RD-COST REGRESSION SPEED UP TECHNIQUE FOR VVC INTRA BLOCK PARTITIONING

M.E.A. Kherchouche[1,2]*, F. Galpin*[1]*, T. Dumas*[1]*, D. Menard*[2]*, L. Zhang*[2]

[1] InterDigital, R&I, 845a Avenue des Champs Blancs, 35510 Cesson-Sevigne
[2]Univ Rennes, INSA Rennes, CNRS, IETR - UMR 6164, F-35000 Rennes, France

ABSTRACT

The last standard Versatile Video Codec (VVC) aims to improve the compression efficiency by saving around 50% of bitrate at the same quality compared to its predecessor High Efficiency Video Codec (HEVC). However, this comes with higher encoding complexity mainly due to a much larger number of block splits to be tested on the encoder side. This paper proposes an acceleration of the VVC partitioning based on a multi-output regression model that predicts a suitable split mode for 32×32 Coding Unit (CU). Experimental results show that our approach improves complexity trade-offs flexibility while adding better complexity trade-offs points compared to the original encoder.

***Index Terms*—** Video compression, deep learning, multi-output regression.

1. INTRODUCTION

The rapid proliferation of video content driven by increasing online platforms, streaming services, and user-generated content, leads to an unprecedented surge in video traffic. This explosion in video consumption poses significant challenges and causes a crucial need for novel coding techniques. The Joint Video Experts Team (JVET) tackles this problem by introducing the new standardized video codec VVC [1], and its reference software VVC Test Model (VTM) used to implement and test the coding tools defined by the VVC standard. The encoding process can be considered as a combinatorial optimization problem and involves several stages: frame partitioning, intra/inter prediction, transform, quantization, and entropy coding. This work addresses the partitioning part in intra coded slice at the encoder side. In this paper, we propose a regression-based complexity reduction method for VVC tackling the exhaustive Rate-Distortion Optimization (RDO) search. This technique relies on a Convolutional Neural Network (CNN) that takes a 32×32 CU of the luma component and the associated Quantization Parameter (QP) to return a vector containing 6 rate-distortion costs, one for each split mode. Then, $n \leq 6$ split modes can be tested on this CU via RDO, n depending on pre-determined thresholds. Section 2 presents the state-of-the-art (SOTA) on partitioning speed-ups at the encoder side. Section 3 details the proposed multi-output regression model. Section 4 shows the mean BD-rate versus encoding complexity trade-offs of VVC with our model. Finally, Section 5 concludes the paper.

2. BACKGROUND

2.1. Encoding complexity

The input video sequence is divided into frames, and each frame is recursively split into blocks. In HEVC [2], each frame is divided into Coding Tree Units (CTUs). As a given CTU/CU can only be split into 4 non-overlapping square sub-CUs of same size, a.k.a Quad-Tree (QT) split, the recursive split of a CTU yields square CUs of potentially various sizes. For instance, in HEVC, for an intra slice, a full causal RDO on a 64×64 CTU tests at most 341 blocks. This corresponds to the reconstruction of 20480 pixels per mode tested. In addition to QT and no split, VVC allows the partition of a CU into rectangular sub-CUs via Multi-Type Tree (MTT). MTT comprises Binary Tree (BT) and Ternary Tree (TT). In BT, the CU is divided into 2 non-overlapping sub-CUs of same size, supporting both horizontal (BTH) and vertical (BTV) splits. In TT, the CU is divided into 3 non-overlapping sub-CUs with size ratio 1:2:1, supporting both horizontal (TTH) and vertical (TTV) splits. For example, in VVC, for an intra slice, a full causal RDO on a 64×64 CTU tests at most $721k$ blocks. This corresponds to the reconstruction of $19M$ pixels per mode tested. Thus, from the HEVC encoder to the VVC encoder, for worst cases, ~ 2000 times more blocks are to be tested and ~ 1000 times more pixels are to be reconstructed.

2.2. Related works

Researchers have already addressed the high complexity of the RDO via classification-based solutions. These solutions are presented in two groups, those viewed as supervised learning involving Convolutional Neural Networks (CNNs) and the others (based on either heuristics or Reinforcement Learning (RL)). Our proposition belongs to the first group.

Regarding the first group, in HEVC, Xu et al. [3] create an Early Terminated Hierarchical CNN that returns 21 predictions mapping the CTU/CU splits. In VP9 [4], Mukherjee *et al.* [5] design a Hierarchical Fully Convolutional Network predicting macro-block partition trees via a bottom-up approach. In HEVC with QTBT (HEVC with enhanced partitioning, prior to VVC) Galpin *et al.* [6] introduce a CNN analyzing the texture of a block, either luma or chroma component, to predict possible sub-block splits. For instance, in luma, the CNN is fed with the luma component of a 64×64 CU along with one-line left and above borders, forming a 65×65 input, plus QP, to return the probabilities of the 480 boundaries associated to each 4×4 sub-block. These boundary probabilities are turned into decisions of skipping split tests via hand-crafted heuristics. In VVC, Li *et al.* [7] propose a Multi-Stage Exit CNN combined with an early-exit mechanism to determine the CU partition. In VVC, Tissier *et al.* [8] build upon [6], replacing the hand-crafted heuristics by a decision tree. Finally, Feng *et al.* [9], propose a Down-Up-CNN to predict a partition map of an input block of 64×64 in VVC intra configuration. Note that, unlike [3, 5–9], our proposed approach relies on regression-based training.

Regarding the second group, in VVC, Qiang *et al.* [10] employ an algorithm for fast QTMTT based on the Scharr operator-derived gradient for texture description and sub-block edge difference for structural information. In HEVC, Na *et al.* [11] developed an early termination classifier trained using trajectories of CU decisions at varying depths. This is achieved through an end-to-end actor-critic RL algorithm, rendering the classifier depth-independent. In [12], the authors employ Deep Reinforcement Learning (DRL) to optimize the decision process of 32×32 CUs in VVC.

3. PROPOSED APPROACH

3.1. Motivations

In VVC, for an intra slice, the largest CU is 128×128, initially forcefully divided by QT into 4 64×64 CUs. Statistically, the split decisions on 32×32 CUs appear to be the most crucial. Indeed, for a larger CU, the split tends to be QT while, for smaller CUs, the number of combinations to be tested tend to be smaller than that for 32×32 CUs. Therefore, this study focuses on the prediction of the splits of 32×32 CUs.

Compared to previous works, our approach targets the prediction of a variable computed before the split decision. Indeed, instead of predicting the split decision, normalized RD costs are predicted. This gives higher flexibility to decide the split strategy. For instance, the differences in RD costs of different splits can help evaluate which splits to explore. Unlike classification task, with jumps from one class to another, the training by regressing on RD costs must better reflect the smoothness of the relative evolutions of the RD costs by varying texture and coding parameters. As illustration, Fig. 1 reveals the smoothness of the relative evolutions of the RD costs w.r.t. QP. It shows the evolution of the RD cost with QP for each split mode on a particular 32×32 block. At high bitrate, i.e. low QP, the best split mode is QT. But, at low bitrate, i.e. high QP, BTV surpasses QT as best split mode. Finally, at very low bitrate, No Split (NS) becomes the best split mode.

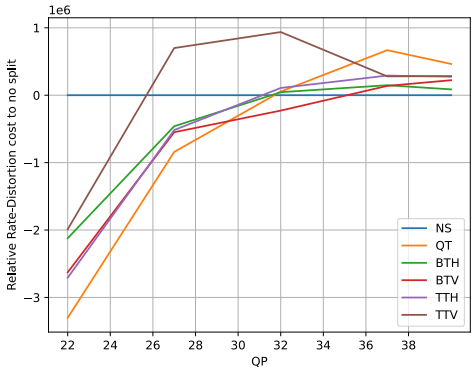

Fig. 1. Rate-distortion costs of a 32×32 CU for each splitting mode at different quantization parameters values. NS RD-cost is subtracted from all RD-costs.

3.2. Design and training

The proposed method is a CNN-based acceleration technique. A deep learning model takes a 32×32 CU to predict 6 RD cost values, one for each split mode in { NS, QT, BTH, BTV, TTH, TTV }. The model can be integrated at the encoder side of a hybrid-block-based video codec to skip unnecessary split modes evaluation, hence accelerating the RDO process. Fig. 2 shows the diagram of the proposed method. The CNN architecture is inspired by ResNet [13], being composed of convolutional layers with a kernel size of 3×3 followed by a ReLU activation function and maxpooling layers. The ConvBlock is made of 4 3×3 convolutional layers with n filters and one shortcut connection, see Fig. 2(b). After the last max-pooling layer, the QP value is concatenated with the resulting vector. Then, the model produces 6 outputs which represent the normalized RD cost of each split. The model contains $\sim 100k$ parameters, making it relatively light and fast at inference time.

The training dataset is generated through intra-mode encoding configuration of BVI-DVC sequences [14] utilizing VTM 18.0. Two versions are tested: the default RDO and the RDO with heuristics related to splits of 32×32 blocks deactivated (i.e. all partitioning modes on a 32×32 CU are tested). Each encoding is conducted across 4 QPs: 22, 27, 32 and 37. With 800 sequences, a balanced dataset of more than 6 million patches (1 million samples per split mode) is generated. The 32×32 CU of index i is represented by a 36×36 patch x_i^0, encompassing the block itself and a 4-pixel-wide

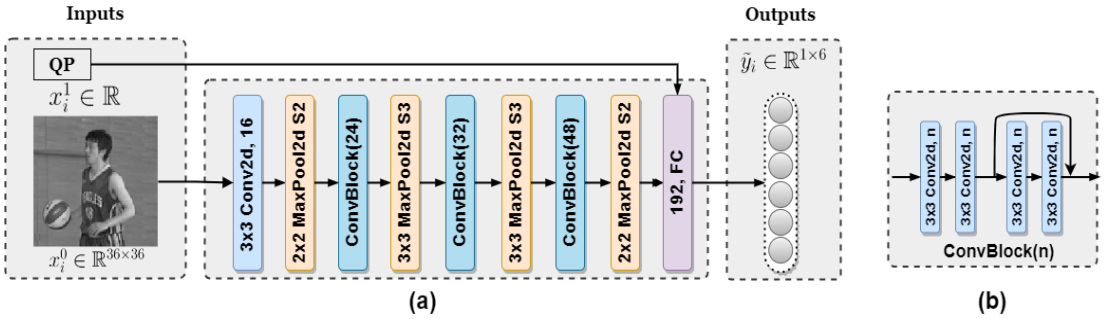

Fig. 2. (a) CNN model architecture overview. (b) Details the ConvBlock layers

causal border. Additionally, the QP, denoted x_i^1, of the CU of index i is extracted. The ground truth vector y_i corresponds to the RD costs selection determined by the RDO process applied to the 32×32 CU of index i. y_i stores 6 values, one RD cost for each tested split mode.

The normalization of ground truth values in continuous regression tasks is a crucial step. Thus, in our work, for the example of index i, the ground-truth $y'_{i,j}$ is a vector of 6 RD costs normalized by the NS RD cost $y_{i,0}$, see Eq. (1).

$$y'_{i,j} = \frac{y_{i,j}}{y_{i,0}}, j \in [|0, 5]. \quad (1)$$

j indexes the 6 split modes. For the example of index i, $y_{i,j}$ denotes the original RD cost.

The function f_θ, parameterized by θ, defines the CNN. f_θ takes x_i^0 and x_i^1 to calculate a prediction $\tilde{y_i}$ of the 6 normalized RD costs, see Eq. (2).

$$\tilde{y_i} = f_\theta(x_i^0, x_i^1) \quad (2)$$

During the training step, a loss function $\mathcal{L}$ computes the error between the predicted vector of RD costs and the ground-truth vector of normalized RD costs, averaged over all examples, plus a regularization term corresponding to the L2 norm of the weights W of the CNN, $W \subset \theta$, see Eq. (3).

$$\mathcal{L}(\mathcal{D}; \theta) = \frac{1}{N} \sum_{i=0}^{N-1} (y'_i - f_\theta(x_i^0, x_i^1))^2 + \lambda \sum_{k=1}^{K} W_k^2 \quad (3)$$

N denotes the total number of examples in the training set $\mathcal{D} = \{(x_0^0, x_0^1, y'_0), ..., (x_{N-1}^0, x_{N-1}^1, y'_{N-1})\}$. K denotes the total number of weights of the CNN. The CNN is trained within the Pytorch deep learning framework. The Adam optimizer is employed with a learning rate of 10^{-4}, λ regularization factor of 10^{-5}, and a batch size of 256. Model training spanned 10 epochs and is completed in 4 hours, using a single GPU (NVIDIA Tesla V100 16GB).

4. EXPERIMENTAL RESULTS

Our method is compared with VTM 18.0 by considering two main setups (with and without heuristics) and several MTT depths, see Fig. 3, which correspond to several complexity points of the encoder. The values 1, 2, 3 and 4 correspond to the maximum MTT Hierarchy Depth of the luma blocks. The first setup represents the anchor where the default VTM heuristics are activated for 32×32 CUs to generate the database using BVI-DVC sequences [14] as discussed in Section 3 and using the default MTT depth of 3 to train the CNN. In this setup, as the heuristics are activated, the VTM sometimes skips certain partitioning modes, leading to missing cost values in our labels in the training dataset. To fill in the missing values, we replace a missing value by the maximum value multiplied by a constant α (for instance $\alpha = 2$).

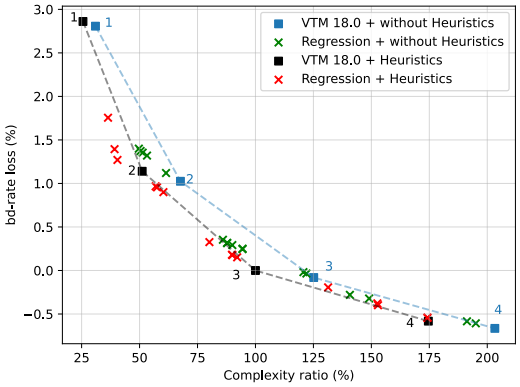

Fig. 3. BD-rate versus complexity speedup ratio comparison in All Intra (AI) configuration with three MTT depth configurations with/without activating heuristics for 32×32 CUs.

Thanks to this penalty, the model predicts relatively high costs for non tested splits. In the second setup, the heuristics are deactivated for 32×32 CUs, allowing the VTM to test all split modes. In this case, there is no missing information when generating the training dataset using the default MTT depth of 3 and, for each sample, all cost values exist. The same architecture illustrated in Fig. 2 is used to train the models of both setups using the appropriate training dataset for each CNN model. The model trained on dataset with heuristics activated is integrated in the first setup while the

Table 1. ΔC and bd-rate results of the first setup model in AI coding and default MTT depth configuration

Class	Sequence	bd-rate%	ΔC%
Class A1	Tango2	1,41	59,49
	FoodMarket4	1,61	57,23
	Campfire	0,63	59,37
	Average	**1,22**	**58,70**
Class A2	CatRobot1	0,94	57,72
	DaylightRoad2	1,41	55,68
	ParkRunning3	0,71	59,34
	Average	**1,02**	**57,58**
Class B	MarketPlace	1,15	58,29
	RitualDance	1,17	56,06
	Cactus	0,97	58,71
	BasketballDrive	1,95	56,79
	BQTerrace	0,74	64,49
	Average	**1,20**	**58,87**
Class C	RaceHorses	0,44	60,15
	BQMall	0,85	57,98
	PartyScene	0,28	63,73
	BasketballDrill	0,78	59,33
	Average	**0,59**	**60,30**
Class D	RaceHorses	0,26	66,91
	BQSquare	0,39	69,73
	BlowingBubbles	0,13	67,97
	BasketballPass	0,91	61,92
	Average	**0,42**	**66,63**
	Average A-B-C-D	**0,88**	**60,57**
Class E	FourPeople	1,22	56,49
	Johnny	1,16	59,38
	KristenAndSara	0,85	59,11
	Average	**1,08**	**58,33**
Class F	TencentAOV5	1,00	57,89
	BasketBallDrillText	0,80	59,84
	SlideEditing	0,92	62,81
	SlideShow	0,90	61,19
	Average	**0,90**	**60,43**
	Average A-B-C-D-E-F	**0,91**	**60,29**

second model trained on dataset without heuristics is integrated in the second setup after converting them from torch model to SADL model [15]. The SADL implementation allows a stand-alone integration of the model in the codec.

During the inference, for each 32×32 CU, the models predict 6 RD cost values and the final set of splitting modes to be checked by the classical RDO process is selected. The predicted values are compared by taking the splitting modes with costs lower than a given threshold, which can be changed. For each threshold, this gives a BD-rate loss and a speed-up compared to the anchor. Test sequences from the JVET Common Test Conditions [16] are used to evaluate our approach including all classes (i.e. class A to class F). The complexity reduction is considered as the difference of the total number of treated pixels during the encoding process with and without the CNN model and is calculated as follows:

$$\Delta C = \frac{1}{4} \sum_{QP_i \in \{22,27,32,37\}} \frac{T_a(QP_i) - T_p(QP_i)}{T_a(QP_i)} \quad (4)$$

where $T_p(QP_i), T_a(QP_i)$ are the complexities with and without the model integrated, respectively. Table 1. summarizes the BD-rate loss and complexity ratio for each sequence.

Fig. 3 shows the speed up ratio versus the BD-rate loss of the two setups with and without the models at three MTT configurations. The dashed line in black represents the first anchor with activated heuristics whereas the dashed line in blue refers to the second anchor without activating the heuristics. The red crosses represent our approach with the heuristics activated whereas the green crosses represent our method without activating heuristics. Our solution enables the inclusion of intermediate trade-offs when compared to a basic MTT depth.

One can notice that, several SOTA methods usually evaluate their approach at only one complexity point (typically the point of default depth 3, see Table 1). However, for lower complexity, for example a 2 times speed-up w.r.t. anchor, the point of depth 2 already has a trade-off of 50% complexity for a BD-rate loss of $\sim 1.1\%$, which is close to several SOTA results [7] [8].

The complexity reduction varies from 43% to 10% for 0.96% and 0.15% as BD-rate loss respectively in MTT depth of 3 for the first setup with the model compared to the model without heuristics which can do the same as the first anchor with the heuristics activated with a complexity reduction of 15% for 0.35% of BD-rate loss.

Based on the two setups, our approach with activated heuristics can accelerate the RDO process while maintaining the existing heuristics with a maximum complexity reduction of 43% for 0.96% BD-rate loss in MTT depth of 3. The model of the second setup allows for the addition of extra points within the gaps between MTT depth transitions and can achieve the performance of the anchor with activated heuristics at MTT depth of 3 with 15% complexity reduction for 0.35% BD-rate loss.

Finally, both models are trained on datasets with a default MTT depth of 3, enabling their applicability to other MTT depths for generating new trade-offs and accelerating the RDO process.

5. CONCLUSION

This paper proposes the first rate-distortion cost regression-based approach to speedup the rate-distortion search on top of VTM-18.0. In this approach, a CNN predicts rate-distortion costs of all split modes of the current inputs. The regression can help the model get relative understanding between the predicted rate-distortion costs and choose a suitable split.

Experimental results show that the generation of a complete dataset without heuristics (all costs are available) can be efficient for the CNN training step and help in getting additional complexity trade-offs during the inference step.

Finally, combining our CNN models with the associated setup can speedup the RDO process at different complexity trade-offs, for example at different MTT depths.

6. REFERENCES

[1] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J. Sullivan, and Jens-Rainer Ohm, "Overview of the versatile video coding (vvc) standard and its applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3736–3764, 2021.

[2] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[3] Mai Xu, Tianyi Li, Zulin Wang, Xin Deng, Ren Yang, and Zhenyu Guan, "Reducing complexity of hevc: A deep learning approach," *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 5044–5059, 2018.

[4] Debargha Mukherjee, Jingning Han, Jim Bankoski, Ronald Bultje, Adrian Grange, John Koleszar, Paul Wilkins, and Yaowu Xu, "A technical overview of vp9 – the latest open-source video codec," in *SMPTE 2013 Annual Technical Conference & Exhibition*, 2013, pp. 1–17.

[5] Somdyuti Paul, Andrey Norkin, and Alan C. Bovik, "Speeding up vp9 intra encoder with hierarchical deep learning-based partition prediction," *IEEE Transactions on Image Processing*, vol. 29, pp. 8134–8148, 2020.

[6] Franck Galpin, Fabien Racapé, Sunil Jaiswal, Philippe Bordes, Fabrice Le Léannec, and Edouard François, "Cnn-based driving of block partitioning for intra slices encoding," in *2019 Data Compression Conference (DCC)*, 2019, pp. 162–171.

[7] Tianyi Li, Mai Xu, Runzhi Tang, Ying Chen, and Qunliang Xing, "Deepqtmt: A deep learning approach for fast qtmt-based cu partition of intra-mode vvc," *IEEE Transactions on Image Processing*, vol. 30, pp. 5377–5390, 2021.

[8] Alexandre Tissier, Wassim Hamidouche, Souhaiel Belhadj Dit Mdalsi, Jarno Vanne, Franck Galpin, and Daniel Menard, "Machine learning based efficient qt-mtt partitioning scheme for vvc intra encoders," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 8, pp. 4279–4293, 2023.

[9] Aolin Feng, Kang Liu, Dong Liu, Li Li, and Feng Wu, "Partition map prediction for fast block partitioning in vvc intra-frame coding," *IEEE Transactions on Image Processing*, vol. 32, pp. 2237–2251, 2023.

[10] Qiang, Hui Li, Ya Meng, and Li, "Texture-based fast qtmt partition algorithm in vvc intra coding," *Signal, Image and Video Processing*, vol. 17, no. 4, pp. 1581–1589, 2023.

[11] Na Li, Yun Zhang, Linwei Zhu, Wenhan Luo, and Sam Kwong, "Reinforcement learning based coding unit early termination algorithm for high efficiency video coding," *Journal of Visual Communication and Image Representation*, vol. 60, pp. 276–286, 2019.

[12] Jinchao Zhao, Yihan Wang, Mingying Li, and Qiuwen Zhang, "Fast coding unit size decision based on deep reinforcement learning for versatile video coding," *Multimedia Tools Appl.*, vol. 81, no. 12, pp. 16371–16387, May 2022.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," 2015.

[14] Di Ma, Fan Zhang, and David R. Bull, "BVI-DVC: A training database for deep video compression," *IEEE Transactions on Multimedia*, vol. 24, pp. 3847–3858, 2022.

[15] Franck Galpin, Pavel Nikitin, Thierry Dumas, and Philippe Bordes, "SADL small adhoc deep-learning library," Tech. Rep. JVET-W0181, InterDigital, July 2021.

[16] Karczewicz Marta and Ye Yan, "Common test conditions and evaluation procedures for enhanced compression tool testing," *WG 05 MPEG Joint Video Coding Team(s) with ITU-T SG 16,* 30^{th} *meeting, Antalya*, June, 2023.