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.](image-url)
2. RELATED WORKS

In recent years, several approaches have dealt 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 terms 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 $64 \times 64$ 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 $64 \times 64$ CU partitioning.

The CNN input is a $65 \times 65$ luminance block composed of a $64 \times 64$ 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 $4 \times 4$ block boundaries of the $64 \times 64$ CU partitioning. The input and output sizes of the CNN correspond to the maximum CU size of the luminance in AI configuration ($64 \times 64$). 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 $4 \times 4$ block.
3.2. Training process

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

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

where \(y\) is the ground truth probability vector, \(\hat{y}\) 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 \(65 \times 65\) luminance blocks and its corresponding ground truth vectors is conceived. The \(65 \times 65\) 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 \(65 \times 65\) 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 \(64 \times 64\) 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 \(P(S1), P(S2), P(S3), \) and \(P(S4)\), respectively. The probability of the BTH split, denoted by \(P(BTH)\), is the minimum between the probabilities of \(S2\) and \(S3\) as defined in

$$P(BTH) = \min(P(S2), P(S3)).$$

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

$$P(QT) = \min(P(BTH), P(BTV)).$$

In all these cases, a split \(S \in \{QT, BTH, BTV, TTH, TTV\}\) is skipped if \(P(S)\) is below a predefined threshold value \(\beta\). By means of this threshold, our solution is able to offer different rate-distortion-complexity characteristics. The higher the threshold value \(\beta\), 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 \(\Delta Encoding Time (\Delta ET)\) that is determined as

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

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 $\beta$ in comparison with state-of-the-art techniques.

<table>
<thead>
<tr>
<th>Class</th>
<th>Lei et al. [13], VTM3.0</th>
<th>Park et al. [9], VTM4.0</th>
<th>Proposed $\beta = 10$</th>
<th>Proposed $\beta = 20$</th>
<th>Proposed $\beta = 30$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BD-BR</td>
<td>$\Delta ET$</td>
<td>BD-BR</td>
<td>$\Delta ET$</td>
<td>BD-BR</td>
</tr>
<tr>
<td>Class A1</td>
<td>0.79%</td>
<td>44.9%</td>
<td>0.67%</td>
<td>32.0%</td>
<td>0.35%</td>
</tr>
<tr>
<td>Class A2</td>
<td>0.96%</td>
<td>39.5%</td>
<td>1.07%</td>
<td>33.0%</td>
<td>0.30%</td>
</tr>
<tr>
<td>Class B</td>
<td>1.06%</td>
<td>45.1%</td>
<td>0.98%</td>
<td>33.0%</td>
<td>0.26%</td>
</tr>
<tr>
<td>Class C</td>
<td>1.09%</td>
<td>48.3%</td>
<td>1.17%</td>
<td>35.0%</td>
<td>0.15%</td>
</tr>
<tr>
<td>Class D</td>
<td>0.97%</td>
<td>44.2%</td>
<td>0.88%</td>
<td>35.0%</td>
<td>0.08%</td>
</tr>
<tr>
<td>Class E</td>
<td>1.32%</td>
<td>47.9%</td>
<td>1.34%</td>
<td>34.0%</td>
<td>0.42%</td>
</tr>
<tr>
<td>Mean</td>
<td>1.03%</td>
<td>45.0%</td>
<td>1.02%</td>
<td>33.7%</td>
<td>0.26%</td>
</tr>
<tr>
<td>Class F</td>
<td>Nan</td>
<td>Nan</td>
<td>Nan</td>
<td>Nan</td>
<td>0.21%</td>
</tr>
</tbody>
</table>

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.

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 $64 \times 64$ coding block and predict vector probabilities for $4 \times 4$ 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.

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

The 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


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


