolution video sequences take more time to extract features and predict CU partitions than low-resolution video sequences.

![Figure 9. The time consumption percentage of the proposed algorithm.](image)

In order to assess the RD performance of the proposed algorithm, the video sequences “FourPeople” and “Johnny” are tested. Figure 10 shows the RD curves of the two test
sequences at four QPs. It can be seen that the RD curve of the proposed algorithm and the VTM7.0 anchor algorithm almost overlap in the high QP encoding, and the RD curve slightly deviates in low QP encoding, which means that the proposed algorithm and the anchor algorithm have similar RD performance.

Figure 10. The RD curves: (a) the RD curve of FourPeople; (b) the RD curve of Johnny.

5. Conclusions

In this paper, we have proposed an SVM-based fast CU partition decision algorithm to reduce the coding complexity. Based on the statistical analysis of the proportion of the split mode of CU, we extract the standard deviation (SD) and the edge point ratio (EPR) in CU and the maximum ratio of the SD and the EPR in sub-CUs. Then, features with higher correlation are used as target features for the SVM. Finally, the proposed SVM models are trained offline for decision CU partition. The proposed algorithm was integrated into VTM 7.0 for testing. Experimental results show that the proposed algorithm can save 54.05% coding time and increase 1.54% BDBR compared with the VTM7.0, which indicates that the proposed algorithm achieves a good compromise between complexity and efficiency.

Author Contributions: Conceptualization, J.Z. and A.W.; methodology, J.Z.; software, A.W.; validation, J.Z., Q.Z. and A.W.; formal analysis, A.W.; investigation, A.W.; resources, Q.Z.; data curation, A.W.; writing—original draft, A.W.; writing—review and editing, J.Z.; visualization, J.Z.; 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 research was funded by the National Natural Science Foundation of China, grant numbers 61771432 and 61302118; the Basic Research Projects of Education Department of Henan, grant numbers 21zx003 and 20A880004; the Postgraduate education reform and quality improvement project of Henan Province, grant number YJS2021KC12.

Conflicts of Interest: The authors declare no conflict 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. Sullivan, G.J.; Ohm, J.-R.; Han, W.-J.; Wiegand, T. Overview of the High Efficiency Video Coding (HEVC) Standard. IEEE Trans. Circuits Syst. Video Technol. 2012, 22, 1649–1668. [CrossRef]
3. Versatile Video Coding Test Model (VTM) Reference Software of the JVET of ITU-T VCEG and ISO/IEC MPEG. Available online: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM (accessed on 13 June 2021).
4. High Efficiency Video Coding Test Model (HM) Reference Software of the JCT-VC of ITU-T VCEG and ISO/IEC MPEG. Available online: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/ (accessed on 13 June 2021).
5. 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]
6. 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]

7. 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]

8. Min, B.; Cheung, R.C.C. A Fast CU Size Decision Algorithm for the HEVC Intra Encoder. *IEEE Trans. Circuits Syst. Video Technol.* 2015, 25, 892–896. [CrossRef]

9. Shen, L.; Zhang, Z.; Liu, Z. Effective CU Size Decision for HEVC Intracoding. *IEEE Trans. Image Process.* 2014, 23, 4232–4241. [CrossRef] [PubMed]

10. Cho, S.; Kim, M. Fast CU Splitting and Pruning for Suboptimal CU Partitioning in HEVC Intra Coding. *IEEE Trans. Circuits Syst. Video Technol.* 2013, 23, 1555–1564. [CrossRef]

11. Zhang, M.; Lai, D.; Liu, Z.; An, C. A novel adaptive fast partition algorithm based on CU complexity analysis in HEVC. *Multimed. Tools Appl.* 2019, 78, 1035–1051. [CrossRef]

12. Grellert, M.; Bampi, S.; Correa, G.; Zatt, B.; da Silva Cruz, L.A. Learning-Based Complexity Reduction and Scaling for HEVC Encoders. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018; pp. 1208–1212. [CrossRef]

13. Correa, G.; Dall’Oglio, P.; Palomino, D.; Agostini, L. Fast Block Size Decision for HEVC Encoders with On-the-Fly Trained Classifiers. In Proceedings of the 2020 28th European Signal Processing Conference (EUSIPCO), Amsterdam, The Netherlands, 18–21 January 2021; pp. 540–544. [CrossRef]

14. Sun, C.; Fan, X.; Zhao, D. A Fast Intra CU Size Decision Algorithm Based on Canny Operator and SVM Classifier. In Proceedings of the 2018 IEEE International Conference on Image Processing (ICIP), Athens, Greece, 7–10 October 2018; pp. 1787–1791. [CrossRef]

15. Yin, J.; Yang, X.; Lin, J.; Chen, Y.; Fang, R. A Fast Block Partitioning Algorithm Based on SVM for HEVC Intra Coding. In Proceedings of the 2018 2nd International Conference on Video and Image Processing (ICVIP 2018), Hong Kong, China, 29–31 December 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 176–181. [CrossRef]

16. Erabadda, B.; Mallikarachchi, T.; Kulupana, G.; Fernando, A. iCUS: Intelligent CU Size Selection for HEVC Inter Prediction. *IEEE Access* 2020, 8, 141143–141158. [CrossRef]

17. Shen, X.; Yu, L. CU splitting early termination based on weighted SVM. *EURASIP J. Image Video Process.* 2013, 2013, 4. [CrossRef]

18. Bouaafia, S.; Khemiri, R.; Sayadi, F.E.; Atri, M. Fast CU partition-based machine learning approach for reducing HEVC complexity. *J. Real-Time Image Process.* 2020, 17, 185–196. [CrossRef]

19. Chen, Z.; Shi, J.; Li, W. Learned Fast HEVC Intra Coding. *IEEE Trans. Image Process.* 2020, 29, 5431–5446. [CrossRef] [PubMed]

20. Feng, Z.; Liu, P.; Jia, K.; Duan, K. HEVC Fast Intra Coding Based CTU Depth Range Prediction. In Proceedings of the 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), Chongqing, China, 27–29 June 2018; pp. 551–555. [CrossRef]

21. Xu, M.; Li, T.; Wang, Z.; Deng, X.; Yang, R.; Guan, Z. Reducing Complexity of HEVC: A Deep Learning Approach. *IEEE Trans. Image Process.* 2018, 27, 5044–5059. [CrossRef] [PubMed]

22. Fan, Y.; Chen, J.; Sun, H.; Katto, J.; Jing, M. A Fast QTMT Partition Decision Strategy for VTC Intra Prediction. *IEEE Access* 2020, 8, 107990–107911. [CrossRef] [PubMed]

23. 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]

24. Cui, J.; Zhang, T.; Gu, C.; Zhang, X.; Ma, S. Gradient-Based Early Termination of CU Partition in VTC Intra Coding. In Proceedings of the 2020 Data Compression Conference (DCC), Snowbird, UT, USA, 24–27 March 2020; pp. 103–112. [CrossRef]

25. Amestoy, T.; Mercat, A.; Hamidouche, W.; Menard, D.; Bergeron, C. Tunable VTC Frame Partitioning Based on Lightweight Machine Learning. *IEEE Trans. Image Process.* 2020, 29, 1313–1328. [CrossRef] [PubMed]

26. Zhang, Q.; Wang, Y.; Huang, L.; Jiang, B. Fast CU Partition and Intra Mode Decision Method for H.266/VVC. *IEEE Access* 2020, 8, 117539–117550. [CrossRef]

27. 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]

28. Wang, Z.; Wang, S.; Zhang, J.; Wang, S.; Ma, S. Effective Quadtree Plus Binary Tree Block Partition Decision for Future Video Coding. In Proceedings of the 2017 Data Compression Conference (DCC), Snowbird, UT, USA, 4–7 April 2017; pp. 23–32. [CrossRef]

29. Wu, G.; Huang, Y.; Zhu, C.; Song, L.; Zhang, W. SVM Based Fast CU Partitioning Algorithm for VVC Intra Coding. In Proceedings of the 2021 IEEE International Symposium on Circuits and Systems (ISCAS), Daegu, Korea, 22–28 May 2021; pp. 1–5. [CrossRef]

30. 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]

31. Tissier, A.; Hamidouche, W.; Vanne, J.; Galpin, F.; Menard, D. CNN Oriented Complexity Reduction of VTC Intra Encoder. In Proceedings of the 2020 IEEE International Conference on Image Processing (ICIP), Abu Dhabi, United Arab Emirates, 25–28 October 2020; pp. 3139–3143. [CrossRef]

32. Park, S.-H.; Kang, J. Fast Multi-type Tree Partitioning for Versatile Video Coding Using a Lightweight Neural Network. *IEEE Trans. Multimed.* 2020, 23, 4388–4399. [CrossRef]
33. Bossen, F.; Boyce, J.; Suehring, K.; Li, X.; Seregin, V. JVET common test conditions and software reference configurations for SDR video JVET-N1010-v1. In Proceedings of the 14th Meeting of the Joint Video Exploration Team (JVET), Geneva, Switzerland, 19–27 March 2019; pp. 1–6.

34. Agustsson, E.; Timofte, R. Ntire 2017 challenge on single image super-resolution: Dataset and study. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Honolulu, HI, USA, 21–26 July 2017; pp. 126–135.