Bagged Tree And ResNet Based Joint End-to-End Fast CTU Partition Decision Algorithm For Video Intra Coding

Yixiao Li

Beijing University of Posts and Telecommunications

Lixiang Li (✉ li_lixiang2006@163.com)

Beijing University of Posts and Telecommunications <https://orcid.org/0000-0001-9949-8731>

Yuan Fang

Beijing University of Posts and Telecommunications

Haipeng Peng

Beijing University of Posts and Telecommunications

Nam Ling

Santa Clara University

Research

Keywords: video coding, fast coding unit (CU) partition, residual network (ResNet), high efficiency video coding (HEVC), intra coding

Posted Date: December 20th, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-1119619/v1>

License: This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Bagged Tree and ResNet Based Joint End-to-End Fast CTU Partition Decision Algorithm for Video Intra Coding

Yixiao Li¹, Lixiang Li^{1*}, Yuan Fang¹, Haipeng Peng¹ and Nam Ling²

*Correspondence:

li_lixiang2006@163.com

¹ Information Security Center,
State Key Laboratory of
Networking and Switching
Technology, National Engineering
Laboratory for Disaster Backup
and Recovery, Beijing University of
Posts and Telecommunications,
Beijing, 100876, China
Full list of author information is
available at the end of the article

In the development of video coding standards, advanced ones have greatly improved the bit rate compared with those of previous generation, but also brought a huge increase in coding complexity. Coding standards, such as high efficiency video coding (HEVC), versatile video coding (VVC) and AOMedia video 2 (AV2), get the optimal encoding performance by traversing all possible combinations of coding unit (CU) partition and selecting the combination with minimum coding cost. This process of searching for the best makes up a large part of encoding complexity. To reduce the complexity of coding block partition for many video coding standards, this paper proposes an end-to-end fast algorithm for partition structure decision of coding tree unit (CTU) in intra coding. It can be extended to various coding standards with fine tuning, and is applied to the intra coding of HEVC reference software HM16.7 as an example. In the proposed method, the splitting decision of a CTU is made by a well designed bagged tree model firstly. Then, the partition problem of a 32×32 sized CU is modeled as a 17-output classification task and solved by a well trained residual network (ResNet). Jointly using bagged tree and ResNet, the proposed fast CTU partition algorithm is able to generate the partition quad-tree structure of a CTU through an end-to-end prediction process, instead of multiple decision making procedures at depth level. Besides, several effective and representative datasets are also conducted in this paper to lay the foundation of high prediction accuracy. Compared with the original HM16.7 encoder, experimental results show that the proposed algorithm can reduce the encoding time by 59.79% on average, while the BD-rate loss is as less as 2.02%, which outperforms the results of most of state-of-the-art approaches in the fast intra CU partition area.

Keywords: video coding; fast coding unit (CU) partition; residual network (ResNet); high efficiency video coding (HEVC); intra coding

Introduction

Video coding standards have been continuously developed and updated for decades to meet the increasing need of video market for videos with higher definition. In the past many years, various coding standards have been invented and replaced, such as advanced video coding (AVC), high efficiency video coding (HEVC), versatile

video coding (VVC), audio video coding standard (AVS) and AOMedia video 1 (AV1). They all brought considerable coding benefits in their time, for example, developed by the Joint Collaborative Team on Video Coding (JCT-VC), VVC and HEVC both are able to achieve about 50% less bit rate than their predecessors while maintaining the same video quality [1]. However, with the transition of video compression rate again and again, the complexity of various video coding standards has increased dramatically due to the introduction of many efficient but complicated coding tools.

Block based coding is one of the most important coding techniques and is commonly used in many popular video coding standards. Taking HEVC and VVC for example, HEVC uses quad-tree structure to partition a coding unit (CU), and VVC uses quad-tree plus binary tree (QTBT) structure to finish block partitioning. In block based video coding standards, each CU can be iteratively further split into sub-CUs according to a specific partition structure. A flexible CU partition rule brings various size combinations for a CU. With many CU sizes and modes to be selected, rate-distortion optimization (RDO) is adopted to select the optimal CU partition structure for a CTU along with prediction modes. During the encoding process, many video coding standards traverse all the possible CU sizes and prediction modes, then RDO is used to select the best combination with the minimum cost. As a result, RDO brings the most encoding complexity burden by exhaustive calculation on the CU size decision process, and it restricts the application of video coding standards in many real-time scenes. Thus, it is very necessary and meaningful to design a fast CU partition algorithm for existing video coding standards.

In the past several years, many approaches have been proposed to reduce the massive encoding complexity in various standards, such as HEVC and VVC. Heuristic based CU depth decision approaches were first proposed and widely studied [2, 3, 4, 5, 6]. For example, Liu et al. [3] proposed a fast CU depth decision algorithm based on statistical analysis. They used a three-stage method to make the splitting result decision according to prior information. Shen et al. [7] proposed an early determination and a bypass strategy for CU size decision by using the texture property of current CU and coding information from neighboring CUs. In some approaches, CU depth range was shorted, and some CU depth levels were skipped according to statistical information [8, 9].

Although the heuristic based fast methods have achieved many acceptable results, they can not properly consider the partition property of various video sequences. In other words, there are too many factors to influence the partition result. And these factors may change with different sequences so that people usually do not know that which combination of these factors has the best performance upon implementation. Generally, we only consider several key factors which have close correlation with CU partition, and small number of the considered factors may lead to a poor result.

The technique of classical machine learning algorithms is introduced in order to overcome the drawback of statistical information based methods. Support vector machine (SVM) models with three outputs are used to get a trade-off between bit distortion and encoding complexity [10, 11]. Besides, to further improve the prediction accuracy, two or more SVM models are employed in each CU depth. Zhang et al. [12] employed two SVMs at each depth to perform the decisions of early CU split

and early CU termination. Zhu et al. [13] used the cascaded SVM and defined a misclassification cost and a risk area to jointly make a CU partition decision. Meanwhile, Grellert et al. [14] and Zhang et al. [15] also proposed SVM based approaches which focused on features analysis. Besides, decision tree or data mining methods were also used to reduce the encoding complexity [16, 17, 18, 19, 20]. Also, Fisher's linear discriminant analysis and the k -nearest neighbors classifier were employed in order to fastly decide a CU partition [21], and Kim et al. [22] proposed a joint online and offline Bayesian decision rule based fast CU partition algorithm. Moreover, Yang et al. [23] proposed an efficient low complexity intra coding algorithm for versatile video coding (VVC) by using learning based classifiers.

Although the classical machine model based methods outperform the heuristic based ones due to their advantages on dealing high dimensional problems, their inputs (i.e. features) play a very important role on the encoding results. Besides, the features designing is all manual and it requires much experience. Additionally, there are many key features which are commonly used in existing works. Hence, in order to significantly improve the encoding performance, researchers should find more efficient features, which is usually quite difficult. Furthermore, generally one or more classifiers are needed to be designed for each depth, which requires much work and training time. The algorithms using convolution neural network (CNN) were proposed to address these problems[24, 25, 26]. Due to the property of CNN on automatic region feature extraction, these algorithms have enormous advantages in image processing. CNN based algorithms also have achieved many good encoding results.

Liu et al. [27] devised the convolution neural network (CNN) based fast algorithm to prune no less than two partition modes for the RDO processing on each CTU. However, their CNN structure is too shallow to fully learn the relation between the image data and the partition structure. Besides, considering all available partition modes for a CU, only no less than two CU partition modes are pruned, which is not enough.

Besides, the algorithm proposed by Kim et al.[28] used image data and encoding information based vector data to train a CNN for the prediction of the CTU depth. However in their algorithm, not only image data but also vector data should be collected before the prediction phase, which requires more pre-encoding time. Besides, three kinds of CNN structures should be constructed, and each is designed for a CU in a certain depth. Therefore, at least three CNNs are needed for one video sequence.

Furthermore, Xu et al. [29] proposed an approach by using CNN and long short-term memory (LSTM) network. Specifically, the CNN was used to predict CU partition of intra coding, and the LSTM was used to predict the CU partition for inter coding. In their algorithm, the CNN especially the LSTM part is very complex and needs much time to get trained and refined.

As we can see, all these CNN based methods still complete the CU partition prediction at the depth level, which needs one or more CNN models for each CU depth. It means that at least three deep learning models are needed for a single video sequence. In other words, existing works still model the CU partition as a binary classification problem, and it is a great waste of the CNN ability. In

this paper, we construct a deep structure to explore a sufficient learning capacity of ResNet. Meanwhile, we propose an end-to-end solution for the CTU partition. What's important is that the proposed algorithm is a general solution to the partitioning problem in many block based coding standards. It models the CU partition as a multi-classification problem unprecedentedly. To verify the effectiveness of our proposed method, we implement it into HEVC intra coding as an example. First, bagged tree models are employed to classify a CTU to sharply shorten the classification categories. Then an end-to-end ResNet is trained to predict the final partition structure of a 32×32 CU instead of predicting the splitting decision of a CU in a certain depth. The main contributions of this paper are presented as follows:

- 1) Using the great superiority of ResNet, this paper proposes a deep neural network structure to predict the partition structure of a CU. It explores more sufficient relations between the image data and the partition.
- 2) Adopting a two-stage prediction strategy with a combination of bagged tree and ResNet, the proposed algorithm can achieve more accurate prediction results. And it reduces CU partition categories sharply and effectively.
- 3) For the first time, the CU partition task is modeled as a 17-class problem. An end-to-end CNN prediction model is designed. As a result, only one CNN model is needed for a single video sequence instead of more than three CNN models.
- 4) A general solution to CU partitioning in video intra coding is proposed, and is verified on HEVC for an example. In a similar way, it can be implemented on other video coding standards and achieve gains in coding complexity reduction.

This paper is organized as follows. Section 2 introduces the quad-tree based partition structure and gives a brief review of bagged tree and ResNet. Section 3 describes the proposed bagged tree and ResNet based end-to-end joint fast CTU partition algorithm. Section 4 reports the experimental results. The conclusions are summarized in Section 5.

Background

In this section, firstly we describe the CU partition technique in HEVC. Then, we give a brief review about bagged tree method, followed by a short introduction of ResNet model.

CU Partition of HEVC

To encode a video sequence, each frame among a sequence is divided into multiple non-overlapping squares, whose size is 64×64 . As the largest coding unit (CU), a CU of size 64×64 is also called a coding tree unit (CTU). As is known to all, larger CU can save more semantic information while smaller CU can achieve more precise pixel prediction values. Thus, HEVC employs a recursive splitting process to traverse all the possible partition results of a CU. Fig. 1 shows the recursive splitting process of a CTU, and this process terminates when it reaches the smallest CU, whose size is usually configured before encoding and default to be 8×8 . Therefore, there are totally 4 kinds of CUs, which have different sizes, i.e. 64×64 , 32×32 , 16×16 and 8×8 , shown as squares of different colors in Fig. 1.

According to a quad-tree structure, HEVC employs RDO to make the optimal partition decision for a CTU. The left part of Fig. 2 shows a partition example of a CTU, and the right part of Fig. 2 is the corresponding quad-tree structure. As we can calculate, there are two options for a CU of size 16×16 (i.e. nonsplit or split to four sub-CUs of size 8×8). However, the options for a CU of size 32×32 increase to 17 ($1+2^4$), and for a CTU the options go up to 83522 ($1+17^4$). Performing RDO on CTUs extremely increases the encoding complexity which is too large to be solved by multi-classification techniques.

Bagged Tree

Traditional machine learning methods have shown great advantages in classification tasks. As one of the most famous classical machine learning models, the decision tree is widely used in various application scenarios, such as image recognition and data mining. Just like its name, the decision tree employs a tree structure to make a classification decision. For a classification task, each leaf node in a decision tree model represents a target class, and each father node contains an attribute along with a threshold. In this way, the decision making process goes deeper, and the predicted labels are generated.

Among traditional machine learning methods, prediction efficiency, time complexity and implementation difficulty vary a lot. However, compared with other models, the tree model has its unique advantages. First, it accords with people's common sense among some classification problems. Thus it can achieve a quite high accuracy among some particular problems through proper training. Second, it is much easier to get trained and takes less time to finish prediction, which brings negligible time overhead. Third, it is also easier to be implemented into existing works due to the simple "if...else..." structure based prediction process.

However, the decision tree is much likely to get overfitting on training set. The bagging technique is introduced to address this problem. As a result, the bagged tree model is generated, and it is an ensemble of many decision tree classifiers, each of which is trained with a random subset of the training dataset. After training, the final prediction of the a bagged tree model for an instance is made by taking the majority vote of the predictions from all the individual decision trees on the input sample. Fig. 3 shows the structure of bagged tree model consisting of n decision trees, and Fig. 3 also illustrates how the bagged tree generates the final prediction for an instance. Since the bagging technique decreases the variance of the model without increasing the bias, the performance of bagged tree model is better than that of a single decision tree model. Therefore, in the proposed joint fast CU partition algorithm, the bagged tree is used in the first phase to predict the splitting result of a CTU.

One of the most important features of bagged tree is that the importance of features can be evaluated during the training process without having to repeatedly train models with different feature combinations like most feature selection techniques require. Caruana et al. [30] proposed a feature importance measurement technique referred as multiple counting for bagged tree. In multiple counting, the importance value for every feature of a single decision tree model is calculated according to the number of data samples decided by this feature in all decision tree

nodes. To compute the importance value of a feature for the bagged tree model, the importance values of a feature on the individual decision trees consisting of the bagged tree model are summed and normalized by the feature entropy to ensure comparability of features with different number of values. More details about the feature importance measurement technique are available in [30].

Residual Network

In recent years, CNN is one of the hottest topics due to its excellent performance on image problems solving, such as image recognition and object detection. Compared with traditional machine learning methods, CNN extracts effective features by using the filters of various sizes instead of hand-crafted ones. Using the convolution, CNN also reduces massive parameters required in a neural network. However, when deeper networks start converging, the researchers found that with the increment of network depth, the training error gets higher, and the accuracy degrades rapidly [31].

To address this problem, He et al. [31] proposed ResNet by using a deep residual learning framework. Fig. 4(a) shows the structure of a basic building block in ResNet. With the introduction of a shortcut in Fig. 4(a), a deep network, constructed by many building blocks, can converge easily, and the degradation problem is solved. By a stack of numbers of building blocks like Fig. 4(a), a ResNet is constructed, and He et al. [31] provided two kinds of build blocks (Fig. 4(b-c)), by which a well performed ResNet of expected depth can be generated. The building block in Fig. 4(b) is used to build a relative deep network (ResNet-34). The building block in Fig. 4(c), called “bottleneck”, is used to stack ResNet-50/101/152, which represents ResNets containing 50, 101 and 152 layers, respectively.

ResNet can be as deep as having 152 layers or contain even more layers while maintaining a good convergence performance. As a result, it is more suitable for ResNet than other CNNs to solve the end-to-end CU partition problem. Therefore, the CU partition problem is modeled as a multi-classification problem by the proposed algorithm.

The Proposed Bagged Tree and ResNet Based Joint CTU Partition Method

In this section, we describe the proposed fast CTU partition algorithm for HEVC intra coding. First, we illustrate the overall process, which explains how the bagged tree model and the ResNet model jointly work. Then, effective features are designed to train bagged tree models. Also, the architecture of the ResNet is designed to achieve a good prediction performance. Finally, the databases are constructed to train and validate our bagged tree and ResNet models. Using these databases, we train 20 bagged tree models (each is used for sequences under the same resolution and QP value). Besides, we also train four ResNet models with the same architecture, and each model is generated when a certain quantization parameter (QP) is set as different values, i.e. 22, 27, 32 and 37.

Flowchart of Our Fast Splitting Method

The flowchart of the proposed two-stage CTU partition algorithm is shown in Fig. 5. As can be seen from Fig. 5, there are roughly two phases in the proposed bagged tree and ResNet based joint CTU partition algorithm.

In the first stage, a bagged tree based classifier is used to predict the splitting status of a CTU, and the output is either split or nonsplit. Specifically, if a CTU is predicted to be a nonsplit class, it goes to the CTU partition structure determination process directly shown in Fig. 5, and the final partition is a whole 64×64 CTU without any splitting. On the other hand, if a CTU is predicted to be split by the bagged tree classifier. It will be split into four sub-CUs with size 32×32 immediately. Sequentially, each of these four CUs will be passed to the second stage for further processing.

In the second stage, a 32×32 sized CU is fed to the fine trained ResNet classifier. As a result, one of 17 labels, each of which represents a partition structure for CU with size 32×32 , is the output by ResNet. Similarly, all these four sub-CUs of a CTU predicted to be split in the first stage are classified, and four corresponding predicted labels are generated. Then these four labels as well as the parent CTUs are processed further.

A CTU partition structure determination process is adopted by following the above-mentioned stages 1-2. According to those four labels generated by ResNet in stage 2, the partition structure of the corresponding parent CTU is decided finally. Fig. 6 shows the flowchart of the proposed algorithm. PL , whose value is from 1 to 17, is the predicted label output by ResNet for a CU of size 32×32 . The corresponding partition quad-tree structure is illustrated in Fig. 6, where the PL values of four sub-CUs (numbered 1, 2, 3 and 4) are assumed to be 1, 5, 17 and 2. Once a partition of a CTU is generated, an optimal rate-distortion cost can be calculated directly without lots of comparisons. Then an encoding process is executed.

Comparing with traditional recursive RDO, the proposed algorithm effectively skips unnecessary search for different combinations of CU sizes. Besides, compared with other existing fast partition methods implemented on depth level, this paper proposes a novel end-to-end two-stage CTU partition algorithm by using the bagged tree and ResNet techniques. In this way, the CTU partition result can be calculated through a well-trained bagged tree model and a fine-tuned ResNet model. As a result, we not only significantly reduce the time spent on RDO, but also spend less time on training required learning models compared with existing works, which take a strategy of training as many as three or more models on the depth level.

Features Design for Bagged Tree Model

The traditional machine learning methods rely heavily on handcrafted features. Many existing works have developed numbers of useful attributes for CU splitting label prediction. Based on these existing features, in our previous work [32] we also designed several novel features proven to be effective. As a result, based on our particular task of this paper we select totally 29 feature candidates used in [32], and they are listed in Table I along with corresponding meanings. These feature candidates are from four fields, which are information from neighboring CTUs, side information during pre-encoding process, statistical information of CTU pixels and information from pixel filtering results. We will give a brief introduction of these feature candidates in the following paragraphs, and more details are available in our previous work [32].

Table 1 feature candidates and corresponding descriptions

Index	Feature Candidates	Feature Description
1	<i>m_varMeanSub</i>	variance of four sub CUs' mean
2	<i>m_varVarSub</i>	variance of four sub CUs' variance
3	<i>m_aveCBF</i>	average value of current CU's Coded Block Flag
4	<i>m_nbCtuAboRd</i>	RD cost of current CU's above CTU
5	<i>m_nbCtuLefRd</i>	RD cost of current CU's left CTU
6	<i>m_nbCtuAblRd</i>	RD cost of current CU's above left CTU
7	<i>m_nbCtuAbrRd</i>	RD cost of current CU's above right CTU
8	<i>m_nbCtuAboDepth</i>	depth of current CU's above CTU
9	<i>m_nbCtuLefDepth</i>	depth of current CU's left CTU
10	<i>m_nbCtuAblDepth</i>	depth of current CU's above left CTU
11	<i>m_nbCtuAbrDepth</i>	depth of current CU's above right CTU
12	<i>totalCost</i>	total cost of current CU encoded with planar
13	<i>totalDistortion</i>	total distortion of current CU encoded with planar
14	<i>totalBins</i>	total bins of current CU encoded with planar
15	<i>m_costHadamard</i>	hadamard cost of planar mode
16	<i>m_sadHadamard</i>	distortion of residual after hadamard transfer
17	<i>m_bitsHadamard</i>	hadamard bits of planar mode
18	<i>m_edgeSobel</i>	edge detection result using Sobel
19	<i>m_nmse</i>	mean square error of neighbor pixels
20	<i>m_dcom</i>	mean of gradients of four directions
21	<i>m_numInterestPoint</i>	number of interesting points of current CU
22	<i>m_haarSumx</i>	sum of horizontal value after haar wavelet transfer
23	<i>m_haarSumy</i>	sum of vertical value after haar wavelet transfer
24	<i>m_haarSumxy</i>	sum of diagonal value after haar wavelet transfer
25	<i>m_haarSumAbsx</i>	sum of horizontal absolute value after haar
26	<i>m_haarSumAbsy</i>	sum of vertical absolute value after haar
27	<i>m_haarSumAbsxy</i>	sum of diagonal absolute value after haar
28	<i>m_meanMain</i>	mean of current CU
29	<i>m_varMain</i>	variance of current CU

Results of [33] suggested that information from neighboring CTUs is useful for the partition decision making of current CU. Thus, the features *m_nbCtuAboRd*, *m_nbCtuLefRd*, *m_nbCtuAblRd* and *m_nbCtuAbrRd* are extracted, and they are the RD cost of the neighboring CTUs locating at the the above, left, above-left and above-right of current CTU, respectively. Besides, for a target CTU, *m_nbCtuAboDepth*, *m_nbCtuLefDepth*, *m_nbCtuAblDepth* and *m_nbCtuAbrDepth* denote the average depth values of its above, left, above-left and above-right neighbored CTU, respectively.

Besides, some bypass results during the encoding process are also important and widely used [34]. Hence, we pre-encode the current CTU with PLANAR mode and use several useful encoding results as key features. As a result, the features *totalCost*, *totalDistortion* and *totalBins* represent the total cost, the total distortion and the number of bits under PLANAR mode, respectively. And *m_aveCBF* is the coded block flag (CBF) of current CTU encoded with PLANAR mode. Moreover, HEVC uses Hadamard transform to fast estimate the encoding performance, so the encoding cost, the distortion and the number of bits generated under Hadamard transform condition are extracted as features, which are denoted as *m_costHadamard*, *m_sadHadamard* and *m_bitsHadamard*, respectively.

Obviously, some classic statistical data, such as mean, variance and gradient, can reflect the content complexity of a CU and the difference among four sub-CUs comprising one CU. Thus for a target CTU, we calculate the mean and the variance of all pixel values as *m_meanMain* and *m_varMain*, respectively. They

are calculated according to the following equations

$$m_meanMain = \frac{1}{n} \sum_{i=1}^n p_i \quad (1)$$

$$m_varMain = \frac{1}{n} \sum_{i=1}^n (p_i - m_meanMain)^2 \quad (2)$$

where p_i is the luminance value of the i th pixel, n is the number of pixels in current CU.

Futhermore, the variance of four sub-CUs means is calculated and denoted as $m_varMeanSub$, and $m_varVarSub$ is the variance of four sub-CUs variances. They are calculated through the following equation

$$m_varMeanSub = \frac{1}{4} \sum_{i=1}^4 (meanSub_i - m_meanMain)^2 \quad (3)$$

$$m_varVarSub = \frac{1}{4} \sum_{i=1}^4 (varSub_i - meanVarSub)^2 \quad (4)$$

where $varSub_i$ represents the variance of luma pixels within the i th CU, and $meanVarSub$ is the mean of all the four sub-CUs variances (i.e. $varSub_1$, $varSub_2$, $varSub_3$ and $varSub_4$).

In addition, we also reuse the features m_nmse , which was designed and proved to be efficient in [11]. m_nmse is the mean of squared errors between each pixel and the mean of its eight neighboring pixels of a CTU. It is calculated as follows:

$$\bar{p}_{i,j} = \frac{1}{8} \begin{pmatrix} p_{i-1,j-1} + p_{i-1,j} + p_{i-1,j+1} \\ + p_{i,j-1} + p_{i,j+1} \\ + p_{i+1,j-1} + p_{i+1,j} + p_{i+1,j+1} \end{pmatrix} \quad (5)$$

$$m_nmse = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N (p_{i,j} - \bar{p}_{i,j})^2 \quad (6)$$

where $p_{i,j}$ is the luma value of the pixel locating at (i, j) in current CU. N is the side length of a CU, and it is 64 in the case of a CTU.

Besides, we also apply several kinds of filters on the pixels of a CTU and process these filter results to obtain useful features, which can reflect the pixel changing, the content complexity and even the edge information of a target CTU. As a result, $m_edgeSobel$ and m_dcom are features calculated from responses of sobel filters widely used in edge detection. Fig. 7 shows four Sobel filters for detecting edges in different directions. Each filter is applied with overlap on every 3×3 square pixel block within current CU. Fig. 7(a) is an example of the 3×3 square pixel block. $m_edgeSobel$ and m_dcom are calculated through the following equations

$$g_h = -a - 2b - c + g + 2h + i \quad (7)$$

$$g_v = -a - 2d - g + c + 2f + i \quad (8)$$

$$g_{45} = 2a + b + d - f - h - 2i \quad (9)$$

$$g_{135} = b + 2c - d + f - 2g - h \quad (10)$$

$$m_edgeSobel = \frac{1}{(N-2)^2} \sum_{k=0}^{(N-2)^2} \left((g_h^k)^2 + (g_v^k)^2 \right)^{1/2} \quad (11)$$

$$m_dcom = \frac{1}{(N-2)^2} \sum_{k=0}^{(N-2)^2} (|g_h^k| + |g_v^k| + |g_{45}^k| + |g_{135}^k|) \quad (12)$$

where N is the pixel number of a CU along a side, and it is 64 in the case of a CTU. k denotes the k th 3×3 square pixel block within current CU.

As we can see, $m_edgeSobel$ can reflect information about horizontal and vertical edges, and m_dcom represents the comprehensive gradient information of current CU from four directions. Moreover, Bay et al. [35] systematically analyzed the significant advantage of Haar wavelet on edges changing and scene shading of a picture. So we perform three Haar filters of different directions on the pixels in a target CTU. Fig. 8 (b-d) are the Haar filters of horizontal, vertical and diagonal directions, respectively. Fig. 8 (a) is an example of a 2×2 square pixel block on which the Haar filters are performed. Specifically, we firstly split a CTU into 2×2 non-overlapped sub-squares. Then, each sub-square is filtered by a same Haar filter to obtain the corresponding filter responses of a certain direction. The responses of a certain filter on a sub-square are calculated according to the following equations

$$d_x = a + b - c - d \quad (13)$$

$$d_y = a - b + c - d \quad (14)$$

$$d_{xy} = a - b - c + d \quad (15)$$

where d_x , d_y and d_{xy} are the filter responses of horizontal, vertical and diagonal directions, respectively.

Based on these responses, we sum them to obtain the features $m_haarSumx$, $m_haarSumy$, $m_haarSumxy$, $m_haarSumAbsx$, $m_haarSumAbsy$, and $m_haarSumAbsxy$ according to the following equations

$$m_haarSumx = \frac{1}{\left(\frac{N}{2}\right)^2} \sum_{k=1}^{\left(\frac{N}{2}\right)^2} d_x^k \quad (16)$$

$$m_haarSumy = \frac{1}{\left(\frac{N}{2}\right)^2} \sum_{k=1}^{\left(\frac{N}{2}\right)^2} d_y^k \quad (17)$$

$$m_haarSumxy = \frac{1}{\left(\frac{N}{2}\right)^2} \sum_{k=1}^{\left(\frac{N}{2}\right)^2} d_{xy}^k \quad (18)$$

$$m_haarSumAbsx = \frac{1}{\left(\frac{N}{2}\right)^2} \sum_{k=1}^{\left(\frac{N}{2}\right)^2} |d_x^k| \quad (19)$$

$$m_haarSumAbsy = \frac{1}{\left(\frac{N}{2}\right)^2} \sum_{k=1}^{\left(\frac{N}{2}\right)^2} |d_y^k| \quad (20)$$

$$m_haarSumAbsxy = \frac{1}{\left(\frac{N}{2}\right)^2} \sum_{k=1}^{\left(\frac{N}{2}\right)^2} |d_{xy}^k| \quad (21)$$

where N is the number of pixels of the current CU along one side, and it is 64 in a case of CTU.

Furthermore, the number of the interest related points within a picture can reflect people's visual perception, while no one notices its influence on the CU splitting. So we extract the interest points related feature $m_numAveInterestPoint$ for a target CTU by using a particular filter on a CTU. The feature $m_numAveInterestPoint$ represents the average number of interest points for each pixel of a CTU. It reflects how much attention people will pay to a CTU and how many details a CTU contains. Three filters as shown in Fig. 9 are used on each of the current CTU, and three corresponding results D_{xx} , D_{yy} , D_{xy} are obtained as the filter responds in the horizontal, vertical, and diagonal directions, respectively. We obtain the final value of the feature $m_numAveInterestPoint$ for current CTU by using the following equations

$$P(i, j) = \left| D_{xx}D_{yy} - (0.9D_{xy})^2 \right| \quad (22)$$

$$B(i, j) = \begin{cases} 0, & P(i, j) < t \\ 1, & P(i, j) \geq t \end{cases} \quad (23)$$

$$m_numAveInterestPoint = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} B(i, j) \quad (24)$$

where $P(i, j)$ is the interest value of the pixel located at (i, j) , and $B(i, j)$ is the Boolean value of being decided to be an interest point for pixel (i, j) . t is the threshold to judge the interest point. N , whose value is 64, represents the size of current CTU. Because the original interest points detection method uses more complicated filters to obtain $P(i, j)$, which is much time consuming, the relative weight 0.9 is used to minimize the error between them.

Features Selection for Bagged Tree Model

These 29 feature candidates are too many to be implemented. Hence, these candidates are ranked by the feature importance measurement technique introduced in Section II-B during the bagged tree training phase.

Table 2 top 10 most important features selected for different QPs and resolution classes with corresponding importance values.

Resolution class A								
QP22			QP27		QP32		QP37	
Rank	Feature	Importance	Feature	Importance	Feature	Importance	Feature	Importance
1	m_aveCBF	0.773	m_aveCBF	0.417	m_aveCBF	0.361	var_var_sub	0.255
2	var_var_sub	0.218	var_var_sub	0.292	var_var_sub	0.242	m_aveCBF	0.198
3	meanMain	0.128	varMain	0.078	varMain	0.107	totalBins	0.124
4	varMain	0.085	m_numAveInterestPoint	0.077	m_haarSumxy	0.078	varMain	0.083
5	m_nbCtuAboDepth	0.083	meanMain	0.066	m_numAveInterestPoint	0.066	m_haarSumxy	0.068
6	m_haarSumy	0.070	m_nbCtuAblDepth	0.063	m_nbCtuAblDepth	0.065	m_haarSumx	0.067
7	m_numAveInterestPoint	0.057	m_haarSumx	0.050	m_haarSumAbsxy	0.060	m_numAveInterestPoint	0.058
8	m_haarSumx	0.056	m_nbCtuLefDepth	0.042	m_nbCtuAbrDepth	0.057	m_nbCtuAboDepth	0.054
9	m_haarSumxy	0.055	m_nbCtuAbrRd	0.041	meanMain	0.049	totalCost	0.051
10	m_haarSumAbsxy	0.042	m_haarSumxy	0.039	totalBins	0.047	m_haarSumy	0.051
Resolution class B								
QP22			QP27		QP32		QP37	
Rank	Feature	Importance	Feature	Importance	Feature	Importance	Feature	Importance
1	m_aveCBF	0.738	m_aveCBF	0.565	m_aveCBF	0.384	m_nbCtuLefDepth	0.282
2	m_nbCtuLefDepth	0.288	m_nbCtuLefDepth	0.213	m_nbCtuLefDepth	0.231	m_aveCBF	0.254
3	m_nbCtuAboDepth	0.150	var_var_sub	0.173	var_var_sub	0.208	var_var_sub	0.201
4	var_var_sub	0.078	m_nbCtuAboDepth	0.095	m_nbCtuAboDepth	0.119	m_nbCtuAboDepth	0.097
5	m_nmse	0.071	totalBins	0.076	totalCost	0.088	totalBins	0.084
6	m_haarSumAbsx	0.065	m_haarSumAbsx	0.060	totalBins	0.080	varMain	0.065
7	m_haarSumx	0.053	m_numAveInterestPoint	0.051	m_numAveInterestPoint	0.048	totalDistortion	0.058
8	meanMain	0.045	totalCost	0.049	m_haarSumAbsx	0.047	m_nmse	0.051
9	totalDistortion	0.045	m_nmse	0.045	meanMain	0.044	meanMain	0.048
10	m_numAveInterestPoint	0.042	meanMain	0.041	m_nmse	0.033	m_numAveInterestPoint	0.046
Resolution class C								
QP22			QP27		QP32		QP37	
Rank	Feature	Importance	Feature	Importance	Feature	Importance	Feature	Importance
1	m_aveCBF	0.767	m_aveCBF	0.625	m_aveCBF	0.509	m_aveCBF	0.463
2	var_var_sub	0.164	var_var_sub	0.203	var_var_sub	0.226	var_var_sub	0.162
3	m_nbCtuAboDepth	0.073	m_haarSumAbsy	0.079	m_haarSumAbsy	0.094	totalBins	0.151
4	m_nmse	0.060	m_nmse	0.059	totalBins	0.093	varMain	0.112
5	var_mean_sub	0.039	m_nbCtuAboDepth	0.044	totalCost	0.064	m_haarSumAbsy	0.100
6	meanMain	0.036	m_haarSumy	0.040	m_nmse	0.060	meanMain	0.066
7	varMain	0.032	totalBins	0.031	varMain	0.040	m_nmse	0.060
8	m_numAveInterestPoint	0.023	var_mean_sub	0.030	meanMain	0.036	m_numAveInterestPoint	0.042
9	totalDistortion	0.021	meanMain	0.030	m_numAveInterestPoint	0.033	m_haarSumAbsxy	0.038
10	m_haarSumAbsxy	0.019	varMain	0.026	totalDistortion	0.030	totalCost	0.034
Resolution class D								
QP22			QP27		QP32		QP37	
Rank	Feature	Importance	Feature	Importance	Feature	Importance	Feature	Importance
1	m_aveCBF	0.500	m_aveCBF	0.260	var_var_sub	0.173	m_aveCBF	0.204
2	var_var_sub	0.089	var_var_sub	0.158	varMain	0.067	totalBins	0.122
3	varMain	0.043	varMain	0.063	m_nmse	0.065	var_var_sub	0.115
4	m_nmse	0.016	m_haarSumAbsx	0.047	m_haarSumAbsxy	0.051	totalCost	0.107
5	meanMain	0.007	m_dcom	0.038	m_aveCBF	0.046	meanMain	0.055
6	totalCost	0.007	totalBins	0.035	totalBins	0.044	m_nmse	0.050
7	m_costHadamard	0.005	var_mean_sub	0.027	totalCost	0.027	totalDistortion	0.034
8	m_sadHadamard	0.005	m_nmse	0.024	totalDistortion	0.026	m_haarSumAbsxy	0.026
9	m_numAveInterestPoint	0.005	m_numAveInterestPoint	0.022	m_haarSumAbsx	0.018	var_mean_sub	0.025
10	totalBins	0.004	m_haarSumAbsxy	0.017	m_sadHadamard	0.016	varMain	0.022
Resolution class E								
QP22			QP27		QP32		QP37	
Rank	Feature	Importance	Feature	Importance	Feature	Importance	Feature	Importance
1	m_aveCBF	0.363	var_var_sub	0.401	var_var_sub	0.413	var_var_sub	0.321
2	var_var_sub	0.262	m_nbCtuLefDepth	0.192	m_nbCtuLefDepth	0.220	m_nbCtuLefDepth	0.219
3	m_nbCtuLefDepth	0.104	m_aveCBF	0.188	m_aveCBF	0.201	m_nbCtuAblDepth	0.168
4	meanMain	0.090	m_nbCtuAboDepth	0.164	meanMain	0.099	meanMain	0.137
5	m_bitsHadamard	0.083	m_nbCtuAbrDepth	0.089	totalBins	0.083	m_aveCBF	0.121
6	m_nbCtuAblRd	0.078	meanMain	0.088	m_nbCtuAboDepth	0.060	m_bitsHadamard	0.100
7	m_nbCtuAboRd	0.059	totalBins	0.067	m_nbCtuAblDepth	0.055	varMain	0.079
8	totalBins	0.058	m_numAveInterestPoint	0.063	m_numAveInterestPoint	0.055	m_nmse	0.070
9	m_nbCtuLefRd	0.052	m_nbCtuAbrRd	0.061	varMain	0.052	totalBins	0.065
10	m_haarSumy	0.049	varMain	0.056	totalDistortion	0.048	totalCost	0.063

According to the ranking results, we find that the importance values are divided between the 10th and the 11th most important features. Besides, the importance values of the top 10 most important features stay in a relatively high level, while the importance values for the rest of feature candidates are generally very low. Thus, we pick the top 10 most important features as our final attributes. Hence, each bagged tree model has its own unique feature set consists of the top 10 most important features.

Table II lists the top 10 features of the classifiers for the five resolution classes (i.e. A, B, C, D and E) and different QP values. The importance values of corresponding features are also shown in Tabel II. As can be seen from Table II, these top 10 features of classifiers for different QPs and resolution classes are almost the same while their ranking changes.

ResNet Model Designing and Training

Four ResNets with the same architecture are used, and they dedicate video sequences under different QP values, i.e. 22, 27, 32 and 37, respectively. The ResNets used in this work are based on the 34-layer ResNet in [31]. Specific architecture of the ResNet is shown in Fig. 10. Each convolution block in Fig. 10 represents a convolution process followed by a normalization process and a ReLU calculation. The ResNet used in this work consists of 34 layers, and it takes a luma CU of size 32×32 as the input. With a fully connected layer at the very end, the ResNet outputs a vector to describe the probability of a CU belonging to each class. Using this probability vector and the ground truth vector of the target CU, the cross entropy loss is calculated according to the following equation

$$\mathcal{L} = \text{loss}(V_1, V_2) = -\log \left(\frac{\exp(V_1 \times V_2)}{\sum_{i=1}^{17} \exp(V_1 \times i)} \right) \quad (25)$$

where V_1 is the probability vector output by ResNet, and it contains the probabilities of a CU belonging to every class, V_2 is the class label ground truth of a CU, and it is in the form of one hot encoding, and $\mathcal{L}$ is the cross entropy loss between V_1 and V_2 . The cross entropy loss is used to measure the difference between the predicted label and the ground truth of the 32×32 sized CU. Since it is a convex function, the global minimum can be easily calculated by taking a partial derivative. Generally, training the ResNet model is finding suitable parameters to minimize the cross entropy loss of all the training samples.

We use the deep learning framework PyTorch to train the ResNets in this paper. Adam optimizer is used with learning rate 0.01. Specifically, the models are trained on 40 epoches with batch size 32.

Databases Generation for Training and Validation

In this paper, the bagged tree and the ResNet are jointly used to make an end-to-end decision of partition structure for a CTU. Specifically, the bagged tree is used to make the splitting decision for a 64×64 sized CU, while the ResNet is used to

Table 3 databases for bagged tree models of each QP and resolution class

Resolution Class	QP22	QP27	QP32	QP37
A (2560×1600)	DB_{A22}	DB_{A27}	DB_{A32}	DB_{A37}
B (1920×1080)	DB_{B22}	DB_{B27}	DB_{B32}	DB_{B37}
C (832×480)	DB_{C22}	DB_{C27}	DB_{C32}	DB_{C37}
D (416×240)	DB_{D22}	DB_{D27}	DB_{D32}	DB_{D37}
E (1280×720)	DB_{E22}	DB_{E27}	DB_{E32}	DB_{E37}

determine the partition structure of a 32×32 sized CU through a single prediction. So two kinds of databases are needed. One is constructed for the bagged tree model and the other one is constructed for the ResNet model.

First, five sequences of different resolutions are picked from the standard test sequences established by JCT-VC [36] as training sequences, which are *Traffic* (2560×2600), *ParkScene* (1920×1080), *BasketballDrill* (832×480), *BQSquare* (416×240) and *FourPeople* (1280×720). Then, these five sequences are encoded by HM16.7 under all intra configurations but four QPs (22, 27, 32, 37).

As for the databases used to train bagged tree models, we extract 10 key features of each CTU among a training sequence under a particular QP and resolution class. Besides, the splitting label of a CTU is also extracted to form the training database. So there are 20 databases in total, each of which is constructed for a QP value and a resolution class as shown in Table III. DB_{xy} represents the database used to train the resolution class x and QP y bagged tree model. For example, DB_{A22} is the database constructed to train the bagged tree model which is used to make the splitting decision of the CTUs in a sequence under resolution A and QP 22.

Specifically, to form the database used to train the bagged tree models, 36,000 CTU samples are randomly picked from the training sequences encoded by HM16.7. To balance the database, in each database the ratio between the samples of label split and nonsplit is 1. When there are not enough samples in the whole encoded sequence, all the nonsplit samples are picked, and the same number of split samples is randomly picked.

As for the databases constructed to train the ResNet, due to the end-to-end prediction structure, each partition structure of a 32×32 sized CU corresponds to a class label. There are 17 classes in total, and these classes together with the corresponding partition structure are shown in Fig. 11. As we can see from Fig. 11 that these classes cover all the possible partition results for a CU of size 32×32 . In this way, a partition structure can be predicted by ResNet through a single prediction.

In the databases for ResNet, each sample includes luma values of a 32×32 sized CU and its ground truth label according to the encoding results. To form the database, all 32×32 sized CUs in a training sequence are collected. Different with 20 bagged tree models, each ResNet model is designed for the sequences encoded under a certain QP, so there are 4 ResNet models needed to be generated totally. Hence, 4 databases are generated (denoted as DB-22, DB-27, DB-32 and DB-37), and they are constructed for the sequences encoded with QP 22, 27, 32 and 37, respectively. Taking the generation of DB-22 for example, we encode five training sequences with

QP 22 firstly. Then, luma values and class labels of 32×32 sized CUs from these five sequences are all collected to make up DB-22. Similarly, DB-27, DB-32 and DB-37 are generated. It is worth noting that, the difference among these four databases is only the labels. The luma pixel values between different databases are the same, because they are all from the same training sequences. In this paper, chroma CUs split just same as luma CUs. Hence, the models and datasets for chroma are not considered.

Experimental results and discussion

In order to verify the performance of the proposed bagged tree and ResNet joint fast algorithm (BTRNFA), we implemented this algorithm in the HEVC reference encoder HM16.7. The original software HM16.7 is available online (<https://hevc.lhi.fraunhofer.de/svn/svn.HEVCSoftware/tags/HM-16.7/>). Besides, the deep learning framework PyTorch is used to perform the training and the prediction of ResNet models. Test sequences were selected from the HEVC common test conditions (CTC) [36]. To analyze the fitting performance and the generalization of the models, BTRNFA is also carried out on five sequences used in the models training.

Experiments were conducted on an Aliyun host which has a windows 64-bit operation system. The host has 4 kernels of Intel(R) Xeon(R) Platinum 8269CY CPU @ 2.50GHz 3.10 GHz with a 16 GB memory. Coding parameters were set as default, and all results were generated under all-intra main configuration. The number of the frames to be encoded was set as the maximal value for each sequence according to the CTC.

There are three versions of the proposed BTRNFA. The first version is denoted as BTRNFA_{DT}, in which only the bagged tree model is active. In this situation, the CUs of depth 0 are partitioned according to the predicted labels output by a bagged tree model, and CUs of depth 1 and 2 are processed by RDO. The second version is BTRNFA_{ResNet}, in which only the ResNet model for the CUs of depths 1 and 2 is active. RDO is performed on the CUs of depth 0. The third version is BTRNFA_{joint}, which is also the most aggressive version. In the BTRNFA_{joint}, both the bagged tree model and the ResNet model are activated. As a result, BTRNFA_{joint} can achieve the most reduction of the encoding time.

First, BTRNFA_{DT}, BTRNFA_{ResNet} and BTRNFA_{joint} were carried out for performance comparison. We analyzed the contributions of the bagged tree model and the ResNet model on the proposed joint algorithm. In addition, we also compared the BTRNFA_{joint} algorithm with the CNN based state-of-the-art CU depth decision algorithms, i.e. the CU partition mode decision algorithm (PMDA) in [27], the complexity reduction method (CRM) in [29] and the fast CTU partition approach (FPA) in [26]. To measure rate-distortion performance, the BD-rate was calculated. Complexity reduction was measured by encoding time saving (denoted as TS), which was calculated according to the following equation

$$TS = \frac{time_{pro} - time_{ref}}{time_{ref}} \times 100\% \quad (26)$$

where $time_{pro}$ is the sequences encoding time of the proposed algorithm, and $time_{ref}$ is the encoding time spent by the reference encoder.

Table 4 BD-rate loss and time saving performance of three versions for the proposed BTRNFA

Class	Sequence	BTRNFA _{DT}		BTRNFA _{ResNet}		BTRNFA _{joint}	
		BD-rate (%)	TS (%)	BD-rate (%)	TS (%)	BD-rate (%)	TS (%)
A	<i>PeopleOnStreet</i>	0.16	17.21	2.19	51.85	2.28	62.21
	Average	0.16	17.21	2.19	51.85	2.28	62.21
B	<i>Kimono</i>	1.12	55.06	1.66	68.65	2.50	79.95
	<i>Cactus</i>	0.34	21.91	2.39	56.13	2.66	66.62
	<i>BasketballDrive</i>	0.98	32.31	3.12	62.14	3.83	73.16
	<i>BQTerrace</i>	0.11	20.24	1.68	52.62	1.77	62.89
	Average	0.64	32.38	2.21	59.88	2.69	70.66
C	<i>BQMall</i>	0.04	14.52	1.73	45.14	1.83	55.16
	<i>PartyScene</i>	0.00	10.79	0.61	33.08	0.62	42.80
	<i>RaceHorses</i>	0.08	18.65	1.38	49.70	1.51	60.50
	Average	0.04	14.65	1.24	42.64	1.32	52.82
D	<i>BasketballPass</i>	0.01	10.17	1.19	38.84	1.23	46.82
	<i>BlowingBubbles</i>	0.00	8.90	0.50	27.16	0.50	34.59
	<i>RaceHorses</i>	0.01	10.50	0.90	36.06	0.92	44.17
	Average	0.01	9.86	0.86	34.02	0.88	41.86
E	<i>Johnny</i>	0.24	39.92	3.24	64.64	3.54	75.59
	<i>KristenAndSara</i>	0.19	30.62	2.89	61.73	3.07	72.80
	Average	0.21	35.27	3.07	63.19	3.30	74.19
Overall Average		0.25	22.37	1.81	49.83	2.02	59.79

Performance of Three Versions of The Proposed Algorithm

In this section, we discuss BD-rate and time saving performance of three versions of the proposed BTRNFA (i.e. BTRNFA_{DT}, BTRNFA_{ResNet} and BTRNFA_{joint}). Experimental results of each version are shown in Table IV.

We can see from Table IV that BTRNFA_{DT} achieves 0.25% BD-rate loss with complexity reduction 22.37% on average. Its BD-rate loss is the least one among these three versions of the proposed algorithm, and its encoding time saving is also the least. The average 0.25% BD-rate loss shows our bagged tree model is trained well and has a high prediction accuracy on test sequences. However, the time saving results are not the highest, because only the bagged tree model is activated in this version of the proposed algorithm, and the encoding complexity among CU partitioning of depths 1 and 2 is not considered to be reduced.

On the contrary, the bagged tree model is disabled in BTRNFA_{ResNet}, and only the ResNet model is active. In this case, RDO is performed on each CTU while the partition structure of CUs with depth 1 is generated by the end-to-end prediction. This version of the proposed algorithm achieves 1.81% BD-rate loss with a time saving of 49.83% on average compared with the original algorithm in HM16.7. We can see from Table IV that, compared with BTRNFA_{DT}, the time saving of BTRNFA_{ResNet} increases about 27.46% with a 1.56% sacrifice in terms of BD-rate. The results of Table IV show that the prediction accuracy of ResNet is not quite high so that more BD-rate loss is caused. On the other hand, we can also see that more complexity can be saved with CU depths 1 and 2 than that in CU with depth 0.

BTRNFA_{joint} is the most aggressive version, in which both the bagged tree model for CTU and the ResNet model for CU in depth 1 are activated. Its results are shown in the last two columns of Table IV. BTRNFA_{joint} achieves as much as 59.79% complexity reduction with only 2.02% BD-rate loss. Since it has the most time saving among these three algorithm versions, its BD-rate loss performance is acceptable.

As we can see, the performance of the proposed BTRNFA_{joint} varies with sequences listed in Table IV. To figure out what kinds of videos are suitable for BTRNFA_{joint} such that the good results can be obtained, we use the ratio of time saving versus BD-rate loss to conduct further analysis. Considering this in Table IV, BTRNFA_{joint} performs better in videos of resolution class C and D than those of A, B and E. It means our method has better performance in videos of low resolution, which maybe due to a high prediction accuracy on CTUs in low resolution videos. Besides, observing performance differences on sequences of the same resolution class, we can see *BQTerrace* shows better results than *BasketballDrive*. Considering BD-rate and time saving jointly, *PartyScene* and *BlowingBubbles* outperform *BQMall* and *BasketballPass*, respectively. It is easy to find the sequences *BasketballDrive*, *BQMall* and *BasketballPass* all contain multiple objects moving fast in scene. Sequences which contain many still or slow-moving contents make our method easy to achieve good results. Thus, we can conclude that videos of low resolution and less moving contents are much suitable for the proposed method.

As for the performance of our method on other QP settings, on the one hand we can use the current models to directly encode videos under other QPs. This does not affect the RD performance, but only affects the benefit of time savings. On the other hand, we can train one model for a particular QP. This will obviously result in better RD performance and more time savings as well, since it is targeted to the QP. In short, experiments on existing QPs have verified the effectiveness of our algorithm. For other QPs, our method can achieve a better trade-off between RD performance and time saving by training the corresponding models.

Prediction Accuracy and Inferring Overhead of the proposed method

To analyze the performance of the proposed BTRNFA_{joint} on each stage, we present the prediction accuracy of bagged tree models and ResNet models as well as the GPU inferring time overhead.

Firstly, Table V shows the prediction accuracy of different bagged tree models for CUs in depth 0. The average accuracy of these 20 bagged tree models is 89.86% on the splitting prediction for CUs in depth 0. Prediction accuracy is quite stable on around 90% for four QP values, which can be observed from the last row of Table V. Besides, we find these four bagged tree models trained for resolution class A all perform below average, perhaps because the video sequence used for training is not representative.

Besides, from different levels, Table VI presents the prediction accuracy of four ResNets. On the overall prediction level, four ResNets show quite stable accuracy performances which distribute between 56% and 58%. Although the overall accuracy is low, their accuracy on different depth levels is quite good. In Table VI, the prediction level depth 1 means the the split flag prediction (i.e. split or non-split) of CUs in depth 1. And the prediction level depth 2 means the splitting decision prediction for CUs in depth 2 which is also a binary decision. As we can see from the third row of Table VI, the highest split flag prediction accuracy for CU in depth 1 is 89.34% of the ResNet trained for QP 27. Even the lowest split flag prediction accuracy for CU in depth 1 is as high as 86.43%. And the accuracy of four ResNets on depth level 2 is similar as that of depth level 1. It can be observed that although

Table 5 prediction accuracy of different Bagged tree models

Resolution Class	QP22	QP27	QP32	QP37
A (2560×1600)	90.66%	94.13%	94.09%	94.79%
B (1920×1080)	79.21%	77.15%	78.25%	78.95%
C (832×480)	95.15%	95.44%	96.42%	96.44%
D (416×240)	98.83%	98.78%	98.12%	96.91%
E (1280×720)	66.11%	83.50%	91.99%	92.26%
Average	85.99%	89.80%	91.78%	91.87%

Table 6 Prediction accuracy from different levels of four ResNets.

Prediction Level	QP22	QP27	QP32	QP37
Overall	57.20%	58.86%	58.12%	56.51%
Depth 1	86.43%	89.34%	88.68%	86.82%
Depth 2	85.36%	86.51%	86.76%	86.64%

the overall prediction accuracy of four ResNet models is not high, the prediction accuracy of splitting decision at each depth level is quite high. This is because the difference between classes is not obvious in the multi-classification task for CU partitioning, and slight prediction error will not lead to the splitting decision error at each depth level. The high prediction accuracy of ResNets at each depth level also verifies the correctness of our algorithm.

This paper aims at reducing the encoding complexity by deciding partitioning structure of a CTU early, so the complexity introduced by the ResNet model must be considered. As a result, an appropriate network structure shown in Fig. 10 is designed, and the encoding time of the proposed method has included the corresponding inference time which is shown in Table VII. It can be observed from Table VII that the inferring overhead of four ResNets is about 1% of the original HM encoder. It is worth noting that the inference process is carried out on GPU, so the consumption is very low.

0.1 Performance Comparison with CNN Based State-Of-The-Art Algorithms.

This section discusses the comparisons of the encoding performance between the proposed BTRNFA_{joint} and other two CNN based intra mode fast decision algorithms. These two algorithms that BTRNFA_{joint} is going to compare with are implemented on reference software of different version. In order to make a fair comparison, we apply the proposed BTRNFA_{joint} to these reference software versions of the two comparison objects respectively. The corresponding results are presented in the following paragraphs.

The PMDA in [27] reduces the hardware complexity of the encoder by decreasing more than two CTU partition modes for RDO. It achieves 60.86% complexity reduction on average, and the BD-rate loss increases by as much as 2.74% on HM12.0. Compared with their results, the proposed BTRNFA_{joint} saves about 0.73% BD-rate loss with a negligible encoding time increment of 0.54% on average. Detailed

Table 7 Inferring overhead compared with the original HM16.7 encoding time.

	QP22	QP27	QP32	QP37
Percentage	0.85%	1.00%	1.12%	1.24%

Table 8 performance comparison between the proposed BTRNFA_{joint} and PMDA under HM12.0 all intra coding

Class	Sequence	PMDA		BTRNFA _{joint}	
		BD-rate (%)	TS (%)	BD-rate (%)	TS (%)
A 2560×1600	<i>PeopleOnStreet</i>	2.27	61.80	2.29	63.30
	Average	2.27	61.80	2.29	63.30
B 1920×1080	<i>Kimono</i>	2.47	62.60	2.51	79.68
	<i>Cactus</i>	2.59	60.40	2.65	66.73
	<i>BasketballDrive</i>	3.49	70.90	3.84	72.23
	<i>BQTerrace</i>	2.26	62.40	1.77	61.83
	Average	2.70	64.08	2.69	70.12
C 832×480	<i>BQMall</i>	2.93	57.80	1.82	56.26
	<i>PartyScene</i>	2.19	51.10	0.61	44.59
	<i>RaceHorses</i>	2.08	56.20	1.50	61.09
	Average	2.40	55.03	1.31	53.98
	<i>BasketballPass</i>	2.89	59.30	1.20	48.24
D 416×240	<i>BlowingBubbles</i>	2.54	53.60	0.49	36.27
	<i>RaceHorses</i>	2.43	53.60	0.89	45.26
	Average	2.62	55.50	0.86	43.26
E 1280×720	<i>Johnny</i>	4.42	72.20	3.55	75.40
	<i>KristenAndSara</i>	3.12	69.30	3.06	73.25
	Average	3.77	70.75	3.30	74.33
Overall Average		2.74	60.86	2.01	60.32

results shown in Table VIII indicate that BTRNFA_{joint} exceeds PMDA on BD-rate with a quite close time saving.

As for the encoding complexity reducing approach CRM in [29], an early terminated hierarchical CNN is proposed to decide the CU partitioning. The encoding complexity on HEVC intra mode decision is dramatically reduced by replacing the RDO with the predicted partitioning result. On average, 61.91% of encoding complexity is saved with 2.24% of BD-rate loss increment compared with the original HM16.5. Comparison results are shown in Table IX. Compared with CRM, the proposed BTRNFA_{joint} achieves 0.22% less BD-rate loss to 2.02% with only 2.60% more encoding time. Besides, only one ResNet model is required in the BTRNFA_{joint}, while the number of CNN models used in CRM is as large as 3. It means that the proposed algorithm has advantages no matter on models training or encoder implementation.

Generally speaking, the proposed BTRNFA_{joint} outperforms two CNN based state-of-the-art algorithms. On average, additionally 0.73% and 0.22% BD-rates are saved, compared with PMDA and CRM respectively. Though the time saving of BTRNFA_{joint} is very close to theirs, our approach shows great advance in terms of the model number required in encoding by introducing an end-to-end partitioning solution. Thus, the complexity of model training and implementation is quite low and competitive for the proposed BTRNFA_{joint}.

Comparisons of CU Partition Results

To compare the differences of the CU partition results between the original algorithm in HM16.7 and the proposed BTRNFA_{joint} algorithm, we encode the 180th frame of the sequence *RaceHorses* (416×240) with QP 22 by using these two algorithms. Fig. 12 presents the CU partition results of the original HM16.7. In Fig. 12, black line represents the CU boundaries. Fig. 13 shows the CU partition results of the proposed BTRNFA_{joint}. In Fig. 13, we use the gold and red lines to rep-

Table 9 performance comparison between the proposed BTRNFA_{joint} and CRM under HM16.5 all intra coding

Class	Sequence	CRM		BTRNFA _{joint}	
		BD-rate (%)	TS (%)	BD-rate (%)	TS (%)
A 2560×1600	<i>PeopleOnStreet</i>	2.37	61.00	2.28	62.08
	Average	2.37	61.00	2.28	62.08
B 1920×1080	<i>Kimono</i>	2.59	83.54	2.53	79.78
	<i>Cactus</i>	2.27	60.96	2.65	65.98
	<i>BasketballDrive</i>	4.27	76.32	3.84	71.71
	<i>BQTerrace</i>	1.84	64.72	1.77	60.73
	Average	2.74	71.39	2.69	69.55
C 832×480	<i>BQMall</i>	2.09	58.42	1.82	54.80
	<i>PartyScene</i>	0.66	44.50	0.61	42.48
	<i>RaceHorses</i>	1.97	57.12	1.50	60.21
	Average	1.57	53.35	1.31	52.50
	<i>BasketballPass</i>	1.84	56.42	1.25	46.79
D 416×240	<i>BlowingBubbles</i>	0.62	40.54	0.50	34.43
	<i>RaceHorses</i>	1.32	55.76	0.93	44.11
	Average	1.26	50.91	0.89	41.78
E 1280×720	<i>Johnny</i>	3.82	70.68	3.55	75.15
	<i>KristenAndSara</i>	3.46	74.86	3.05	72.74
	Average	3.64	72.77	3.30	73.95
Overall Average		2.24	61.91	2.02	59.31

resent the differences of the CU boundaries between the original HM16.7 and the BTRNFA_{joint}.

It can be observed from Fig. 13 that the same CU partition results take up a quite large portion. Setting the results in Fig. 12 as the baseline, we observe that almost all the partition results upon background and plain areas stay the same as those in Fig. 13. This observation indicates that our bagged tree model and ResNet model achieve a high accuracy on CUs with flat contents. Besides, as we can see, the differences between Fig. 12 and Fig. 13 mainly exist among the edge areas. It means that our algorithm needs should be improved on the prediction upon some particular CUs which cross the boundaries of objects.

In general, compared with the optimal CU partition results generated by the original HM16.7 as shown in Fig. 12, there are not much differences existing in the CU partition results generated by the proposed BTRNFA_{joint}. Our BTRNFA_{joint} algorithm performs well, and its CU partition results are satisfactory.

Conclusion

In this paper, we propose a bagged tree and ResNet based joint end-to-end CTU partition structure prediction algorithm for HEVC intra coding. First, we construct effective datasets for our models. Then the bagged tree model is designed to predict the splitting flag of a CTU. Besides, a ResNet based model is designed and used on each CU of size 32×32, so that the partition structure can be determined through a single prediction. In this way, the traditional brute-force RDO is replaced, so that lots of encoding complexity is saved. Compared with the original HM16.7, the proposed algorithm saves as much as 59.79% encoding time with only 2.02% BD-rate loss on average. Furthermore, our approach defeats three CNN based state-of-the-art fast algorithms for intra mode decision, which shows great application prospect of our approach.

Abbreviations

HEVC: high efficiency video coding, VVC: versatile video coding, AV2: AOMedia video 2, CTU: coding tree unit, ResNet: residual network, CU: coding unit, AVC: advanced video coding, AVS: audio video coding standard, AV1: AOMedia video 1, JCT-VC: Joint Collaborative Team on Video Coding, QTBT: quad-tree plus binary tree, RDO: rate-distortion optimization, SVM: support vector machine, CNN: convolutional neural network, LSTM: long-short term memory, QP: quantization parameter, CBF: coded block flag, BTRNFA: bagged tree and ResNet joint fast algorithm, CTC: common test conditions, PMDA: partition mode decision algorithm, CRM: complexity reduction method, FPA: fast CTU partition approach.

Declarations

Availability of data and materials

The datasets used and analysed during the current study are available from the corresponding author on reasonable request.

Competing interests

The authors declare that they have no competing interests.

Funding

This work is supported in part by the National Key Research and Development Program of China (Grant No. 2020YFB1805403), the National Natural Science Foundation of China (Grant Nos. 62032002 and 61932005) and the 111 Project (Grant No. B21049).

Authors' contributions

YL proposed the framework of the paper and wrote the first draft. LL and NL improved the paper framework. YF assisted in completing the algorithm experiment. LL and HP revised the manuscript, and all authors conducted the final proofreading of the manuscript.

Acknowledgements

Not applicable.

Authors' information

YL received the B.S. degree in information engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2015. Now he is pursuing his Ph.D. degree in computer science at Beijing University of Posts and Telecommunications, Beijing, China. He was a visiting scholar to Santa Clara University, CA, USA, in 2017-2019. His research directions are video coding, machine learning and image processing.

LL received the M.S. degree in circuit and system from Yanshan University, Qinhuangdao, China, in 2003, and the Ph.D. degree in signal and information processing from Beijing University of Posts and Telecommunications, Beijing, China, in 2006. She is currently a professor at the School of CyberSpace Security, Beijing University of Posts and Telecommunications, China. The winner of National Excellent Doctoral Thesis, the New Century Excellent Talents in University, the winner of Henry Folk Education Foundation, the winner of Hong Kong Scholar Award, the winner of Beijing Higher Education Program for Young Talents, the winner of Outstanding Youth Award of Chinese Association for Cryptology Research. Visiting Potsdam Institute for Climate Impact Research, Potsdam, Germany from July 2011 to June 2012. Engaged in research of compressive sensing, complex networks, swarm intelligence and network security; having published more than 100 papers.

YF received the B.S. degree in network engineering from Hebei university of technology, Tianjin, China, in 2016. Now she is working towards the Ph.D. degree in cyberspace security at Beijing University of Posts and Telecommunications, Beijing, China. Her major interests are compressive sensing, image encryption and deep learning.

HP received the M.S. degree in system engineering from Shenyang University of Technology, Shenyang, China, in 2006, and the Ph.D. degree in signal and information processing from Beijing University of Posts and Telecommunications, Beijing, China, in 2010. He is currently a professor at the School of CyberSpace Security, Beijing University of Posts and Telecommunications, China. His research interests include information security, network security, complex networks and control of dynamical systems. Dr. H. Peng is the co-author of over 50 scientific papers.

NL (S'88-M'90-SM'99-F'08) received the B.Eng. degree from the National University of Singapore, Singapore, in 1981, and the M.S. and Ph.D. degrees from the University of Louisiana at Lafayette, Lafayette, LA, USA, in 1985 and 1989, respectively. From 2002 to 2010, he was an Associate Dean with the School of Engineering, Santa Clara University, Santa Clara, CA, USA. He was the Sanfilippo Family Chair Professor, and is currently the Wilmot J. Nicholson Family Chair Professor and the Chair with the Department of Computer Science and Engineering, Santa Clara University. He is/was also a Consulting Professor with the National University of Singapore, a Guest Professor with Tianjin University, Tianjin, China, a Guest Professor with Shanghai Jiao Tong University, Shanghai, China, a Cuiying Chair Professor with Lanzhou University, Lanzhou, China, a Chair Professor and Minjiang Scholar with Fuzhou University, Fuzhou, China, a Distinguished Professor with the Xi'an University of Posts and

Telecommunications, Xi'an, China, a Guest Professor with the Zhongyuan University of Technology, Zhengzhou, China, and an Outstanding Overseas Scholar with the Shanghai University of Electric Power, Shanghai, China. He has authored or coauthored over 220 publications and seven adopted standard contributions. He has been granted nearly 20 U.S. patents so far. Dr. Ling is an IEEE Fellow due to his contributions to video coding algorithms and architectures. He is also an IET Fellow. He was named as an IEEE Distinguished Lecturer twice and was also an APSIPA Distinguished Lecturer. He was a recipient of the IEEE ICCE Best Paper Award (First Place) and the Umedia Best/Excellent Paper Award three times, six awards from Santa Clara University, four at the University level (Outstanding Achievement, Recent Achievement in Scholarship, President's Recognition, and Sustained Excellence in Scholarship), and two at the School/College level (Researcher of the Year and Teaching Excellence). He was a Keynote Speaker for IEEE APCCAS, VCVP (twice), JCPC, IEEE ICAST, IEEE ICIEA, IET FC Umedia, IEEE Umedia, IEEE ICCIT, and Workshop at XUPT (twice), and a Distinguished Speaker for IEEE ICIEA. He has served as a General Chair/Co-Chair for IEEE Hot Chips, VCVP (twice), IEEE ICME, Umedia (seven times), and IEEE SiPS. He was an Honorary Co-Chair for IEEE Umedia 2017. He has also served as a Technical Program Co-Chair for IEEE ISCAS, APSIPA ASC, IEEE APCCAS, IEEE SiPS (twice), DCV, and IEEE VCIP. He was a Technical Committee Chair for IEEE CASCOM TC and IEEE TCMM, and has served as a Guest Editor or an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—REGULAR PAPERS, the IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING, IEEE ACCESS, Springer JSPS, and Springer MSSP.

Author details

¹ Information Security Center, State Key Laboratory of Networking and Switching Technology, National Engineering Laboratory for Disaster Backup and Recovery, Beijing University of Posts and Telecommunications, Beijing, 100876, China. ²Department of Computer Science and Engineering, Santa Clara University, Santa Clara, CA 95053, USA.

References

- Sullivan, G.J., Ohm, J., Han, W., Wiegand, T.: Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on Circuits and Systems for Video Technology* **22**(12), 1649–1668 (2012)
- Lee, D., Jeong, J.: Fast intra coding unit decision for high efficiency video coding based on statistical information. *Signal Processing: Image Communication* **55**, 121–129 (2017)
- Liu, Z., Lin, T.-L., Chou, C.-C.: HEVC coding-unit decision algorithm using tree-block classification and statistical data analysis. *Multimedia Tools and Applications* **76**(6), 9051–9072 (2017)
- Ahn, S., Lee, B., Kim, M.: A novel fast cu encoding scheme based on spatiotemporal encoding parameters for HEVC inter coding. *IEEE Transactions on Circuits and Systems for Video Technology* **25**(3), 422–435 (2014)
- Min, B., Cheung, R.C.: A fast cu size decision algorithm for the HEVC intra encoder. *IEEE Transactions on Circuits and Systems for Video Technology* **25**(5), 892–896 (2014)
- Xiong, J., Li, H., Wu, Q., Meng, F.: A fast HEVC inter cu selection method based on pyramid motion divergence. *IEEE Transactions on Multimedia* **16**(2), 559–564 (2013)
- Shen, L., Zhang, Z., Liu, Z.: Effective cu size decision for HEVC intracoding. *IEEE Transactions on Image Processing* **23**(10), 4232–4241 (2014)
- Shen, L., Liu, Z., Zhang, X., Zhao, W., Zhang, Z.: An effective cu size decision method for HEVC encoders. *IEEE Transactions on Multimedia* **15**(2), 465–470 (2012)
- Shen, L., Zhang, Z., An, P.: Fast cu size decision and mode decision algorithm for HEVC intra coding. *IEEE Transactions on Consumer Electronics* **59**(1), 207–213 (2013)
- Zhang, Y., Pan, Z., Li, N., Wang, X., Jiang, G., Kwong, S.: Effective data driven coding unit size decision approaches for HEVC intra coding. *IEEE Transactions on Circuits and Systems for Video Technology* **28**(11), 3208–3222 (2017)
- Liu, X., Li, Y., Liu, D., Wang, P., Yang, L.T.: An adaptive cu size decision algorithm for HEVC intra prediction based on complexity classification using machine learning. *IEEE Transactions on Circuits and Systems for Video Technology* **29**(1), 144–155 (2017)
- Zhang, T., Sun, M.-T., Zhao, D., Gao, W.: Fast intra-mode and cu size decision for HEVC. *IEEE Transactions on Circuits and Systems for Video Technology* **27**(8), 1714–1726 (2016)
- Zhu, L., Zhang, Y., Kwong, S., Wang, X., Zhao, T.: Fuzzy svm-based coding unit decision in HEVC. *IEEE Transactions on Broadcasting* **64**(3), 681–694 (2017)
- Grellert, M., Zatt, B., Bampi, S., da Silva Cruz, L.A.: Fast coding unit partition decision for HEVC using support vector machines. *IEEE Transactions on Circuits and Systems for Video Technology* **29**(6), 1741–1753 (2018)
- Zhang, Y., Kwong, S., Wang, X., Yuan, H., Pan, Z., Xu, L.: Machine learning-based coding unit depth decisions for flexible complexity allocation in high efficiency video coding. *IEEE Transactions on Image Processing* **24**(7), 2225–2238 (2015)
- Qin, J., Bai, H., Zhang, M., Zhao, Y.: Fast intra coding algorithm for HEVC based on decision tree. *IEEE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* **100**(5), 1274–1278 (2017)
- Zhang, D., Duan, X., Zang, D.: Decision tree based fast cu partition for HEVC lossless compression of medical image sequences. In: 2017 9th International Conference on Wireless Communications and Signal Processing (WCSP), pp. 1–6 (2017). IEEE
- Correa, G., Assuncao, P.A., Agostini, L.V., da Silva Cruz, L.A.: Fast HEVC encoding decisions using data mining. *IEEE Transactions on Circuits and Systems for Video Technology* **25**(4), 660–673 (2014)
- Ruiz, D., Fernández-Escribano, G., Adzic, V., Kalva, H., Martínez, J.L., Cuenca, P.: Fast cu partitioning algorithm for HEVC intra coding using data mining. *Multimedia tools and applications* **76**(1), 861–894 (2017)
- Zhu, S., Zhang, C.: A fast algorithm of intra prediction modes pruning for HEVC based on decision trees and a new three-step search. *Multimedia Tools and Applications* **76**(20), 21707–21728 (2017)
- Lee, D., Jeong, J.: Fast cu size decision algorithm using machine learning for HEVC intra coding. *Signal Processing: Image Communication* **62**, 33–41 (2018)

CUpartitionStructure.PNG

Figure 1 Recursive partition process of a CTU in the HEVC intra coding.

CUpartitionSample.PNG

Figure 2 A partition example of a CTU and the corresponding quad-tree splitting structure in the HEVC intra coding.

theStructureOfBaggedTree.PNG

Figure 3 The structure of bagged tree. An illustration of how the final prediction for a sample input is generated by a bagged tree model, which is composed of multiple decision trees, each trained on a random subset of data.

ResNetIntroduction.PNG

Figure 4 Structure of a building block in ResNet and two examples of basic block for ResNet construction of different depth. (a) the basic structure with a shot cut for ResNet. (b) an example of a basic block used to build a shallow ResNet. (c) the “bottleneck” used by stacking to construct a deep ResNet.

flowChartOftheProposedAlgorithm.PNG

Figure 5 Flowchart of the proposed bagged tree and ResNet based joint fast CTU partition algorithm.

22. Kim, H.-S., Park, R.-H.: Fast cu partitioning algorithm for HEVC using an online-learning-based bayesian decision rule. *IEEE Transactions on Circuits and Systems for Video Technology* **26**(1), 130–138 (2015)
23. 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 Transactions on Circuits and Systems for Video Technology* (2019)
24. Galpin, F., Racapé, F., Jaiswal, S., Bordes, P., Le Lannec, F., François, E.: Cnn-based driving of block partitioning for intra slices encoding. In: 2019 Data Compression Conference (DCC), pp. 162–171 (2019). IEEE
25. Huang, C., Peng, Z., Chen, F., Jiang, Q., Jiang, G., Hu, Q.: Efficient cu and pu decision based on neural network and gray level co-occurrence matrix for intra prediction of screen content coding. *IEEE Access* **6**, 46643–46655 (2018)
26. Shi, J., Gao, C., Chen, Z.: Asymmetric-kernel cnn based fast ctu partition for HEVC intra coding. In: 2019 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5 (2019). IEEE
27. Liu, Z., Yu, X., Gao, Y., Chen, S., Ji, X., Wang, D.: Cu partition mode decision for HEVC hardwired intra encoder using convolution neural network. *IEEE Transactions on Image Processing* **25**(11), 5088–5103 (2016)
28. Kim, K., Ro, W.W.: Fast cu depth decision for HEVC using neural networks. *IEEE Transactions on Circuits and Systems for Video Technology* **29**(5), 1462–1473 (2018)
29. Xu, M., Li, T., Wang, Z., Deng, X., Yang, R., Guan, Z.: Reducing complexity of HEVC: A deep learning approach. *IEEE Transactions on Image Processing* **27**(10), 5044–5059 (2018)
30. Caruana, R., Elhawary, M., Munson, A., Riedewald, M., Sorokina, D., Fink, D., Hochachka, W.M., Kelling, S.: Mining citizen science data to predict prevalence of wild bird species. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 909–915 (2006)
31. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
32. Li, Y., Li, L., Fang, Y., Peng, H., Yang, Y.: Bagged tree based frame-wise beforehand prediction approach for HEVC intra-coding unit partitioning. *Electronics* **9**(9), 1523 (2020)
33. Shen, X., Yu, L.: Cu splitting early termination based on weighted svm. *EURASIP journal on image and video processing* **2013**(1), 4 (2013)
34. Grellert, M., Bampi, S., Correa, G., Zatt, B., da Silva Cruz, L.A.: Learning-based complexity reduction and scaling for HEVC encoders. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1208–1212 (2018). IEEE
35. Bay, H., Tuytelaars, T., Van Gool, L.: Surf: Speeded up robust features. In: European Conference on Computer Vision, pp. 404–417 (2006). Springer
36. Bossen, F., et al.: Common test conditions and software reference configurations. *JCTVC-L1100* **12**, 7 (2013)

CTUpartitionStructureDetermination.PNG

Figure 6 An example of CTU partition structure determination process with the corresponding partition quad-tree. PL is the predicted label for a 32×32 sized CU output by ResNet. PL is set as an integer from 1 to 17.

sobelFilters.PNG

Figure 7 Sobel filters for different directions. (a) example of a 3×3 square pixel block to be filtered by Sobel filters. (b) Sobel filter for horizontal direction. (c) Sobel filter for vertical direction. (d, e) Sobel filters for 45° and 135° , respectively.

haarFilters.PNG

Figure 8 Haar filters for different directions. (a) example of a 2×2 square pixel block to be filtered by Haar filters. (b) Haar filter for horizontal direction. (c) Haar filter for vertical direction. (d) Haar filters for diagonal direction.

filtersForInterestPointDetection.PNG

Figure 9 Three filters used for interest point detection in horizontal, vertical and diagonal directions. (a) the filter used in the horizontal direction. (b) the filter used in the vertical direction. (c) the filter used in the diagonal direction. (a), (b) and (c) are performed on each pixel of a CTU, and the corresponding filtering responses are D_{xx} , D_{yy} and D_{xy} , respectively.

ResNetArchitecture.PNG

Figure 10 Architecture of the ResNet used in this paper. The dotted lines represent the shortcuts which increase the dimensions.

ResNetClassificationLabels.PNG

Figure 11 Class labels and corresponding partition structures of a 32×32 sized CU for ResNet multi-classification. Variate $label$ is the class index of different partitions for a CU of size 32×32 .

partitionResultRaceHorses416x240frame180original.PNG

Figure 12 Partition results of the 180th frame in the sequence *RaceHorses* (416×240), which is encoded by the original HM 16.7 with QP 22. Black line represents the splitting boundaries between CUs.

partitionResultRaceHorses416x240frame180BTRNFAjointedited.PNG

Figure 13 Partition results of the 180th frame in the sequence *RaceHorses* (416×240), which is encoded by the third version BTRNFA_{joint} of the proposed algorithm with QP 22. Black line represents the same partition results as those of the original HM16.7. Gold line represents the boundaries of CUs, which are split by the original HM16.7 but are not split by the BTRNFA_{joint}. The boundaries of CUs, which are decided non-split by the original HM16.7 but are split by the BTRNFA_{joint}, are shown with red line.

Figures

a CU in depth 0

Figure 1

Recursive partition process of a CTU in the HEVC intra coding.

Figure 2

A partition example of a CTU and the corresponding quad-tree splitting structure in the HEVC intra coding.

Figure 3

The structure of bagged tree. An illustration of how the final prediction for a sample input is generated by a bagged tree model, which is composed of multiple decision trees, each trained on a random subset of data.

Figure 4

Structure of a building block in ResNet and two examples of basic block for ResNet construction of different depth. (a) the basic structure with a shot cut for ResNet. (b) an example of a basic block used to build a shallow ResNet. (c) the “bottleneck” used by stacking to construct a deep ResNet.

Figure 5

Flowchart of the proposed bagged tree and ResNet based joint fast CTU partition algorithm.

Figure 6

An example of CTU partition structure determination process with the corresponding partition quad-tree. P L is the predicted label for a 32×32 sized CU output by ResNet. P L is set as an integer from 1 to 17.

Figure 7

Sobel filters for different directions. (a) example of a 3 × 3 square pixel block to be filtered by Sobel filters. (b) Sobel filter for horizontal direction. (c) Sobel filter for vertical direction. (d, e) Sobel filters for 45.

(a)

(b)

(c)

(d)

Figure 8

Haar filters for different directions. (a) example of a 2 × 2 square pixel block to be filtered by Haar filters. (b) Haar filter for horizontal direction. (c) Haar filter for vertical direction. (d) Haar filters for diagonal

direction.

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
1	1	1	-2	-2	-2	1	1	1
1	1	1	-2	-2	-2	1	1	1
1	1	1	-2	-2	-2	1	1	1
1	1	1	-2	-2	-2	1	1	1
1	1	1	-2	-2	-2	1	1	1
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

(a)

0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	-2	-2	-2	-2	-2	0	0
0	0	-2	-2	-2	-2	-2	0	0
0	0	-2	-2	-2	-2	-2	0	0
0	0	-2	-2	-2	-2	-2	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0

(b)

0	0	0	0	0	0	0	0	0
0	1	1	1	0	-1	-1	-1	0
0	1	1	1	0	-1	-1	-1	0
0	1	1	1	0	-1	-1	-1	0
0	0	0	0	0	0	0	0	0
0	-1	-1	-1	0	1	1	1	0
0	-1	-1	-1	0	1	1	1	0
0	-1	-1	-1	0	1	1	1	0
0	0	0	0	0	0	0	0	0

(c)

Figure 9

Three filters used for interest point detection in horizontal, vertical and diagonal directions. (a) the filter used in the horizontal direction. (b) the filter used in the vertical direction. (c) the filter used in the diagonal

direction. (a), (b) and (c) are performed on each pixel of a CTU, and the corresponding filtering responses are Dxx, Dyy and Dxy, respectively.

Figure 10

Architecture of the ResNet used in this paper. The dotted lines represent the shortcuts which increase the dimensions.

Figure 11

Class labels and corresponding partition structures of a 32x32 sized CU for ResNet multi-classification. Variate label is the class index of different partitions for a CU of size 32x32.

Figure 12

Partition results of the 180th frame in the sequence RaceHorses (416×240), which is encoded by the original HM 16.7 with QP 22. Black line represents the splitting boundaries between CUs.

Figure 13

Partition results of the 180th frame in the sequence RaceHorses (416×240), which is encoded by the third version BTRNFAjoint of the proposed algorithm with QP 22. Black line represents the same partition results as those of the original HM16.7. Gold line represents the boundaries of CUs, which are split by the original HM16.7 but are not split by the BTRNFAjoint. The boundaries of CUs, which are decided non-split by the original HM16.7 but are split by the BTRNFAjoint, are shown with red line