CNN ORIENTED COMPLEXITY REDUCTION OF VVC INTRA ENCODER

A. Tissier[⋆]*, W. Hamidouche*[⋆]*, J. Vanne*[†]*, F. Galpin*[‡] *and D. Menard*[⋆]

[⋆] Univ Rennes, INSA Rennes, CNRS, IETR - UMR 6164, Rennes, France.
[†] Computing Sciences, Tampere University, Tampere, Finland.
[‡] InterDigital, Cesson-Sévigné, France.
Emails: alexandre.tissier2@insa-rennes.fr, whamidou@insa-rennes.fr,
jarno.vanne@tuni.fi and dmenard@insa-rennes.fr

ABSTRACT

The Joint Video Expert Team (JVET) is currently developing the next-generation MPEG/ITU video coding standard called Versatile Video Coding (VVC) and their ultimate goal is to double the coding efficiency over the state-of-the-art HEVC standard.The latest version of the VVC reference encoder, VTM6.1, is able to improve the intra coding efficiency by 24% over the HEVC reference encoder HM16.20, but at the expense of 27 times the encoding time. The complexity overhead of VVC primarily stems from its novel block partitioning scheme that complements Quad-Tree (QT) split with Multi-Type Tree (MTT) partitioning in order to better fit the local variations of the video signal. This work reduces the block partitioning complexity of VTM6.1 through the use of Convolutional Neural Networks (CNNs). For each 64×64 Coding Unit (CU), the CNN is trained to predict a probability vector that speeds up coding block partitioning in encoding. Our solution is shown to decrease the intra encoding complexity of VTM6.1 by 51.5% with a bitrate increase of only 1.45%.

***Index Terms*—** Versatile Video Coding (VVC), Convolutional Neural Network (CNN), Multi-Type Tree (MTT), Complexity

1. INTRODUCTION

The explosion of IP video traffic [1] alongside emerging video formats like 4K/8K and 360-degree videos call for novel video codecs whose coding efficiency goes beyond the limits of the current High Efficiency Video Coding (HEVC) standard [2]. The International Telecommunication Union (ITU) and ISO/IEC Moving Picture Experts Group (MPEG) formed the Joint Video Expert Team (JVET) to address this new challenge by introducing a video coding standard called Versatile Video Coding (VVC) [3]. JVET is also developing a VVC reference software called VVC Test Model (VTM) that implements all normative VVC coding tools for practical rate-distortion-complexity evaluation and conformance testing. The latest version of VTM is VTM6.1.

This work addresses the All Intra (AI) coding configuration of VTM6.1. It is shown to improve intra coding efficiency by 24% [4] over that of the HEVC reference implementation HEVC test Model (HM)16.20. According to [5], around one third of this coding gain (8.5%) is obtained by extending a Quad-Tree (QT) block partitioning scheme of HEVC with a MTT partitioning, which also supports Binary-Tree (BT) and Ternary-Tree (TT) splits.

This work is partially supported by the French FUI project EFIGI, the REACTIVE project funded by Brittany region, the ANR CominLabs Labex, and the Academy of Finland (decision no. 301820)

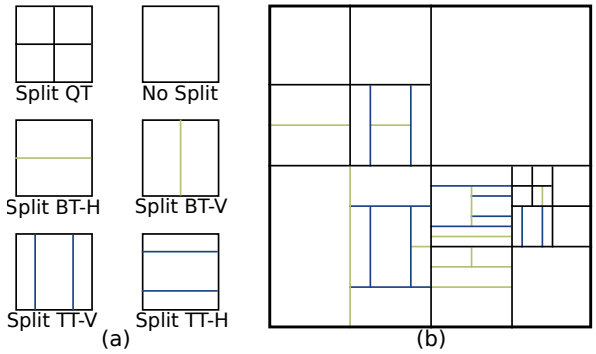

Fig. 1: Coding Tree Unit (CTU) partitioning in VVC. (a) VVC split types. (b) Example.

Fig. 1(a) illustrates the split types specified in VVC. As in HEVC, the QT split divides a Coding Unit (CU) into four square sub-CUs of the same size. In addition, VVC specifies the BT and TT split types that allow rectangular CU shapes. The BT split divides a CU into two sub-CUs of equal size whereas the TT split yields three sub-CUs with the ratio 1:2:1. The BT and TT splits are allowed in both horizontal (BTH and TTH) and vertical directions (BTV and TTV). However, the QT split is forbidden after either one BT or TT split has been initiated in the partitioning. The VTM encoder relies on the Rate-Distortion Optimization (RDO) process that calculates Rate-Distortion (RD)-cost for each split and finally selects the block partitioning with the lowest RD-cost. Fig. 1(b) shows a possible CTU partitioning with the QT, BT, and TT splits.

In this paper, we propose an efficient, CNN-based complexity reduction technique for VTM6.1 intra encoder. The CNN is fed with a 64×64 pixels luminance CU and it creates a vector that gives probabilities of having edges at the 4×4 boundaries of the block. This probability vector is further exploited by the encoder to skip unlikely splits. Our previous CNN based technique was already proposed in [6] to reduce the encoding complexity of an earlier version of VVC reference software with obsolete split types. This work extends our previous approach [6] to the latest VVC split types as well as the newly introduced coding tools in VTM6.1. To the best of our knowledge, no complexity reduction technique is currently based on a CNN for complexity reduction of the VTM encoder.

The remainder of this paper is organized as follows. Section 2 briefly describes state-of-the-art techniques. Section 3 introduces the proposed method and its integration into the VTM6.1 encoder. Section 4 presents the experimental setup and assesses the complexity

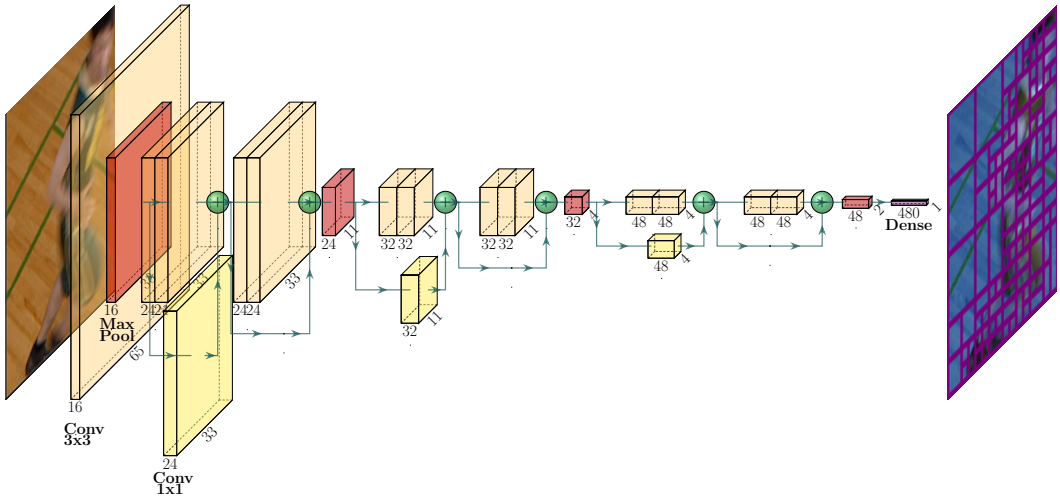

Fig. 2: The CNN structure with convolution layers in orange and yellow, max-pooling layers in red and fully connected layer in purple.

reduction under the VVC Common Test Conditions (CTC). Finally, Section 5 concludes the paper.

2. RELATED WORKS

In recent years, several approaches have dealed with VVC complexity reduction. The existing techniques reduce the MTT partitioning search space of QT, BT and TT splits [7], [8]-[9] or evaluate a reduced set of intra prediction modes [10], [11]. Additionally, our previous study [12] presents the complexity reduction opportunities in the VVC intra encoder of intra mode prediction, multiple transform choice and MTT partitioning. It shows that the MTT partitioning has the largest opportunities in term of complexity reduction.

Fu et al. [8] proposed two distinct fast block partition techniques through Bayesian decision rule. The first one explores information of previously tested BTH split to skip vertical split with a binary classification. The second technique intends to skip TTH split, depending on both sub-CUs partitioning and intra prediction mode. TTH split can also be skipped based on RD-cost difference between BTH and BTV. Yang et al. [10] focused first on MTT partitioning search through several binary classification problems. They used global texture information, local texture information, and context information for the classification. Secondly, they proposed a fast intra mode decision using one-dimensional gradient descent search. Lei et al. [13] proposed a technique to pre-process prediction information in order to skip redundant partition modes. Hadamard cost determines first the intra prediction mode for sub-CUs of the BT split. The optimal intra mode is further evaluated by using accumulated RD-cost of sub-CUs as an estimate of the RD-cost of their parent partition in order to prune sub-optimal modes. Park et al. [9] used a probabilistic approach and exploited the RD-cost of a previously encoded CU with BT splits to skip TT splits The decision to skip the RDO process of TT splits depends on the difference of RD-cost between BTH and BTV.

Compared with prior-art, our solution takes a step forward and reduces the complexity of VTM6.1. Unlike the others, it uses a CNN to predict optimal coding partitions and reduce the complexity of MTT partitioning search.

3. PROPOSED METHOD

In the VTM encoder, MTT partitioning has high complexity due to its exhaustive RDO search process, which goes through all possible splits to get the optimal partitioning. This section presents the CNN structure, the CNN training, and its integration into VTM6.1.

3.1. CNN structure

The BT and TT splits introduced in the VTM push machine learning techniques to multiply the number of classifiers [7], [10]. For instance, in [10], the authors proposed a cascade decision framework through 5 binary classifiers. This multiplication of classifiers can be an issue with their high computational inference time. CNN is computationally intensive that makes it necessary to limit the number of inferences as well as the CNN structure size. The objective of our solution is to limit the access to the CNN by predicting output probabilities for all recursive splits within 64×64 CU at a time.

The CNN illustrated in Fig. 2 is inspired by ResNet [14]. The orange layers represent a convolution with 3x3 kernel (Conv 3x3), whereas the yellow layers denote a convolution with 1x1 kernel (Conv 1x1) that allows the addition (marked in green) between layers with different dimensions. The red layers represent a max-pooling with a window of 2x2 (Max Pool). Finally, the purple layer is a fully connected layer (Dense) that predicts the vector of 480 probabilities which represents the 64×64 CU partitioning.

The CNN input is a 65×65 luminance block composed of a 64×64 CU plus one additional line on the top and left of the CU for intra mode computation. The prediction is a probability vector that corresponds to the 4×4 block boundaries of the 64×64 CU partitioning. The input and output sizes of the CNN correspond to the maximum CU size of the luminance in AI configuration (64×64). Fig. 3 presents the matching between the CU partitioning and the probability vector. For instance, the first value of the vector corresponds to the first bottom boundary of the top-left 4×4 block.

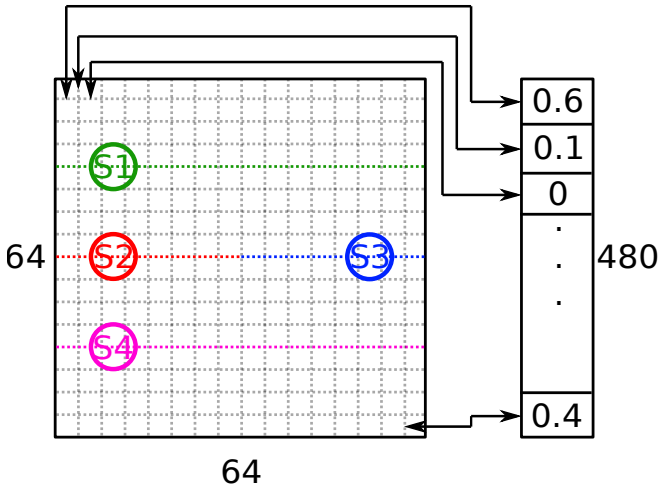

Fig. 3: Correspondence between a predicted probability vector and a partitioning of 64×64 CU.

3.2. Training process

The training process optimizes the CNN weights in order to minimize the loss function, which is defined as

$$\mathcal{L} = ||y - \tilde{y}||_2^2 + \lambda \sum_k c_k, \tag{1}$$

where y is the ground truth probability vector, $\tilde{y}$ is the predicted probability vector, $||\cdot||_2$ stands for the l_2 norm, and $\lambda = 10^{-5}$. Regularizers are applied on the kernels of convolutional layer c_k. These penalties are incorporated in the loss function.

Optimization of the CNN is performed using Adam optimizer with the default parameters provided in [15]. The training is carried out under Python3.6 with the Keras framework [16] running on top of Tensorflow [17]. The model is trained for 10 epochs with a batch size of 256 on a GPU (RTX 2080 Ti).

An input dataset of 65×65 luminance blocks and its corresponding ground truth vectors is conceived. The 65×65 luminance blocks are extracted from Div2k [18] and 4K image [19] datasets. These datasets are only composed of still images as our solution focuses on AI configuration. This brings more diversity than a training with video datasets only. No CTC [20] sequences are used for the training. The CNN takes as an input a 65×65 luminance block normalized between $[0\,,1]$. The input dataset is encoded by VTM6.1 anchor under AI configuration in order to establish the corresponding ground truth. The MTT partitioning information is gathered for each 64×64 CU and converted to the output format of the CNN, i.e. ground truth vectors of 480 probabilities composed of one (split) and zero (not split).

3.3. Inference in VTM

The proposed solution is implemented in the VTM6.1. The CNN inference is carried out in C++ through the frugally-deep library [21] with a trained Python model. This framework offers a conversion from the model trained in Python to a file interpretable in C++.

Fig. 3 illustrates the skip process for the horizontal splits, i.e. BTH and TTH splits. Their probabilities are deduced as follows. The mean probabilities are first computed on the segments S1, S2, S3, and S4, denoted by $\mathbb{P}(S1)$, $\mathbb{P}(S2)$, $\mathbb{P}(S3)$, and $\mathbb{P}(S4)$, respectively. The probability of the BTH split, denoted by $\mathbb{P}(BTH)$, is the minimum between the probabilities of S2 and S3 as defined in

$$\mathbb{P}(BTH) = min(\mathbb{P}(S2), \mathbb{P}(S3)). \tag{2}$$

The probability of the TTH split is computed with its respective segments as $(min(\mathbb{P}(S1), \mathbb{P}(S4))$. The vertical splits, i.e. BTV and TTV splits, follow the same procedure but with vertical segments. Regarding the QT split probability, $\mathbb{P}(QT)$ is calculated from the probabilities of the BTH and BTV splits as

$$\mathbb{P}(QT) = min(\mathbb{P}(BTH), \mathbb{P}(BTV)). \tag{3}$$

In all these cases, a split $S \in \{QT, BTH, BTV, TTH, TTV\}$ is skipped if $\mathbb{P}(S)$ is below a predefined threshold value β. By means of this threshold, our solution is able to offer different rate-distortion-complexity characteristics. The higher the threshold value β, the less splits take place, which results in both higher complexity reduction and coding loss. Our solution supports a threshold value range of $\beta \in \{0, 100\}$.

4. EXPERIMENTAL RESULTS

This section details the experimental setup and analyses our results over the state-of-the-art techniques. Several configurations of our solution are assessed in order to propose different rate-distortion-complexity trade-offs.

4.1. Experimental setup

All our experiments were conducted under AI configuration with VTM version 6.1. Each encoding and CNN prediction was carried out individually on Intel Xeon E5-2603 v4 processor running at 1.70 GHz on Ubuntu 16.04.5 operating system. A test set was composed of CTC sequences specified by JVET. The CTC test set was selected because it is widely used and it contains a wide range of resolutions, textures, bit depths, and motions. Altogether, there are 26 sequences separated into six classes: A (3840x2160), B (1920x1080), C (832x480), D (416x240), E (1280x720), and F (832x480 to 1920x1080). They are encoded with four Quantization Parameter (QP) values: 22, 27, 32, and 37.

The coding quality is measured with Bjøntegaard Delta Bit Rate (BD-BR) [22] and complexity reduction with Δ Encoding Time (ΔET) that is determined as

$$\Delta ET = \frac{1}{4} \sum_{QP_i \in \{22,27,32,37\}} \frac{T_R(QP_i) - T_C(QP_i)}{T_R(QP_i)}, \tag{4}$$

where T_R is the reference encoding time of the VTM6.1 anchor and T_C encoding time of VTM6.1 with the proposed complexity reduction techniques. The execution time of the CNN is not included in T_C as it highly depends on the processor performance. However, for the sake of comparison, the CNN is able to compute all inferences for benchmarked 4K test sequences in less than 2% of the VTM encoding time. Furthermore, it can be done in parallel with the RDO process.

4.2. Results and analysis

There are several existing techniques that seek to reduce the complexity of MTT partitioning search in VTM. The most competitive ones have been implemented by Lei et al. [13] in VTM3.0 and Park

Table 1: Performance of the proposed solution in VTM6.1 with different threshold values β in comparison with state-of-the-art techniques.

Class	Lei et al. [13], VTM3.0		Park et al. [9], VTM4.0		Proposed $\beta = 10$		Proposed $\beta = 20$		Proposed $\beta = 30$	
	BD-BR	ΔET	BD-BR	ΔET	BD-BR	ΔET	BD-BR	ΔET	BD-BR	ΔET
Class A1	0.79%	44.9%	0.67%	32.0%	0.35%	45.3%	0.87%	56.3%	1.55%	62.9%
Class A2	0.96%	39.5%	1.07%	33.0%	0.30%	40.1%	0.83%	52.6%	1.47%	60.0%
Class B	1.06%	45.1%	0.98%	33.0%	0.26%	36.9%	0.75%	51.5%	1.41%	61.1%
Class C	1.09%	48.3%	1.17%	35.0%	0.15%	13.9%	0.56%	26.4%	1.20%	37.9%
Class D	0.97%	44.2%	0.88%	35.0%	0.08%	13.0%	0.33%	22.7%	0.83%	32.5%
Class E	1.32%	47.9%	1.34%	34.0%	0.42%	29.5%	1.18%	43.8%	2.29%	54.4%
Mean	1.03%	45.0%	1.02%	33.7%	0.26%	29.8%	**0.75%**	**42.2%**	1.45%	51.5%
Class F	Nan	Nan	Nan	Nan	0.21%	15.2%	0.75%	25.4%	1.61%	36.3%

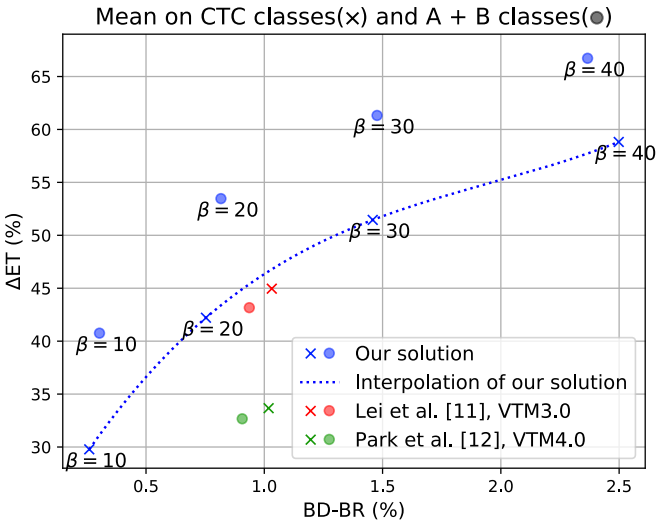

Fig. 4: Performance comparison between the proposed solution and state-of-the-art techniques. Circles are results of Full Hd and 4K sequences. Crosses are results of CTC classes without class F.

et al. [9] in VTM4.0. Thereafter, various new coding tools have been adopted to VTM, such as low frequency non-separable transforms [23], intra sub-partitioning [24], and joint chroma residual coding [25]. However, the partitioning search is kept unchanged, so the comparison between our technique in VTM6.1 and techniques in [13], [9] remains still relevant and fair.

Tab. 1 presents our results with $\beta = 10, 20, 30$ over the techniques in [13], [9]. The first proposed configuration with $\beta = 10$ limits the BD-BR increase to only 0.26% with 29.8% complexity reduction. The second configuration with $\beta = 20$ still features a negligible BD-BR increase of 0.75% and a high complexity reduction of 42.2%. Our last configuration with $\beta = 30$ halves the complexity for 1.45% BD-BR increase.

In VTM3.0, Lei et al. [13] achieved quite the same complexity reduction but for an increase of 0.28% BD-BR compared to our second configuration. Park et al. [9] obtained 33.7% complexity saving for 1.02% BD-BR increase, so it lags behind our second solution both in terms of complexity and BD-BR.

Our solution achieves better coding performance on high resolution sequences especially on classes A-B. For instance, our proposal with $\beta = 10$ almost halved the complexity with only 0.35% BD-BR increase for class A1. For $\beta = 30$, the complexity reduction reaches 62.9% with 1.55% BD-BR increase. For low-resolution C-D classes, our solution is able to reduce the complexity by 35.2% with 1.01% BD-BR increase. The training database is mainly composed of 4K and Full HD images, so our CNN performs better with them.

Fig. 4 plots the relative complexity reduction (ΔET) versus BD-BR increase over the CTC classes A-E (marked with crosses) for the proposed four configurations ($\beta = 10, 20, 30, 40$) and related works [13], [9]. Tackling the VVC coding complexity is of particular importance to higher resolutions so the corresponding results are also separately given for classes A-B (marked with circles).

With high-resolution sequences, the complexity saving of our proposal ranges from 40.8% to 66.7% and BD-BR increase from 0.30% to 2.37%. With $\beta = 20$, the overall complexity reduction is 42.2% for only 0.75% BD-BR increase. For classes A-B, the respective metrics are 53.5% and 0.82%. Compared with previous configuration ($\beta = 20$), [13] achieves 10.3% lower complexity reduction for a BD-BR increase of 0.12%. It also outperforms [9] in terms of both BD-BR and complexity reduction.

To conclude, our solution is able to achieve higher complexity reduction and BD-BR gain than [13], [9] with high-resolution sequences, comparable results with smaller resolutions, and averagely better coding performance with the entire CTC test set. Multiple threshold values also make our implementation more configurable to various operating points, like practical video encoders with several presets.

5. CONCLUSION

This paper presented a CNN-based complexity reduction technique for a VVC reference encoder VTM6.1. The CNN is used to analyze the texture inside each 64×64 coding block and predict vector probabilities for 4×4 boundaries inside these blocks. From the probability of boundaries, a split probability is deduced and compared with a predefined threshold. The execution time of the CNN is negligible compared to the VTM encoding time. In VTM6.1 intra coding, the proposed solution enables 42.2% complexity reduction for a slight BD-BR increase of 0.75%. With high-resolution sequences, the speedup is even higher, up to 54.5% at a cost of 0.85% BD-BR overhead. Our proposal allows several configurations and the majority of them overcome the state-of-the-art techniques.

These promising results motivate us to push our approach a step further and examine the database disparity and the CNN performance. A statistical behaviour of threshold values will also be investigated through in-depth statistical analysis.

6. REFERENCES

[1] Cisco, "Cisco Visual Networking Index : Forecast and Trends, 2017-2022," Tech. Rep., 2019.

[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, Dec. 2012.

[3] Benjamin Bross, Jianle Chen, and Shan Liu, "Versatile Video Coding (Draft 6)," *JVET-O2001*, July 2019.

[4] Frank Bossen, Xiang Li, and Karsten Suehring, "AHG report: Test model software development (AHG3)," *JVET-P0003*, Oct. 2019.

[5] A. Wieckowski, J. Ma, H. Schwarz, D. Marpe, and T. Wiegand, "Fast Partitioning Decision Strategies for The Upcoming Versatile Video Coding (VVC) Standard," in *2019 IEEE International Conference on Image Processing (ICIP)*, Sept. 2019, pp. 4130–4134.

[6] F. Galpin, F. Racapé, S. Jaiswal, P. Bordes, F. Le Léannec, and E. François, "CNN-Based Driving of Block Partitioning for Intra Slices Encoding," in *2019 Data Compression Conference (DCC)*, Mar. 2019, pp. 162–171.

[7] Thomas Amestoy, Alexandre Mercat, Wassim Hamidouche, Daniel Menard, and Cyril Bergeron, "Tunable VVC Frame Partitioning Based on Lightweight Machine Learning," *IEEE Transactions on Image Processing*, vol. 29, pp. 1313–1328, 2020.

[8] Ting Fu, Hao Zhang, Fan Mu, and Huanbang Chen, "Fast CU Partitioning Algorithm for H.266/VVC Intra-Frame Coding," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*, Shanghai, China, July 2019, pp. 55–60, IEEE.

[9] Sang-Hyo Park and Je-Won Kang, "Context-Based Ternary Tree Decision Method in Versatile Video Coding for Fast Intra Coding," *IEEE Access*, vol. 7, pp. 172597–172605, 2019.

[10] 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 Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2019.

[11] S. Ryu and J. Kang, "Machine Learning-Based Fast Angular Prediction Mode Decision Technique in Video Coding," *IEEE Transactions on Image Processing*, vol. 27, no. 11, pp. 5525–5538, Nov. 2018.

[12] A. Tissier, A. Mercat, T. Amestoy, W. Hamidouche, J. Vanne, and D. Menard, "Complexity Reduction Opportunities in the Future VVC Intra Encoder," in *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, Kuala Lumpur, Malaysia, Sept. 2019, pp. 1–6, IEEE.

[13] M. Lei, F. Luo, X. Zhang, S. Wang, and S. Ma, "Look-Ahead Prediction Based Coding Unit Size Pruning for VVC Intra Coding," in *2019 IEEE International Conference on Image Processing (ICIP)*, Sept. 2019, pp. 4120–4124.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015, arXiv: 1512.03385.

[15] Diederik P. Kingma and Jimmy Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017, arXiv: 1412.6980.

[16] François Chollet et al, *Keras*, 2015.

[17] Martın Abadi et al, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," p. 19.

[18] Eirikur Agustsson and Radu Timofte, "NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Honolulu, HI, USA, July 2017, pp. 1122–1131, IEEE.

[19] Evgeniu Makov, *Dataset image 4k*, 2019.

[20] F. Bossen, J. Boyce, K. Suehring, X. Li, and V. Seregin, "JVET common test conditions and software reference configurations for SDR video," *JVET-M1010*, Jan. 2019.

[21] Tobias Hermann, *Frugally Deep*, 2018.

[22] Gisle Bjontegaard, "Calculation of average PSNR differences between RD-curves," *VCEG-M33*, Apr. 2001.

[23] Moonmo Koo, Jaehyun Lim, Mehdi Salehifar, and Seung Hwan Kim, "Low frequency non separable transform," *JVET-N0193*, Mar. 2019.

[24] S. De-Luxán-Hernández, V. George, J. Ma, T. Nguyen, H. Schwarz, D. Marpe, and T. Wiegand, "An Intra Subpartition Coding Mode for VVC," in *2019 IEEE International Conference on Image Processing (ICIP)*, Sept. 2019, pp. 1203–1207.

[25] Christian Helmrich, Heiko Schwarz, Tung Nguyen, Christian Rudat, Detlev Marpe, Thomas Wiegand, Bappaditya Ray, Geert Van der Auwera, Adarsh Ramasubramonian, Muhammed Coban, and Marta Karczewicz, "Joint chroma residual coding with multiple modes," *JVET-O0105*, July 2019.