A Machine Learning Method for Optimizing Partition of Prediction Block in Coding Unit in H.265/HEVC

Wenchan Jiang[1], Ming Yang[2], Ying Xie[2], and Zhigang Li[2]
wenchan.jiang@americold.com, ming.yang@kennesaw.edu,
ying.xie@kennesaw.edu, zhigang.li@kennesaw.edu

[1]Americold Logistics, Atlanta, GA, USA

[2]College of Computing and Software Engineering
Kennesaw State University, Marietta, GA 30060, USA

Abstract: In the latest generation of video coding standard – H.265/HEVC, the partition of Coding Tree Unit (CTU) into CU (Coding Unit) is a very critical yet time-consuming component. Traditional methods find the optimum partition mode for each CTU through iterative and exhaustive search, which is a very time-consuming process and hinders its application to real-time video streaming scenarios. In this research, we explored and implemented a machine learning based method to avoid the exhaustive search and to improve the performance of encode/decode by optimizing the partition of prediction block in the coding unit. Our results in coding unit split pattern prediction show a significant performance improvement in terms of processing time.

Keywords: Machine Learning, CNN, CTU, Partitioning, H.265/HEVC, Coding Unit.

1 Introduction

Following the previous video coding standard H.264/AVC, High Efficiency Video Coding (H.265/HEVC) has been deemed as the newest video coding standard of the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group [1]. HEVC has the potential to deliver better performance than earlier standards such as H.264/AVC. H.265/HEVC has introduced a refined content-adaptive approach for Coding Block Partitioning (corresponding to fix-sized macroblock in previous standards), which has significantly improved the compression efficiency. However, the implementation in H.265/HEVC reference software HM explore all the possibilities in a traversal and exhaustive manner to find the best partition and merge pattern for a specific prediction unit. It is a time-consuming process and will be difficult, if not impossible, to stream ultra HD video contents in real time using the new HEVC standard.

Over the past decade, machine learning has become one of the top trending information technologies deeply integrated into our life. With the ever-increasing amounts of data becoming available, it's reasonable to believe that smart data analysis will become even more pervasive as a necessary ingredient for technological progress. [2]. In this paper, we utilized the machine learning technique in Keras framework to speed up the process of encoding by improving the partition speed of prediction block in coding block. This approach avoids the

EAI MOBIMEDIA 2020, August 27-28, Harbin, People's Republic of China

DOI 10.4108/eai.27-8-2020.2297985

exhaustive searching process and effectively speeds up the coding process to enable its application in real-time video streaming scenarios.

2 Background

2.1 High Efficiency Video Coding (H.265)

The new standard of HEVC, or H.265, promises a 50% storage reduction as its algorithm uses efficient coding by encoding video at the lowest possible bit rate while maintaining a high image quality level. H.265 still uses the widely accepted hybrid coding framework since H.264, including intra-frame prediction, inter-frame prediction based on motion compensation, transformation, entropy coding, and quantization. Table 1 summarizes the improvements of H.265/HEVC has made over H.264/AVC [3].

Table 1. Main Differences between H.264 and H.265

Category	H.264	H.265
Partition Size	Macroblock 16 × 16	Coding Unit 8 × 8 to 64 × 64
Partitioning	Sub-block down to 4 × 4	Prediction Unit Quadtree down to 4 × 4 square,
Intra Prediction	Up to 9 predictions	35 predictions
Transform	Integer DCT 8×, 4×4	Transform Unit square IDCT from 32×32 to 4×4 + DST Luma intra 4×4
Filters	Deblocking filter	Deblocking filter, Sample Adaptive Offset
Motion Prediction	Spatial Median (3 blocks)	Advanced Motion Neighbor Vector Prediction (AMVP) for both spatial and temporal
Entropy Coding	CABAC, CAVLC	CABAC

2.2 CU Partitioning in HM

Coding Tree Units **(**CTU) and Coding Unit (CU) are the most important concepts in our research. The CTU is the basic processing unit similar to MacroBlock (MB) in previous generation of coding standards such as H.264/AVC. The size N of the CTUs is chosen by the encoder, ranging from 16 × 16, 32 × 32, to 64 × 64, with 64x64 usually being the default. CTU may be too big to decide whether we should perform inter-picture prediction or intra-picture prediction. Thus, each CTU can be differently split into multiple CUs and each CU becomes the decision-making point of prediction type. The size of the CU can range from the same size as the CTU to a minimum size of (8 × 8). CTU corresponds to MB in previous standards, which has a fixed size of 16x16. In the past decade, we have much higher frame sizes to deal with since 4K production became practical. Therefore, larger macroblocks are

needed to efficiently implement motion estimation and motion compensation for these frame sizes. On the other hand, blocks at the granularity of 4×4 are also essential to process prediction and transformation of small details.

As mentioned above, the CTU is further partitioned into multiple CUs to adapt to various local characteristics. Rate-Distortion analysis is utilized to determine the optimum partition. Such recursive procedure will be repeated for all the CUs in a CTU, as well as for all CTUs in one fame, and finally the optimal split mode for each CU within one video sequence will be obtained. This processing order of CUs can be interpreted as a depth-first traversing in a Zig-Zag order in the coding tree structure as shown in Figure 1 below. Figure 2 also illustrates a broader view of how CTU partitioning works within a frame.

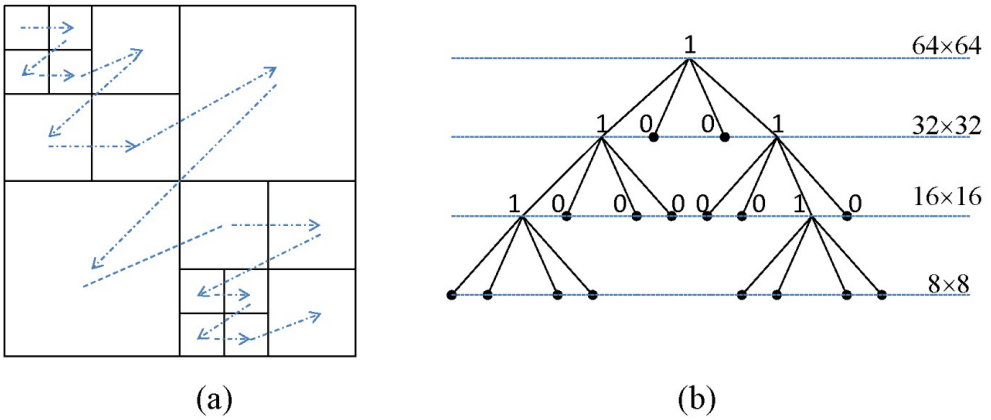

Fig. 1. Example of CTU partitioning and processing order when size of CTU is equal to 64×64 and minimum CU size is equal to 8×8. (a) CTU partitioning. (b) Corresponding coding tree structure [4]

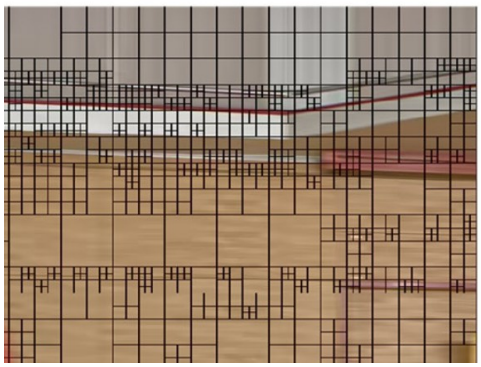

Fig. 2. Illustration of CTU splits into CUs [5]

The split pattern of a specific CTU in the reference software HM is exhaustive and recursive. It means HM will calculate the rate distortion cost of every possible split pattern (i.e., 17 combinations in total), and further compare these cost values recursively to obtain the best split pattern based on the minimum rate distortion cost. Understandably, it is a very time-consuming process, and it will be unrealistic to compress and transfer videos of ultra-high definition integrating H.265 standard in a real-time streaming manner. There will be a lot of overhead upfront that is unbearable and unfortunate to apply HEVC new standard in a broader spectrum.

3 Literature Review

Different approaches have been proposed to speed up the coding process of H.265/HEVC. The common goal is to avoid the exhaustive search process in coding decision making and speed up the encoding process. Momcilovic et al proposed a novel fast Coding Tree Unit partitioning for H.265 encoder [6]. The proposed approach decouples the requirement of any pre-training and yields a high adaptivity to the dynamic changes in video contents. The proposed methodology has reduced the encoding time for up to 65% with negligible rate-distortion penalties. In another study the authors proposed a machine learning based method using features that describe CU statistics and sub-CU homogeneity [7]. Comparatively, the experiment results can achieve 36.8% complexity reduction on average with only 3.0% bit-rate increase. In Alam's work [8], a fast Convolutional-Neural-Network (CNN) based quantization strategy for HEVC was proposed. They utilized the contrast gain control model to develop a structural facilitation model to capture effects of recognizable structures on distortion visibility. Liu et al. proposed a fast algorithm based on convolution neural network to decrease no less than two CU partition modes in each CTU for full rate-distortion optimization (RDO) processing, therefore reducing the encoder's hardware complexity [9] [10]. The proposed algorithm can save 63% Intra encoding time at the cost of average 2.66% BDBR increase. A fast coding unit (CU) depth decision algorithm for intra coding of HEVC using an artificial neural network (ANN) and a support vector machine (SVM) was proposed by Chen et al [11]. In their methodology, machine learning provided a systematic approach for developing a fast algorithm for early CU splitting or termination to reduce intra coding computational complexity.

4 Proposed Approach and Experimental Settings

The procedures for splitting CTUs into CUs in the reference software HM for HEVC is very ineffective and insufficient, thus making it unsuited for real-world streaming service. In this study, we implemented CNN in place of the split procedure of CTUs in the reference software.

4.1 Training Data Set Generation

The training and testing data consist of 13 video clips. For each video, 80% of frames are used as training dataset and the rest 20% as testing dataset. Each frame is divided into 32 × 32 sized CTUs. For a typical frame in our experiment, it is divided into 99 CTUs.

The luma samples for each CTU will be used as the input of the CNN, and the output of CNN will be the split pattern/intra prediction mode. For a 32 × 32 CTU with minimum 4 × 4 CU, there are altogether 17 possible split patterns. The CNN network will be trained to generate a compiled model that will be used to predict the labels for new inputs, and exhaustive search for the optimum split patterns will be avoided. As long as comparable levels of accuracy can be achieved, the CNN will be deemed as fully trained.

One of the most powerful and easy-to-use Python libraries for developing and evaluating deep learning models is Keras. It wraps the efficient numerical computation libraries Theano and TensorFlow. The advantage of this is mainly that you can get started with neural networks in an easy and fun way. In our experiment, we use Python as the language and Theano as the backend.

4.2 Convolutional Neural Network Design

In deep learning, selection of optimal number of layers and neurons is also one of the hyperparameters that can be fine-tuned. A method is to add layers until it starts to overfit the training set. Then it is time to add dropout or another regularization method. The idea is that once your network overfits you're sure that it is powerful enough for your task. The dropout helps to prevent feature co-adaptation and therefore avoid over-fitting. The number of neurons in each layer is not really sensible. Usually a bit more or as much neurons should be put on the first layer than inputs, and the number should decrease slowly as we approach the output layer.

In our research, we tried different combinations of parameters and keep the one with the lowest loss value or better accuracy on the validation set. We tried two possibilities of 2 and 3 fully connected layers. For 2 layers, it has 512 and 17 hidden units for the fully connected layers. Whereas for 3 layers, it has 512, 128, and 17 hidden units for each layer.

For the dataset that we trained, the accuracy is slightly higher for all video clips except one when we use 2 fully connected layers versus 3 layers. Another benefit of using less fully connected layers is less amount of execution time. In our case, average execution time for 3-layer is 7 seconds per epoch, versus 5 seconds per epoch for 2-layer network, which results in 28% decrease of time when we choose 2-layer. Therefore, we will use 2 fully connected layers across our experiments.

After fine tuning the parameters, we have decided on the CNN's structure shown in Figure 4 below. There are 2 main layers in the network with 1 convolution layer and 1 max pooling layer in each main layer. Each convolution layer consists of 32 filters in the size of 3 × 3 to extract the feature map. The activation function is ReLU across the boarder. The border mode is set as "valid", which means there is no padding around input or feature map. For max pooling layer, the pool size is 2 × 2 with strides of 2 × 2. After two main layers, it comes with the fully connected layers where the hidden units will be "flattened" and directed to the output of 17 labels.

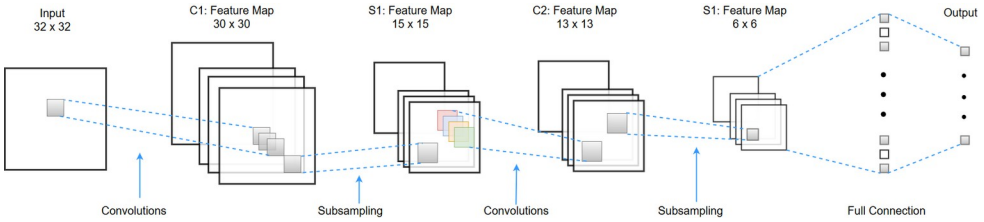

Fig. 1. CNN Structure

5 Results and Discussion

5.1 Max Pooling

The purpose of max pooling is to down-sample an input representation to reduce its dimensionality and allow for assumptions to be made about features contained in the sub-regions binned. Max pooling is employed in our experiments for split pattern prediction after each layer by using a 2 × 2 filter. Although the network can achieve similar accuracy either with or without employing max pooling layer, with max pooling layer in place, it only needs about half of execution time. Similar trends are observed in other clips, and we take one of the clips as an example and summarize the observations in Table 2.

Table 2. Comparison of execution time for max pooling

	with max pooling	w/o max pooling
time of each epoch (s)	10	23
accuracy of 30 epochs (%)	70.8	71.4

5.2 Dropout

In our experiment, dropout is applied to reduce the complexity of the model and prevent overfitting. Also, training will be faster with dropout set. We tuned dropout ratios in our experiments. Three different dropout ratios were tested: 0.5, 0.25, and 0.1. Dropout ratio of 0.5 generates the lowest accuracy, while dropout ratio 0.1 generates the highest accuracy. Small dropout ratio obviously has higher level of computational overhead, compared to the cases of greater dropout ratio. However, in our experiments, such kind of cost is marginal compared to the extra accuracy achieved. Thus, we set dropout ratio as 0.1 by taking all the factors into account.

It should also be noted that dropout is only applied during training, and we need to rescale the remaining neuron activations. Specifically, if 50% of the activations in a given layer is set to zero, we need to scale up the remaining ones by a factor of 2.

5.3 Training and Testing Results

For a typical sample video clip with 300 frames and each frame with size of 352 × 288 pixel, there will be 11 × 9 × 300 = 29700 CTUs generated by reference software HM and available to use. Specifically, the input will be the matrix of pixel value of 32 × 32 CTU as demonstrated in figure below. During the process of building the model, such big data set will be split into 80/20 portion of training/test data set. Therefore, the test data set can be used to validate the training results.

During the training process, the input data is fed into the Keras model with the parameters configured as mentioned in the previous section. For some sample video clips, 50 iterations, which takes roughly 25 minutes with the test machine can yield ~ 90% training accuracy. In the experiment, the video clips of bus, mobile, tempete, and flower can quickly achieve high training accuracy as summarized below (Figure 6). The reason is because these video clips have relatively smooth scenes, without many variations in terms of pixel changes and ranges. So, the model can quickly learn the relationship between the input pattern and split pattern with relatively high accuracy.

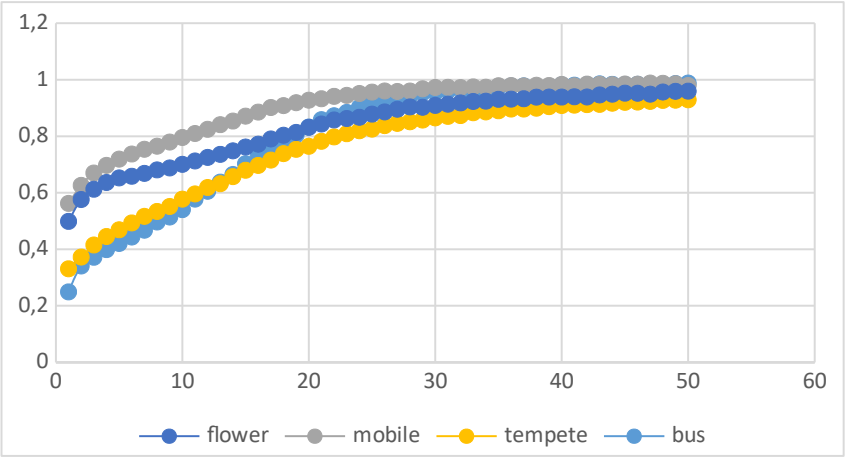

Fig. 2. Training accuracy of ~ 90% in 50 cycles

For the other video clips, it generally requires about 100, sometimes even 200 iterations to achieve ~ 90% training accuracy (Figure 7). These video clips generally have more variations and more complex scenes. Thus, it takes longer time for the CNN to adapt to these changes and train itself to achieve a high level of prediction accuracy.

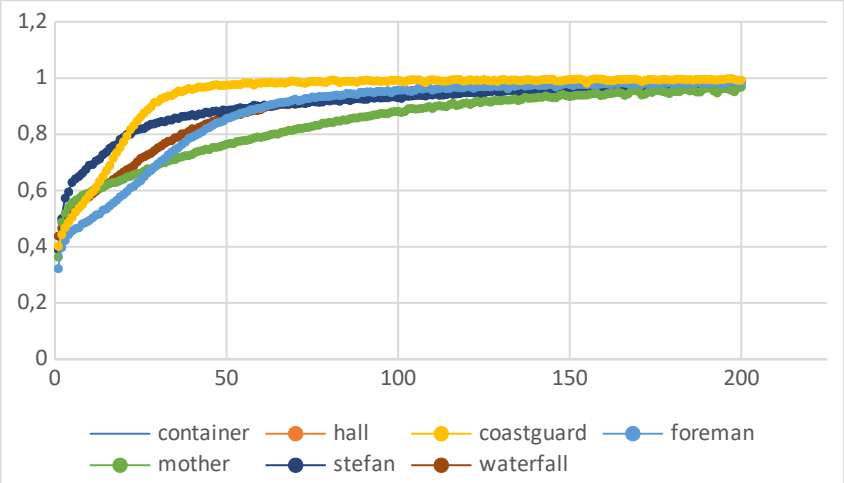

Fig. 3. Training accuracy of above 90% in 200 cycles

After the network is trained, we start to use the CNN to predict split patterns for the testing data set. Most of the video sample clips achieved reasonably high accuracy of around 90% (Figure 8). Another observation is that more training data set in each video sequence will in general lead to higher accuracy. The reason is because more frames will help the network to adapt to the variation in the video sequence.

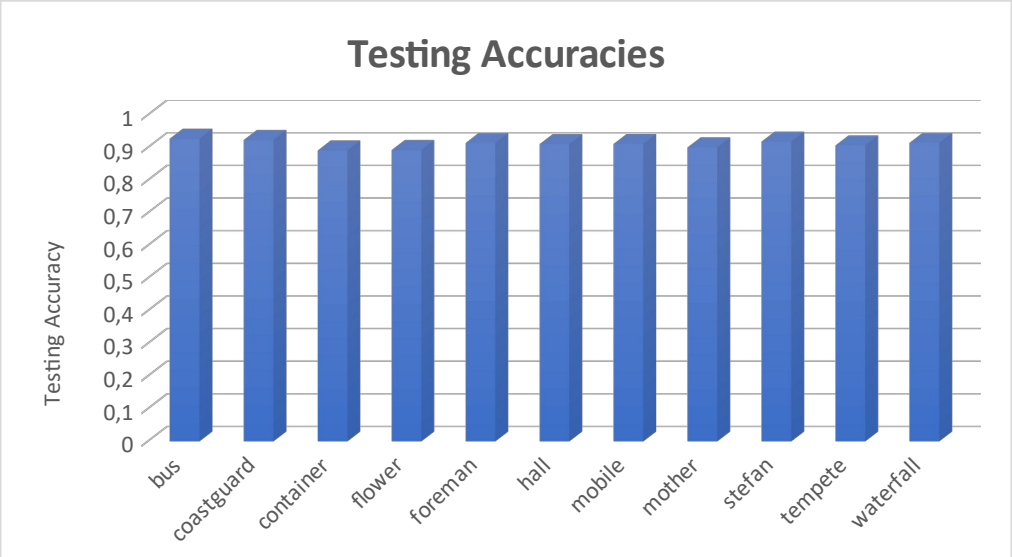

Fig. 4. Comparison of testing accuracies for different sample video

In addition to the good training accuracy for split pattern prediction, the trained network also significantly improved the speed of splitting the CTUs. The results from our experiment indicated that, the trained model is able to predict the split pattern for a specific CTU with over 90% accuracy within 400 microseconds, which is a significant improvement from 4000 microseconds per CTU in HM.

6 Conclusion

As discussed earlier in this paper, the CU split pattern prediction method in the HEVC reference software is ineffective and not well-suited for real-world use cases. In this study, we took advantage of the convolutional neural network to improve the prediction speed of the CU split pattern. Our results demonstrated a prediction accuracy over 90% and a significant 90% improvement in prediction speed. In the future, we are planning to apply CNN based methodology to replace other computationally intensive modules in H.265/HEVC encoder and speed up the coding process of Ultra-HD video.

7 References

[1] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. 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.

[2] A. Smola and S. Vishwanathan, "Introduction," in *Introduction to Machine Learning*, A. Smola and S. Vishwanathan, Eds. Cambridge University Press, 2008.

[3] Narang, Nitin, "What is the difference between HEVC (H.265) and H.264 (MPEG-4 AVC)," *M&E Industry Trends, Technology and Research*, 01-Oct-2013. [Online]. Available: https://www.mediaentertainmentinfo.com/2013/10/4-concept-series-what-is-the-difference-between-hevc-h-265-and-h-264-mpeg-4-avc.html/. [Accessed: 23-Jan-2019].

[4] I.-K. Kim, J. Min, T. Lee, W.-J. Han, and J. Park, "Block Partitioning Structure in the HEVC Standard," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, pp. 1697–1706, 2012.

[5] M. Zhang, X. Zhai, and Z. Liu, "Fast and adaptive mode decision and CU partition early termination algorithm for intra-prediction in HEVC," *EURASIP Journal on Image and Video Processing*, pp. 1–11, 2017.

[6] S. Momcilovic, N. Roma, L. Sousa, and I. Milentijevic, "Run-Time Machine Learning for HEVC/H.265 Fast Partitioning Decision," in *2015 IEEE International Symposium on Multimedia*, 2015.

[7] F. Duanmu, Z. Ma, and Y. Wang, "Fast CU partition decision using machine learning for screen content compression," in *2015 IEEE International Conference on Image Processing (ICIP)*, 2015, pp. 4972–4976, doi: 10.1109/ICIP.2015.7351753.

[8] M. M. Alam, T. Nguyen, M. Hagan, and D. Chandler, "A perceptual quantization strategy for HEVC based on a convolutional neural network trained on natural images," 2015, p. 959918, doi: 10.1117/12.2188913.

[9] Z. Liu, X. Yu, S. Chen, and D. Wang, "CNN oriented fast HEVC intra CU mode decision," in *2016 IEEE International Symposium on Circuits and Systems*, 2016.

[10] Z. Liu, X. Yu, Y. Gao, S. Chen, X. Ji, and D. Wang, “CU Partition Mode Decision for HEVC Hardwired Intra Encoder Using Convolution Neural Network,” *IEEE Transactions on Image Processing*, 2016.
[11] Z.-Y. Chen, J.-T. Fang, Y.-C. Liu, and P.-C. Chang, “Machine Learning-based Fast Intra Coding Unit Depth Decision for High Efficiency Video Coding,” *Journal of Information Science and Engineering*, 2016.
[12] R. Song, D. Liu, H. Li, and F. Wu, “Neural network-based arithmetic coding of intra prediction modes in HEVC,” *2017 IEEE Visual Communications and Image Processing (VCIP)*, Dec. 2017, doi: 10.1109/vcip.2017.8305104.
[13] Y. Li *et al.*, “Convolutional Neural Network-Based Block Up-sampling for Intra Frame Coding,” *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, pp. 1–14, 2017.
[14] A. Abramowski, “A survey over possible intra prediction optimizations in the H.265/HEVC encoder,” in *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments*, Wilga, Poland, 2016.