A DEEP CONVOLUTIONAL NEURAL NETWORK APPROACH FOR COMPLEXITY REDUCTION ON INTRA-MODE HEVC

Tianyi Li[†], *Mai Xu*[†*] *and Xin Deng*[‡]

[†]School of Electronic and Information Engineering, Beihang University, Beijing 100191, China
[‡]Communications and Signal Processing (CSP) Group, Imperial College London, SW7 2AZ, UK

ABSTRACT

The High Efficiency Video Coding (HEVC) standard significantly saves coding bit-rate over the proceeding H.264 standard, but at the expense of extremely high encoding complexity. In fact, the coding tree unit (CTU) partition consumes a large proportion of HEVC encoding complexity, due to the brute-force search for rate-distortion optimization (RDO). Therefore, we propose in this paper a complexity reduction approach for intra-mode HEVC, which learns a deep convolutional neural network (CNN) model to predict CTU partition instead of RDO. Firstly, we establish a large-scale database with diversiform patterns of CTU partition. Secondly, we model the partition as a three-level classification problem. Then, for solving the classification problem, we develop a deep CNN structure with various sizes of convolutional kernels and extensive trainable parameters, which can be learnt from the established database. Finally, experimental results show that our approach reduces intra-mode encoding time by 62.25% and 69.06% with negligible Bjøntegaard delta bit-rate of 2.12% and 1.38%, over the test sequences and images respectively, superior to other state-of-the-art approaches.

Index Terms— High efficiency video coding, intra-mode coding, complexity reduction, convolutional neural network

1. INTRODUCTION

The latest High Efficiency Video Coding (HEVC) standard [1] saves nearly 50% bit-rate over its predecessor H.264/Advanced Video Coding (AVC) standard with similar video quality, by adopting several sophisticated video coding techniques like coding tree unit (CTU) partition. However, these techniques lead to extremely high encoding complexity of HEVC. As investigated by [2], the HEVC encoding time is higher than H.264/AVC by $9\% \sim 502\%$, making it intractable to be implemented on real-time applications. Therefore, it is necessary to reduce the encoding complexity of HEVC to an acceptable level with ignorable loss on rate-distortion (RD) performance. In the past few years, great improvement has been seen in the research of HEVC complexity reduction, with various effective approaches emerging. While most of these approaches are concerned with inter-mode [3–13], it is also necessary to reduce intra-mode coding complexity of HEVC, which is 3.2 times higher than that of H.264/AVC [14]. Since the CTU partition contributes to the largest proportion of encoding complexity in HEVC (at least 80% in the reference software HM [15]), most of existing intra-mode HEVC complexity reduction works aim to reduce the complexity by simplifying the process of CTU partition [16–21].

Generally, the works for HEVC complexity reduction can be classified into two categories: heuristic and learning-based methods. In heuristic methods, some intermediate features during encoding are properly explored to early determine the CTU partition before checking all the possibilities of partition. To be more specific, Cho *et al.* [16] developed a CU splitting and pruning method with a Bayesian decision rule, based on full and low-complexity RD costs. Kim *et al.* [17] proposed a method to decide whether a CU is split, according to the number of high-frequency key-points in each CU. In addition to CU-based complexity reduction, some methods were proposed to relieve the complexity of prediction unit (PU) partition. For example, Khan *et al.* [18] proposed a fast PU size decision method by adaptively combining smaller PUs into larger PUs on the basis of video frame contents.

Despite effective, the heuristic methods possess the disadvantage that the criteria to determine the CTU partition have to be developed manually. It may lead to arbitrariness to some extent and is intricate to find correlations among multiple intermediate features, making it difficult to achieve desirable RD performance. To solve such a problem, learning-based methods, with the ability to learn from extensive data and build an optimal model, have emerged to reduce video coding complexity, especially for HEVC. For example, for intra-mode HEVC coding, Hu *et al.* [19] modelled the CU partition process as a binary classification problem with logistic regression, and Liu *et al.* [20] made the classification with the support vector machine (SVM). As a result, the computational time on CTU partition can be significantly reduced using well-trained classification models instead of the brute-force search. For inter-mode HEVC coding, Corrêa *et al.* [12] proposed three early termination schemes with data mining techniques, to simplify the decision of optimal CTU structures. Zhang *et*

*Corresponding author: Mai Xu (maixu@buaa.edu.cn). This work was supported by the NSFC projects under Grants 61573037 and 61202139, and Fok Ying-Tong education foundation under grant 151061.

al. [13] proposed a CU depth decision method with a joint classifier of SVM to make a trade-off between encoding complexity and RD performance. In all those methods, some handcrafted features, such as RD cost, quantization parameter (QP) and the texture complexity, have to be manually extracted to predict the patterns of CTU partition. Those handcrafted features rely heavily on prior knowledge about their relationships with CTU partition results, which remain elusive. Actually, exploration on handcrafted features for CTU partition may benefit from the latest convolutional neural networks (CNN), which can automatically learn features for making decisions on CTU structure. The most recent work [21] has developed a CNN approach to predict CTU structure. However, the architecture of CNN in [21] is shallow, as it only contains two convolutional layers with 6 and 16 3×3 kernels. This shallow architecture can avoid over-fitting when the training data is not sufficient, but it can hardly learn enough features to accurately predict the CTU partition.

In this paper, we propose a deep CNN approach to reduce the complexity of HEVC intra-coding, by learning to split CTU instead of the conventional brute-force search on CTU partition. To this end, we first establish a large-scale database for CTU partition of intra-mode HEVC, which contains 2,000 HEVC encoded images with CTU partition available at 4 different QPs. With enough training data from our database, the architecture of CNN is able to "go deeper", so that extensive parameters can be learnt for handling diverse patterns of CTU partition. Accordingly, we develop a new CNN architecture adaptive to CTU characteristics of HEVC, in which various sizes of convolutional kernels are chosen according to the sizes of all possible CUs. In addition, the stride of kernels in our deep CNN satisfies the CTU splitting, in which non-overlapping is enabled for convolution. More importantly, different from the shallow CNN architecture of [21], a great number of parameters (both on convolutional kernels and fully connected layers) are embedded in our CNN architecture. As a result, our approach is capable of generalizing various kinds of partition patterns in HEVC intra-mode coding, such that the CTU partition can be accurately predicted. It is worth mentioning that the shallow structure of [21] cannot handle the prediction of splitting from 64×64 to 32×32, and thus our approach can reduce more encoding complexity than [21]. By avoiding brute-force search for optimal partition of CTU, our approach is effective and efficient in significantly reducing complexity of HEVC intra-mode coding, which takes advantage of the accurate prediction of CTU partition.

2. CTU PARTITION DATABASE

2.1. Overview of CTU Partition

The CTU partition structure [1] is one of the major contributions in HEVC standard. The size of CTU is 64×64 pixels by default, also the maximum allowed size in standard HEVC. A CTU can either contain a single CU or be recursively split into multiple smaller CUs, based on a quad-tree structure. The minimum size of CU is usually configured before encoding, with the default size of 8×8. Thus, the sizes of CUs in CTU are diverse, ranging from 64×64 to 8×8.

The numbers and sizes of CUs in each CTU are determined by a brute-force rate-distortion optimization (RDO) search, which includes a top-down checking process and a bottom-up comparing process. Figure 1 illustrates the RD cost checking and comparing between a parent CU and its sub-CUs, respectively. In the checking process, the encoder checks the RD cost of the whole CTU, followed by checking its sub-CUs, until reaching the minimum CU size. In Figure 1, the RD cost of a parent CU is denoted as R_p, and the RD costs of its sub-CUs are denoted as $R_{s,m}$, where $m \in \{1, 2, 3, 4\}$ is the index of each sub-CU. Note that when the parent CU is non-split, the RD cost of split flag is included in R_p. However, when split, the cost of encoding split flag as "true" needs to be considered additionally, denoted as R_{sft}. Afterwards, the comparing process based on the RD cost is conducted to determine whether or not a parent CU should be split. As shown in Figure 1 (b), if $R_{sft} + \sum_{m=1}^{4} R_{s,m} < R_p$, the parent CU needs to be split, and otherwise non-split. After the whole RDO search, the CTU partition with the minimum RD cost is selected.

It is worth pointing out that the RDO search is extremely time-consuming, mainly attributed to the brute-force recursive checking process. In a 64×64 CTU, 85 possible CUs are checked, i.e., 1, 4, 4^2 and 4^3 CUs with size of 64×64, 32×32, 16×16 and 8×8 respectively. In order to check the RD cost of each CU, the encoder needs to execute pre-coding for the CU, in which the possible prediction and transformation modes have to be encoded. More importantly, the pre-coding has to be conducted for all 85 possible CUs in standard HEVC intra-coding, thus consuming the largest proportion of encoding time. However, in the final CTU partition, only part of CUs are selected, from 1 (if the 64×64 CU is not split) to 64 (if the whole CTU is split into 8×8 CUs). Therefore, the pre-coding of at most 84 CUs can be avoided by CTU partition prediction (i.e., 85-1), when the whole CTU is not split. The pre-coding of at least 21 CUs can be saved by CTU partition prediction (i.e., 85-64), once the sizes of all CUs in the CTU are 8×8.

2.2. Database Establishment

Now, we establish a large-scale database for CTU Partition of Intra-mode HEVC, namely CPIH database. To our best knowledge, our database[1] is the first one on CTU partition patterns. First, 2000 images at resolution 4928×3264 are selected from Raw Images Dataset (RAISE) [22]. These 2000 images are randomly divided into training (1700 images), validation (100 images) and test (200 images) sets. Furthermore, each set is equally divided into four subsets: one subset is with original resolution and the other three

[1]Our database is available at https://github.com/HEVC-Projects/CPIH

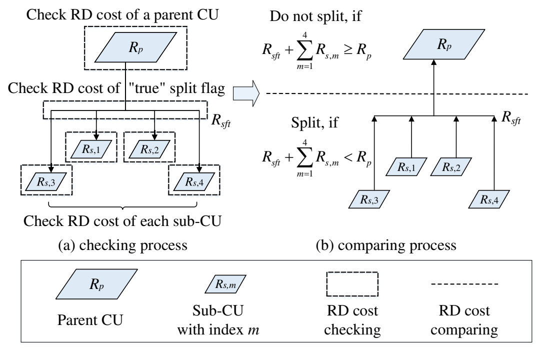

Fig. 1: Illustration of checking and comparing RD cost between a parent CU and its sub-CUs. Note that this illustration can be applied to the splitting of $64\times 64 \rightarrow 32\times 32$, $32\times 32 \rightarrow 16\times 16$ or $16\times 16 \rightarrow 8\times 8$.

Table 1: Splitting and non-splitting samples in CPIH database

CU Size	QP	Number of Samples		
		Splitting	Non-splitting	Total
64×64	22	2,439,479	439,021	2,878,500
	27	2,160,455	718,045	2,878,500
	32	1,982,486	896,014	2,878,500
	37	1,870,295	1,008,205	2,878,500
32×32	22	5,334,843	4,423,073	9,757,916
	27	4,043,975	4,597,845	8,641,820
	32	3,636,461	4,293,483	7,929,944
	37	3,254,952	4,226,228	7,481,180
16×16	22	10,908,631	10,430,741	21,339,372
	27	7,719,222	8,456,678	16,175,900
	32	6,236,637	8,309,207	14,545,844
	37	4,709,216	8,310,592	13,019,808
Total		54,296,652	56,109,132	110,405,784

subsets are down-sampled to be 2880×1920, 1536×1024 and 768×512. As such, our CPIH database contains images at different resolutions. This ensures sufficient and diverse training data for learning to predict CTU partition. Next, all images are encoded by the HEVC reference software HM [15]. Specifically, four QPs $\{22, 27, 32, 37\}$ are applied for encoding with the configuration file *encoder_intra_main.cfg* in the common test conditions [23]. After encoding, the binary labels indicating splitting (=1) and non-splitting (=0) are obtained for all CUs, and each CU with its corresponding binary label can be seen as a sample in our database. Finally, the CPIH database is obtained, which contains 12 sub-databases according to QP and CU size, on account that 4 QPs are applied and CUs with 3 different sizes (64×64, 32×32 and 16×16) are allowed to be split.

Table 1 shows the numbers of splitting and non-splitting CUs from 12 sub-databases in our CPIH database. In total, 110,405,784 samples are gathered, ensuring the sufficiency of training data, and the percentages of splitting and non-splitting CUs are 49.2% and 50.8%, respectively.

3. PROPOSED METHOD

3.1. Three-level CU Classifier

According to CTU partition structure in HEVC, a maximum of four different CU sizes are supported, i.e., 64×64, 32×32,

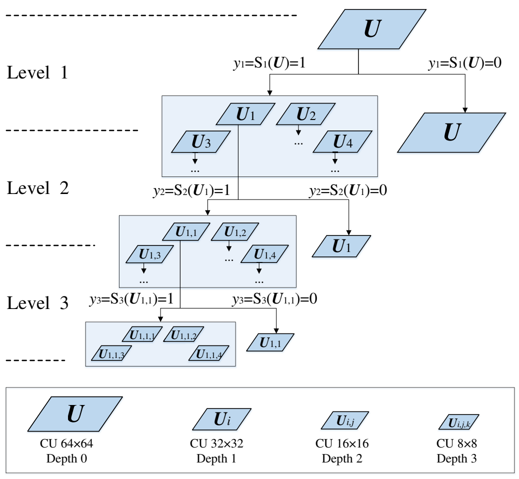
Level 1
Level 2
Level 3
y_1=S$_1$($\boldsymbol{U}$)=1
y_1=S$_1$($\boldsymbol{U}$)=0
y_2=S$_2$($\boldsymbol{U}_1$)=1
y_2=S$_2$($\boldsymbol{U}_1$)=0
y_3=S$_3$($\boldsymbol{U}_{1,1}$)=1
y_3=S$_3$($\boldsymbol{U}_{1,1}$)=0
CU 64×64 Depth 0
CU 32×32 Depth 1
CU 16×16 Depth 2
CU 8×8 Depth 3

Fig. 2: Illustration of three-level CU classifier

16×16 and 8×8, corresponding to CU depth 0, 1, 2 and 3. Note that a CU with size $\geq 16\times 16$ can be either split or non-split. As illustrated in Figure 2, the overall CTU partition can be regarded as a combination of binary classifiers $\{S_l\}_{l=1}^{3}$ at three levels, where $l \in \{1, 2, 3\}$ represents three levels of decisions on whether to split a parent CU into smaller CUs. In particular, $l = 1$ indicates the level of decision for a 64×64 CU into 32×32 CUs. Similarly, $l = 2$ means the level of decision for 32×32 into 16×16, and $l = 3$ stands for 16×16 into 8×8. Given a CTU, we assume that luminance CUs with depth 0, 1, 2 and 3 are denoted as $\boldsymbol{U}$, $\boldsymbol{U}_i$, $\boldsymbol{U}_{i,j}$ and $\boldsymbol{U}_{i,j,k}$, and the subscripts $i, j, k \in \{1, 2, 3, 4\}$ are the indices of the sub-CUs split from $\boldsymbol{U}$, $\boldsymbol{U}_i$ and $\boldsymbol{U}_{i,j}$ respectively. For a CU with size $\geq 16\times 16$, the binary classifier S_l yields the output y_l indicating whether this CU is split ($y_l = 1$) or not ($y_l = 0$), represented by downward arrows with two branches in Figure 2. Considering all binary classifiers, the overall partition of the CTU is extremely complex, because of the quad-tree based CTU structure and the recursive splitting processes. Specifically, for a 16×16 CU, only 2 patterns are possible: splitting and non-splitting. For a 32×32 CU, 2^4 splitting patterns are possible, as we have to decide whether or not to split for its four 16×16 CUs. With the addition of non-splitting pattern on this 32×32 CU, the total number of partition patterns is $2^4 + 1 = 17$. Furthermore, for a 64×64 CU, there exist 17^4 splitting patterns for four 32×32 CUs, such that the total number of the CTU partition patterns is $17^4 + 1 = 83522$.

In HEVC, the classifiers $\{S_l\}_{l=1}^{3}$ are obtained with time-consuming RDO process as mentioned in Section 2.1. In fact, they can be predicted at a much faster speed by machine learning. However, due to the multitudinous patterns of the CTU partition problem (83522 classes in total), it is intractable to be directly predicted by a multi-class classification. Instead, a separate binary classifier is adopted at each decision level $l \in \{1, 2, 3\}$ to predict the binary classification y_l given input $\boldsymbol{U}$, $\boldsymbol{U}_i$ or $\boldsymbol{U}_{i,j}$. Mathematically, there exists a prediction function for our three-level CU

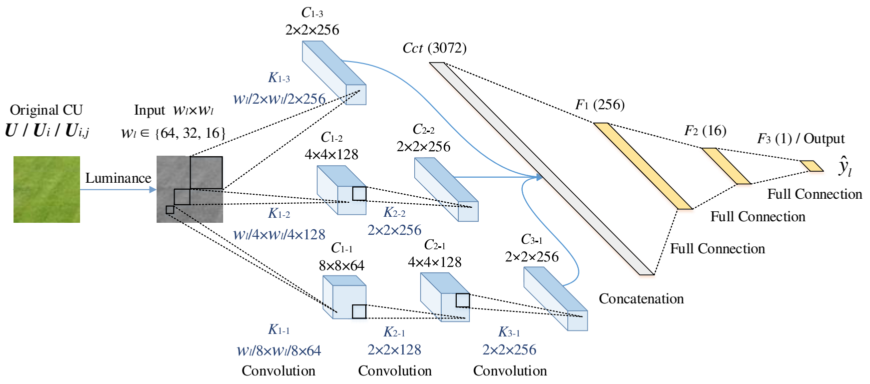

Fig. 3: Structure of the proposed deep CNN. Input to CNN is luminance CU i.e., $\boldsymbol{U}$, $\boldsymbol{U}_i$ and $\boldsymbol{U}_{i,j}$ with size being $w_l \times w_l$. K_{p-q} $(w_k \times w_k \times n_f)$ stands for n_f convolutional filters and their kernel sizes are $w_k \times w_k$, in the q-th branch of the p-th convolutional layer. The stride of each filter is equivalent to the width of kernel size w_k, meaning all the convolutions are non-overlap. After convolution, C_{p-q} $(w_f \times w_f \times n_f)$ is n_f feature maps with size of $w_f \times w_f$. Cct (n_f) denotes the concatenation layer with n_f features, and F_m (n_f) indicates the m-th fully-connected layer with n_f features.

classifier:

$$\hat{y}_l = \begin{cases} S_l(\boldsymbol{U}), & l = 1 \\ S_l(\boldsymbol{U}_i), & l = 2, \quad i \in \{1,2,3,4\} \\ S_l(\boldsymbol{U}_{i,j}), & l = 3, \quad i,j \in \{1,2,3,4\} \end{cases} \tag{1}$$

where $\hat{y}_l$ denotes the predicted y_l. In the following, we focus on developing a deep CNN approach for learning the three-level CU classifier of (1). It is worth mentioning that the proposed classifier can reduce the encoding complexity remarkably through omitting the redundant RD cost checking in the original HEVC standard.

3.2. Proposed Deep CNN Structure

We propose a deep CNN structure adaptive to CTU partition in HEVC, for learning the three-level CU classifier of (1). The structure is composed of an input layer, three convolutional layers, a concatenation layer and three fully-connected layers. By sharing a uniform deep CNN structure, three separate CNN models are learnt for obtaining the classifiers $\{S_l\}_{l=1}^3$ at three levels. The only difference among the three separate CNN models is kernel sizes of the first convolutional layer, pertinent to different sizes of CUs $\boldsymbol{U}$, $\boldsymbol{U}_i$ and $\boldsymbol{U}_{i,j}$. We show in Figure 3 more details about the uniform deep CNN structure. In this figure, w_l is the width of input CU, i.e., 64 for $\boldsymbol{U}$, 32 for $\boldsymbol{U}_i$ and 16 for $\boldsymbol{U}_{i,j}$. To be more specific, all layers in our CNN structure are described as follows.

- **Input layer.** The input layer is luminance CUs of $\boldsymbol{U}$, $\boldsymbol{U}_i$ or $\boldsymbol{U}_{i,j}$, corresponding to the classifiers $S_1(\boldsymbol{U})$, $S_2(\boldsymbol{U}_i)$ and $S_3(\boldsymbol{U}_{i,j})$. Therefore, the input to one CNN model is the $w_l \times w_l$ matrices, where $w_l \in \{64, 32, 16\}$ equals the width of $\boldsymbol{U}$, $\boldsymbol{U}_i$ or $\boldsymbol{U}_{i,j}$ for classifier S_l. Note that all elements of input matrices are normalized to be in $[0, 1]$.

- **Convolutional layers.** For the 1-st convolutional layer, three branches of filters C_{1-1}, C_{1-2} and C_{1-3} with kernel sizes of $\frac{w_l}{8} \times \frac{w_l}{8}$, $\frac{w_l}{4} \times \frac{w_l}{4}$ and $\frac{w_l}{2} \times \frac{w_l}{2}$ are applied in parallel to extract low-level features of CU splitting. We set the strides the same as the sizes of those filters for non-overlap convolutions. The above design of the 1-st convolutional layer is in accordance with all possible non-overlap CUs at different sizes for CTU partition. Following the 1-st convolutional layer, feature maps are half-scaled by convoluting with non-overlapping 2×2 kernels, until the size of final feature maps reaching 2×2.

- **Other layers**. All feature maps, yielded from the last convolutional layer, are concatenated together and then converted into a vector, through the concatenation layer. Next, all features in the concatenated vector flow through three fully-connected layers, including two hidden layers and one output layer. Between the 2-nd fully-connected and output layers, features are randomly dropped out [24] with probability of 50% during CNN training. It is worth mentioning that all convolutional layers and hidden fully-connected layers are activated with rectified linear units (ReLU). The output layer is activated with sigmoid function, since the target output $\hat{y}_l$ for splitting or non-splitting is binary.

Table 2: Configuration of proposed deep CNN

Layer	Filter Size	Number of Filters	Number of Trainable Parameters
C_{1-1}	$\frac{w_l}{8} \times \frac{w_l}{8}$	64	w_l^2
C_{1-2}	$\frac{w_l}{4} \times \frac{w_l}{4}$	128	$8w_l^2$
C_{1-3}	$\frac{w_l}{2} \times \frac{w_l}{2}$	256	$64w_l^2$
C_{2-1}	2×2	128	32,768
C_{2-2}	2×2	256	131,072
C_{3-1}	2×2	256	131,072
F_1	1×1	256	786,432
F_2	1×1	16	4,096
F_3	1×1	1	16
Total Number of Parameters	$73w_l^2 + 1{,}085{,}456 = \begin{cases} 1{,}384{,}464 & w_l = 64 \\ 1{,}160{,}208 & w_l = 32 \\ 1{,}104{,}144 & w_l = 16 \end{cases}$		

The specific configuration of CNN structure of Figure 3 is presented in Table 2. We can see from this table that there are in total $1,384,464$ / $1,160,208$ / $1,104,144$

Table 3: Results of BD-BR (%), BD-PSNR (dB) and ΔT (%) for video sequences

Class	Sequence	Appr.	BD-BR	BD-PSNR	ΔT			
					QP=22	QP=27	QP=32	QP=37
A	PeopleOnStreet	[13]	16.02	-0.86	**-63.39**	-58.99	-57.01	-56.96
		[21]	4.08	-0.23	-50.79	-53.87	-56.58	-61.15
		Our	**2.16**	**-0.12**	-60.83	**-61.10**	**-63.58**	**-66.39**
	Traffic	[13]	11.26	-0.58	-60.84	-60.25	-58.94	-60.16
		[21]	5.17	-0.28	-53.86	-59.08	-63.54	-66.88
		Our	**2.76**	**-0.15**	**-65.39**	**-69.98**	**-70.56**	**-74.21**
B	BasketballDrive	[13]	11.62	-0.27	-59.79	-60.03	-60.90	-62.50
		[21]	6.34	-0.15	**-68.50**	-68.55	-70.30	-70.70
		Our	**4.60**	**-0.11**	-66.70	**-75.03**	**-79.25**	**-82.15**
	BQTerrace	[13]	13.05	-0.77	**-58.26**	-58.38	-57.33	-57.94
		[21]	5.01	-0.31	-53.35	-56.81	-60.26	-61.13
		Our	**1.47**	**-0.10**	-55.20	**-59.35**	**-61.21**	**-64.38**
	Cactus	[13]	13.89	-0.50	**-60.59**	-58.89	-58.06	-58.60
		[21]	6.38	-0.24	-58.18	-61.01	-64.94	-67.78
		Our	**2.11**	**-0.08**	-60.53	**-64.42**	**-68.18**	**-72.35**
	Kimono	[13]	2.42	-0.09	-65.34	-62.93	-62.43	-64.03
		[21]	2.38	-0.08	-70.66	-72.75	-73.62	-73.86
		Our	**2.15**	**-0.08**	**-82.21**	**-83.53**	**-84.11**	**-84.70**
	ParkScene	[13]	6.53	-0.27	-61.48	-60.90	-60.59	-62.12
		[21]	3.81	-0.16	-60.27	-65.10	-68.57	-70.16
		Our	**2.13**	**-0.09**	**-66.12**	**-69.48**	**-71.58**	**-75.96**
C	BasketballDrill	[13]	22.48	-0.98	-58.21	-57.25	-53.76	-53.20
		[21]	12.69	-0.58	**-60.29**	**-62.35**	**-64.48**	-67.20
		Our	**2.97**	**-0.14**	-49.69	-53.80	-61.03	**-67.99**
	BQMall	[13]	22.01	-1.27	**-57.25**	-54.39	**-55.40**	-56.71
		[21]	8.52	-0.53	-47.08	-51.15	-53.26	-57.05
		Our	**1.27**	**-0.08**	-49.29	**-55.44**	-52.96	**-61.07**
	PartyScene	[13]	14.97	-1.10	**-58.77**	-56.49	-48.97	-54.01
		[21]	9.87	-0.76	-52.72	**-57.51**	**-59.77**	**-64.98**
		Our	**0.50**	**-0.04**	-37.33	-37.94	-41.45	-43.94
	RaceHorses	[13]	12.89	-0.80	**-59.20**	-57.16	-54.96	-56.75
		[21]	4.65	-0.30	-50.52	**-59.30**	**-59.81**	-63.15
		Our	**1.75**	**-0.11**	-52.67	-54.70	-58.10	**-64.52**
D	BasketballPass	[13]	18.35	-0.98	-56.02	-54.06	-46.17	-53.81
		[21]	8.80	-0.50	**-60.24**	**-62.89**	**-64.31**	**-66.67**
		Our	**2.20**	**-0.13**	-51.26	-55.38	-58.05	-65.30
	BlowingBubbles	[13]	13.99	-0.80	**-57.26**	-54.88	-48.72	-56.45
		[21]	8.76	-0.52	-54.62	**-60.45**	**-62.55**	**-65.48**
		Our	**0.68**	**-0.04**	-41.68	-44.23	-48.11	-54.43
	BQSquare	[13]	21.55	-1.71	**-53.73**	**-52.82**	-47.41	**-49.93**
		[21]	2.72	-0.24	-42.55	-46.05	**-48.37**	-49.89
		Our	**0.19**	**-0.02**	-46.60	-46.94	-47.49	-46.91
	RaceHorses	[13]	17.08	-1.12	**-57.23**	-54.30	-50.45	-53.73
		[21]	5.19	-0.36	-52.86	**-57.32**	**-58.80**	**-60.20**
		Our	**1.23**	**-0.08**	-45.38	-49.76	-52.60	-55.99
E	FourPeople	[13]	17.57	-0.96	-59.49	-57.58	-58.01	-58.67
		[21]	8.44	-0.49	-54.79	-59.79	-64.39	-67.17
		Our	**3.12**	**-0.18**	**-61.83**	**-66.59**	**-69.11**	**-72.43**
	Johnny	[13]	23.09	-0.88	-59.84	-63.53	-62.41	-63.62
		[21]	8.09	-0.32	-62.92	-65.51	-67.71	-70.05
		Our	**3.60**	**-0.15**	**-72.56**	**-75.74**	**-77.51**	**-79.46**
	KristenAndSara	[13]	24.36	-1.16	-61.35	-60.74	-59.16	-62.43
		[21]	5.57	-0.28	-61.24	-64.61	-65.82	-67.21
		Our	**3.21**	**-0.16**	**-70.47**	**-73.01**	**-75.14**	**-77.42**
Average		[13]	15.73	-0.84	**-59.34**	-57.98	-55.59	-57.87
		[21]	6.47	-0.35	-56.41	-60.23	-62.62	-65.04
		Our	**2.12**	**-0.10**	-57.54	**-60.91**	**-63.33**	**-67.20**

trainable parameters for the CNN models, corresponding to classification of $S_1(\boldsymbol{U})$, $S_2(\boldsymbol{U}_i)$ and $S_3(\boldsymbol{U}_{i,j})$. Thus, our CNN structure provides high capacity of learning, in comparison to only 1,224 trainable parameters in [21] that may cause under-fitting issue. Such huge numbers of trainable parameters also benefit from extensive training samples from our CPIH database (see Table 1 for the number of training samples). For each sample, luminance CU ,i.e., $\boldsymbol{U}, \boldsymbol{U}_i$ or $\boldsymbol{U}_{i,j}$, is fed into the input layer of its corresponding CNN, and its ground-truth classification label y_l is seen as the target output.

In our method, CNNs are trained using batches, the size of which is R. The sets of ground-truth classification labels and predicted outputs are denoted as $\{y_l^{(r)}\}_{r=1}^{R}$ and $\{\hat{y}_l^{(r)}\}_{r=1}^{R}$, respectively, where $y_l^{(r)}$ represents the ground-truth of the r-th training sample and $\hat{y}_l^{(r)}$ is its corresponding prediction. Considering that S_l is a binary classifier, its loss function L_l is based on the following cross-entropy:

$$L_l = -\frac{1}{R}\sum_{r=1}^{R}[y_l^{(r)}\log\hat{y}_l^{(r)} + (1-y_l^{(r)})\log(1-\hat{y}_l^{(r)})]. \tag{2}$$

During training, L_l is minimized with an optimizer implementing the stochastic gradient descent algorithm with momentum.

4. EXPERIMENTAL RESULTS

In this section, we evaluate the performance of our approach, by comparing it with two state-of-the-art approaches, the SVM based approach [13] and the latest CNN approach [21]. All the three approaches were implemented on the HEVC reference software HM [15]. In HM, the all-intra (AI) mode was applied by using the configuration file *encoder_intra_main.cfg* [23], since our approach mainly focuses on complexity reduction of intra-coding of HEVC. Here, four QP values $\in \{22, 27, 32, 37\}$ were chosen for compressing images or videos, with other parameters being set by default. In our experiments, we test on all 200 images of the training set of our CPIH database, as well as all 18 video sequences of the Joint Collaborative Team on Video Coding (JCT-VC) standard test set [14]. The Bjøntegaard delta bit-rate (BD-BR), Bjøntegaard delta PSNR (BD-PSNR) [25] and ΔT are used for evaluating performance of complexity reduction, where ΔT denotes the encoding time-saving rate of complexity reduction approaches over the original HM.

For our approach, 12 deep CNN models were trained for 3 classifiers $\{S_l\}_{l=1}^{3}$ at 4 QP values ($= 22, 27, 32, 37$). When training the models, hyperparameters were tuned on CPIH validation set. As a result, the size of batch for training is 64, and the momentum of optimizer is set to 0.9. In addition, the initial learning rate is 10^{-5} for classifier S_1, 10^{-4} for S_2 and S_3, and it decreases by 1% exponentially at every 1,000 epochs. In total, 500,000 epoches were iterated for training each model.

Evaluation on complexity reduction. First of all, we compare the complexity reduction of the three approaches. Tables 3 and 4 tabulate the encoding complexity reduction results, in terms of complexity reduction rate over the original HM. We can see from this table that our deep CNN approach saves more time for most sequences at four QPs. In average, our approach ($-60.91\%, -63.33\%$ and -67.20%) outperforms other two approaches ($-57.98\%, -55.59\%$ and -57.87% for [13]; $-60.23\%, -62.62\%$ and -65.04% for [21]) in complexity reduction at QP $= 27, 32$ and 37. Note that our deep CNN approach consumes less time than the shallow CNN approach [21], since [21] requires RDO search for decision on splitting from 64×64 to 32×32. We can further find that the gap of time saving between our and other two approaches becomes larger when QP increases. This is because only one or two deep CNN models are applied in our three-level classifier (corresponding to classifier S_1 or S_2) for more CTUs at decreasing bit-rate, which leads to more large CUs. In addition, we can see from Table 4 that our approach is able to averagely reduce more time on encoding all images of the test set from our CPIH database, compared to [13] and [21]. However, for low resolution images encoded at a high bit-rate, our approach may have a higher computational

Table 4: Results of BD-BR (%), BD-PSNR (dB) and ΔT (%) for images from our CPIH test set)

Image Source	Resolution	Appr.	BD-BR	BD-PSNR	ΔT			
					QP=22	QP=27	QP=32	QP=37
CPIH Test Set	768×512	[13]	9.04	-0.63	**-60.47**	**-60.56**	-58.38	-61.30
		[21]	3.02	-0.22	-54.97	-58.78	**-61.78**	-64.41
		Our	**1.05**	**-0.08**	-54.74	-57.84	-59.58	**-65.19**
	1536×1024	[13]	10.50	-0.68	**-60.95**	-61.19	-61.13	-62.81
		[21]	3.28	-0.22	-55.84	-59.46	-62.43	-64.17
		Our	**1.28**	**-0.09**	-59.08	**-62.25**	**-63.47**	**-67.68**
	2880×1920	[13]	4.85	-0.26	-65.86	-65.64	-64.93	-66.70
		[21]	2.23	-0.12	-59.95	-63.14	-68.07	-69.46
		Our	**1.67**	**-0.09**	**-69.63**	**-73.34**	**-75.92**	**-78.48**
	4928×3264	[13]	4.47	-0.21	-66.81	-66.68	-66.26	-66.09
		[21]	1.95	-0.09	-61.43	-65.27	-68.70	-71.00
		Our	**1.51**	**-0.07**	**-76.01**	**-79.56**	**-81.19**	**-81.05**
Average		[13]	7.22	-0.44	-63.52	-63.52	-62.67	-64.22
		[21]	2.62	-0.16	-58.05	-61.66	-65.25	-67.26
		Our	**1.38**	**-0.08**	**-64.86**	**-68.25**	**-70.04**	**-73.10**

complexity than [13], as shown in Table 3. The reason is that CTU prefers to have small CUs, when resolution is low and bit-rate is high. In a word, our approach is capable of improving the time efficiency of HEVC intra-mode coding.

Evaluation on RD performance. Next, we compare the RD performance of our and other two approaches, in terms of BD-BR and BD-PSNR. Tables 3 and 4 report BD-BR increment and BD-PSNR reduction of three approaches, with the original HM as anchor. We can see from these tables that the BD-BR increment of our deep CNN approach is averagely 2.12% for videos and 1.38% for images, which significantly outperforms [13] (6.47% for videos and 2.62% for images) and [21] (15.73% for videos and 7.22% for images). In addition, our approach incurs 0.1 dB and 0.08 dB PSNR degradation for videos and images, respectively, far better than those of [13] and [21]. Thus, our approach performs best among the three approaches. The improvement of RD performance by our approach mainly attributes to the high prediction accuracy of CTU partition, benefiting from the deep CNN structure with sufficient parameters learnt from our large-scale CPIH database.

5. CONCLUSIONS

In this paper, we propose a deep CNN approach to reduce the encoding complexity on intra-mode HEVC, by learning to predict the optimal CTU partition instead of conventional brute-force RDO search. The CNN utilizes image textures of CUs and is adaptive to CTU partition. In addition, a large-scale CPIH database is established, with diversiform CTU partition patterns sufficient for CNN training. Compared with the original HM, our approach reduces encoding time by 62.25% and 69.06% with negligible 2.12% and 1.38% BD-BR, on the JCT-VC standard test sequences and our CPIH test images, respectively.

6. REFERENCES

[1] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE TCSVT*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[2] G. Corrêa, P. Assuncao, L. Agostini, and L. A. da Silva Cruz, "Performance and computational complexity assessment of high-efficiency video encoders," *IEEE TCSVT*, vol. 22, no. 12, pp. 1899–1909, Dec. 2012.

[3] J. Leng, L. Sun, T. Ikenaga and S. Sakaida, "Content Based Hierarchical Fast Coding Unit Decision Algorithm for HEVC," in *CMSP*, 2011.

[4] X. Shen, L. Yu and J. Chen, "Fast coding unit size selection for HEVC based on Bayesian decision rule," in *PCS*, 2012.

[5] J. Xiong, H. Li, Q. Wu, and F. Meng, "A fast HEVC inter CU selection method based on pyramid motion divergence," *IEEE TMM*, vol. 16, no. 2, pp. 559–564, Feb. 2014.

[6] H. M. Yoo and J. W. Suh, "Fast coding unit decision algorithm based on inter and intra prediction unit termination for HEVC," in *ICCE*, 2013.

[7] C. Kiho and E. S. Jang, "Early TU decision method for fast video encoding in high efficiency video coding," *Electron. Lett*, vol. 48, no. 12, pp. 689–691, Jun. 2012.

[8] J. Vanne, M. Viitanen, and T. Hamalainen, "Efficient Mode Decision Schemes for HEVC Inter Prediction," *IEEE TCSVT*, vol. 24, no. 9, pp. 1579–1593, Sept. 2014.

[9] K. Miyazawa, T. Murakami, A. Minezawa and H. Sakate, "Complexity reduction of in-loop filtering for compressed image restoration in HEVC," in *PCS*, 2012.

[10] T. Shanableh, E. Peixoto, and E. Izquierdo, "MPEG-2 to HEVC video transcoding with content-based modeling," *IEEE TCSVT*, vol. 23, no. 7, pp. 1191–1196, Jul. 2013.

[11] E. Peixoto, T. Shanableh, and E. Izquierdo, "H.264/AVC to HEVC video transcoder based on dynamic thresholding and content modeling," *IEEE TCSVT*, vol. 24, no. 1, pp. 99–112, Jan. 2014.

[12] G. Corrêa, P. A. Assuncao, L. V. Agostini and L. A. da Silva Cruz, "Fast HEVC Encoding Decisions Using Data Mining," *IEEE TCSVT*, vol. 25, no. 4, pp. 660–673, Apr. 2015.

[13] Y. Zhang, S. Kwong, X. Wang, H. Yuan, Z. Pan and L. Xu, "Machine Learning-Based Coding Unit Depth Decisions for Flexible Complexity Allocation in High Efficiency Video Coding," *IEEE TIP*, vol. 24, no. 7, pp. 2225–2238, Jul. 2015.

[14] J. Vanne, M. Viitanen, T. D. H am al ainen, and A. Hallapuro, "Comparative rate-distortion-complexity analysis of hevc and avc video codecs," *IEEE TCSVT*, vol. 22, no. 12, pp. 1885–1898, 2012.

[15] JCT-VC, "HM Software," [Online]. Available: `https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-12.0/`, 2014, [Accessed 5-Nov.-2016].

[16] S. Cho and M. Kim, "Fast CU Splitting and Pruning for Suboptimal CU Partitioning in HEVC Intra Coding," *IEEE TCSVT*, vol. 23, no. 9, pp. 1555–1564, Sept. 2013.

[17] N. Kim, S. Jeon, H. J. Shim, B. Jeon, S. C. Lim and H. Ko, "Adaptive keypoint-based CU depth decision for HEVC intra coding," in *IEEE BMSB*, 2016.

[18] M. U. K. Khan, M. Shafique and J. Henkel, "An adaptive complexity reduction scheme with fast prediction unit decision for HEVC intra encoding," in *IEEE ICIP*, 2013.

[19] Q. Hu, Z. Shi, X. Zhang and Z. Gao, "Fast HEVC intra mode decision based on logistic regression classification," in *IEEE BMSB*, 2016.

[20] D. Liu, X. Liu and Y. Li, "Fast CU Size Decisions for HEVC Intra Frame Coding Based on Support Vector Machines," in *IEEE DASC*, 2016.

[21] Z. Liu, X. Yu, S. Chen and D. Wang, "CNN oriented fast HEVC intra CU mode decision," in *IEEE ISCAS*, 2016.

[22] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato, "RAISE - A Raw Images Dateset for Digital Image Forensics," in *ACM MM*, 2015.

[23] F. Bossen, "Common Test Conditions and Software Reference Configurations," Joint Collaborative Team on Video Coding, document Rec. JCTVC-L1100, Jan. 2013.

[24] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov, "Improving Neural Networks by Preventing Co-adaptation of Feature Detectors," *Computer Science*, vol. 3, no. 4, pp. 212–223, 2012.

[25] G. Bjøntegaard, "Calculation of Avarage PSNR Difference Between RD-Curves," in *ITU-T, VCEG-M33, Austin, TX, USA*, Apr. 2001.