LIGHT-WEIGHT CNN-BASED VVC INTER PARTITIONING ACCELERATION

Yiqun Liu[⋆†] *Mohsen Abdoli*[⋆] *Thomas Guionnet*[⋆] *Christine Guillemot*[†] *Aline Roumy*[†]
[⋆] ATEME, Rennes, France
[†] INRIA, Rennes, France

ABSTRACT

The Versatile Video Coding (VVC) standard has been finalized by Joint Video Exploration Team (JVET) in 2020. Compared to the High Efficiency Video Coding (HEVC) standard, VVC offers about 50% compression efficiency gain, in terms of Bjontegaard Delta-Rate (BD-rate), at the cost of about 10x more encoder complexity [1]. In this paper, we propose a Convolutional Neural Network (CNN)-based method to speed up inter partitioning in VVC. Our method operates at the Coding Tree Unit (CTU) level, by splitting each CTU into a fixed grid of 8×8 blocks. Then each cell in this grid is associated with information about the partitioning depth within that area. A lightweight network for predicting this grid is employed during the rate-distortion optimization to limit the Quaternary Tree (QT)-split search and avoid partitions that are unlikely to be selected. Experiments show that the proposed method can achieve acceleration ranging from 17% to 30% in the RandomAccess Group Of Picture 32 (RAGOP32) mode of VVC Test Model (VTM)10 with a reasonable efficiency drop ranging from 0.37% to 1.18% in terms of BD-rate increase.

Index Terms— Inter block partitioning acceleration, VVC, lightweight CNN, depth map

1. INTRODUCTION

The design of efficient compression methods remains critical due to the continuously increasing demand for video transmission. In order to improve coding efficiency, a novel partition structure, called quadtree with nested multi-type tree (QTMT) [2], has been integrated into VVC. In addition, multiple new inter coding modes [3] [4] [5] have been adopted in VVC. Thus, the inter prediction has been largely improved at the cost of increased complexity of rate-distortion optimization (RDO) of split types in inter coding. As a result, the newly adopted QTMT and inter coding modes imply significant complexity for the partition search algorithm.

Various machine learning based methods [6] [7] [8] [9] have been proposed to speed up the RDO search of inter partitioning in VVC. They can be categorized into Random Forest (RF)-based and CNN-based methods. In the first category, the work by Amestoy *et al.* [6] is a cascade of binary skip decisions predicted by trained RFs. This method has been later improved by Kulupana *et al.* [7], by combining it with a handcrafted early termination rule for the Ternary-type Tree (TT) splits. For CNN-based methods, Pan *et al.* propose in [8] a multi-branch CNN, namely MF-CNN, to perform a binary classification of the "Partition" or "Non-partition" at the Coding Unit (CU) level, in order to early terminate the partitioning. In this paper, different models are trained for different CU sizes. Moreover, Yeo *et al.* propose a method in [9], where a variant of the Branch Convolutional Neural Network (B-CNN) [10], called multi-level tree CNN (MLT-CNN), is used to predict the partitioning mode of the CTU. The key feature of this CNN is that its outputs correspond to different decisions at different levels of a split mode tree.

The aforementioned methods suffer from some limitations. RF-based methods are complex in terms of number of trained models involved. For example, 18 models are needed to be trained separately for the different possible CU sizes with obvious implication in terms of training and implementation complexity. The main disadvantage of CNN methods is that their trade-offs between encoding complexity reduction and coding loss are much worse than that of RF methods.

To tackle these issues, we develop a lightweight CNN-based inter partitioning acceleration method for VVC in this paper. Given that the partitioning syntax of VVC can be specified as a serie of QT-then-MT splits, the proposed method predicts the QT-depth of each 8×8 block. The prediction is performed by a CNN learned from a large dataset of encoded videos, and is integrated into the VTM with the help of a threshold to control the trade-off between complexity reduction and coding performance. To achieve accurate QT-depth prediction, we propose to base the decision on not only luminance values but also motion fields, and motion compensated residues.

Compared to RF methods, our proposed method achieves similar performance. The main advantage is that our method is based on a unique model trained on CTU level, which significantly eases the method implementation. Moreover, our lightweight method is scalable since we can adjust a threshold to obtain different levels of acceleration. Finally, the performance of the proposed method is significantly better than the one of related CNN-based approaches. In addition, experiments show that our solution can be applied to different versions of the VTM with steady performance.

The rest of this paper is organized as follows. Section II describes the proposed method in detail, while Section III presents results and finally Section IV concludes the paper.

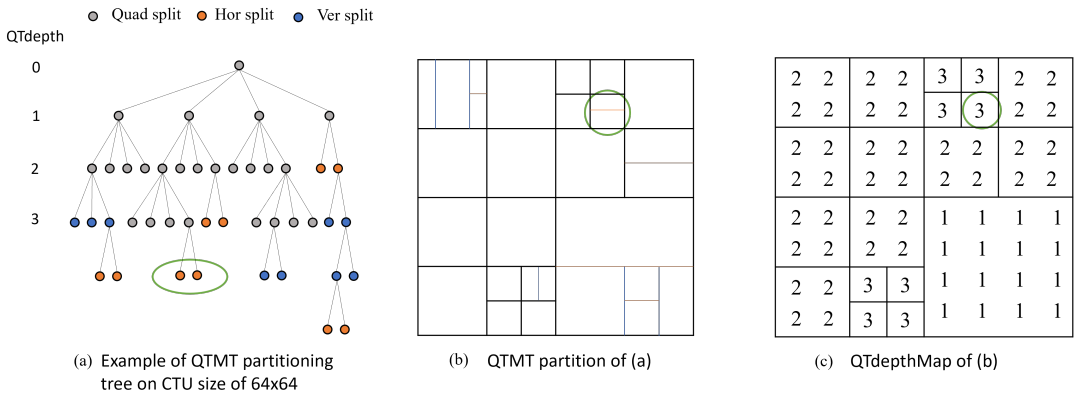

Fig. 1: Example of partitioned 64x64 CTU with its Quad Tree depth map

2. PROPOSED METHOD

2.1. Quad-tree depth map Representation

Starting from the CTU structure, VVC performs RDO for all possible split options as well as the No Split (NS) option, at each level of the partitioning tree, in order to find the partition which best exploits spatial and temporal redundancy. The QTMT partitioning scheme in VVC consists in splitting the CU using either a QT split or a Multi-type Tree (MT) split. In MT mode, VVC uses four split types: Horizontal Binary Tree (HBT) split, Vertical Binary Tree (VBT) split, Horizontal Ternary Tree (HTT) split, Vertical Ternary Tree (VTT) split.

Fig. 1 (a) shows an example of a QTMT partition tree, with its corresponding partition shown in Fig. 1 (b). Without loss of generality and for the sake of simplicity, the example in this figure is shown for a CTU of size 64×64 and QTdepth values in $\{0, 1, 2, 3\}$, while in Common Test Condition (CTC) of VVC, the CTU of size 128×128 results in QTdepth values in $\{0, 1, 2, 3, 4\}$. In Fig. 1 (a), the red horizontal arc-shaped lines indicate the border between the last QT split and the first MT split (if any). According to the QT-then-MT scheme adopted in VVC, once the first MT split is chosen, the QT split becomes forbidden for the following nodes. Therefore, one can represent partitioning series for reaching any arbitrary CU, as N consecutive QT splits, followed by M consecutive MT splits. In the proposed method, we use the values of N that are associated to the leaf nodes (*i.e.* final partitions) and form a grid called Quad Tree depth map (QTdepthMap). Fig. 1 (c) shows this map computed for the above partitioning example. The goal of the paper is to propose a CNN architecture, that predicts this QTdepthMap.

2.2. Network architecture for QT depth map prediction

The CNN proposed in this paper, called Multi-Branch Multi-Pooling CNN (MBMP-CNN), is depicted in Fig. 2. This network structure merges inputs of different types.

For the first input, luma values of CTU are concatenated with residual blocks obtained by motion-compensated prediction of current CTU from the nearest frame. The second input is formed by the motion vectors calculated on each 4x4 sub-block in reference to the closest frame. The reason for choosing the residual block and the motion field as input is that these features reflect temporal correlation between frames as depicted in multiple papers [6] [8] [9] [11]. Finally, the Quantization Parameter (QP) value and temporal ID form the third input. With this third input, the proposed network achieves a good tradeoff between prediction accuracy and memory consumption of the network. Indeed, a single learned network provides one prediction per QP value and per position of the frame in the GOP.

Tensors of same width and height corresponding to multiple input branches are concatenated before being fed into the main branch of the network architecture.The main branch begins with three residual blocks [12] followed by the multi-pooling layer introduced in [13]. The different kernel sizes are consistent with the CU sizes at different QTdepths. Finally, three convolutional layers shrink the tensor dimension, which leads to the predicted 16×16 QTdepthMap constituted by QTdepths on different 8×8 sub-blocks. Each value equals to the QTdepth of the CU containing the corresponding sub-block, as described earlier in Fig. 1 (c).

The inference within the VTM is performed by the CNN on each CTU before entering the RDO decision loop. By doing so, the encoder can use the acceleration algorithm presented in the next section to speed up the partition search. The CPU overhead of inference of this lightweight CNN increases the encoding time by only 0.21% when integrated in VTM10.

2.3. Partitioning acceleration algorithm

The main idea of the proposed acceleration algorithm is to avoid unnecessary MT and NS split checks in the QT-phase of the partitioning. Our algorithm navigates through the tree depicted in Fig. 1 (a) from top to bottom. Then, for each

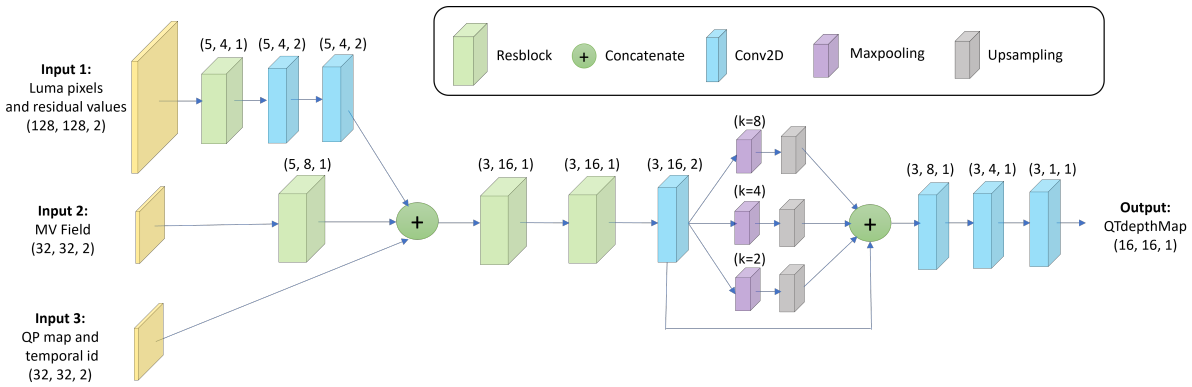

Fig. 2: Structure of the proposed MBMP-CNN. The vector of three elements on top of Resblocks and Conv2D layers represents kernel size, number of filters and stride respectively. Value "k" denotes pooling size for Maxpooling layers.

Algorithm 1 MT and NS early skipping

Input: QTdepthMap; $\text{QTdepth}_{\text{cur}}$; Th; Size_{CU}; Pos_{CU}
Output: SkipMT_NS: Boolean to decide whether to skip MT and NS split types or not

1: Compute the $\text{QTdepth}_{\text{avg}}$ based on Size_{CU}, Pos_{CU} and QTdepthMap
2: **if** $\text{QTdepth}_{\text{avg}} > (\text{QTdepth}_{\text{cur}} + \text{Th})$ **then**
3: SkipMT_NS = True
4: **else**
5: SkipMT_NS = False
6: **end if**

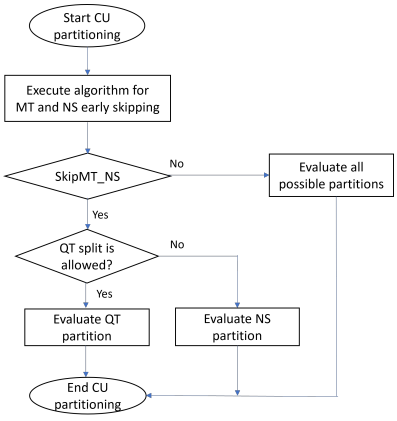

Fig. 3: Flow chart of proposed acceleration method

possible CU of the tree, if the QTdepthMap predicts that a QT split must be performed, then we remove from the RDO check, all MT and NS split checks.

More precisely, let us consider a CU at depth $\text{QTdepth}_{\text{cur}}$. Our predicted QTdepthMap provides a QTdepth value for all 8x8 subblocks of the CTU. Therefore, for the CU under consideration, we compute $\text{QTdepth}_{\text{avg}}$, the average of the QTdepth values given by the predicted QTdepthMap of all 8x8 subblocks inside the CU. Then, the predicted QTdepth of the CU is given by this $\text{QTdepth}_{\text{avg}}$- Threshold. Finally, if the predicted QTdepth value (*i.e.* $\text{QTdepth}_{\text{avg}}$- Threshold) is greater than the QTdepth value of the CU ($\text{QTdepth}_{\text{cur}}$), then a QT split is performed and all MT and NS split checks are avoided.

The Threshold value introduced above and denoted Th, has two advantages. First, it allows to compensate for outliers in the predicted QTdepth map. Second, it allows to produce a scalable tradeoff between compression performance and acceleration. Indeed, a high Threshold value, leads to less skips in the RDO checks. Therefore, compression performance will less be affected, but acceleration will be reduced. The opposite effect occurs for small Threshold value. Note that such scalability can not be achieved with the methods based on minimum and maximum values of [13]. The proposed method is summarized in Fig. 3, where the early skipping strategy is detailed in Algorithm 1.

2.4. Dataset and Training Details

The proposed method has been implemented in various VTM versions using the Frugally deep library [14] to load the trained model and to perform the inference on the CPU. 800 sequences of different resolutions, namely 240p, 540p, 720p and 1080p, have been encoded using the VVC Test Model 10 (VTM10) with the RAGOP32 configuration. From this set, 600 sequences of 64 frames come from the database BVI-DVC [15], while the remaining sequences of 600 frames have been selected from the Youtube UGC dataset [16]. We have collected the CTU-level features, mentioned in previous section, together with its QTdepthMap. 200k CTU samples have been randomly chosen for each resolution. The model has been trained using Keras on GTX1080ti with AMD 3700x processor. We used the L1 loss function and the Adam optimizer [17]. The learning rate has been set to $1e^{-3}$ for the first 20 epoches, then decreased by 10% every 10 epochs. The batch size set for training is 200. The model converges after 30 epoches.

Table 1: BD-rate and Time Saving tradeoff for the proposed method and state-of-the-art CNN-based methods. (The average of TS is geometric mean and that of BD-rate is arithmetic mean. Unit: %)

Average on Class	Pan [8] (VTM6)		Yeo [9] (VTM11)		Proposed (Th=0, VTM6)		Proposed (Th=0.1, VTM11)	
	BD-rate	TS	BD-rate	TS	BD-rate	TS	BD-rate	TS
A	3.85	36.46	1.95	16.99	2.12	35.88	1.47	26.05
B	3.76	30.27	0.95	16.19	1.59	31.79	1.06	23.34
C	2.56	25.77	0.09	4.05	0.39	22.94	0.29	17.29
D	2.35	21.53	0.11	2.34	0.19	12.34	0.15	8.82
E	2.81	35.15	NA	NA	1.15	29.18	0.80	14.23
Total average	**3.18**	**30.63**	**0.71**	**7.12**	**1.20**	**27.85**	**0.83**	**19.34**

3. EXPERIMENTAL RESULTS

The tests are conducted on CTC sequences with the RandomAccess (RA) configuration. BD-rate [18] and Time Saving (TS) have been used to assess the performance. The formula for computing TS is given in Equation 1. T_{Test} indicates the encoding time of the proposed method and T_{VTM} indicates for the encoding time of encoder in the same condition.

$$TS = \frac{1}{4} \sum_{QP \in \{22,27,32,37\}} \frac{T_{VTM}(QP) - T_{\text{Test}}(QP)}{T_{VTM}(QP)} \quad (1)$$

First, when test is conducted with VTM10, we obtain encoding time acceleration ranging from 17% to 30% for only 0.37% to 1.18% BD-rate loss in RAGOP32. Second, since performances of different VTMs vary significantly [19], and in order to perform fair comparison with state of the art methods, the tests have been also conducted for other VTM versions. Note however, that our CNN is trained on a dataset generated from VTM10, which works against our method.

In Table 1, the proposed method is compared with the two CNN-based inter partitioning methods in [8] and [9]. For a fair comparison, we have implemented our method in VTM6 and VTM11. The proposed method offers much better tradeoff between time saving and performance drop than the state of the art methods. More precisely, when we set the threshold Th in Algorithm 1 to 0, we obtain a similar acceleration as the method developed in [8] with a significantly lower loss. Similarly, when the threshold Th equals 0.1, acceleration is tripled and the loss remains the same as in [9].

The two other acceleration methods for inter partitioning of VVC, notably the RF-based works presented in [6] and [7], generate their results differently. The test set used in these papers is different from the JVET CTC. Besides, the encoder configuration is modified. Indeed, the maximum of MT split depth has been reduced from its default value 3 to 2 in [6] and [7]. We refer to this setup as VTM8 with MT2 in the following. Fig. 4 presents a performance comparison of RF solutions with our method implemented in VTM8 and VTM8 with MT2. For a fair comparison, results in this figure are calculated on all CTC sequences included in the test set of [7]. The label "Th" indicate the value of the threshold.

Interestingly, the proposed method achieves similar results to these of the RF based methods. This is remarkable as our method requires a unique network to predict the QT-depthmap for any QP value and any frame position in the GOP, and the learning has been performed for a different VTM version as the one used for testing.

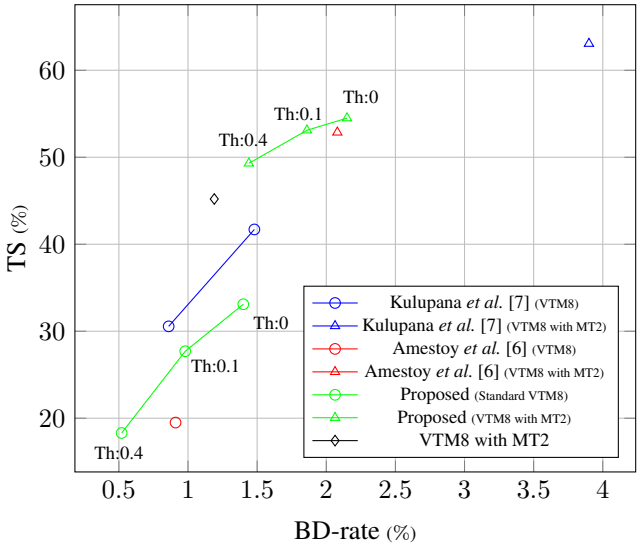

Fig. 4: Complexity gains versus BD rate loss in comparison with RF based methods.

4. CONCLUSION

In this paper, we have proposed a lightweight CNN-based CU partition acceleration algorithm specialised for inter coding in VVC. The proposed method adopts CNN-based skip rule to effectively speed up the RDO of MT and NS split types. This scalable method succeeds in speeding up the VTM10 efficiently from 17% to 30% for only 0.37% to 1.18% rate increase in RAGOP32 configuration, which largely outperforms CNN approaches. Compared with RF based methods, we have attained the same level of performance with a method which is easier to implement.

5. REFERENCES

[1] B. Bross et al., "Overview of the versatile video coding (VVC) standard and its applications," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3736–3764, 2021.

[2] Y. Huang et al., "Block partitioning structure in the VVC standard," *IEEE Trans. on Circuits and Systems for Video Technology*, 2021.

[3] Z. Wang et al., "Adaptive motion vector resolution scheme for enhanced video coding," in *Data Compression Conference (DCC)*. IEEE, 2016, pp. 101–110.

[4] L. Li et al., "An efficient four-parameter affine motion model for video coding," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 28, no. 8, pp. 1934–1948, 2017.

[5] A. Alshin et al., "Bi-directional optical flow for improving motion compensation," in *28th Picture Coding Symposium*. IEEE, 2010, pp. 422–425.

[6] T. Amestoy et al., "Tunable VVC frame partitioning based on lightweight machine learning," *IEEE Trans. on Image Processing*, vol. 29, pp. 1313–1328, 2019.

[7] G. Kulupana et al., "Fast Versatile Video Coding using Specialised Decision Trees," in *Picture Coding Symposium (PCS)*. IEEE, 2021, pp. 1–5.

[8] Z. Pan et al., "A CNN-Based Fast Inter Coding Method for VVC," *IEEE Signal Processing Letters*, 2021.

[9] W. Yeo et al., "CNN-based Fast Split Mode Decision Algorithm for Versatile Video Coding (VVC) Inter Prediction," *Journal of Multimedia Information System*, vol. 8, no. 3, pp. 147–158, 2021.

[10] X. Zhu et al., "B-CNN: branch convolutional neural network for hierarchical classification," *arXiv preprint arXiv:1709.09890*, 2017.

[11] Z. Wang et al., "Fast QTBT partitioning decision for interframe coding with convolution neural network," in *IEEE Int. Conf. on Image Processing (ICIP)*. IEEE, 2018, pp. 2550–2554.

[12] K. He et al., "Deep residual learning for image recognition," in *IEEE Conf. on computer vision and pattern recognition (CVPR)*, 2016, pp. 770–778.

[13] A. Feng et al., "Cnn-Based Depth Map Prediction for Fast Block Partitioning in HEVC Intra Coding," in *IEEE Int. Conf. on Multimedia and Expo (ICME)*, 2021, pp. 1–6.

[14] Tobias Hermann, "Frugally-Deep," https://github.com/Dobiasd/frugally-deep, 2018.

[15] D. Ma et al., "BVI-DVC: a training database for deep video compression," *IEEE Trans. on Multimedia*, 2021.

[16] Y. Wang et al., "YouTube UGC dataset for video compression research," in *IEEE Int. Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2019, pp. 1–5.

[17] "Adam: A method for stochastic optimization, author=Kingma, D.P. and others," *arXiv preprint arXiv:1412.6980*, 2014.

[18] G. Bjontegaard, "Calculation of average PSNR differences between rd-curves," *VCEG-M33*, 2001.

[19] Gary Sullivan, "Versatile Video Coding (VVC) Delivers: Coding Efficiency and Beyond, DCC," 2021.