Fast QTMT for H.266/VVC Intra Prediction using Early-Terminated Hierarchical CNN model

Xiem HoangVan, Sang NguyenQuang, Minh DinhBao, Minh DoNgoc, Dinh Trieu Duong
Faculty of Electronics and Telecommunications, University of Engineering and Technology, Vietnam National University, Hanoi
xiemhoang@vnu.edu.vn, ngsang998@gmail.com, minhdinh@vnu.edu.vn, ngocminhc2nc1@gmail.com, duongdt@vnu.edu.vn

Abstract— **Versatile Video Coding (VVC) has been standardization in July 2020. Compared to previous High Efficiency Video Coding (HEVC) standard, VVC saves up to 50% bitrate for equal perceptual video quality. To reach this efficiency, Joint Video Experts Team (JVET) has introduced a number of improvement techniques to VVC model. As a result, the complexity of VVC encoding also greatly increases. One of the new techniques affects to the growing of complexity is the quad-tree nested multi-type tree (QTMT) including binary split and ternary splits, which lead to a block in VVC with various shapes in both square and rectangle. Based on the aforementioned information we propose in this paper a new deep learning based fast QTMT method. We use a learned convolutional neural network (CNN) model namely Early-Terminated Hierarchical CNN to predict the coding unit map and then fed into the VVC encoder to early terminate the block partitioning process. Experimental results show that the proposed method can save 30.29% encoding time with a negligible BD-Rate increase.**

Keywords—VVC, Early-Terminate Hierarchical, CNN.

I. Introduction

Nowadays, digital videos have been covering a very wide range of applications from multimedia messaging, video telephony, video conferencing and display in high resolution formats like high definition (HD), Full HD, 2K, 4K, and 360-degree. The progression of video formats and resolutions requires a new video coding standard with better performance compared with previous standards. However, to achieve the high performance, the state-of-the-art video codec makes computational complexity greatly increase.

Block partitioning is one of the most important techniques in video coding in which the video picture is adaptively divided into smaller blocks for encoding. In H.264/AVC [1], each of frames is partitioned into macro-blocks (MB) with a size of 16×16 Luma samples while HEVC [2] introduces a new technique called quad-tree partitioning (QT). In QT, each frame is split into coding tree units (CTUs) with a size of 64×64. After that, CTUs are divided into smaller coding units (CUs) of different sizes. CUs in HEVC are always square and their size can be from 8x8 Luma samples up to the largest coding units (LCUs).

In the newest video coding standard, namely Versatile Video Coding [3], Joint Video Exploration Team (JVET) also applies block-based hybrid mechanism to get a flexible hierarchy unit representation. To get better coding efficiency, VVC has adopted many new coding tools, such as quad-tree nested multi-type tree (QTMT) in the block partitioning process and the largest supported size is up to 128x128. CUs in VVC can be partitioned into smaller square or rectangular block using quad-tree, binary split or ternary split as shown in Fig. 1.

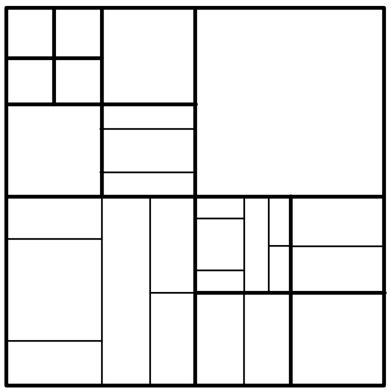
Fig.1. An example of QTMT in VVC

Because of using QTMT, the complexity of VVC has also increased greatly. In the newest document, JVET reports that computational complexity of VVC increase more than 30 times compared to HEVC [13]. Therefore, it is necessary to reduce the processing time of the codec, especially at the encoder side.

Artificial intelligence, notably the deep learning technique has recently been used in a wide range of vision and image processing applications [10]. Following this direction, Mai Xu et al. [6] introduced a deep learning based method to reduce complexity of the HEVC encoder. Inspired from this work, we propose a combined network to predict the CU partition for the inter prediction mode in VVC encoder. In this paper, deep early-terminated hierarchical convolutional neural network (ETH-CNN) structure is developed to learn the hierarchical CU partition map for predicting the CU partition of intra-mode VVC, and early-terminated hierarchical long and short-term memory network (ETH-LSTM) is proposed to learn the temporal correlation of the CU partition. In this paper, we introduced a deep learning approach to reduce the complexity of the CU partition in VVC standard by applying ETH-CNN.

To demonstrate the efficiency of proposed method, the rest of this paper is organized as follows. Section II briefly introduces the VVC block partitioning and related fast block partitioning algorithms. Section III presents the proposed ETH-

CNN model while Section IV discusses the experimental results. Finally, Section V gives the conclusion of this paper.

II. Background and Related Works

A. Block partitioning in VVC

In this section, we review the CU partition in VVC standard, which have many different from that in previous video coding standard. Fig.2 shows an image region which is partitioned and encoded by VVC (left) and HEVC (right). We can observe that VVC supports both square and rectangle partition shapes while blocks in HEVC are only square shape.

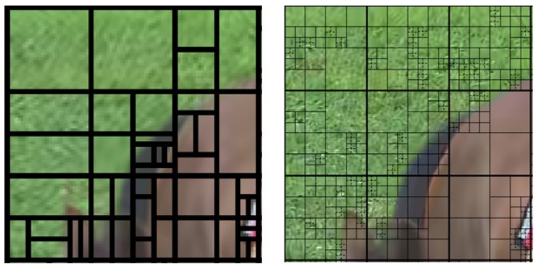
Fig.2. Block partitioning in H.266/VVC (left) and H.265/HEVC (right)

In HEVC [2], input frames are divided into the basic coding structure called coding tree unit – CTU, with the maximum allowed luma block size is 64×64. After that, CTU with size of 2N×2N is recursively partitioned into smaller coding unit – CU via the quad-tree. By using quad-tree structure, CTU and CU are split into four sub-CU with size of N×N [9].

Inspired from the HEVC structure, the raw image in VVC is divided into a sequence of CTUs with the same concept in HEVC [7, 8]. However, VVC allows the maximum size of the Luma block in a CTU is 128×128.

Firstly, in depth = 0, if CTU size is larger than maximum allowed binary and ternary tree size, only quad-tree structure is applied to partition CTU into four smaller CU with size of 64×64. After that, while depth >= 1, Quad-tree with nested multi-type tree (QTMT) containing binary and ternary splits is used to partition current block into sub blocks. Fig. 3 shows four splitting types of multi-type tree structure. A CU can be split into sub CUs by vertical binary splitting (SPLIT_BT_VER), horizontal binary splitting (SPLIT_BT_HOR), vertical ternary splitting (SPLIT_TT_VER), and horizontal ternary splitting (SPLIT_TT_HOR) mode. In most case, if the width or height of the color component of the CU is smaller than maximum supported transform length, CU, PU and TU have the same block shape and size in the QTMT coding block structure. In case that the width or height of the coding block is larger than the maximum transform width or height, the coding block is automatically split in vertical and/or horizontal direction to meet the supported transform size.

If a part of the current CU exceeds the bottom or right picture boundary, it is forced to be split until all samples of every coded CU are not located outside the picture boundaries. For example, when a portion of the current CU exceeds both the bottom and the right picture boundaries, it is forced to be split with QT split mode if the sub-CU size is larger than the minimum QT size, otherwise, the block is forced to be split with SPLIT_BT_HOR mode.

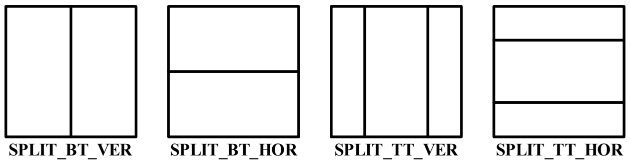

Fig. 3. Binary and Ternary Split in H.266/VVC

Due to the use of multi-type tree, there are some different splitting patterns give the same results of coding block structure. Therefore, in VVC, some of special splitting patterns are not allowed. Fig. 4 describes one of the special cases. Assume that current CU is a square shape, the encoder will use SPLIT_BT_VER mode in both current CU and sub-CUs, instead of use SPLIT_TT_VER and use SPLIT_BT_VER for the central sub-CU in next step.

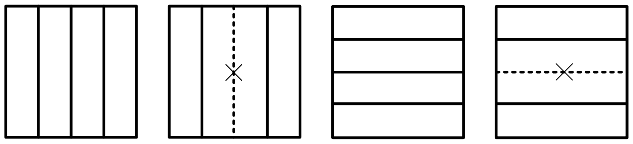
Fig. 4. A special case of multi-type tree

B. Fast block partitioning algorithm in VVC

In general video codec, the encoder will evaluate every possible partition structure, and choose the best partition structure with the minimum *RDCost*, computed in equation (1).

$$RDCost = D + \lambda \times R \tag{1}$$

Where, *D* is the difference between original and the reconstructed information. *R* represents the bitrate which is needed to encode the CU and λ is a Lagrange multiplier.

Targeting the VVC complexity reduction, Zhang *et al.* proposed in [4] a texture based fast CU partitioning method. This research includes two algorithms: i) early terminate the splitting process based on texture energy and ii) using texture direction to decide the splitting mode of CUs. In [5], Jin *et al.* introduced a novel fast QTBT partition decision method based on Convolutional Neural Network (CNN). First, the current frame is divided into a set of 32×32 blocks and CNN model will predict the minimum and maximum partition depth for QTBT partition. In the depth 0, if the classification result is ‘‘0’’ for anyone patches within CTU, it means that the CTU is smooth and no longer split. Otherwise, divid it into smaller blocks. In step 2, the current depth is 1, if all patches of the current CU are smooth and predicted with label ‘‘0’’ or ‘‘1’’, processor calculates RD cost of the 64×64, and then calculates *RDCost* of each sub-CUs in next step. If the classification results of CNN are bigger than “1”, it means that current 64x64 CU has some detail texture, so that processor directly divides it into four sub-CUs using quad-tree partition without calculating RD cost. In step 3, the current size of block is 32×32, the encoder will calculate RD cost for each partition depth within the candidate depth range, and finally determine the optimal QTBT partition structure through RDO process.

III. Proposed Method

A. Fast QTMT structure

Because of using quad-tree nested multi type tree, the existing fast algorithms for block partitioning proposed for HEVC cannot be directly and fully implemented in VVC. So that, we only applied ETH-CNN to square block and quad-tree structure to reduce complexity of the intra mode in VVC.

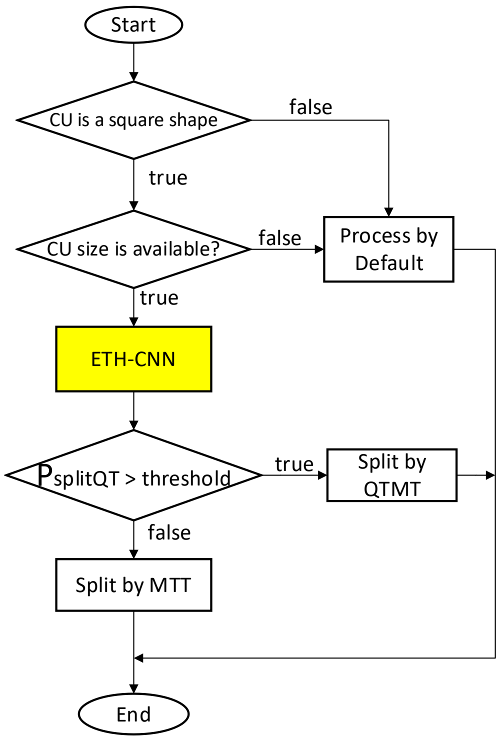

Fig. 5. Proposed Method

Fig. 5 specifically described our proposed method. The input of the process is a CU. The algorithm flow is listed as follows:

1) If the size of CU is 128×128, it will be applied quad-tree to split into 4 square sub-CUs.

2) If the current CU is a square block and the size of width and height are available, the ETH-CNN model predicts the probability ($P_{splitQT}$) of splitting current CU by quad-tree.

3) The encoder compares this probability with a threshold (α_l= 0.5) and decides whether or not the current CU use quad-tree structure to split into sub-CUs. If this probability larger than threshold, the current CU will be split by quad-tree nested multi type tree and if this probability smaller than threshold, current CU is split into 2 or 3 sub-CUs by using multi type tree.

4) Return to step 1 to process the next CU or sub-CU.

B. ETH - CNN model

ETH-CNN was originally introduced by Mai Xu *et al.* [6], to reduce the computational complexity of the prior HEVC standard. In this method, ETH-CNN is used for learning to predicting the CU partition of intra-mode instead of using conventional brute-force RDO search. According to the mechanism of CU partitioning, the ETH-CNN structure is divided into three branches. ETH-CNN has an input as CTU which is divided into sub-CU with size of 64×64, 32×32,16×16, corresponding to all predictions of HCPM at three levels (for more information about HCPM, we suggest reading [6] carefully). Then, the data is passed through the layers to be processed and the result is the probability of the split of that CU.

Inspired from this work, we propose to early select the CU size and partition following a trained CNN model as shown in Fig. 5. In this structure, three parallel branches of ETH-CNN structure will represent the three levels of CU depth, respectively. The level of CU depth in each branch will be changed from [6], specifically B1 corresponding to depth 1, B2 corresponding to depth 2 and B3 corresponding to depth 3. The input of these branches is the Y channel of CU size 64×64 (denoted by U) taken from the original CU size 64×64 (depth level is 1). The output of ETH-CNN is saved as a hierarchical

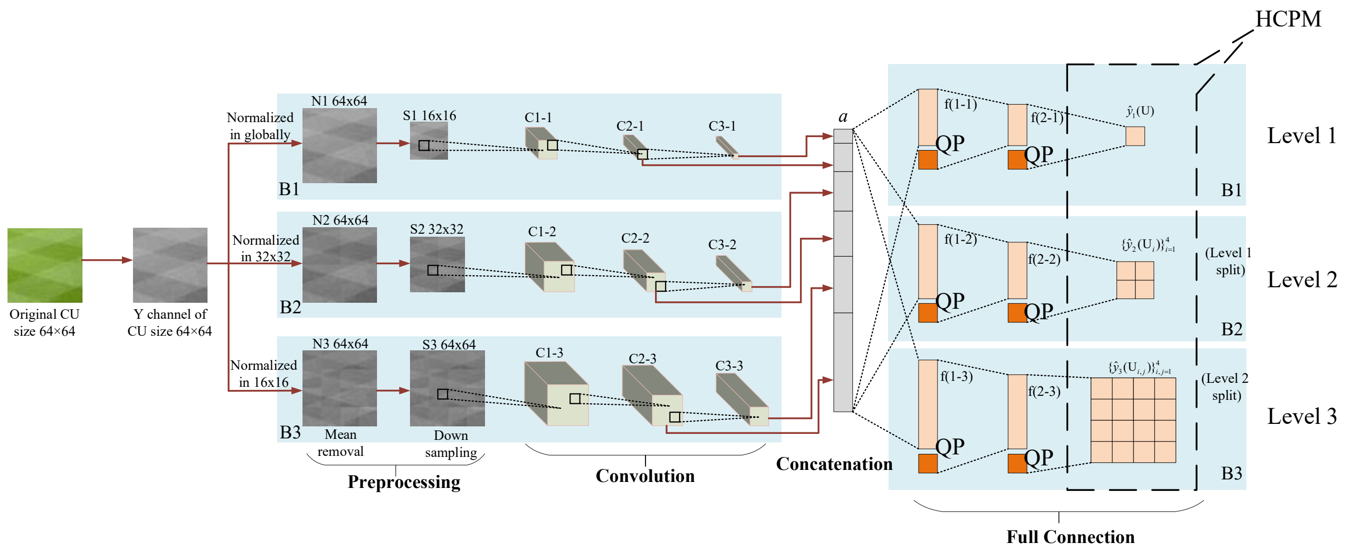

Fig. 6. ETH-CNN structure

CU partition map (HCPM) which as same as the quad-tree structure in HEVC and has three levels from 1 to 3 corresponding to CU size 64×64, 32×32 and 16×16.

Each branch contain two Preprocessing layers, three Convolution layers, one Concatenating layers and three Fully Connected layers. In which, the 3 Fully Connected layers on branch 2 and branch 3 of ETH-CNN can be skipped if the one in the upper branch makes a prediction that the CU will not be split.

Since the input of three branches is Y channel of CU at depth 1, this data is firstly normalized its size to fit with the CU depth of each branch: 64×64 for branch B1, 32×32 for branch B2 and 16×16 for branch B3. The normalized data is the input of two Preprocessing layers and in which, the raw data is processed by mean removal and down-sampling to make input data on each branch become more relevant to that branch's output requirements.

After that, in each branch, the corresponding preprocessed data flow through three Convolutional layers to extract feature. In this paper, we used the same parameter as [6], that is kernel sizes of convolution layer on 3 branches are same with 4×4 kernels (16 filters) for first layer, 2×2 kernels (24 filters) for second layer and 2×2 (32 filters) for final layer.

Then, the features extracted from Convolutional layers are concatenated together and then flattened into a vector *a* in concatenating layer.

Finally, all features contained in vector *a* will be processed on three branches corresponding to 3 levels of HCPM. In each branch, the feature extracted flow through 3 fully connected layers, containing 2 hidden layers and 1 output layer to predict the level of quad-tree structure. The feature vectors $\{\mathbf{f}_{1-l}\}_{l=1}^{3}$ are generated by 2 hidden layers, so that the outputs in B1, B2 and B3 have 1, 4 and 16 elements, serving as the predicted binary labels of quad-tree structure at the three levels ($\hat{y}_1(\mathbf{U})$ in 1x1, $\{\hat{y}_2(\mathbf{U}_i)\}_{i=1}^{4}$ in 2x2 and $\{\hat{y}_3(\mathbf{U}_{i,j})\}_{i,j=1}^{4}$ in 4x4). Besides, the feature vectors $\{\mathbf{f}_{1-l}\}_{l=1}^{3}$ are combined with external features to exploit the QP information, so that ETH-CNN can adaptive to various QPs in predicting quad-tree structure. Especially, if the branch B1 predicts that current CU is not split, the B2 and B3 will be skipped, or if the branch B2 predicts that current CU is not split, B3 will be skipped. Therefore, the complexity of ETH-CNN is reduced.

After collecting the predicted information for quad-tree structure, (the binary labels $\hat{y}_1(\mathbf{U})$, $\hat{y}_2(\mathbf{U}_i)$ và $\hat{y}_3(\mathbf{U}_{i,j})$), probability that the label is 1 is calculated, and then compared with a threshold (α_l) to decide that current CU is split or not. Since the bi-threshold used in [6] α_l and $\bar{\alpha}_l$ all set equal 0.5, we will use only threshold α_l= 0.5 in this paper.

Here, we use cross-entropy loss function for training model. Loss function L_r of each sample in a dataset contains R training samples can be calculated by (2):

$$L_r = H\left(y_1^r(U_i), \hat{y}_1^r(U_i)\right) + \sum_{\substack{i\in\{1,2,3,4\}\\ y_2^r(U_i)\neq null}} H\left(y_2^r(U_i), \hat{y}_2^r(U_i)\right) + \sum_{\substack{i,j\in\{1,2,3,4\}\\ y_3^r(U_{i,j})\neq null}} H\left(y_3^r(U_{i,j}), \hat{y}_3^r(U_{i,j})\right) \quad (2)$$

where the groundtruth dataset are $\{y_1^r(U), \{y_2^r(U_i)\}_{i=1}^4$ and $\{y_3^r(U_{i,j})\}_{i,j=1}^4\}_{r=1}^R$ and predict dataset are $\{\hat{y}_1^r(U), \{\hat{y}_2^r(U_i)\}_{i=1}^4$ and $\{\hat{y}_3^r(U_{i,j})\}_{i,j=1}^4\}_{r=1}^R$.

Then, the ETH-CNN model computes the loss function overall training samples by (3):

$$L = \frac{1}{R}\sum_{r=1}^{R} L_r \quad (3)$$

Because the loss function is calculated on each sample, to optimize the ETH-CNN model, the stochastic gradient descent algorithm with momentum is applied.

IV. EXPRERIMENT AND PERFORMANCE EVALUATION

A. Dataset and test condition

The proposed method is evaluated on 9 standard test sequences in Table 1 with different setting the Quantization Parameters (QPs) to 22, 27, 32 and 37 under All-Intra main configuration, respectively.

To obtain the ETH-CNN model, we adopted a similar training set from [14] where 2000 image were compressed at four QPs.

To show the complexity reduction performance, the ΔT is defined by the following equation (4):

$$\Delta T = \frac{T_{Proposed} - T_{VVC}}{T_{VVC}} \times 100\% \quad (4)$$

where ΔT represents the run time saving by using proposed method, T_{VVC} represents the total run time of the VTM-12.1 [11], $T_{Proposed}$ represents the total run time of the proposed method.

BDBR parameter is used to represent the average bit rate differences [12].

TABLE I. THE DETAILS OF TEST SEQUENCES

Sequence	Spatial Resolution	Number of frames	Frame Rate	Content type
PeopleOnStreet	2560×1600	150	30 Hz	Surveillance
Traffic	2560×1600	150	30 Hz	Surveillance
Kimono	1920×1080	240	24 Hz	Natural
ParkScene	1920×1080	240	24 Hz	Natural
FourPeople	1280×720	600	60 Hz	Conference
KristenAndSara	1280×720	600	60 Hz	Conference
Johnny	1280×720	600	60 Hz	Conference
SlideShow	1280×720	500	20 Hz	Screen content
SlideEditting	1280×720	300	30 Hz	Screen content

B. Results and discussion

The experiment results of time saving, BDBR of all test sequences are shown in Table II while Fig.7 illustrates RD performance of the proposed method.

TABLE II. ENCODING TIME SAVING

Sequence	Time Saving (ΔT %)				BDBR (%)
	QP 22	QP 27	QP 32	QP 37	
PeopleOnStreet	-33.05	-32.21	-30.34	-24.55	1.20
Traffic	-30.81	-29.91	-29.49	-23.33	0.88
Kimono	-34.17	-28.15	-18.74	-13.05	0.21
ParkScene	-35.51	-43.80	-39.83	-31.54	0.71
FourPeople	-32.47	-34.48	-35.64	-33.01	1.17
KristenAndSara	-28.69	-22.00	-19.78	-12.65	1.92
Johnny	-36.14	-34.59	-33.37	-30.52	1.15
SlideShow	-32.28	-33.48	-32.57	-37.01	3.0
SlideEditting	-19.79	-24.08	-40.24	-39.01	2.28
Average	***-31.43***	***-31.41***	***-31.11***	***-27.19***	***1.39***

From the obtained results in Table II and Fig.7, some conclusions can be given as:

- The proposed method provides a low complexity solution for the state-of-the-art video coding standard.
- The proposed method can reduce encoding complexity for high resolution video using H.266/VVC encoder.
- The total encoding time can be cut by proposed method for All-Intra configuration ranges from -12.65% to -40.24%, with an average of -30.29%. Meanwhile, the BDBR is from 0.21% to 3.0%, with 1.39% on average.
- The proposed method can be applied to a variety of video content (i.e. natural, surveillance, conference, screen content).
- Quantization parameter does not affect the encoding time saving.

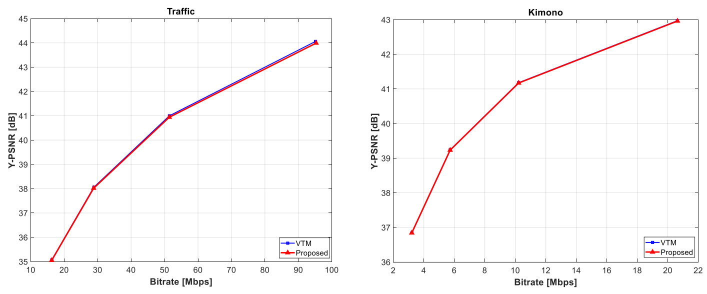

Fig. 7. RD performance of sequence Traffic and Kimono

The CU structure partition of the proposed algorithm is visualized in Fig. 8. The figures are cropped from the first frame of sequence *KristenAndSara* under QP 37. We can observe that the CU structure and video quality of the proposed algorithm is very similar to the default VTM encoder.

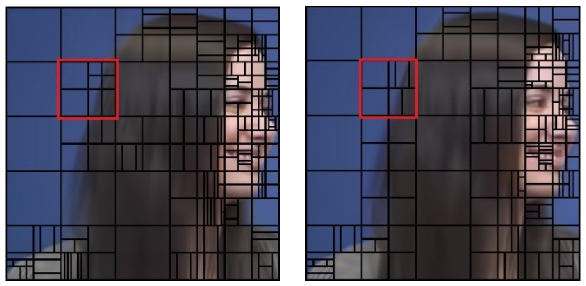

Fig. 8. The CU structure partition result of the default VTM (left) and proposed method (right)

V. CONCLUSIONS

In this paper, a fast QTMT method is proposed for VVC intra coding using Early-Terminated Hierarchical CNN model. The experiment results show that the proposed algorithm can achieve 30.29% time saving on average while paying for only 1.39% BD-rate increase over the VVC standard test sequences. In the future, we plan to adapt this model for binary and ternary split process.

ACKNOWLEDGMENT

Minh DinhBao was funded by Vingroup Joint Stock Company and supported by the Domestic Master/ PhD Scholarship Programme of Vingroup Innovation Foundation (VINIF), Vingroup Big Data Institute (VINBIGDATA), code VINIF.2020.ThS.43

REFERENCES

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra, "Overview of the H.264/AVC video coding standard," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560-576, July 2003.

[2] G. J. Sullivan, J. Ohm, W. Han and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1649-1668, Dec. 2012.

[3] B. Bross, J. Chen, J. -R. Ohm, G. J. Sullivan and Y. -K. Wang, "Developments in International Video Coding Standardization After AVC, With an Overview of Versatile Video Coding (VVC)," in Proceedings of the IEEE, 2020.

[4] Q. Zhang, Y. Zhao, B. Jiang, L. Huang and T. Wei, "Fast CU Partition Decision Method Based on Texture Characteristics for H.266/VVC," in IEEE Access, vol. 8, pp. 203516-203524, 2020.

[5] Z. Jin, P. An, C. Yang and L. Shen, "Fast QTBT Partition Algorithm for Intra Frame Coding through Convolutional Neural Network," in IEEE Access, vol. 6, pp. 54660-54673, 2018.

[6] 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.

[7] Jianle Chen, Yan Ye, Seung Hwan Kim, "Algorithm description for Versatile Video Coding and Test Model 12 (VTM 12)," document JVET-U2002, 21st JVET Meeting, by teleconference, 6–15 Jan. 2021.

[8] High Efficiency Video Coding (HEVC), Rec. ITU-T H.265 and ISO/IEC 23008-2, Jan. 2013 (and later editions).

[9] I. Kim, J. Min, T. Lee, W. Han and J. Park, "Block Partitioning Structure in the HEVC Standard," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1697-1706, Dec. 2012

[10] S. Dargan, et al., "A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning," in Arch Computat Methods Eng vol. 27, pp. 1071–1092, 2020.

[11] VVCSoftware_VTM-12.1. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/tree/VTM-12.1.

[12] G. Bjontegaard, “Calculation of average PSNR differences between RD curves,” document VCEG-M33, 13th ITU-T VCEG Meeting, VCEG, Austin, TX, USA, Apr. 2001.

[13] Frank Bossen, et al., “AHG report: Test model software development (AHG3)”, document JVET-V0003-v1, 22nd JVET Meeting, by teleconference, 20–28 Apr. 2021.

[14] [Online]. Available: https://github.com/HEVC-Projects/CPH