Pictures Decoding Time Estimation for Low-Power VVC Software Decoding

Pierre-Loup Cabarat, Oussama Hammami, Daniel Menard, Hafssa Boujida

Pictures Decoding Time Estimation for Low-Power VVC Software Decoding

Pierre-Loup Cabarat, Oussama Hammani, Hafssa Boujida and Daniel Menard
University of Rennes, INSA Rennes, CNRS, IETR - UMR 6164
Rennes, France
firstname.lastname@insa-rennes.fr

***Abstract*—This paper addresses the issue of decoding time estimation aiming to reduce the power consumption of software decoding via Dynamic Voltage and Frequency Scaling (DVFS) techniques. Some high level syntax elements information is extracted alongside with each picture decoding time measured from a real-time software video decoder. Relationships between these syntax elements and decoding time are analyzed to select appropriate parameters for regression methods. Finally various regression methods are compared with regards to practical inference time and estimation accuracy required by a theoretical VVC Video decoding DVFS setup.**

***Index Terms*—VVC, DVFS, decoding time, FPS...**

I. Introduction

With the advent of video services such as Video-on-Demand, web TV, video sharing sites and live streaming, digital video has become ubiquitous in our lives. Technological advances make it possible to capture, broadcast and view digital video anytime, anywhere. As new video formats emerge, the amount of video data will continue to grow over time. According to a Cisco study [1], video traffic has quadrupled in five years to 81% of all Internet traffic. This massive increase in video traffic has led the Motion Picture Experts Group (MPEG) of ISO to propose the new Versatile Video Coding (VVC/H.266) standard [2] in 2020. This standard shows bit rate distortion (BD-BR) gains of up to 40% over HEVC [3] [4]. However, this bitrate reduction comes with the introduction of advanced tools that increase the overall complexity of the codec. In fact, the complexity of VVC is estimated to be up to 27 times greater than HEVC on the encoder side, and 2 times greater on the decoder side [4]. The decoding process being carried out at each viewing, minimizing the decoding energy consumption is of the utmost importance.

The most energy efficient solution for video decoding in mobile devices is to integrate a hardware (HW) decoder in the system on a chip (SoC). However, from a sustainability perspective, energy-efficient software decoding might become an interesting substitute. This latter allows to extend the life of the handset and avoid technological obsolescence by providing a software decoder whenever a given codec is not supported by the HW decoder. The challenge is therefore to provide energy-efficient software VVC decoding. Energy consumption can be reduced by exploiting the processor's different levels of parallelism to lessen the overall time required to decode the video. Energy savings can also be achieved by running the decoder at lower power levels during periods of low processing demand and scaling the processor frequency its actual workload. However the decoding time may fluctuate considerably from one picture to another due to variations in the underlying content. This variability annihilates the efficiency of classical CPU frequency governors available in operating systems. Nevertheless if the decoding time of each picture could be estimated, the decoder could intelligently adjust its frequency based on an estimation of the complexity of the upcoming pictures sequence, enabling efficient frequency scaling and optimizing power consumption. This approach requires the development of an accurate estimation model that estimates the decoding time of each picture, taking into account the various factors that influence the decoding process.

In [5], two models for decoding energy are constructed on the characteristic of each Coding Unit (CU). Among the respectively 230 and 67 proposed features most of them correspond to syntax elements only accessible during picture decoding process, which prevents from an a priori picture decoding time estimation. To counter this issue, [7] proposes some VVC green metadata to be computed by the encoder and embedded within the bitstream. Provided with the green metadata content, the decoder can use an appropriate model to estimate picture decoding time.

This paper proposes a new lightweight VVC decoding time estimator which provides a good balance between the inference time and the estimation quality. Compared to the state-of-the-art, this estimator requires no metadata, temporal information are exploited and only a few features are used. These features are associated with each picture and extracted by parsing the high-level syntax of the bitstream, allowing estimation of the decoding time of a picture before its actual decoding. This estimator may consist in a key element to a closed-loop system adjusting the processor clock frequency to minimize the decoder energy consumption.

This paper is structured as follows: Section 2 presents the background and the the existing approaches. Section 3 gives an overview of the data generation and preparation process. Section 4 presents decoding time statistics. The tested models to estimate the decoding time are detailed in Section 5. Finally, Section 6 concludes this paper.

This work has been conducted in the context of the 3EMS-2 project funded by Région Bretagne, Rennes Metropole, co-funded by EU FEDER program and supported by Images et Reseaux cluster.

II. Background and State of the Art

A. Dynamic Voltage and Frequency Scaling approaches

In modern Systems-on-a-Chip (SoC), the high computing capabilities make room for reducing the power consumption of a decoder by dynamically tuning its processor's voltage and frequency. Dynamic Voltage and Frequency Scaling (DVFS) techniques optimize power consumption by adjusting in real-time the processor operating voltage and frequency according to changing workload and processing requirements. Nevertheless, typical DVFS strategies fail to adapt efficiently the processor clock frequency to the decoder load due to the high variation of the decoding time from one picture to another. As shown in our experiment (see Figure 2), for a given resolution, frame-rate and bitrate, the decoding time can vary up to a factor of 50 between the lowest and the highest decoding time. To overcome, this dynamic behavior, a hypothetical closed-loop system in which the processor clock is controlled according to the decoding process load is described in Figure 1.

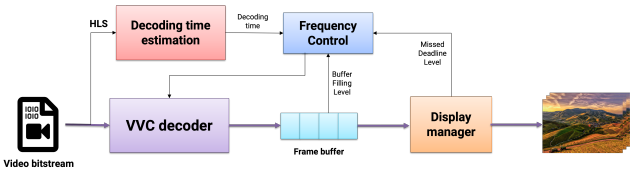


Fig. 1: Block Diagram of a hypothetical DVFS System Integration inside a video Player System

The decoder reconstructs each picture of the video from the data contained in the bitstream. Meanwhile, the display manager expects the decoded pictures for presentation at a fixed rate F_d set by the system. To find the minimum working frequency, and thus, run as slow as possible, estimating the incoming picture decoding time is essential. The following section describes the different approaches proposed to estimate the decoding time.

B. Techniques for decoding time estimation

In [7], the use of green metadata to vary the operating frequency and reduce decoder power consumption is discussed. Complexity Metrics (CM) metadata are proposed to feed the decoder with information about the complexity of decoding a picture. CM metadata may include information such as the level of details in video pictures, the amount of motion in the scene, and the complexity of the coding algorithms. In [5] the objective of minimizing decoder power consumption is addressed by developing precise models based on bitstream features. These models can be used for decoding-energy-rate-distortion optimization (DERDO) [6] to reduce the decoder energy demand. This research paper introduces two models to estimate the decoding energy of the VVC Test Model (VTM) decoder: the feature-based versatile model (FV) and the feature-based simple versatile model (FVS). These models incorporate a series of features classified according to different aspects of video coding, including intra-prediction, inter-prediction, transform, and loop filter. The FV model comprises 230 features, while the FVS model comprises 67 features. It should be noted that these models, even though have achieved significant results, are primarily focused on energy consumption reduction. Real-time decoding would require additional considerations beyond the scope of these models.

In [8], a machine-learning model that estimates the decoding time of individual pictures is proposed. This model is built on the ExtRaTrees regressor that accurately estimates the decoding time of 1080p video pictures with a low relative error of 5.58% and a high R2 score of 94%. This model entails the utilization of picture-related data that is readily reachable by the decoder, allowing estimating the decoding time of a picture just before its decoding.

Compared to [8], the model proposed in this paper is not dedicated for a specific resolution but is valid for different resolutions. In addition to features extracted from the "high-level syntax", the temporal information of previous pictures decoding times are also considered. Finally, the model inference time has been taken into account and evaluated with an optimized C source code.

III. Dataset constitution

This section describes the coding configurations used in our experiments. The first subsection discusses the video sequences datasets and coding configuration selection, while the second section details the picture decoding time measurements and feature extraction procedure.

A. Coded Video Data Sequences Generation

TABLE I: Dataset Coding Parameters

Resolution		***Target rates***	***Frame-rate***	***CTU sizes***
UHD 4K	(3840x2160)	1 Mbps	120 Hz	128
FHD 1080p	(1920x1080)	2 Mbps	60 Hz	64
HD 720p	(1280x720)	4 Mbps	30 Hz	32
SD 544p	(960x544)	8 Mbps		
Video Sequences				
BVI-HFR			UVG DataSet	
bobblehead	guitarfocus	pond	Beauty	
books	hamster	pour	Bosphorus	
bouncyball	joggers	sparkler	HoneyBee	
catch	lamppost	typing	Jockey	
catchtrack	leaveswall	waterripples	ReadySetGo	
cyclist	library	watersplashing	ShakeNDry	
flowers	martialarts		YachtRide	
golfside	plasma			

To achieve sufficient diversity in both spatial and temporal characteristics various video sequences have been used. The original video sequences considered in this work were taken from both the UVG DataSet [10] and BVI_HFR [11]. To increase the diversity of content UHD (3840×2160) sequences with a frame-rate of 120 FPS were down-sampled to FHD 1080p (1920×1080), HD 720p (1280×720) and SD 544p (960×544) at both 60 and 30 Hz using FFmpeg. The VVEnc encoder [12] with slow preset was selected as a good trade-off

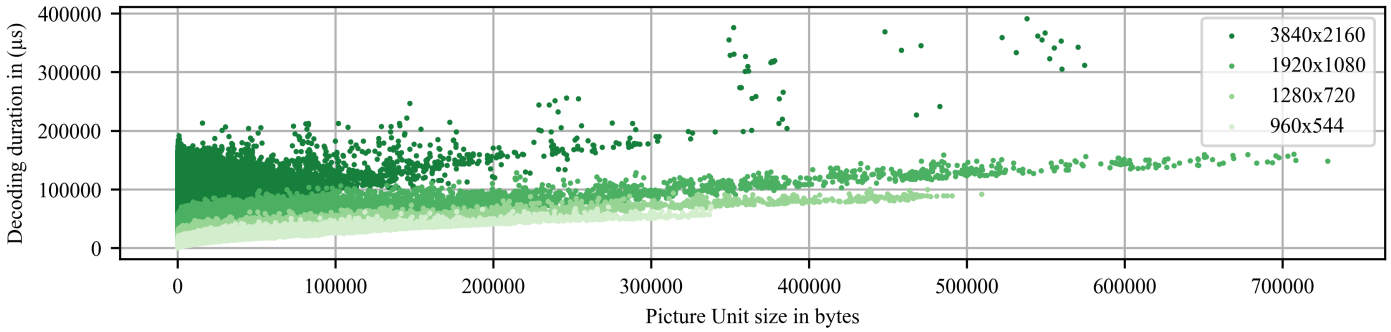


Fig. 2: Evolution of the picture decoding time according to Picture Unit size in bytes for the different resolutions

between the encoding speed and bit-rate performances. Indeed, most of VVC tools are exploited and this encoder is faster than the VTM reference software. Thus, such an encoder is more likely to be used at a large scale than the reference software when it comes to video content distribution.

VVC Common Testing Condition (CTCs) [13] uses fixed Quantized Parameters (QP), to assess compression efficiency of a specific coder tool. However, when it comes to decoder performance assessment use cases, it is more common to target a specific bit-rate range where a rate control algorithm selects a Quantization Parameter (QP) at a thinner grain. The aim is to achieve the best compromise for the Rate Distortion Optimization (RDO) process targeting a specific bit rate. This is why four arbitrary rate control targets were set to 1, 2, 4 and 8 Mbps. As allowed by the VVC standard, Coding Tree Unit (CTU) of sizes equal to 32, 64 or 128 have been considered. The list of selected sequences as well as a summary of the coding parameters used in the dataset composition can be found in table III-B.

B. Decoded Video Data Information

To provide machine learning algorithms with some data to derive decoding time estimators, the OpenVVC decoder [9] has been modified to extract the data needed for the regression process. For each picture, the decoding times (in µs) were computed from the difference between timestamps taken at the start and the end of the PU decoding. Note that the PU decoding start and end times may differ from the PU entrance and the reconstructed picture output times because of the delay implied by Random Access pictures reordering. Indeed, in this configuration the reordering delay may cause the decoding time of a picture to also include the decoding time of all leading pictures which entered the decoder after this picture in display order.

Alongside picture decoding time, information related to the picture decoding is extracted by parsing the high-level syntax elements. Extracted information contains: 1) *pu_size* : the sum of the number of bytes read in the Picture Unit (PU) NALUs; 2) *alf_flags* : gathering together the activation status of the ALF filter on the luma and chroma planes; 3) picture resolution (in our set the sequences are consistent but this may change in case VVC Reference Picture Rescaling (RPR) tool is activated or if streams of different resolution are concatenated such as in Dynamic Adaptive Streaming over HTTP using a bitrate ladder); 4) the Slice Quantization Parameter (QP) selected by rate control; 5) the NAL Unit Type to get information on refresh picture (IDR CRA) 6) the Temporal Layer Id of each PU

The picture decoding times were measured on an Intel Core i7-7700 running @ 3,6 GHz. To avoid interference on time measurements the decoder was configured to use only one thread and TurboBoost (which is a DVFS tool) was disabled from the OS.

Next section discusses this data and last section compares estimators efficiency in terms of estimation accuracy and inference time.

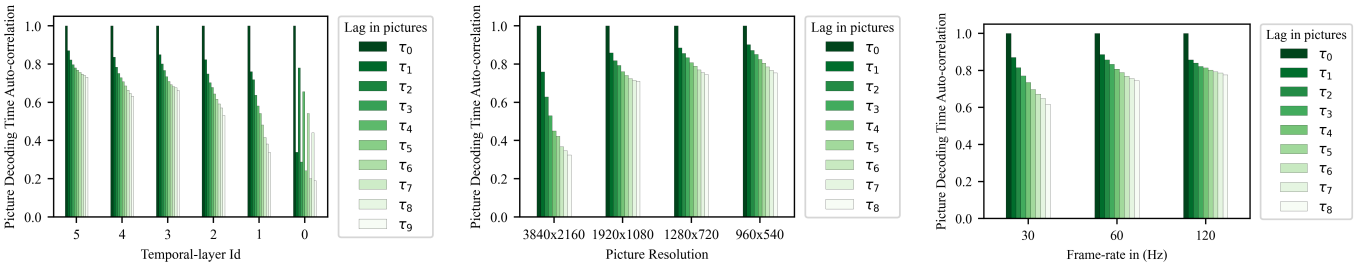
IV. Decoding statistics

A. Picture Unit Size and Picture decoding times

In this section, the relation between the decoding time and the Picture Unit size (pu_size) is analyzed. The pu_size corresponds to the number of bits occupied by the encoded data of the picture unit. In figure 2, the decoding time of a picture is represented according to the pu_size of the associated encoded picture for different resolutions. For low resolutions (HD 720p, SD 544p), there is a clear relation between the decoding time and the pu_size. For higher resolutions with HD 1080 & UHD 4K, a relation between the decoding time and the pu_size remains visible for high values of pu_size. For low pu_size value, the decoding time and the pu_size are weakly correlated. This phenomenon is particularly pronounced for UHD 4K resolution. In conclusion, the correlation between the decoding time and the pu_size decreases when the resolution increases and the pu_size decreases

B. Temporal characteristics

In this section, the temporal characteristics of decoding time are observed through the decoding time auto-correlation. A key element to analyze the temporal characteristics of decoding time is the Group of Picture (GoP) structure and the hierarchy of picture for the inter prediction process. Each picture is associated with a Temporal-layer Identifier (TID)



(a) per *temporal_layer_id* (b) per resolution (c) per frame-rate

Fig. 3: Evolution of the picture decoding time auto-correlation for 10 lag values according to different temporal layer id (a), resolutions (b) and frame-rates (c)

which defines the temporal layer of the considered picture in a video coding hierarchy. The pictures belonging to a temporal layer x only use as reference for inter prediction pictures belonging to temporal layers lower than x. In the considered experiments, the GoP uses an Intra-Picture period of 32, 64 and 128 pictures according to the sequences frame-rates.

Figure 3.a compares the auto-correlation of the decoding times by temporal layer for videos with a fixed resolution of 1280x720 and a fixed frame-rate of 60 fps. Results show that the first temporal layer is a specific case compared to other temporal layer for which the temporal correlation is low. Indeed, pictures from temporal layer 0 correspond to I-Pictures (using intra prediction only) and these pictures are the furthest away. From temporal layer 1 to temporal layer 5, pictures rely more on inter-prediction. Their correlation is relatively higher and decreases when the lag τ increases. The higher the temporal layer, the closer in time are the reference images used for prediction. This increase in correlation according to the layer shows that the decoding times of picture are similar when the pictures are close in display order.

Figure 3.b compares the evolution of the auto-correlation of the decoding time by their resolution for a fixed frame-rate of 60 fps and from a fixed temporal layer 3. The results show that the correlation decreases when the resolution increases and especially for 4K UHD resolution for which the correlation levels are significantly lower.

Figure 3.c compares the auto-correlation of the decoding time by their frame-rate for a fixed resolution of 1280x720p and a fixed temporal layer 3. The results show that the correlation decreases when the frame-rate diminishes. Indeed, when the frame-rate is lower, the pictures are spaced further apart in time and thus, the correlation is weaker.

V. Models for decoding time estimation

In our approach, three supervised machine learning algorithms were considered to estimate the decoding time : ExtRa-Trees, Support Vector Regressor and XGBoost Regressor. ExtRa-Trees (Extremely Randomized Trees) [14] and XGBoost Regressor [15] both belong to ensemble supervised machine learning methods that uses decision trees. With the ExtRa-Trees algorithm, many decision trees are created and the sampling for each tree is random, without replacement. With XGBoost Regressor from the gradient boosting frameworks, the model is based on an ensemble of weak estimation models which are considered for building a stronger global model. Support vector regression (SVR) [16] is a regression analysis exploiting Support Vector Machines (SVM) to fit a model to data. SVR searches the hyperplane in a high-dimensional feature space that maximally separates the data points by minimizing its distance to the data points.

The model selection is driven by two aspects: the estimation model accuracy and the model inference time. The model accuracy is evaluated through the Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE). With regards to a decoding frame-rate of 120 Hz a picture decoding must be done within a time of less than 8.33ms and 16.66ms considering 60 FPS sequences. The model inference must consume significantly less resource compared to the decoding process. Thus, the inference time have to be limited to a small fraction of the actual decoding time. The three supervised machine learning algorithms considered in this paper have been selected based on their ability to provide an optimized C-source code library able to limit the model inference time.

The number of features considered by the machine learning process has been adapted to explore the trade-off between the inference time and the estimation quality. To take into account the statistical characteristics and especially the correlation between pictures having the same temporal layer, the models were provided with the decoding times of the 7 previously decoded pictures. In addition to the temporal informations, 3 to 8 features, extracted from the HLS associated with the picture, have been tested. For a considered number of features n, an exhaustive search is performed to select the n best features leading to the minimal MAPE.

To evaluate estimator accuracy we used both MAE an MAPE distortion metrics. Selecting either MAE or MAPE depends on the precision for the DVFS. While MAPE gives information regarding the estimator accuracy when working with independent pictures, MAE stays useful when it comes to keep the actual decoding time target in check. Indeed when working with pictures of different decoding time magnitudes (this is especially true for picture of temporal layer 0 against higher temporal layers) what can be regarded as a small error

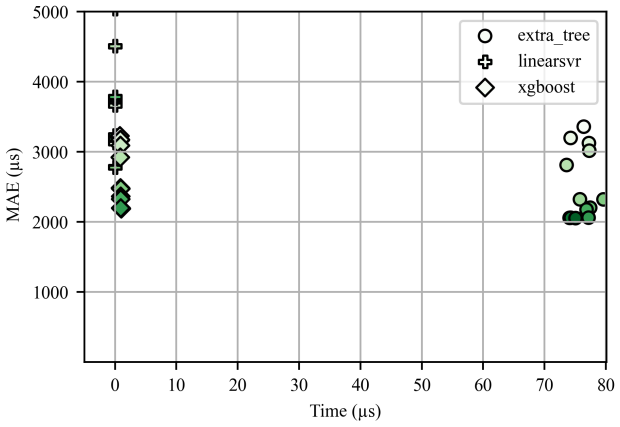


Fig. 4: Estimation Mean Absolute Error (MAE) according to the inference time

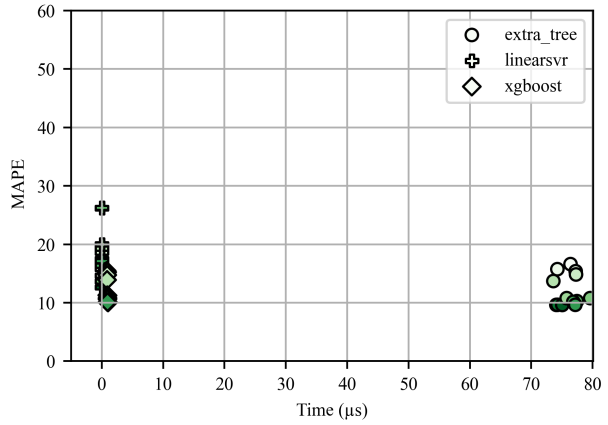


Fig. 5: Estimation Mean Absolute Percentage Error (MAPE) according to the inference time

percentage on a picture with a large duration can dominate the estimation error measured on a picture with a smaller duration even if the error percentage on this latter was significant.

All models were trained using 1% of our dataset as a training set. The complement of the training set was used as a validation set. The C objects were obtained by exporting the various models via the *m2Cgen* module of scikit-learn and built with gcc before inclusion to a C program. The whole validation set was then used to measure the estimation errors on each points while also measuring the actual inference times. Figure 4 and 5 display respectively the Mean Absolute Percentage Error and the Mean Absolute Error of the tested estimators in function of the inference time. ExtRa-Trees based methods perform well when it comes to the actual estimation accuracy, ranging from 9.60 to 16.59% in MAPE however they are less performant when it comes to inference times with regards to XGBoost and SVR performances. The SVRs outperform all other methods when it comes to inference speed with an average inference time of 0.03 µs. However it performs the worst when it comes to both MAPE and MAE results. The XGBoost Regressors seem to work well on both distortion metrics while always maintaining average inference times lower than 1 µs. Given these results among tested methods XGboost Regressor seems to satisfy our requirements for a practical VVC Video Decoding DVFS setup.

VI. Conclusion and future Work

This work paves the way to an actual VVC DVFS setup in a video player. After studying the picture decoding time statistics in a typical video distribution configuration. estimators were tested with particular attention given to the practical inference time needed. After multiple estimators comparison it appears XGBoost Regressor seems to match the needs for a practical DVFS setup in a real time decoder such as OpenVVC achieving an MAPE accuracy of 10.04%, while maintaining an inference time of 1.03 µs. There is still work to be done considering the challenge of decoding time estimation in a threaded environment when decoding multiple pictures in parallel.

References

[1] Cisco Annual Internet Report (2018–2023) White Paper. (2022, January 23). Cisco.

[2] W. Hamidouche et al., "Versatile Video Coding Standard: A Review From Coding Tools to Consumers Deployment," in IEEE Consumer Electronics Magazine, vol. 11, no. 5, pp. 10-24, 1 Sept. 2022, doi: 10.1109/MCE.2022.3144545.

[3] Sullivan, G., Ohm, J., Han, W. M., and Wiegand, T. Overview of the High Efficiency Video Coding (HEVC) Standard. IEEE Transactions on Circuits and Systems for Video Technology, 22(12), 1649–1668, Sept. 2012

[4] Bossen, F., Li, X., and Suehring, K., "JVET AHG report: Test model software development (AHG3)," JVET Document, P0003, 2019.10

[5] Kränzler, M., Herglots, C., and Kaup, A. Decoding energy modeling for versatile video coding. 2020 IEEE International Conference on Image Processing (ICIP).

[6] Herglotz, C., Heindel, A., and Kaup, A. Decoding-Energy-Rate-Distortion Optimization for Video Coding. IEEE Transactions on Circuits and Systems for Video Technology, 29(1), 171–182, Jan 2019

[7] C. Herglotz, et al. AHG9: Green Metadata SEI message for VVC, document JVET-W0071, Jul. 2021

[8] H. Boujida, P. -L. Cabarat and D. Menard, Decoding Time Prediction for Versatile Video Coding, 2023 IEEE 25th International Workshop on Multimedia Signal Processing (MMSP), Poitiers, France, 2023, pp. 1-6

[9] Amestoy, T., Cabarat, P., Gautier, G., Hamidouche, W., and Ménard, D. (2022, May 24). OpenVVC: a Lightweight Software Decoder for the Versatile Video Coding Standard. arXiv.org. https://arxiv.org/abs/2205.12217

[10] Ultra Video Group (UVG) dataset, https://ultravideo.fi/dataset.html

[11] A. Mackin, F. Zhang and D. R. Bull. A study of subjective video quality at various frame rates, In Proceedings of the IEEE International Conference on Image Processing (ICIP), Quebec City, QC, Canada, 2015, pp. 3407-3411,

[12] A. Wieckowski et al. VVenC: An Open And Optimized VVC Encoder Implementation. In Proceedings of the IEEE International Conference on Multimedia and Expo Workshops (ICMEW), Shenzhen, China, IEEE, July 2021, pp. 1-2,

[13] J. Boyce, K. Suehring, X. Li and V. Seregin. JVET-J1010: JVET common test conditions and software reference configurations. Apr. 2018

[14] P. Geurts, D. Ernst and L. Wehenkel. Extremely randomized trees. Machine learning, 63, (2006) 3-42.

[15] T. Chen et al. Xgboost: extreme gradient boosting. R package, (2015): 1-4.

[16] H. Drucker, C. J. Burges, L. Kaufman, A. Smola and V. Vapnik. Support vector regression machines. Advances in neural information processing systems, 9, 1996