Deep learning-based intra prediction mode decision for HEVC

Thorsten Laude and Jörn Ostermann
Institut für Informationsverarbeitung, Leibniz Universität Hannover
Appelstraße 9a, 30167 Hannover, Germany
Email: {laude, office}@tnt.uni-hannover.de

***Abstract*—The High Efficiency Video Coding standard and its screen content coding extension provide superior coding efficiency compared to predecessor standards. However, this coding efficiency is achieved at the expense of very complex encoders. One major complexity driver is the comprehensive rate distortion (RD) optimization. In this paper, we present a deep learning-based encoder control which replaces the conventional RD optimization for the intra prediction mode with deep convolutional neural network (CNN) classifiers. Thereby, we save the RD optimization complexity. Our classifiers operate independently of any encoder decisions and reconstructed sample values. Thus, no additional systematic latency is introduced. Furthermore, the loss in coding efficiency is negligible with an average value of 0.52% over HM-16.6+SCM-5.2.**

I. Introduction

During the last decades a tremendous improvement of video coding algorithms was observed. In January 2013, the Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG finished the technical work for the latest video coding standard, High Efficiency Video Coding (HEVC) [1]. It achieves the same visual quality at half the bit rate compared to the predecessor standard Advanced Video Coding (AVC) [2], [3]. After finalizing HEVC version 1, the JCT-VC continued with the standardization of several extensions addressing specific application scenarios.

Among these extensions is the HEVC screen content coding extension, which is often referred to as HEVC SCC [4]. This extension was finalized in February 2016 and brings new coding tools addressing several key characteristics of screen content (small number of different colors, recurrent patterns, RGB source material, no noise, etc.).

The superior coding efficiency of HEVC and its extensions is achieved at the expense of very complex encoders. Bossen et al. analyzed in [5] that HEVC encoders are several times more complex than AVC encoders. One main complexity driver for HEVC encoders is the comprehensive rate-distortion (RD) optimization which is indispensable to fully exploit all benefits of the HEVC standard. A major disadvantage of the RD optimization is that encoders which cannot afford a comprehensive RD optimization will likely not accomplish the optimal coding efficiency. The RD optimization consists in the evaluation of all combination possibilities (coding modes, parameter for these coding modes, partitioning, etc.) and the selection of the combination with the smallest RD costs. In case of the intra prediction, the RD optimization determines the intra prediction mode. Specifically, for HEVC, there are 33 angular intra prediction modes, the DC mode and the planar mode [6].

Therefore, to overcome the described disadvantage, we aim at avoiding the RD optimization complexity for the intra prediction mode decision. Taking into account that the intra prediction mode decision can be formulated as a classification problem with the different intra prediction modes forming the classes, machine learning approaches suggest themselves as solution. Deep learning is a very active topic in the machine learning community [7]. It is evident that deep learning approaches provide superior results for classification problems by utilizing deep convolutional neural networks (CNNs) [8].

For this reason, we use CNNs for the intra prediction mode decision. Additionally, we limit ourselves to original input sample values of the encoded video. This way, the decisions of the CNNs are computable independently from any previous encoder decisions and reconstructed sample values. By decoupling the encoding decision from the actual encoding process all decisions could be carried out in parallel for all blocks. Hence, no additional systematic latency is introduced. Furthermore, CNNs are very suitable for screen content coding systems such as servers.

Our contribution in this paper is a novel deep learning-based video encoder control. It is based on the deployment of CNN classifiers as part of the encoding process. By means of this encoding process, we avoid the RD optimization complexity.

The remainder of this paper is organized as follows: In Section II we analyze related work and discuss the differences of our approach. Our novel deep learning-based encoder is presented in Section III. Section IV describes the evaluation of our method and Section V concludes the paper.

II. Related work

Numerous methods for fast video encoders are known from the literature. In this section we discuss the state-of-the-art and elaborate on the distinguishing features of our contribution. Since we propose a machine learning-based fast intra prediction mode decision algorithm, we discuss related work in two categories: machine learning-based fast encoding algorithms and fast intra prediction mode decision algorithms.

Recently, approaches to speed up encoders by employing machine learning algorithms have emerged. The commonality of the related works in this first category is the encoder

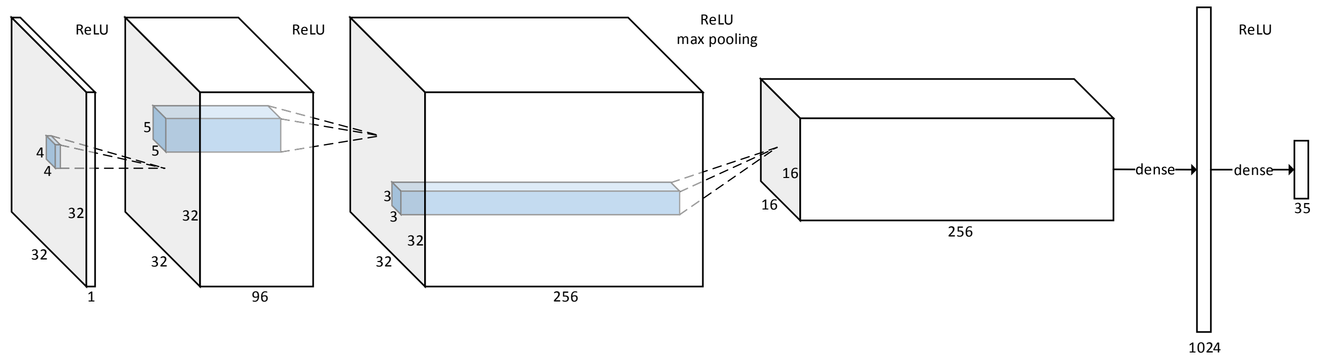


Fig. 1: CNN model for the classification of 32×32 blocks. Each block is fed through two convolutional, one max pooling, and two fully-connected layers. The capacity of the model is chosen such that a good trade-off between a low training error and a good generalization is achieved. In the final layer a classification into 35 classes (i.e. intra prediction modes) is carried out. Rectified linear units (ReLU) are not plotted for the sake of easy readability.

acceleration by using fast partitioning algorithms. Ruiz-Coll et al. [9] design a decision tree based on manually defined features and use data mining classifiers for a fast partitioning. In [10], Duanmu et al. speed up the partitioning for screen content coding by manually deriving features and by applying various machine learning technologies. In addition to logistic regression, linear perceptron classifiers, and support vector machines, they also use CNNs as in our approach. Closest to our method is a work by Yu et al. [11] in which the authors present a fast partitioning algorithm that utilizes CNNs. As main difference to these works, we apply machine learning methods for the intra prediction mode selection rather than for the partitioning. Another differentiation of our method to some of the related works is that we do not rely on manually designed features. Instead, our features are determined by the learning process of our CNNs. Thereby, complex properties of the video signal which are hardly captured by manually designed features can be taken into consideration. Additionally, we do only rely on original input sample values instead of previous encoder decisions. Thus, our learning models can be applied in parallel to the video encoding process without introducing additional systematic latency.

The works in the second category have in common that they accelerate the encoding process by pruning the list of intra prediction modes which are evaluated during the RD optimization. For instance, Gao et al. [12] take advantage of the correlation between the intra prediction mode of the current block and the intra prediction modes of adjacent blocks. In another work, Jamali et al. [13] use edge detection to find dominant edges in the signal and restrict the tested intra prediction modes to directions near to these dominant edges. A different approach is chosen by Lee et al. [14]. In their work, the authors apply local binary patterns to reduce the number of tested intra prediction modes. One recent work by Shang et al. [15] addresses both, fast partitioning and fast intra prediction mode decision. As one aspect, they use the depth information of neighboring blocks to assist the partitioning decision of the current block. As another aspect, they utilize intra prediction modes from higher layer (i.e. bigger) blocks to speed up the intra prediction mode decision for corresponding smaller blocks. In contrast to these methods, we deploy machine learning technologies to prune the list of tested intra prediction modes rather than using manually determined criteria. Furthermore, our method does not require previous encoder decisions as input.

III. Deep learning-based intra prediction mode decision

In this section, we describe our novel deep learning-based intra prediction mode decision. By using CNNs we are able to save the computational expensive RD optimization to find the best intra prediction mode.

Conceptually, the original input samples of the current block are parsed through the CNN of a classifier. This design principle of using only original input sample values decouples the deep learning-based classification process from the actual video encoder. Hence, the classifier does not depend on any decision made by the encoder or any reconstructed sample values of the encoded video. In consequence, the input blocks can be classified in a parallel track of the entire coding system without introducing any additional systematic latency. Additionally, only the luma component of the input blocks is processed. This is motivated by the observation that the chroma components contain only little structural information which could be beneficial for the intra prediction mode decision.

The 35 different intra prediction modes (33 angular modes, the DC mode, and the planar mode) are considered as different classes for the classification process. Thereby, the i-th class maps to the i-th intra mode as defined by the HEVC standard. For example, class 26 maps to the intra mode 26 (i.e. the vertical angular mode). Consequently, the input blocks are classified into these classes.

The selection of suitable parameters for the model and the solver is of crucial importance for machine learning approaches. For this reason, we discuss our model set-up following Figure 1 (CNN model illustration) and Table I (model

TABLE I: Parameters for the CNN models. The network includes two convolutional, three rectified linear unit (ReLU), one max pooling, and two fully-connected layers. Deviating parameters for blocks with a size smaller than 16×16 are noted in brackets.

Layer	Type	#outputs	Filter size	Stride	Weight fill	Bias fill
1	convolutional	96	4×4	1	gaussian, std=0.01	constant, 0
2	ReLU					
3	convolutional	256	5×5 (3×3)	1	gaussian, std=0.01	constant, 1
4	ReLU					
5	MAX pooling	256	3×3	2		
6	fully-connected	1024			gaussian, std=0.005	constant, 1
7	ReLU					
8	fully-connected	35			gaussian, std=0.01	constant, 0

parameters) for the example of 32×32 blocks. Variations for other block sizes are highlighted if applicable. All parameters were chosen to achieve a good trade-off between a low training error and a good generalization to previously unseen data.

Our overall CNN structure is roughly inspired by the award-winning CNN of Krizhevsky et al. [8]. However, we drastically reduced the capacity of the network (e.g. by eliminating some layers and reducing the filter sizes) to cope with the smaller input blocks and fewer classes in our application scenario. Thereby, we avoid overfitting which occurs if the capacity of the network is too high for a given set of training samples. Each input block is fed through two convolutional, one max pooling, and two fully connected layers. Rectified linear units (ReLU) are chosen as hidden units for the activation of the neurons in the hidden layers. They implement the non-linear activation function $f(x) = \max(0, x)$ for the input x. Taking into account this activation function, it is apparent that these hidden units are very easy to optimize because of their similarity to linear units [7].

The first convolutional layer takes the luma component of the input block (i.e. dimension 32×32×1) and filters it with 96 filters of size 4×4 and a stride of one. Thus, the dimension of the succeeding layer is 32×32×96. Afterwards, in the second convolutional layer, the data is filtered by 256 filters of size 5×5. As before, the spatial resolution remains unchanged, i.e. the resulting dimension is 32×32×256. The parameters for this layer vary for blocks smaller than 16×16: the filter size is reduced to 3×3 to cope with the smaller block size. Subsequently, the spatial resolution is halved by a max pooling layer. The reduced size after this layer is 16×16×256. Finally, the data is fed through two fully-connected layers. The first one has 1024 outputs while the second one leads to a softmax output over the 35 classes (i.e. intra prediction modes). The parameters of all layers (which include the filter coefficients) are initialized with common values [8]: the weights are initialized with Gaussian distributed values and the bias is initialized with constant values.

Given the described CNN architecture, we use the well-known caffe framework [16] to learn the CNNs. The *caffe* framework was selected because it provides the tool chain for the entire learning process starting from the training of CNNs on GPUs to the deployment of the trained CNNs as part of an application. Additionally, results are easily reproducible by configuring caffe according to the description in the remainder of the paper. Taking into account that the distribution of selected intra prediction modes depends on the block size, we learn separate CNNs for each block size. The parameters for the learning process are summarized in Table II. Similar to our CNN architecture, our learning strategy is inspired by Krizhevsky et al. [8] with modifications as appropriate for our application.

TABLE II: Parameters for the learning of our CNNs. Deviating parameters for blocks with a size smaller than 16×16 are noted in brackets.

Parameter	Value
optimization method	stochastic gradient descent
base learning rate (λ)	10^{-5} (10^{-2})
learning rate policy	step [8]
step size	10^5
learning rate update	0.1
momentum (μ)	0.9
weight decay (δ)	$5 \cdot 10^{-4}$

To optimize our CNN weights, we use the back propagation algorithm which is based on a stochastic gradient descent solver. Let μ denote the momentum, λ the learning rate, and i and $i-1$ the current and previous learning iteration, respectively. Additionally, let $\nabla L(w_{i-1})$ be the derivative of the objective function (softmax multinomial logistic loss) and δ the weight decay. With this notation, the weight w_i in iteration i is calculated based on the weight in the preceding iteration w_{i-1} and the previous weight update

$$\Delta w_{i-1} = w_{i-1} - w_{i-2} \tag{1}$$

as follows:

$$w_i = w_{i-1} + \mu \cdot \Delta w_{i-1} - \lambda \left(\delta \cdot w_{i-1} + \nabla L(w_{i-1}) \right). \tag{2}$$

For a better fine tuning at the end of the learning process, a step learning rate policy with step size 10^5 and learning rate update 0.1 is applied as in [8]. Thereby, the learning rate is multiplied by 0.1 every 10^5 iterations.

TABLE III: Luma BD-rates for the proposed deep learning-based intra prediction mode classification. Positive numbers indicate a coding efficiency loss. It can be observed that only negligible losses of in average 0.52% are introduced. The baseline results are achieved by random intra mode decision. It is evident that our method considerably outperforms the baseline.

		BD-rate	
Category	**Sequence**	Ours	Baseline
Text & graphics with motion, 1080p	Flying Graphics	0.25%	5.08%
	Desktop	0.01%	2.12%
	Console	0.25%	2.10%
	Chinese Editing	0.05%	2.27%
Text & graphics with motion, 720p	Web Browsing	0.17%	5.03%
	Map	0.54%	10.68%
	Programming	0.32%	7.81%
	Slide Show	1.40%	13.43%
Mixed content, 1440p	Basketball Screen	0.64%	12.32%
	Mission Control Clip 2	0.75%	11.87%
Mixed content, 1080p	Mission Control Clip 3	0.65%	11.31%
Animation, 720p	Robot	1.73%	18.22%
Animation, 768p	China Speed	0.82%	12.84%
Mean text & graphics with motion		**0.37%**	**6.07%**
Mean mixed content		**0.68%**	**11.83%**
Mean animation		**1.28%**	**15.53%**
screen content cross validation	CAD Waveform	0.09%	0.25%
	Video Conferencing	0.04%	0.32%
	Slide Editing	1.08%	4.15%
	Social Network Map	0.11%	0.53%
	Twist Tunnel	0.34%	1.26%
	Word Editing	0.72%	1.19%
Mean cross validation		**0.40%**	**1.28%**
Mean all		**0.52%**	**6.46%**

IV. Evaluation

In this section, the proposed deep learning-based intra prediction mode decision process is evaluated. For this purpose, the CNNs were learned as described in the previous section. The training and test data with ground truth information for the learning process were generated by encoding the first ten pictures of the sequences listed in Table III (excluding the sequences which are listed in the cross validation section of the table). This way, the encoder decision for each encoded block was used to label the data. Furthermore, due to the high temporal and spatial variations in the test sequences, this is a good training set, although the same sequences were used for the evaluation.

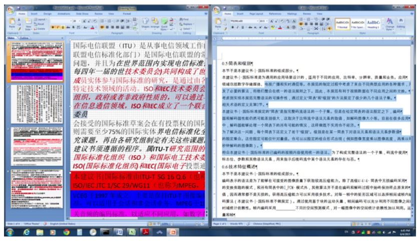
The novel intra prediction mode classification process by CNNs was implemented into the HEVC SCC reference software HM-16.6+SCM-5.2 [17]. Moreover, the encoder was configured to the all-intra configuration as defined by the common test conditions (CTC) [18]. For the purpose of determining the worst-case coding efficiency loss by our introduced method, we used the full intra prediction mode RD optimization as anchor in our evaluation. Four commonly used quantization parameters from the CTC (22, 27, 32, 37) were used for the encoding process. A set of 13 JCT-VC test sequences as defined in Table III, covering a variety of different content characteristics in multiple categories (*text & graphics with motion*, *mixed content* and *animation*), was encoded. Since the first ten pictures were used for the CNN learning, they were skipped for the evaluation. In addition, taking into account the all-intra encoder configuration, we decided that more valuable results are achieved by encoding few pictures from many sequences rather than encoding many pictures from few sequences. Thus, the pictures 10-109 from each sequence were encoded.

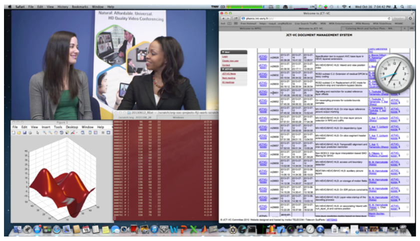
The Bjøntegaard-Delta (BD)-rate as defined in [19] was calculated to evaluate the coding efficiency. Table III summarizes the resulting luma BD-rates. All losses are negligible with an average value of 0.52%. For two sequences, *Desktop* and *Chinese Editing*, the loss is barely noticeable (<0.1%). Furthermore, a breakdown by categories reveals that the proposed method performs better for *text & graphics with motion* sequences than for *mixed content* sequences. In turn, *mixed content* sequences have better results than the *animation* sequences. The reason for this gradation is based in the different complexity of the structures of the sequences in these categories. Along with the losses, the complexity of the structures increases. Thereby, the correct intra prediction mode is more difficult to predict. Examples for sequences in the three categories are illustrated in Figure 2.

As baseline for the evaluation we use the BD-rate losses that are caused by randomly selecting an intra mode instead of conducting the rate distortion optimization. With this comparison we are able to rank our results between optimal approximation (0% BD-rate loss) and worst case (random baseline). These BD-rates are listed in Table III as well. It is noteworthy that the losses for random intra mode selection vary considerably across the different sequences. For sequences with easy and/or only few structural parts, especially in the *text & graphics with motion* category, losses of slightly more than 2% were measured. For complex content, e.g. *Robot*, losses of more than 18% can be observed. For all sequences, our deep learning-based method clearly outperforms the baseline. On average, the losses for the baseline are 12 times higher than the losses for out method. Thus, it can be concluded that the CNNs have learned to predict the correct intra mode.

The state-of-the-art works for fast intra prediction mode decision, which were introduced in Section II, present average BD-rate losses of 1.0% [12], 1.07% [13], 0.69% [14], and 0.66% [15], respectively. Therefore, it can be concluded that our deep learning-based method achieves competitive losses compared to conventional encoder optimizations. This observation is reasonable because deep CNNs can learn to process image characteristics which are only hardly captured by manually designed fast mode decision algorithms.

We evaluated the generalization of the learned CNNs by encoding previously unseen sequences. The results are listed in the *cross validation* section of Table III. The average BD-rate loss is 0.4% which is better than the average result. This indicates an excellent generalization for screen content. In addition, we encoded the pure natural content sequence *Basketball Drive*. Taking into account that the CNNs never saw a natural content sequence during the learning process, a BD-rate loss of 2.74% for *Basketball Drive* suggests that the proposed method is also applicable to natural content. The loss for natural content would drop if the CNNs would see natural content during the training.


(a) Chinese Editing


(b) Mission Control Clip 3

(c) Robot

Fig. 2: Exemplary sequences for the categories *text & graphics with motion* (a), *mixed content* (b) and *animation* (c)

This paper is of explorative nature. Its purpose is to study the applicability of CNNs for encoder decisions. In the light of the presented results it can be concluded that CNNs are very suitable to approximate encoder decisions. Additionally, it is believed that CNNs will be implementable very efficiently in the next few years due to the very regular architecture of filtering operations which can be highly parallelized. For instance, CNN chips could be incorporated into real world devices. However, for this paper we use a classification deployment server based on *caffe* which receives requests from the HM encoder. Given this architecture, run times are not meaningful. Moreover, it is worth mentioning that the proposed method does not introduce additional systematic latency because only original sample values are required. Hence, the CNNs can operate independently of previous encoder decisions and reconstructed sample values.

V. Conclusions

In this paper, we presented a novel deep learning intra prediction mode decision process for HEVC. It is based on feeding the original input sample values of the block to be coded through a deep convolutional neural network. Thereby, the decision over the selected intra prediction mode is formulated as a classification problem without RD optimization of all available modes. BD-rate losses are negligible with an average value of 0.52% over the HEVC SCC reference software. Thus, it can be concluded that CNNs are suitable for making video encoder decisions. Additionally, no systematic latency is introduced. Our method is beneficial for application scenarios in which multiple representation of a video are encoded (e.g. for streaming providers) since the encoder control solely operates on original samples. Thus, a single classification for the best intra prediction mode can be used by all encoders.

References

[1] "ITU-T Recommendation H.265/ ISO/IEC 23008-2:2013 MPEG-H Part 2: High Efficiency Video Coding (HEVC)," 2013.

[2] ISO/IEC 1449610, "Coding of Audiovisual Objects-Part 10: Advanced Video Coding/ITU-T Recommendation H.264 Advanced video coding for generic audiovisual services," 2003.

[3] Philippe Hanhart, Martin Rerabek, Francesca De Simone, and Touradj Ebrahimi, "Subjective quality evaluation of the upcoming HEVC video compression standard," in *SPIE Optical Engineering + Applications*, oct 2012, p. 84990V.

[4] R. Joshi, S. Liu, G. J. Sullivan, Y.-K. Wang, J. Xu, and Y. Ye, "JCT-VC V1005: High Efficiency Video Coding (HEVC) Screen Content Coding: Draft 5. 22th Meeting of the Joint Collaborative Team on Video Coding (JCT-VC), Geneva, CH," 2015.

[5] Frank Bossen, Benjamin Bross, Karsten Suhring, and David Flynn, "HEVC Complexity and Implementation Analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1685–1696, dec 2012.

[6] Jani Lainema, Frank Bossen, Woo-Jin Han, Junghye Min, and Kemal Ugur, "Intra Coding of the HEVC Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1792–1801, dec 2012.

[7] Yoshua Bengio, Ian J. Goodfellow, and Aaron Courville, *Deep Learning*, Book in preparation for MIT Press. Current version available at http://www.deeplearningbook.org/., 2016.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[9] Damian Ruiz-Coll, Velibor Adzic, Gerardo Fernandez-Escribano, Hari Kalva, Jose Luis Martinez, and Pedro Cuenca, "Fast partitioning algorithm for HEVC Intra frame coding using machine learning," in *IEEE International Conference on Image Processing (ICIP)*. oct 2014, pp. 4112–4116, IEEE.

[10] Fanyi Duanmu, Zhan Ma, and Yao Wang, "Fast CU partition decision using machine learning for screen content compression," in *2015 IEEE International Conference on Image Processing (ICIP)*. sep 2015, pp. 4972–4976, IEEE.

[11] Xianyu Yu, Zhenyu Liu, Junjie Liu, Yuan Gao, and Dongsheng Wang, "VLSI friendly fast CU/PU mode decision for HEVC intra encoding: leveraging convolution neural networks," in *IEEE International Conference on Image Processing (ICIP)*, Quebec, Canada, 2015.

[12] Longfei Gao, Shengfu Dong, Wenmin Wang, Ronggang Wang, and Wen Gao, "Fast intra mode decision algorithm based on refinement in HEVC," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. may 2015, pp. 517–520, IEEE.

[13] Mohammadreza Jamali, Stephane Coulombe, and Francois Caron, "Fast HEVC Intra Mode Decision Based on Edge Detection and SATD Costs Classification," in *2015 Data Compression Conference*. apr 2015, pp. 43–52, IEEE.

[14] Jong-Hyeok Lee, Kyung-Soon Jang, Byung-Gyu Kim, Seyoon Jeong, and Jin Soo Choi, "Fast intra mode decision algorithm based on local binary patterns in High Efficiency Video Coding (HEVC)," in *2015 IEEE International Conference on Consumer Electronics (ICCE)*. jan 2015, pp. 270–272, IEEE.

[15] Xiwu Shang, Guozhong Wang, Tao Fan, and Yan Li, "Fast CU size decision and PU mode decision algorithm in HEVC intra coding," in *2015 IEEE International Conference on Image Processing (ICIP)*. sep 2015, pp. 1593–1597, IEEE.

[16] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[17] JCT-VC, "HEVC Screen Content Coding reference software (HM-16.6+SCM-5.2). Available at https://hevc.hhi.fraunhofer.de," 2015.

[18] H. Yu, R. Cohen, K. Rapaka, and J. Xu, "JCT-VC S1015: Common Test Conditions for Screen Content Coding. 21th Meeting of the Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11. Warsaw, PL," 2015.

[19] Gisle Bjøntegaard, "VCEG-AI11: Improvements of the BD-PSNR model. ITU-T Study Group 16 Question 6. 35th Meeting, Berlin, Germany," 2008.