FAST LEARNING-BASED SPLIT TYPE PREDICTION ALGORITHM FOR VVC

Dayong Wang[1], Liulin Chen[2], Xin Lu[3], Frederic Dufaux[4], Weisheng Li[5], Ce Zhu[6]

[1]Chongqing Key Laboratory on Big Data for Bio Intelligence, [2]School of Computer Science and Technology, [5]Chongqing Key Laboratory of Image Cognition, Chongqing University of Posts and Telecommunications.
[3]Faculty of Computing, Engineering and Media (CEM), De Montfort University,
[4]Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire des signaux et systèmes,
[6]University of Electronic Science and Technology of China
wangdayong@cqupt.edu.cn, 1179483700@qq.com, xin.lu@dmu.ac.uk,
frederic.dufaux@l2s.centralesupelec.fr, liws@cqupt.edu.cn, eczhu@uestc.edu.cn

ABSTRACT

As the latest video coding standard, Versatile Video Coding (VVC) is highly efficient at the cost of very high coding complexity, which seriously hinders its widespread application. Therefore, it is very crucial to improve its coding speed. In this paper, we propose a learning-based fast split type (ST) prediction algorithm for VVC using a deep learning approach. We first construct a large-scale database containing sufficient STs with diverse video resolution and content. Next, since the ST distributions of coding units (CUs) of different sizes are significantly distinct, so we separately design neural networks for all different CU sizes. Then, we merge ambiguous STs into four merged classes (MCs) to train models to obtain probabilities of MCs and skip unlikely ones. Experimental results demonstrate that the proposed algorithm can reduce the encoding time of VVC by 67.53% with 1.89% increase in Bjøntegaard delta bit-rate (BDBR) on average.

Index Terms— VVC, split type, deep learning

1. INTRODUCTION

With the rapid development of information technology, various video applications have been widely used in our daily lives, such as network broadcasting and video conferencing. Meanwhile, users are demanding higher video quality and various ultra-high definition (UHD) video applications are becoming popular, such as 4K/8K video and virtual reality (VR) video, causing the explosive growth of visual data. The previous generation of coding standard, High-Efficiency Video Coding (HEVC) [1], has gradually failed to meet the market demand for more efficient coding efficiency targets for high-resolution video. Therefore, the Joint Video Exploration Team (JVET) has developed the latest generation standard, Versatile Video Coding (VVC) [2]. Thanks to a series of new tools, VVC achieves significantly higher coding efficiency compared with HEVC [3]. However, its computational complexity is also greatly increased, making VVC unsuitable for real-time applications [4]. Therefore, it is necessary to develop fast algorithms to reduce encoding complexity.

To this end, we propose a fast split type (ST) prediction algorithm for VVC in this paper. Since the ST distributions of different-sized coding units (CUs) are significantly different, we design neural networks suitable for all different-sized CUs, and merge ambiguous STs into four merged classes (MCs) to train models for all different-sized CUs to predict candidate MCs. Comprehensive experiments show that the proposed approach can significantly reduce encoding time with a reasonable increase in Bjøntegaard delta bit-rate (BDBR). In particular, it outperforms recent competing methods.

2. RELATED WORK

In order to improve coding speed, numerous deep learning-based methods have been proposed. Liu et al. [5] proposed a convolutional neural network (CNN)-based coding tree unit (CTU) structure decision to skip unlikely CU and prediction unit (PU) modes to speed up the encoding of HEVC. Laude et al. [6] used a CNN classifier to select likely prediction modes in HEVC. Kim et al. [7] selected CU's image values and vector data from the encoding information of CU, and then used CNN to predict likely coding depths. Li et al. [8] modelled CTU partitioning as a three-level classification problem, and then used CNN to predict CTU partitioning. Xu et al. [9] used CNN and long short-term memory (LSTM) networks, respectively, to predict HEVC's intra-encoding CU partitioning and inter-encoding CU partitioning. The above algorithms were

This work was supported in part by the Natural Science Foundation of Chongqing under Grant cstc2020jcyj-msxmX0766; in part by the Science and Technology Research Program of Chongqing Municipal Education Commission under Grant KJZD-K202100604; in part by the Jiangxi Provincial Natural Science Foundation under Grant 20202BABL202006.

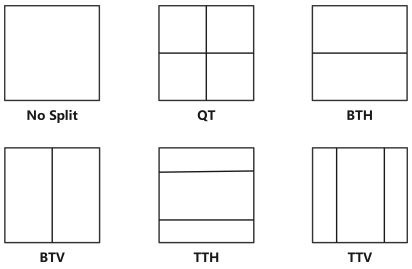


Fig. 1. VVC split types.

developed for HEVC. Since the structure of HEVC is very different from that of VVC, these HEVC-oriented methods cannot be directly applied to VVC.

Given the complex quad-tree plus multi-type tree (QTMT) partition structure of VVC, Li et al. [10] used a multi-stage with an early exit mechanism to predict candidate STs for VVC. However, errors in feature extraction can spread across multiple stages, thus affecting prediction accuracy, especially for small CU sizes. In addition, the convolution operation of residual units is complex, hindering improvements in coding speed. Park et al. [11] selected two useful types of features - explicit VVC features and derived VVC features, and then used a lightweight neural network model to terminate the nested ternary tree (TT) block structure based on these features. As only the TT partitions were terminated early, the coding speed can hardly be improved significantly. Furthermore, this work does not take into account the ST distribution of different CU sizes, which may also degrade the coding performance. Therefore, there is still room for further coding speed improvements.

Based on the above analysis, HEVC-oriented approaches cannot be directly applied in VVC, and VVC-oriented approaches also have some drawbacks. In order to address these issues, we propose a fast ST prediction algorithm to further improve encoding speed.

3. LEARNING-BASED SPLIT TYPE PREDICTION

In VVC, there are at most six partition types: no split, QT, binary-tree-horizontal (BTH), binary-tree-vertical (BTV), ternarytree-horizontal (TTH) and ternary-tree-vertical (TTV), as shown in Fig. 1. In addition, the numbers of STs may vary from 2 to 6 in different CUs and is listed in the second column of Table 1.

CNNs are widely used for image and video processing and can achieve excellent performance. To efficiently predict candidate STs, we use CNNs to extract features. More specifically, we first build a database to train and test the CNN models to predict VVC STs. Then, we design the structure of the CNN models. Finally, we use these models to obtain the probabilities of STs for the current CU and select likely STs based on the probabilities. The proposed approach is described in detail below.

Table 1. STs and their corresponding merged classes of different CUs

CU Size	Split Types	Merged classes
64×64	No split, QT	NSC, QC
32×32	No split, QT, BTH, BTV, TTH, TTV	NSC, QC, HSC, VSC
32×16/16×32	No split, BTH, BTV, TTH, TTV	NSC, HSC, VSC
32×8/8×32	No split, BTH, BTV, TTV/TTH	NSC, HSC, VSC
32×4/4×32	No split, BTV/BTH, TTV/TTH	NSC, VSC/HSC
16×16	No split, QT, BTH, BTV, TTH, TTV	NSC, QC, HSC, VSC
16×8/8×16	No split, BTH, BTV, TTV/TTH	NSC, HSC, VSC
16×4/4×16	No split, BTV/BTH, TTV/TTH	NSC, VSC/HSC
8×8	No split, BTH, BTV	NSC, HSC, VSC
8×4/4×8	No split, BTV/BTH	NSC, VSC/HSC

3.1. Database Construction

To train and test the neural network models, we need to build a large-scale database for the ST prediction of VVC Intra coding (called the SPVIC database). As both image data and quantization parameters (QPs) are closely related to the ST selection, we choose both of them as components of the database. The image data are derived from 204 original video sequences [10], which come with different resolutions and content. These video sequences were divided into three non-overlapping sets, including a training set (160 sequences), a validation set (22 sequences) and a test set (22 sequences).

According to the Common Test Conditions (CTC) [12], the QPs are set to 22, 27, 32 and 37 with All-Intra (AI) configuration for testing. All video sequences and images are encoded by the VVC reference software VTM-10.2. The ground truth STs of the CUs can be obtained accordingly. The SPVIC database is constructed through the process described above. Each sample in the database includes the image data, CU's QP value, as well as its corresponding ground truth ST.

3.2. Design of the Neural Network Architecture

Before designing the Neural Network, we first investigate the ST distribution of CUs. Extensive experimental results show that the ST distributions of all different CUs are also different. Clearly, if all CUs use the same model to predict candidate STs, the corresponding predictions cannot always obtain the best performance. Therefore, we should train separate models for different CUs.

In VVC, there are many CUs have their corresponding transposed CUs, such as 32×16 CU and 16×32 CU. To avoid training redundant models, we only train the CUs whose widths are larger than their heights.

Fig. 2 is the proposed Neural network architecture of ST prediction for training and inferencing. This structure first uses the convolutional layers and pooling layers for the image data, and finally, uses fully connected layers for the vector

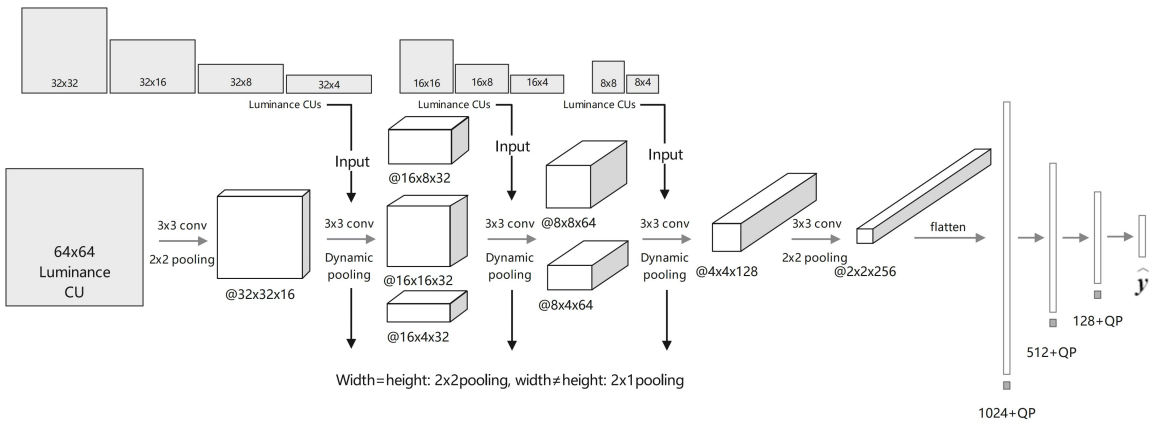


Fig. 2. Proposed neural network architecture for ST prediction(conv and pooling refer to the convolutional layer and pooling layers, respectively).

data. Only the 3×3 convolution kernel with stride 1 and pad 1 is used in convolutional layers. The pooling layer is implemented to reduce the size of their representation. Since there are square CUs and rectangular CUs, a dynamic max-pooling layer is implemented. More specifically, the 2×2 pooling kernel is used for square CUs, and the 2×1 pooling kernel is used for rectangular CUs. Through the 2×1 pooling kernel, the ratio between the width and height of a rectangular CU is reduced by 2. Through a series of 2×1 pooling kernels, a rectangular CU can be reduced to a smaller square. For example, through the 2×1 pooling kernel, a 32×8 CU is reduced into a 16×8 CU. By performing the process once, the 16x8 CU is reduced into an 8x8 CU.

After being processed in a series of convolutional layers and pooling layers, the vector data is concatenated to feed into the fully connected layer. Furthermore, QPs are closely related to the partitioning of CU. For small QPs, CUs are more likely to be partitioned, and vice versa. Therefore, QPs need to be considered a neuron. In addition, all convolutional layers and fully connected layers are activated with rectified linear units (ReLU). Finally, we obtain the predicted value $\widehat{y}$ by the softmax output function. The number of $\widehat{y}$ depends on the size of the CU itself and ranges between 2 and 6.

3.3. Training and Testing the Neural Network Architecture

We construct the database and designed the neural network architecture through the process discussed above. A model can be learned by repeating the training and testing processes. Using the trained model, six STs of 32×32 CUs are predicted, and their corresponding test accuracies are shown in Table 2.

In Table 2, the rows and columns represent predicted STs and actual STs, respectively. From Table 2, we can find that it is difficult to identify BT and TT in the horizontal and vertical direction. For example, 26% of CUs use TTH when the predicted ST is BTH; 24% of CUs use TTV when the predicted ST is BTV. To solve this problem, we merged the ambiguous STs into one class. Specifically, we merged BTH and TTH into the horizontal split class (HSC), and BTV and TTV into the vertical split class (VSC). Accordingly, we grouped the six STs into four classes, namely, no split class (NSC), QT class (QC), HSC and VSC. We merged the ambiguous STs for CUs of sizes 64×64 to 8×4, and their corresponding MCs are listed in the third column of Table 1.

Table 2. Test accuracy (%) of STs of 32×32 CUs

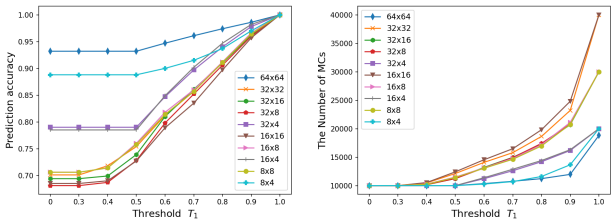
Predicted ST \ Actual ST	No split	QT	BTH	BTV	TTH	TTV
No split	74	1	9	8	6	8
QT	1	75	8	11	14	15
BTH	12	13	52	14	26	12
BTV	6	9	10	47	7	24
TTH	5	1	20	1	46	1
TTV	2	1	1	19	1	40

3.4. Probability-based MC selection

For MCs, we retrain MC-based models. Using these model, we can obtain the probabilities of the MCs. Obviously, MCs with high probabilities are more likely to be selected, and vice versa. Therefore, we sort MCs according to their probabilities from highest to lowest. a_i denotes the probability of the ith MC of the current CU, the sum of the probability of the first n MCs, s_1, is:

$$s_1 = \sum_{i=1}^{n} a_i \tag{1}$$

If s_1 is greater than or equal to a threshold value denoted as T_1, we can early terminate ST seletion. How to choose the best T_1 is the key. On the validation data, the prediction



(a) Prediction accuracy (b) The Number of MCs

Fig. 3. Prediction accuracy and the number of MCs under different T_1 on the validation data.

accuracies and the numbers of MCs under different T_1 are shown in Fig. 3.

From Fig. 3, we can see that both the prediction accuracy and the number of MCs increase as T_1 increases. While high prediction accuracy leads to high coding efficiency, the high number of MCs also results in high coding complexity. Therefore, there is a trade-off between coding efficiency and coding complexity. Considering this trade-off, we set 0.7 as the threshold. The MC selection can be written as follows:

$$\sum_{i=1}^{n} a_i \geq 0.7 \qquad \sum_{i=1}^{n-1} a_i < 0.7 \tag{2}$$

If the above conditions are met, only the first n MCs can be selected and the later ones skipped to increase coding speed and maintain coding efficiency.

4. EXPERIMENTAL RESULTS

To verify the performance of the proposed VVC fast coding algorithm, we performed the evaluations on a server with an Intel(R) 2.0 GHz CPU and 30 GB of RAM using the reference software VTM 10.2. For a fair comparison, we use the GeForce RTX 3090 GPU to speed up the training only but disable it when testing the coding performance. When training the CNN model, we set all weights and bias parameters randomly with Xavier initialization [13]. For each model trained from scratch, we perform 200,000 iterations with a batch size of 128. In every 2000 iterations, we initially set the learning rate to 10^{-4} and then reduced it exponentially by 1%. During the training process, we use the Adam algorithm to optimize the parameters of the trainable components [14], while keeping other parameters unchanged. We use the deep learning framework PyTorch to train the CNN models, and then embed them into the VTM encoder during the test.

According to the CTC [12], 22 video sequences in classes A to E are selected in our experiments. All these sequences are tested with the AI configuration with QP values (22, 27, 32, 37). The performance of our proposed algorithm is evaluated by coding efficiency and computational complexity. Coding efficiency refers to the visual quality and its corresponding bit rate together, which is measured by BDBR [15]. Computational complexity is measured by the percentage of coding time saved, denoted as TS.

Table 3. Overall performance comparisons of three algorithms

Class	Sequence	Proposed		DQTMT[10]:"faster"		LCCPI[16]	
		BDBR(%)	TS(%)	BDBR(%)	TS(%)	BDBR(%)	TS(%)
A1	Tango2	1.66	66.68	3.49	54.35	1.63	45.28
	Foodmarket4	1.49	54.48	2.78	57.91	5.15	60.49
	Campfire	1.65	65	4.17	66.52	2.64	50.12
A2	CatRobot	1.99	64.94	4.88	62.87	1.13	47.17
	DaylightRoad2	1.97	72.63	2.78	65.63	2.18	52.45
	ParkRunning3	1.45	58.05	2.68	63.01	1.25	45.58
B	MarketPlace	1.03	74.66	1.89	65.15	4.2	57.97
	RitualDance	1.63	64.79	2.69	62.98	3.73	61.93
	Cactus	1.71	71.37	2.85	67.7	1.32	60.68
	BasketballDrive	2.2	74.75	3.87	68.5	2.19	55.07
	BQTerrace	2.58	70.45	2.57	64.39	5.25	58.96
C	BasketballDrill	3.76	63.84	4.72	60.98	4.29	60.09
	BQMall	1.88	69.84	3.1	67.45	2.84	55.04
	PartyScene	1.3	65.81	1.86	64.64	2.79	57.2
	RaceHorses	1.41	67.47	2.5	65.68	2.39	54.91
D	BasketballPass	2.13	67.14	3.66	62.62	1.88	54.69
	BQSquare	1.76	66.77	2.04	62.52	1.37	51.36
	BlowingBubbles	1.4	65.02	2.38	61.85	3.17	52.42
	RaceHorses	1.75	65.35	2.92	61.29	1.19	45.47
E	FourPeople	2.18	72.71	3.3	66.91	1.66	51.51
	Johnny	2.58	72.01	5.08	64.35	2.43	57.49
	KristenAndSara	2.14	71.94	3.93	66.11	3.78	58.85
Average		1.89	67.53	3.19	63.79	2.66	54.31

We compare the performance of the proposed approach with the DQTMT algorithm [10] and the LCCPI algorithm [16], the corresponding overall performance comparisons are listed in Table 3. From Table 3, we find that the averages of BDBR and TS of the proposed algorithm are 1.89% and 67.53% respectively, and those of the DQTMT algorithm ("faster" mode) are 3.19% and 63.79%, and those of the LCCPI algorithm are 2.66% and 54.31%. Obviously, compared with these two algorithms, the proposed algorithm has a smaller average BDBR and greater average TS. Therefore, we can conclude that the proposed algorithm performs much better in both coding speed and coding efficiency compared with the DQTMT and LCCPI algorithms.

5. CONCLUSION

In this paper, we proposed a fast learning-based ST prediction algorithm for VVC. Since split type (ST) distributions of different-sized CUs are significantly different, we design Neural Networks and train models for all different-sized CUs to predict candidate MCs. Comprehensive experimental results show that our approach can outperform other state-of-the-art approaches.

6. REFERENCES

[1] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1648–1667, Dec. 2012.

[2] B. Bross, Y. K. Wang, Y. Ye, S. Liu, J. L. Chen, G. J. Sullivan, and J. R. Ohm, "Overview of the Versatile Video Coding (VVC) Standard and its Applications," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3736-3764, Oct. 2021.

[3] F. Bossen, X. Li, A. Norkin, K. Sühring, "JVET AHG report: Test model software development (AHG3)," *Joint Video Experts Team (JVET) of ITU-T and ISO/IEC*, Document JVET-O0003, Jul. 2019.

[4] A. Tissier, A. Mercat, T. Amestoy, W. Hamidouche, J. Vanne, and D. Menard, "Complexity reduction opportunities in the future VVC intra encoder," in *Proc. IEEE 21st Int. Workshop Multimedia Signal Process. (MMSP)*, Sep. 2019, pp. 1–6.

[5] Z. Liu, X. Yu, Y. Gao, S. Chen, X. Ji, and D. Wang, "CU partition mode decision for HEVC hardwired intra encoder using convolution neural network," *IEEE Trans. Image Process.*, vol. 25, no. 11, pp. 5088–5103, Nov. 2016.

[6] T. Laude and J. Ostermann, "Deep learning-based intra prediction mode decision for HEVC," in *Proc. Picture Coding Symp. (PCS)*, 2016, pp. 1–5.

[7] K. Kim and W. W. Ro, "Fast CU Depth Decision for HEVC Using Neural Networks," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 5, pp. 1462-1473, May 2019.

[8] T. Li, M. Xu and X. Deng, "A deep convolutional neural network approach for complexity reduction on intra-mode HEVC," *2017 IEEE International Conference on Multimedia and Expo (ICME)*, 2017, pp. 1255-1260.

[9] M. Xu, T. Li, Z. Wang, X. Deng, R. Yang and Z. Guan, "Reducing Complexity of HEVC: A Deep Learning Approach," in *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 5044-5059, Oct. 2018.

[10] T. Li, M. Xu, R. Tang, Y. Chen and Q. Xing, "DeepQTMT: A Deep Learning Approach for Fast QTMT-Based CU Partition of Intra-Mode VVC," in *IEEE Transactions on Image Processing*, vol. 30, pp. 5377-5390, 2021.

[11] S. -h. Park and J. -W. Kang, "Fast Multi-Type Tree Partitioning for Versatile Video Coding Using a Lightweight Neural Network," in *IEEE Transactions on Multimedia*, vol. 23, pp. 4388-4399, 2021.

[12] J. Boyce, K. Suehring, X. Li, and V. Seregin, *JVET Common Test Conditions and Software Reference Configurations*, document JVETJ1010, San Diego, CA, USA, Apr. 2018.

[13] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. AISTATS*, vol. 9, 2010, pp. 249–256.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent*. (ICLR), May 2015, pp. 1–15.

[15] G. Bjontegaard, "Calculation of average psnr difference between rd-curves," in*Proc. 13th VCEG-M33 Meet*. IEEE, 2001, pp. 1–4.

[16] H. Yang, L. Shen, X. Dong, Q. Ding, P. An, and G. Jiang, "Low complexity CTU partition structure decision and fast intra mode decision for versatile video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 6, pp. 1668–1682, Jun. 2020.