Reducing Complexity of HEVC: A Deep Learning Approach

Mai Xu, *Senior Member, IEEE*, Tianyi Li, Zulin Wang, Xin Deng, Ren Yang, and Zhenyu Guan

***Abstract*— High efficiency video coding (HEVC) significantly reduces bit rates over the preceding H.264 standard but at the expense of extremely high encoding complexity. In HEVC, the quad-tree partition of the coding unit (CU) consumes a large proportion of the HEVC encoding complexity, due to the brute-force search for rate-distortion optimization (RDO). Therefore, this paper proposes a deep learning approach to predict the CU partition for reducing the HEVC complexity at both intra- and inter-modes, which is based on convolutional neural network (CNN) and long- and short-term memory (LSTM) network. First, we establish a large-scale database including substantial CU partition data for the HEVC intra- and inter-modes. This enables deep learning on the CU partition. Second, we represent the CU partition of an entire coding tree unit in the form of a hierarchical CU partition map (HCPM). Then, we propose an early terminated hierarchical CNN (ETH-CNN) for learning to predict the HCPM. Consequently, the encoding complexity of intra-mode HEVC can be drastically reduced by replacing the brute-force search with ETH-CNN to decide the CU partition. Third, an ETH-LSTM is proposed to learn the temporal correlation of the CU partition. Then, we combine the ETH-LSTM and the ETH-CNN to predict the CU partition for reducing the HEVC complexity at inter-mode. Finally, experimental results show that our approach outperforms the other state-of-the-art approaches in reducing the HEVC complexity at both intra- and inter-modes.**

***Index Terms*— High efficiency video coding, complexity reduction, deep learning, convolutional neural network, long- and short-term memory network.**

I. Introduction

THE High Efficiency Video Coding (HEVC) standard [1] saves approximately 50% of the bit-rate at similar video quality compared to its predecessor, H.264/Advanced Video Coding (AVC). This is achieved by adopting some advanced video coding techniques, e.g., the quad-tree structure of the coding unit (CU) partition. However, these techniques lead to extremely high complexity of HEVC. As investigated by [2], the encoding time of HEVC is higher than H.264/AVC by 253% on average, making it impractical for implementation in multimedia applications. Therefore, it is necessary to significantly reduce the encoding complexity of HEVC with ignorable loss in rate-distortion (RD) performance.

The past half decade has witnessed a great number of approaches on encoding complexity reduction for HEVC. It is found that the recursive process of the quad-tree CU partition contributes to the largest proportion of the encoding time (at least 80% in the reference software HM [3]). Thus, most HEVC complexity reduction approaches attempt to simplify the process of the CU partition. The basic idea of these approaches is to predict the CU partition in advance, instead of the brute-force recursive RD optimization (RDO) search. The early works for predicting the CU partition are heuristic approaches such as in [4]–[6]. These heuristic approaches explore some intermediate features to early determine the CU partition before checking all possible quad-tree patterns. Since 2015, several machine learning approaches [7]–[11] have been proposed to predict the CU partition toward HEVC complexity reduction. For example, to reduce the complexity of inter-mode HEVC, Zhang *et al.* [8] proposed a CU depth decision approach with a three-level joint classifier based on support vector machine (SVM), which predicts the splitting of three-sized CUs in the CU partition. To reduce the complexity of intra-mode HEVC, Liu *et al.* [9] developed a convolutional neural network (CNN) based approach that predicts the CU partition. However, the structure of the CNN in [9] is shallow, with limited learning capacity, and thus, it is insufficient to accurately model the complicated CU partition process. In this paper, we propose a deep CNN structure for learning to predict the intra-mode CU partition, thus reducing the complexity of HEVC at intra-mode. In addition, all existing approaches, including [8] and [9], do not explore the correlation of the CU partition across neighboring frames. This paper develops a long- and short-term memory (LSTM) structure to learn the temporal dependency of the inter-mode CU partition. Then, a deep learning approach is proposed to predict the CU partition at inter-mode, which combines the CNN and LSTM structures. Consequently, the encoding complexity of inter-mode HEVC can be significantly reduced.

Specifically, in this paper, we first establish a large-scale database for the CU partition, boosting studies on deep learning based complexity reduction in HEVC. Our database is available online: https://github.com/HEVC-Projects/CPH. Our database contains the data of the CU partition at both intra-mode (2000 raw images compressed at four QPs) and

Manuscript received August 27, 2017; revised January 2, 2018 and March 1, 2018; accepted June 10, 2018. Date of publication June 13, 2018; date of current version July 12, 2018. This work was supported in part by NSFC under Grant 61573037 and Grant 61702028 and in part by the Fok Ying-Tong Education Foundation under Grant 151061. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Feng Wu. *(Corresponding author: Zhenyu Guan.)*

M. Xu, T. Li, Z. Wang, R. Yang, and Z. Guan are with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China (e-mail: guanzhenyu@buaa.edu.cn).

X. Deng is with the Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, U.K.

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author. The material includes information relevant to the main paper. The total size of the file is 0.107 MB. Contact MaiXu@buaa.edu.cn for further questions about this work.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2018.2847035

inter-mode (111 raw sequences compressed at four QPs). Next, we propose a deep learning based complexity reduction approach for both intra- and inter-mode encoding of HEVC, which learns from our database to split an entire coding tree unit (CTU). More specifically, we propose to efficiently represent the CU partition of a CTU through the hierarchical CU partition map (HCPM). Given sufficient training data and efficient HCPM representation, the deep learning structures of our approach are able to "go deeper", such that extensive parameters can be learned for exploring diverse patterns of the CU partition. Accordingly, our deep learning approach introduces an early-terminated hierarchical CNN (ETH-CNN) structure, which hierarchically generates the structured HCPM with an early-termination mechanism. The early-termination mechanism introduced in ETH-CNN can save computational time when running the CNN. ETH-CNN can be used to reduce the encoding complexity of intra-mode HEVC. Our approach further introduces an early-terminated hierarchical LSTM (ETH-LSTM) structure for inter-mode HEVC. In ETH-LSTM, the temporal correlation of the CU partition is learned in LSTM cells. With the features of the ETH-CNN as the input, ETH-LSTM hierarchically outputs HCPM by incorporating the learnt LSTM cells with the early-termination mechanism. As such, our approach can also be used to reduce the encoding complexity of inter-mode HEVC.

This paper was previously presented at the conference [12], with the following extensions. Rather than the three-level CU splitting labels of [12], HCPM is proposed in this paper to hierarchically represent the structured output of the CU partition. Based on the proposed HCPM representation, we advance the deep CNN structure of [12] by introducing the early-termination mechanism, for intra-mode HEVC complexity reduction. More importantly, this paper further proposes ETH-LSTM to reduce the HEVC complexity at inter-mode. In contrast, our previous work [12] only addresses complexity reduction in intra-mode HEVC. For learning ETH-LSTM, a large-scale database of inter-mode CU partition is established in this paper, whereas [12] only contains the database of intra-mode CU partition. In brief, the main contributions of this paper are summarized as follows.

- We establish a large-scale database for the CU partition of HEVC at both intra- and inter-modes, which may facilitate the applications of deep learning in reducing HEVC complexity.
- We propose a deep CNN structure called ETH-CNN to predict the structured output of the CU partition in the form of HCPM, for reducing the complexity of intra-mode HEVC.
- We develop a deep LSTM structure named ETH-LSTM that learns the spatio-temporal correlation of the CU partition, for reducing the complexity of HEVC at inter-mode.

This paper is organized as follows. Section II reviews the related works on HEVC complexity reduction. Section III presents the established CU partition database. In Sections IV and V, we propose ETH-CNN and ETH-LSTM to reduce the HEVC complexity at intra-mode and inter-mode, respectively. Section VI reports the experimental results, and Section VII concludes this paper.

II. Related Works

The existing HEVC complexity reduction works can be generally classified into two categories: heuristic and learning-based approaches. This section reviews the complexity reduction approaches in these two categories.

In heuristic approaches, the brute-force RDO search of the CU partition can be simplified according to some intermediate features. The representative approaches include [4]–[6], [13], [14]. To be more specific, Leng *et al.* [4] proposed a CU depth decision approach at the frame level, which skips certain CU depths rarely found in the previous frames. At the CU level, Xiong *et al.* [5] and Kim *et al.* [14] proposed deciding whether to split CUs based on the pyramid motion divergence and the number of high-frequency key-points, respectively. Shen *et al.* [6] proposed selecting some crucial and computational-friendly features, e.g., RD cost and inter-mode prediction error, to early determine the splitting of each CU. Then, the CU splitting is determined based on the rule of minimizing the Bayesian risk. Also based on the Bayesian rule, Cho and Kim *et al.* [13] developed a CU splitting and pruning approach according to the features of full and low-complexity RD costs. In addition to simplified the CU partition, various heuristic approaches [15]–[18] were proposed to reduce the complexity of prediction unit (PU) or transform unit (TU) partition. For example, Khan *et al.* [15] proposed a fast PU size decision approach, which adaptively integrates smaller PUs into larger PUs with regard to video content. Yoo *et al.* [16] proposed estimating the PU partition with the maximum probability, on account of the coding block flag (CBF) and RD costs of encoded PUs. In the latest work of [18], the RDO quantization (RDOQ) process is accelerated based on the transform coefficients with a hybrid Laplace distribution. In addition, other components of HEVC, such as intra- and inter-mode prediction and in-loop filtering, are simplified to reduce the encoding complexity in [19]–[23].

Most recently, learning-based approaches [7]–[11], [24]–[33] have flourished for complexity reduction in HEVC, and even for other communication tasks [34]. These approaches utilize machine learning from extensive data to generalize the rules of HEVC encoding components, instead of a brute-force RDO search on these components. For example, for intra-mode HEVC, [24] modeled the CU partition process as a binary classification problem with logistic regression, and [25] proposed using SVM to perform classification in the CU partition process. As a result, the computational time of the CU partition can be decreased using well-trained classification models instead of brute-force search. For inter-mode HEVC, Corrêa *et al.* [7] proposed three early termination schemes with data mining techniques to simplify the decision on the optimal CTU structures. In [8], several HEVC domain features that are correlated with the CU partition were explored. Then, a joint classifier of SVM was proposed to utilize these features in determining CU depths, such that the encoding complexity of HEVC can be reduced because the brute-force RDO search is skipped. Later, a binary and multi-class SVM algorithm [11] was proposed to predict both the CU partition and PU mode with an off-on-line machine learning mechanism. As such, the encoding time of HEVC can be further saved. However, the above learning-based approaches rely heavily on hand-crafted features that are related to the CU partition.

For complexity reduction in HEVC, the CU partition-related features can be automatically extracted by deep learning, rather than manual extraction. The advantage of applying deep learning for HEVC complexity reduction is that it can take advantage of large-scale data to automatically mine extensive features related to the CU partition, rather than the limited hand-crafted features. Unfortunately, there are few deep learning works on the prediction of the CU partition. To the best of our knowledge, the only approaches in this direction are [9], [26], in which the CNN architectures have been developed to predict intra-mode CU partition. The CNN architectures of [9], [26] are not sufficiently deep due to a lack of training data. For example, [9] only contains two convolutional layers with 6 and 16 kernels sized 3×3. In contrast, we propose the deep architectures of ETH-CNN and ETH-LSTM for HEVC complexity reduction at both intra- and inter-modes.

Our approach differs from the above learning-based CU partition prediction approaches in four aspects. (1) Compared with the conventional three-level CU splitting labels as in [8], [9], [11], [26], the HCPM is proposed to efficiently represent the structured output of the CU partition. This can dramatically reduce the computational time of the CU partition, since the complete CU partition in a whole CTU can be obtained in terms of one HCPM by running the trained ETH-CNN/ETH-LSTM model only once. (2) A deep ETH-CNN structure is designed to automatically extract features for predicting the CU partition, instead of the handcrafted feature extraction in [8], [11]. Besides, the deep ETH-CNN structure has much more trainable parameters than the CNN structures of [9] and [26], thus remarkably improving the prediction accuracy of the CU partition. Additionally, the early-termination mechanism proposed in our approach can further save computational time. (3) A deep ETH-LSTM model is further developed for learning long- and short-term dependencies of the CU partition across frames for inter-mode HEVC. To our best knowledge, this is a first attempt to leverage LSTM for predicting CU partition in HEVC complexity reduction. (4) To train the large amount of parameters in ETH-CNN and ETH-LSTM, we establish a large-scale database for the CU partition of HEVC at both intra- and inter-modes. In contrast, other works only rely on the existing JCT-VC database, much smaller than our database. Our database may facilitate the future works of applying deep learning in the CU partition prediction for reducing HEVC complexity.

III. CU Partition Database

A. Overview of CU Partition

The CTU partition structure [1] is one of the major contributions to the HEVC standard, with the CU partition as the core process. The size of a CTU is 64×64 pixels by default, and a CTU can either contain a single CU or be recursively split into multiple smaller CUs, based on the quad-tree structure. The minimum size of a CU is configured before encoding, with the default being 8×8. Thus, the sizes of the CUs in the CTU are diverse, ranging from 64×64 to 8×8. Note that the maximal CTU sizes can be extended to be larger than 64×64, e.g., 128×128.

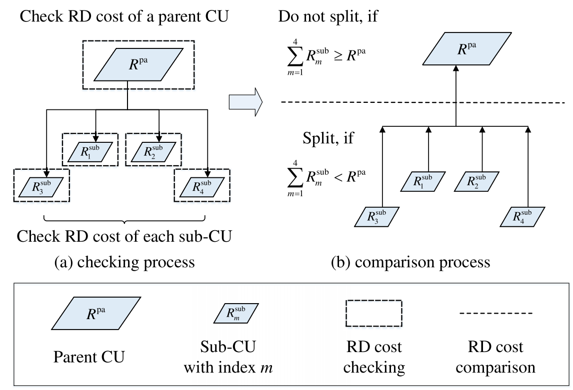


Fig. 1. Illustration of checking and comparing RD cost between a parent CU and its sub-CUs. Note that this illustration can be applied to the splitting of $64 \times 64 \rightarrow 32 \times 32$, $32 \times 32 \rightarrow 16 \times 16$ or $16 \times 16 \rightarrow 8 \times 8$.

As we know, the sizes of the CUs in each CTU are determined using a brute-force RDO search, which includes a top-down checking process and a bottom-up comparison process. Fig. 1 illustrates the RD cost checking and comparison between a parent CU and its sub-CUs. 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 Fig. 1, the RD cost of a parent CU is denoted as R^{pa}, and the RD costs of its sub-CUs are denoted as R_m^{sub}, where $m \in \{1, 2, 3, 4\}$ is the index of each sub-CU. Afterwards, based on the RD costs of CUs and sub-CUs, the comparison process is conducted to determine whether a parent CU should be split. As shown in Fig. 1 (b), if $\sum_{m=1}^{4} R_m^{sub} < R^{pa}$, the parent CU needs to be split; otherwise, it is not split. Note that the RD cost of the split flag is also considered when deciding whether to split the CU. After the full RDO search, the CU partition with the minimum RD cost is selected.

It is worth noting that the RDO search is extremely time-consuming, mainly attributed to the recursive checking process. In a 64×64 CTU, 85 possible CUs are checked, including 1, 4, 4^2 and 4^3 CUs with sizes of 64×64, 32×32, 16×16 and 8×8. 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 the standard HEVC, consuming the largest proportion of the encoding time. However, in the final CU partition, only certain 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 84 CUs (i.e., 85-1) at most and 21 CUs (i.e., 85-64) at least can be avoided through the accurate prediction of the CU partition.

B. Database Establishment for Intra-Mode

In this section, we present our large-scale database for the CU Partition of HEVC at Intra-mode: the CPH-Intra database. To the best of our knowledge, our database is the first database on the CU partition patterns. To establish the CPH-Intra database, 2000 images at resolution 4928×3264 were selected from the Raw Images Dataset

TABLE I
CONFIGURATION OF PROPOSED ETH-CNN

Feature	Size	Number of Parameters	Number of Additions	Number of Multiplications
C_{1-1}	16×16	256	3,840	4,096
C_{1-2}	32×32	256	15,360	16,384
C_{1-3}	64×64	256	61,440	65,536
C_{2-1}	4×4	1,536	4,608	6,144
C_{2-2}	8×8	1,536	18,432	24,576
C_{2-3}	16×16	1,536	73,728	98,304
C_{3-1}	2×2	3,072	2,304	3,072
C_{3-2}	4×4	3,072	9,216	12,288
C_{3-3}	8×8	3,072	36,864	49,152
$\mathbf{f}_{1-1}$	2688	172,032	171,968	172,032
$\mathbf{f}_{1-2}$	2688	344,064	343,936	344,064
$\mathbf{f}_{1-3}$	2688	688,128	687,872	688,128
$\mathbf{f}_{2-1}$	65	3,120	3,072	3,120
$\mathbf{f}_{2-2}$	129	12,384	12,288	12,384
$\mathbf{f}_{2-3}$	257	49,344	49,152	49,344
$\hat{y}_1(\mathbf{U})$	49	49	48	49
$\{\hat{y}_2(\mathbf{U}_i)\}_{i=1}^4$	97	388	384	388
$\{\hat{y}_3(\mathbf{U}_{i,j})\}_{i,j=1}^4$	193	3,088	3,072	3,088
Total	-	1,287,189	1,497,584	1,552,149

(RAISE) [35]. These 2000 images were randomly divided into training (1700 images), validation (100 images) and test (200 images) sets. Furthermore, each set was equally divided into four subsets: one subset was with original resolution, and the other three sets were down-sampled to 2880 × 1920, 1536 × 1024 and 768 × 512. As such, our CPH-Intra database contains images at different resolutions. This ensures sufficient and diverse training data for learning to predict the CU partition.

Next, all images were encoded by the HEVC reference software HM 16.5 [3]. Here, four QPs $\{22, 27, 32, 37\}$ were applied to encode at the All-Intra (AI) configuration with the file *encoder_intra_main.cfg* [36]. After encoding, the binary labels indicating splitting (=1) and non-splitting (=0) can be obtained for all CUs, and each CU with its corresponding binary label is a sample in our database. Finally, there are 110,405,784 samples collected in the CPH-Intra database. These samples are split into 12 sub-databases according to their QP value and CU size, as reported in Table I-(a) of the *Supporting Document*. As shown in this table, the total percentage of split CUs (49.2%) is close to that of non-split CUs (50.8%) over the whole database.

C. Database Establishment for Inter-Mode

We further establish a database for the CU Partition of HEVC at Inter-mode: the CPH-Inter database. For establishing the CPH-Inter database, 111 raw video sequences were selected, therein consisting of 6 sequences at 1080p (1920 × 1080) from [37], 18 sequences of Classes A ∼ E from the Joint Collaborative Team on Video Coding (JCT-VC) standard test set [38], and 87 sequences from Xiph.org [39] at different resolutions. As a result, our CPH-Inter database contains video sequences at various resolutions: SIF (352 × 240), CIF (352 × 288), NTSC (720 × 486), 4CIF (704 × 576), 240p (416 × 240), 480p (832 × 480), 720p (1280 × 720), 1080p, WQXGA (2560 × 1600) and 4K (4096 × 2160). Note that the NTSC sequences were cropped to 720 × 480 by removing the bottom edges of the frames, considering that only resolutions in multiples of 8 × 8 are supported. Moreover, if the durations of the sequences are longer than 10 seconds, they were clipped to be 10 seconds.

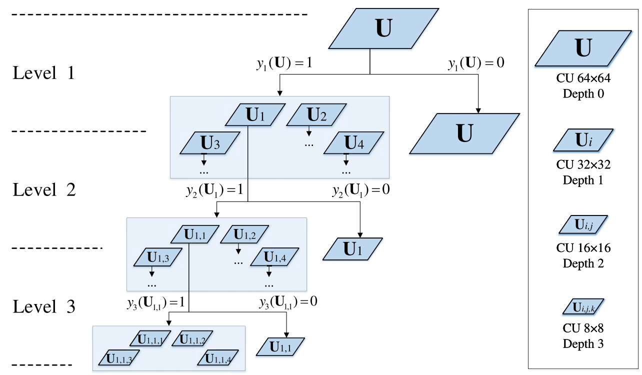


Fig. 2. Illustration of three-level CU classifier.

In our CPH-Inter database, all the above sequences were divided into non-overlapping training (83 sequences), validation (10 sequences) and test (18 sequences) sets. For the test set, all 18 sequences from the JCT-VC set [38] were selected. Similar to the CPH-Intra database, all sequences in our CPH-Inter database were encoded by HM 16.5 [3] at four QPs $\{22, 27, 32, 37\}$. They were compressed with the Low Delay P (LDP) (using *encoder_lowdelay_P_main.cfg*), the Low Delay B (LDB) (using *encoder_lowdelay_main.cfg*) and the Random Access (RA) (using *encoder_randomaccess_main.cfg*) configurations, respectively. Consequently, 12 sub-databases were obtained for each configuration, corresponding to different QPs and CU sizes. As reported in Table I-(b), -(c) and -(d) of the *Supporting Document*, totally 307,831,288 samples were collected for the LDP configuration in our CPH-Inter database. Besides, 275,163,224 and 232,095,164 samples were collected in total in our database for the LDB and RA configurations, respectively.

IV. COMPLEXITY REDUCTION FOR INTRA-MODE HEVC

A. Hierarchical CU Partition Map

According to the CU partition structure in HEVC, four different CU sizes are supported by default, i.e., 64 × 64, 32 × 32, 16 × 16 and 8 × 8, corresponding to four CU depths, 0, 1, 2 and 3. Note that a CU of size ≥ 16 × 16 can be either split or not split. As illustrated in Fig. 2, the overall CU partition can be regarded as a combination of binary labels $\{y_l\}_{l=1}^3$ at three levels, where $l \in \{1, 2, 3\}$ represents the level on which the split decision is made. In particular, $l = 1$ indicates the first level, which determines whether to split a 64 × 64 CU into 32 × 32 CUs. Similarly, $l = 2$ is the second level to decide whether a 32×32 CU is split into 16×16 CUs, and $l = 3$ is for 16 × 16 into 8 × 8.

Given a CTU, we assume that the CU of depth 0 is denoted as $\mathbf{U}$. For $\mathbf{U}$, the first-level label $y_1(\mathbf{U})$ indicates whether $\mathbf{U}$ is split (= 1) or not (= 0). If $\mathbf{U}$ is split, its sub-CUs of depth 1 are denoted as $\{\mathbf{U}_i\}_{i=1}^4$. Then, the second-level labels $\{y_2(\mathbf{U}_i)\}_{i=1}^4$ denote whether $\{\mathbf{U}_i\}_{i=1}^4$ are split (= 1) or not (= 0). For each split $\mathbf{U}_i$, its sub-CUs of depth 2 are represented by $\{\mathbf{U}_{i,j}\}_{j=1}^4$. Similarly, the labels $\{y_3(\mathbf{U}_{i,j})\}_{i,j=1}^4$ denote the split labels of $\{\mathbf{U}_{i,j}\}_{i,j=1}^4$ at the third level, and for each split $\mathbf{U}_{i,j}$, its sub-CUs of depth 3 are $\{\mathbf{U}_{i,j,k}\}_{k=1}^4$. The subscripts $i, j, k \in \{1, 2, 3, 4\}$ are the indices of the sub-CUs split from

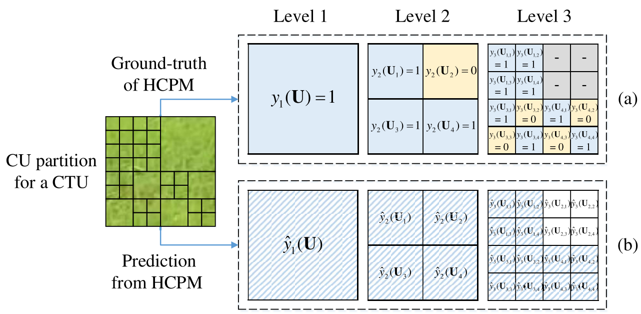


Fig. 3. An example of HCPM. (a) The ground-truth of HCPM for CU splitting. (b) Modeling HCPM for prediction the CU splitting.

$\mathbf{U}$, $\mathbf{U}_i$ and $\mathbf{U}_{i,j}$, respectively. The above hierarchical labels for the CU split are represented by the downward arrows in Fig. 2. The overall CU partition in a CTU is extremely complex, due to the large number of possible pattern combinations. For example, for a 64×64 $\mathbf{U}$, if $y_1(\mathbf{U}) = 1$, it will be split into four 32×32 CUs, i.e., $\{\mathbf{U}_i\}_{i=1}^4$. Since for each $\mathbf{U}_i$ there exist $1 + 2^4 = 17$ splitting patterns in $\{\mathbf{U}_{i,j}\}_{j=1}^4$, the total number of splitting patterns for $\mathbf{U}$ is $1 + 17^4 = 83522$.

In HEVC, the labels $\{y_l\}_{l=1}^3$ are obtained with the time-consuming RDO process, as mentioned in Section III-A. They can be predicted at a much faster speed by machine learning. However, due to the enormous patterns of the CU partition (83,522 patterns, as discussed above), it is intractable to predict the CU partition patterns by a simple multi-class classification in one step. Instead, the prediction should be made step by step in a hierarchy to yield $\hat{y}_1(\mathbf{U})$, $\{\hat{y}_2(\mathbf{U}_i)\}_{i=1}^4$ and $\{\hat{y}_3(\mathbf{U}_{i,j})\}_{i,j=1}^4$, which denote the predicted $y_1(\mathbf{U})$, $\{y_2(\mathbf{U}_i)\}_{i=1}^4$ and $\{y_3(\mathbf{U}_{i,j})\}_{i,j=1}^4$, respectively.

In typical machine learning based methods [8], [9], [24], [25], the binary labels $\hat{y}_1(\mathbf{U})$, $\{\hat{y}_2(\mathbf{U}_i)\}_{i=1}^4$ and $\{\hat{y}_3(\mathbf{U}_{i,j})\}_{i,j=1}^4$ are predicted separately for CUs of size 64×64, 32×32 and 16×16. To determine the CU partition for an entire CTU, the trained model needs to be invoked multiple times, leading to repetitive computation overhead. To overcome such a drawback, we develop an HCPM to efficiently represent CU splitting as the structured output of machine learning, such that the training model is run only once for predicting the partition of the whole CTU. This way, the computational time of determining the CU partition can be dramatically reduced. In addition, the encoding complexity can be decreased remarkably by bypassing the redundant RD cost checking in the conventional HEVC standard.

Fig. 3 shows an example of HCPM, which arranges the labels of the CU splitting as the structured output in a hierarchy. Specifically, HCPM hierarchically contains 1×1, 2×2 and 4×4 binary labels at levels 1, 2 and 3, respectively, corresponding to $y_1(\mathbf{U})$, $\{y_2(\mathbf{U}_i)\}_{i=1}^4$ and $\{y_3(\mathbf{U}_{i,j})\}_{i,j=1}^4$ for ground-truth and $\hat{y}_1(\mathbf{U})$, $\{\hat{y}_2(\mathbf{U}_i)\}_{i=1}^4$ and $\{\hat{y}_3(\mathbf{U}_{i,j})\}_{i,j=1}^4$ for prediction, respectively. Note that when $\mathbf{U}$ or $\mathbf{U}_i$ is not split, the corresponding sub-CUs $\{\mathbf{U}_i\}_{i=1}^4$ or $\{\mathbf{U}_{i,j}\}_{j=1}^4$ do not exist. Accordingly, the labels of $\{y_2(\mathbf{U}_i)\}_{i=1}^4$ or $\{y_3(\mathbf{U}_{i,j})\}_{j=1}^4$ are set to be null, denoted as "-" in the HCPM.

B. ETH-CNN Structure for Learning HCPM

Next, a deep ETH-CNN structure is developed to learn HCPM for predicting the CU partition of intra-mode HEVC. According to the mechanism of the CU partition, the ETH-CNN structure is designed in Fig. 4. We can see from this figure that ETH-CNN is fed with an entire CTU, denoted by $\mathbf{U}$, and is able to hierarchically generate three branches of structured output, representing all predictions $\hat{y}_1(\mathbf{U})$, $\{\hat{y}_2(\mathbf{U}_i)\}_{i=1}^4$ and $\{\hat{y}_3(\mathbf{U}_{i,j})\}_{i,j=1}^4$ of HCPM at three levels. Note that the input CTU is extracted from raw images or sequences in YUV format, and only the Y channel is used in ETH-CNN as this channel contains most visual information. In contrast to the conventional CNN structure, the mechanism of early termination is introduced in ETH-CNN, which may early terminate the three fully connected layers at the second and third branches. In addition to the three fully connected layers, the ETH-CNN structure is composed of two preprocessing layers, three convolutional layers, and one concatenating layer. In the following, we briefly present the details of these layers.

- **Preprocessing layers.** The raw CTU is preprocessed by mean removal and down-sampling in three parallel branches $\{\mathrm{B}_l\}_{l=1}^3$, corresponding to three levels of HCPM. Fig. 4 denotes mean removal and down-sampling by $\{N_l\}_{l=1}^3$ and $\{S_l\}_{l=1}^3$, respectively. For mean removal, at each branch, the input CTUs are subtracted by the mean intensity values to reduce the variation of the input CTU samples. Specifically, at branch B_1, the mean value of $\mathbf{U}$ is removed in accordance with the single output of $\hat{y}_1(\mathbf{U})$ at the first level of HCPM. At branch B_2, four CUs $\{\mathbf{U}_i\}_{i=1}^4$ are subtracted by their corresponding mean values, matching the 2×2 output of $\{\hat{y}_2(\mathbf{U}_i)\}_{i=1}^4$ at the second level of HCPM. Similarly, $\{\mathbf{U}_{i,j}\}_{i,j=1}^4$ at B_3 also remove the mean values in each CU for the 4×4 output $\{\hat{y}_3(\mathbf{U}_{i,j})\}_{i,j=1}^4$ at the third level of HCPM. Next, because CTUs of smaller depths generally possess a smooth texture, $\mathbf{U}$ and $\{\mathbf{U}_i\}_{i=1}^4$ at branches B_1 and B_2 are down-sampled to 16×16 and 32×32, respectively. This way, the outputs of the three-layer convolutions at branches B_1 and B_2 are with the same sizes of HCPM at levels 1 and 2, i.e., 1×1 at level 1 and 2×2 at level 2.
- **Convolutional layers.** In each branch B_l, all preprocessed data flow through three convolutional layers. At each convolutional layer, the kernel sizes are the same across different branches. Specifically, the data are convoluted with 4×4 kernels (16 filters in total) at the first convolutional layer to extract the low-level features $\{C_{1-l}\}_{l=1}^3$ for the CU partition. Recall that l denotes the level of the CU partition. At the second and third layers, feature maps are sequentially convoluted twice with 2×2 kernels (24 filters for the second layer and 32 filters for the third layer) to generate features at a higher level, denoted by $\{C_{2-l}\}_{l=1}^3$ and $\{C_{3-l}\}_{l=1}^3$, respectively. We set the strides of all the above convolutions equal to the widths of the corresponding kernels for non-overlap convolutions, in accordance with non-overlap CU splitting.
- **Concatenating layer.** All feature maps at three branches, yielded from the second and third convolutional layers, are concatenated together and then flattened into a vector $\mathbf{a}$. As shown in Fig. 4, the vectorized features of the concatenating layer are gathered from 6 sources of feature maps in total, i.e., $\{C_{2-l}\}_{l=1}^3$ and $\{C_{3-l}\}_{l=1}^3$, to obtain a variety of both global and local features. With the whole

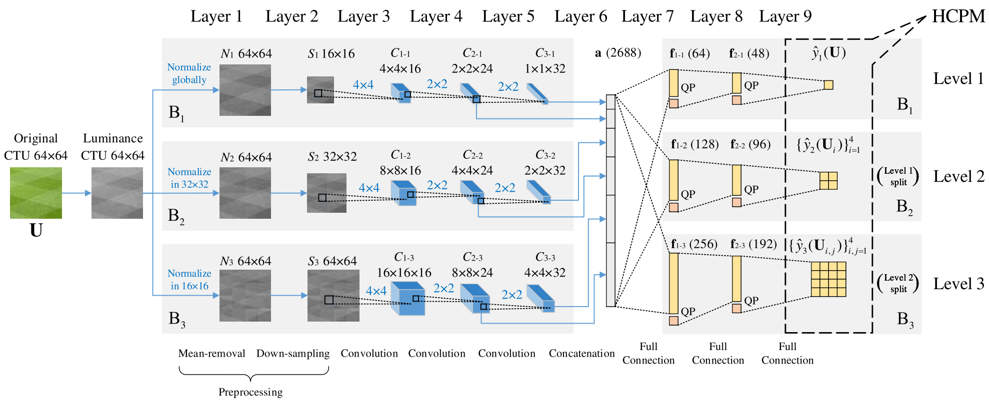


Fig. 4. Structure of ETH-CNN. For preprocessing and convolution, the size of the output at each layer is shown in black font, and the blue font indicates the kernel size for convolution. At concatenation and fully connected layers, the number inside each bracket represents the vector length.

concatenating vector, features generated from the whole CTU are all considered in the following fully connected layers, rather than those only from the CUs in one branch, to predict the CU partition of HCPM at each single level.

- **Fully connected layers.** Next, all features in the concatenated vector are processed in three branches $\{\mathrm{B}_l\}_{l=1}^{3}$, also corresponding to three levels of HCPM. In each branch, the vectorized features of the concatenating layer pass through three fully connected layers, including two hidden layers and one output layer. The two hidden fully connected layers successively generate feature vectors $\{\mathbf{f}_{1-l}\}_{l=1}^{3}$, and the output layer produces HCPM as the output of ETH-CNN. The numbers of features at these layers vary with respect to l, so that the outputs have 1, 4 and 16 elements in B_1, B_2 and B_3, serving as the predicted binary labels of HCPM at the three levels shown in Fig. 3 (b), i.e., $\hat{y}_1(\mathbf{U})$ in 1×1, $\{\hat{y}_2(\mathbf{U}_i)\}_{i=1}^{4}$ in 2×2 and $\{\hat{y}_3(\mathbf{U}_{i,j})\}_{i,j=1}^{4}$ in 4×4. Moreover, QP also has a significant influence on the CU partition. As QP decreases, more CUs tend to be split, and vice versa. Therefore, QP is supplemented as an external feature in the feature vectors $\{\mathbf{f}_{1-l}\}_{l=1}^{3}$ for the full connection, enabling ETH-CNN to be adaptive to various QPs in predicting HCPM. In ETH-CNN, the early termination may result in the calculation of the fully connected layers at levels 2 and 3 being skipped, thus saving computation time. Specifically, if $\mathbf{U}$ is decided not to be split at level 1, the calculation of $\{\hat{y}_2(\mathbf{U}_i)\}_{i=1}^{4}$ of HCPM is terminated early at level 2. If $\{\mathbf{U}_i\}_{i=1}^{4}$ are all not split, the $\{\hat{y}_3(\mathbf{U}_{i,j}, t)\}_{i,j=1}^{4}$ at level 3 do not need to be calculated for the early termination.
- **Other layers.** During the training phase, after the first and second fully connected layers, features are randomly dropped out [40] with probabilities of 50% and 20%, respectively. It is worth mentioning that all convolutional layers and hidden fully connected layers are activated with rectified linear units (ReLU) [41]. Moreover, all the output layers in $\{\mathrm{B}_l\}_{l=1}^{3}$ are activated with the sigmoid function, since all the labels in HCPM are binary.

The specific configuration of the ETH-CNN structure is presented in Table I, which lists the numbers of trainable parameters for obtaining different feature maps and vectors. We can see from this table that there are in total 1,287,189 trainable parameters for ETH-CNN. Thus, the ETH-CNN structure provides a much higher learning capacity, in comparison to the only 1,224 trainable parameters in [9], which may cause under-fitting issues. Such abundant trainable parameters also benefit from the extensive training samples of our CPH-Intra database (see Table I-(a) of the *Supporting Document* for the number of training samples). Another major merit is that all structured labels of HCPM are learned in the ETH-CNN model with shared parameters, to predict $y_1(\mathbf{U})$, $\{y_2(\mathbf{U}_i)\}_{i=1}^{4}$ and $\{y_3(\mathbf{U}_{i,j})\}_{i,j=1}^{4}$. This design efficiently reduces the overall computational complexity for predicting the CU partition, compared with conventional learning-based approaches [8], [9], [24], [25] that sequentially predict splitting labels from 64×64 CU partition to 16×16 CU partition.

C. Loss Function for Training ETH-CNN Model

Given the above ETH-CNN structure, we concentrate on the loss function for training the ETH-CNN model, which is used to predict HCPM. Assume that there are R training samples, the HCPM labels of which are $\{y_1^r(\mathbf{U}), \{y_2^r(\mathbf{U}_i)\}_{i=1}^{4}$ and $\{y_3^r(\mathbf{U}_{i,j})\}_{i,j=1}^{4}\}_{r=1}^{R}$. For each sample, the loss function L_r sums the cross-entropy over all valid elements of HCPM as follows:

$$L_r = H(y_1^r(\mathbf{U}), \hat{y}_1^r(\mathbf{U})) + \sum_{\substack{i \in \{1,2,3,4\} \\ y_2^r(\mathbf{U}_i) \neq \text{null}}} H(y_2^r(\mathbf{U}_i), \hat{y}_2^r(\mathbf{U}_i)) + \sum_{\substack{i,j \in \{1,2,3,4\} \\ y_3^r(\mathbf{U}_{i,j}) \neq \text{null}}} H(y_3^r(\mathbf{U}_{i,j}), \hat{y}_3^r(\mathbf{U}_{i,j})), \quad (1)$$

where $\{\hat{y}_1^r(\mathbf{U}), \{\hat{y}_2^r(\mathbf{U}_i)\}_{i=1}^{4}$ and $\{\hat{y}_3^r(\mathbf{U}_{i,j})\}_{i,j=1}^{4}\}_{r=1}^{R}$ are the labels of HCPM predicted by ETH-CNN. In (1), $H(\cdot, \cdot)$ denotes the cross-entropy between the ground-truth and predicted labels. Considering that some ground-truth labels in HCPM do not exist (such as $\{y_3(\mathbf{U}_{2,j})\}_{j=1}^{4}$ in Fig. 3), only valid labels with $y_2^r(\mathbf{U}_i) \neq$ null and $y_3^r(\mathbf{U}_{i,j}) \neq$ null are counted in the loss of (1).

Then, the ETH-CNN model can be trained by optimizing the loss function over all training samples:

$$L = \frac{1}{R} \sum_{r=1}^{R} L_r. \quad (2)$$

Because the loss function of (1) and (2) is the sum of the cross-entropies, the stochastic gradient descent algorithm with momentum is applied to train the ETH-CNN model. Finally, the trained ETH-CNN model can be used to predict the CU partition of HEVC in the form of HCPM.

D. Bi-Threshold Decision Scheme

Given an input CTU, ETH-CNN yields the probabilities for the binary labels of $\hat{y}_1(\mathbf{U}), \hat{y}_2(\mathbf{U}_i)$ and $\hat{y}_3(\mathbf{U}_{i,j})$, to predict the CU partition. Let $P_1(\mathbf{U})$, $P_2(\mathbf{U}_i)$ and $P_3(\mathbf{U}_{i,j})$ denote the probabilities of $\hat{y}_1(\mathbf{U}) = 1, \hat{y}_2(\mathbf{U}_i) = 1$ and $\hat{y}_3(\mathbf{U}_{i,j}) = 1$. Typically, one decision threshold α_l ($= 0.5$ in this paper) is set for each level l of HCPM. Specifically, a CU with $P_1(\mathbf{U}) > \alpha_1$ is decided as being split into four CUs $\{\mathbf{U}_i\}_{i=1}^{4}$; otherwise, it is not split. Similarly, $P_2(\mathbf{U}_i) > \alpha_2$ and $P_3(\mathbf{U}_{i,j}) > \alpha_3$ make a positive decision on the splitting of $\mathbf{U}_i$ and $\mathbf{U}_{i,j}$. This remarkably reduces the encoding complexity by avoiding all redundant RD cost comparison.

On the other hand, a false prediction of $y_1(\mathbf{U})$, $y_2(\mathbf{U}_i)$ or $y_3(\mathbf{U}_{i,j})$ leads to RD degradation, as the optimal RD performance is not achieved. To ameliorate this issue, we adopt a bi-threshold CU decision scheme [11]. For each level of HCPM, a splitting threshold α_l and a non-splitting threshold $\bar{\alpha}_l$ are set ($\bar{\alpha}_l \leq \alpha_l$). Then, the false prediction rate of the ground-truth labels is reduced by introducing an uncertain zone $[\bar{\alpha}_l, \alpha_l]$, where a parent CU (e.g., $\mathbf{U}, \mathbf{U}_i$ or $\mathbf{U}_{i,j}$) and its four sub-CUs are all checked with the RD cost. Generally speaking, if the uncertain zone becomes wider, more CUs tend to be checked, such that the encoding complexity is increased with decreased misclassification of the CU splitting. Therefore, a trade-off between RD performance and encoding complexity can be achieved through the bi-threshold of $[\bar{\alpha}_l, \alpha_l]$.

The optimal thresholds of the uncertain zone $[\bar{\alpha}_l, \alpha_l]$ can be obtained by traversing the combinations of $\{[\bar{\alpha}_l, \alpha_l]\}_{l=1}^{3}$ to reduce the encoding complexity as much as possible at the constraint of the RD degradation. However, in practice, both complexity reduction and RD degradation vary in a large extent because of different encoding requirements and diverse video contents. Thus, it is hard to find one specific threshold combination of $\{[\bar{\alpha}_l, \alpha_l]\}_{l=1}^{3}$ fitting all encoding requirements and video sequences. Instead, for the better complexity-RD (CRD) performance, our bi-threshold scheme follows [8] on account of the setting that the uncertain zones of the CU partition at lower levels should be narrower. Similar to [8], when the RD degradation is constrained in complexity reduction, the uncertain zones at lower levels of the CU partition in our approach are supposed to be narrower, i.e., $[\bar{\alpha}_1, \alpha_1] \subset [\bar{\alpha}_2, \alpha_2] \subset [\bar{\alpha}_3, \alpha_3]$. In addition,the upper and lower thresholds of our bi-threshold scheme are set based on an assumption that the upper and lower thresholds are symmetric with regard to 0.5, i.e., $\bar{\alpha}_l = 1 - \alpha_l$ for each level l. Note that 0.5 is a widely used threshold for the conventional binary classification problems. Given this assumption, the settings of thresholds can be simplified from six variables to three variables for achieving the CRD trade-off. Moreover, this assumption is reasonable, because the symmetry of the upper and lower thresholds at 0.5 balances the prediction accuracy for both split and non-split CUs.

E. Computational Complexity of ETH-CNN

In this section, we analyze the time complexity of ETH-CNN via counting the number of floating-point operations, including additions and multiplications. Note that all floating-point operations in ETH-CNN are in single precision (32-bit). The last two columns of Table I report the numbers of floating-point operations for each layer of ETH-CNN and the total number of floating-point operations required in ETH-CNN. We can see that ETH-CNN performs a total of 3.05×10^6 floating-point operations, including 1,497,584 additions and 1,552,149 multiplications. ETH-CNN performs fewer floating-point operations than Alex-Net ($\sim 3 \times 10^9$ floating-point operations) or VGG-Net ($\sim\ 4 \times 10^{10}$ floating-point operations) by at least three orders of magnitude.

In Table I, early termination is not considered for counting the total number of floating-point operations of ETH-CNN. In practical compression, the early termination of ETH-CNN can further reduce the time complexity. Here, we record the average number of floating-point operations of ETH-CNN, when compressing 18 standard JCT-VC sequences [38] at intra-mode and QP $=$ 32. We find that 12.6% floating-point operations can be saved due to the early termination mechanism in ETH-CNN. Consequently, for compressing 1080p@30 Hz sequences, our approach requires $\sim$ 40.8 G single-precision floating-point operations per second (FLOPS) for intra-mode HEVC complexity reduction. The FLOPS of our approach is far less than the computational capacity of an Intel(R) Core(TM) i7-7700K CPU @4.2 GHz, which supports $\sim$ 287.9 G double-precision FLOPS, as reported in [42]. In the *Supporting Document*, time complexity of our approach on field programmable gate array (FPGA) implementation is also analyzed. As analyzed, ETH-CNN only consumes less than 3 ms for one frame of 1080p sequences in the FPGA of Virtex UltraScale+ VU13P.

V. Complexity Reduction for Inter-Mode HEVC

In this section, we first analyze the temporal correlation of the CU depth in inter-mode HEVC. Based on our analysis, we then propose to predict the CU partition of inter-mode HEVC, via designing the ETH-LSTM network. Finally, the computational complexity of ETH-LSTM is analyzed.

A. Analysis on Temporal Correlation of CU Depth

Typically, the adjacent video frames exhibit similarity in video content, and such similarity decays as the temporal distance of two adjacent frames increases. The same phenomenon may also hold for the CU partition of inter-mode HEVC. Fig. 5 shows examples of the CU partition across adjacent frames with different temporal distances. In this figure, we consider the first column as the reference. Then, the CU partition of subsequent frames with different distances is compared to the reference. In each CU, we highlight the same CU depth in red. We can see from Fig. 5 that there exists high similarity in the CU partition across neighboring frames, and the similarity drops along with increasing temporal distance.

We further measure the similarity of the CU partition in inter-mode HEVC over all 93 sequences in the training and validation sets of our CPH-Inter database. Specifically, we calculate the correlation of the CU depth between two frames at various distances, ranging from 1 group of pictures (GOP) to 25 GOPs. Such correlation is measured between co-located

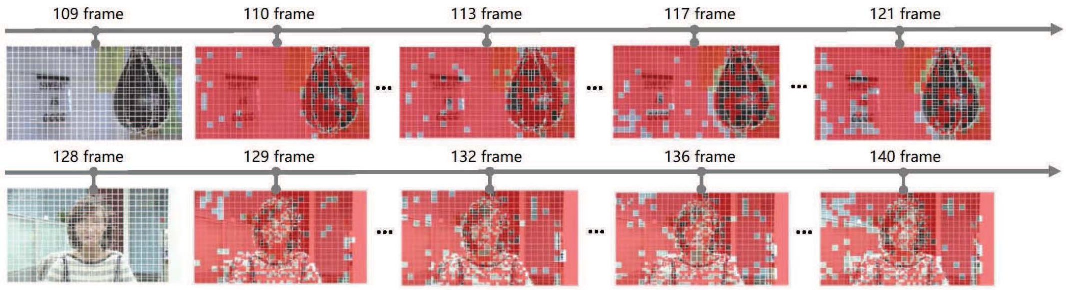


Fig. 5. Two examples illustrating temporal CU partition correlation. Red patches represent CUs of the same depths as in the reference frame.

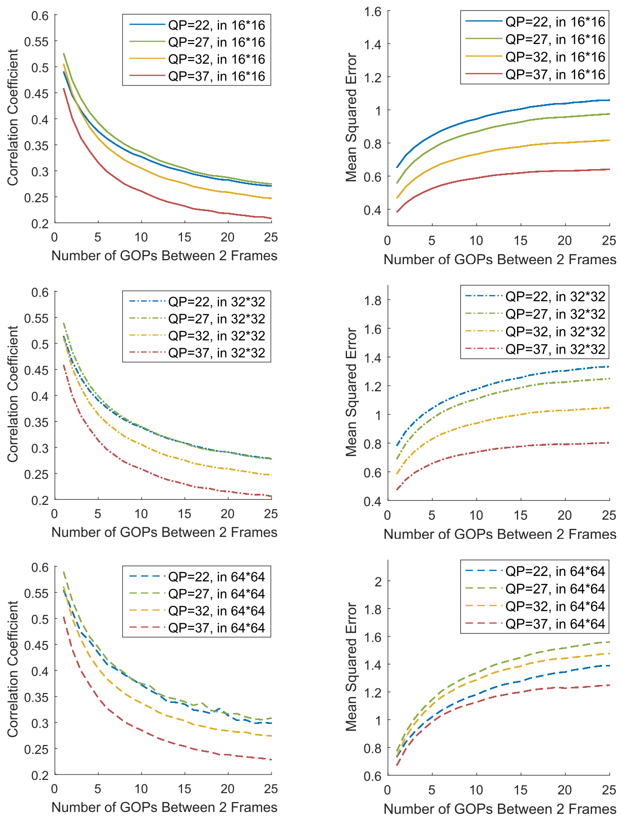


Fig. 6. Temporal CU depth correlation versus distances of two frames for inter-mode HEVC at four QP values. Left: Correlation coefficient. Right: Mean squared error.

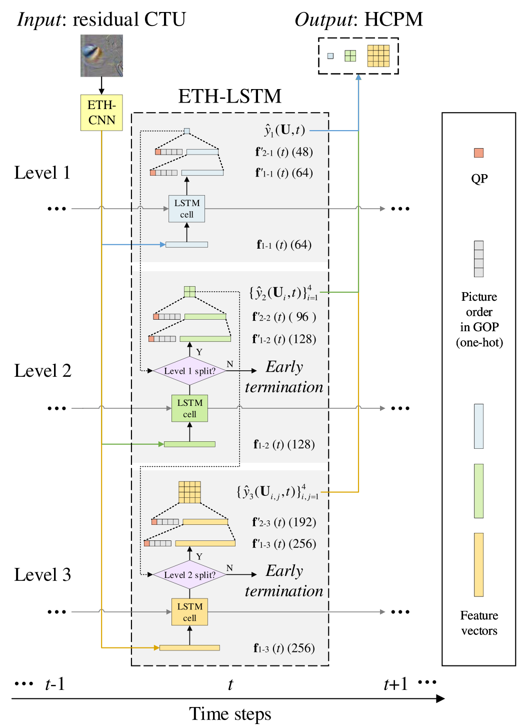


Fig. 7. Framework of ETH-LSTM. The number in each bracket is the dimension of the vector.

units[1] from two frames in terms of the linear correlation coefficient (CC) and mean squared error (MSE). In our analysis, the results for the CC and MSE are averaged over all frames of 93 sequences, which are shown in Fig. 6 for four QPs (QP $=$ 22, 27, 32 and 37).

We can see from Fig. 6 that CC is always much larger than 0, indicating the existence of a positive correlation on the temporal CU depth. Moreover, the CC decreases alongside increasing distance between two frames. Similar results hold for MSE, as can be found in Fig. 6. Therefore, Fig. 6 implies that there exist long- and short-term dependencies of the CU partition across adjacent frames for inter-mode HEVC.

[1]Non-overlapping units with sizes of 64×64, 32×32 and 16×16 are considered, corresponding to splitting depths of 0, 1 and 2. Note that 8×8 units are not measured, as an 8×8 CU is definitely split from a larger CU.

B. Deep LSTM Structure

As analyzed in Section V-A, the CU partition of neighboring frames is correlated with each other. Thus, this section proposes the ETH-LSTM approach, which learns the long- and short-term dependencies of the CU partition across frames. Fig. 7 illustrates the overall framework of ETH-LSTM. As observed in Fig. 7, the input to ETH-LSTM is the residue of each CTU. Here, the residue is obtained by precoding the currently processed frame, in which both the CU and PU sizes are forced to be 64×64. It is worth mentioning that the computational time for precoding consumes less than 3% of the total encoding time, which is considered in our approach as time overhead. Then, the residual CTU is fed into ETH-CNN. For inter-frames, the parameters of ETH-CNN are re-trained over the residue and ground-truth splitting of the training CTUs from the CPH-Inter database. Next, the features $\{\mathbf{f}_{1-l}\}_{l=1}^{3}$ of ETH-CNN are extracted for frame t, and these features are then fed into ETH-LSTM. Recall that $\{\mathbf{f}_{1-l}\}_{l=1}^{3}$ are the features at Layer 7 of ETH-CNN.

Fig. 7 shows that three LSTMs in ETH-LSTM are arranged in a hierarchy for determining the CU depths in the form of HCPM. Specifically, the LSTM cells are at levels 1, 2 and 3, corresponding to three levels of HCPM: $\hat{y}_1(\mathbf{U}, t)$, $\{\hat{y}_2(\mathbf{U}_i, t)\}_{i=1}^4$ and $\{\hat{y}_3(\mathbf{U}_{i,j}, t)\}_{i,j=1}^4$. Here, $\hat{y}_1(\mathbf{U}, t)$ indicates whether the CU $\mathbf{U}$ (size: 64×64, depth $= 0$) at frame t is split. Similarly, $\{\hat{y}_2(\mathbf{U}_i, t)\}_{i=1}^4$ and $\{\hat{y}_3(\mathbf{U}_{i,j}, t)\}_{i,j=1}^4$ denote the splitting labels of CUs $\mathbf{U}_i$ (size: 32×32, depth $= 1$) and $\mathbf{U}_{i,j}$ (size: 16×16, depth $= 2$). At each level, two fully connected layers follow the LSTM cells, which also include the QP value and the order of frame t at GOP. Note that the frame order is represented by a one-hot vector. For level l at frame t, we denote $\mathbf{f}'_{1-l}(t)$ and $\mathbf{f}'_{2-l}(t)$ as the output features of the LSTM cell and the first fully connected layer. Additionally, the output of the second fully connected layer is the probabilities of CU splitting, which are binarized to predict HCPM.[2] However, if the LSTM cell at level 1 decides not to split, the calculation of $\{\hat{y}_2(\mathbf{U}_i, t)\}_{i=1}^4$ of HCPM is terminated early at level 2, via skipping its two fully connected layers. Instead, all indices of $\{\hat{y}_2(\mathbf{U}_i, t)\}_{i=1}^4$ are set to be 0, meaning that the CU depths at level 2 are 0. The calculation of $\{\hat{y}_3(\mathbf{U}_{i,j}, t)\}_{i,j=1}^4$ at level 3 is also terminated early in a similar manner. Consequently, early termination is able to reduce the computational time of ETH-LSTM. Finally, the results of HCPM are output by ETH-LSTM, therein predicting the partition patterns of a CTU at frame t.

When determining the HCPM of each CTU, the partition patterns of the CTUs co-located at the previous frames can be considered in ETH-LSTM. ETH-LSTM learns the long- and short-term correlations of CU depths across frames, via incorporating the LSTM cells at different levels. For learning ETH-LSTM, each LSTM cell is trained separately using its corresponding CUs. Next, we introduce the learning mechanism of ETH-LSTM by taking the LSTM cell of frame t at level l as an example. Such an LSTM cell works with three gates: the input gate $\mathbf{i}_l(t)$, the output gate $\mathbf{o}_l(t)$ and the forget gate $\mathbf{g}_l(t)$. Given the input ETH-CNN feature $\mathbf{f}_{1-l}(t)$ and the output feature $\mathbf{f}'_{1-l}(t-1)$ of the LSTM cell at the last frame, these three gates can be obtained by

$$\begin{cases} \mathbf{i}_l(t) = \sigma(\mathbf{W}_i \cdot [\mathbf{f}_{1-l}(t), \mathbf{f}'_{1-l}(t-1)] + \mathbf{b}_i) \\ \mathbf{o}_l(t) = \sigma(\mathbf{W}_o \cdot [\mathbf{f}_{1-l}(t), \mathbf{f}'_{1-l}(t-1)] + \mathbf{b}_o) \\ \mathbf{g}_l(t) = \sigma(\mathbf{W}_f \cdot [\mathbf{f}_{1-l}(t), \mathbf{f}'_{1-l}(t-1)] + \mathbf{b}_f) \end{cases} \tag{3}$$

where $\sigma(\cdot)$ stands for the sigmoid function. In the above equations, $\mathbf{W}_i$, $\mathbf{W}_o$, $\mathbf{W}_f$ are trainable parameters for three gates, and $\mathbf{b}_i$, $\mathbf{b}_o$ and $\mathbf{b}_f$ are their biases. With these three gates, the LSTM cell updates its state at frame t as

$$\mathbf{c}_l(t) = \mathbf{i}_l(t) \odot \tanh(\mathbf{W}_c \odot [\mathbf{f}_{1-l}(t), \mathbf{f}'_{1-l}(t-1)] + \mathbf{b}_c) + \mathbf{g}_l(t) \odot \mathbf{c}_l(t-1), \tag{4}$$

where $\odot$ represents element-wise multiplication. In (4), $\mathbf{W}_c$ and $\mathbf{b}_c$ are parameters and biases for $\mathbf{c}_l(t)$, which need to be trained. Finally, the output of the LSTM cell $\mathbf{f}'_{1-l}(t)$ can be calculated as follows:

$$\mathbf{f}'_{1-l}(t) = \mathbf{o}_l(t) \odot \mathbf{c}_l(t). \tag{5}$$

[2]The binarization process is the same as that of ETH-CNN in Section IV-B. The bi-threshold decision scheme of Section IV-D can also be applied for the binarization process to achieve a trade-off between prediction accuracy and complexity reduction.

TABLE II

CONFIGURATION OF ETH-LSTM

Feature	Size	Number of Parameters	Number of Additions	Number of Multiplications
$\mathbf{i}_1/\mathbf{o}_1/\mathbf{g}_1(t)$	64	8,192	8,128	8,192
$\mathbf{i}_2/\mathbf{o}_2/\mathbf{g}_2(t)$	128	32,768	32,640	32,768
$\mathbf{i}_3/\mathbf{o}_3/\mathbf{g}_3(t)$	256	131,072	130,816	131,072
$\mathbf{c}_1(t)$	64	8,192	8,255	8,320
$\mathbf{c}_2(t)$	128	32,768	32,895	33,024
$\mathbf{c}_3(t)$	256	131,072	131,327	131,584
$\mathbf{f}'_{1-1}(t)$	64	0	63	64
$\mathbf{f}'_{1-2}(t)$	128	0	127	128
$\mathbf{f}'_{1-3}(t)$	256	0	255	256
$\mathbf{f}'_{2-1}(t)$	48	3,312	3,264	3,312
$\mathbf{f}'_{2-2}(t)$	96	12,768	12,672	12,768
$\mathbf{f}'_{2-3}(t)$	192	50,112	49,920	50,112
$\hat{y}_1(\mathbf{U}, t)$	1	53	52	53
$\{\hat{y}_2(\mathbf{U}_i, t)\}_{i=1}^4$	4	404	400	404
$\{\hat{y}_3(\mathbf{U}_{i,j}, t)\}_{i,j=1}^4$	16	3,152	3,136	3,152
Total	-	757,929	757,118	759,273

Note that both $\mathbf{c}_l(t)$ and $\mathbf{f}'_{1-l}(t)$ are vectors with the same length as $\mathbf{f}_{1-l}(t)$.

The configuration of ETH-LSTM with all trainable parameters is listed in Table II. For training the parameters of the LSTM cells in the above equations, the cross-entropy is applied as the loss function, the same as that for ETH-CNN defined in (1). Let $L_r(t)$ be the loss for the r-th training CU sample at frame t. Then, each LSTM cell among the three levels of ETH-LSTM can be trained by optimizing the loss over all R training samples alongside T frames,

$$L = \frac{1}{RT} \sum_{r=1}^{R} \sum_{t=1}^{T} L_r(t), \tag{6}$$

which is solved by the stochastic gradient descent algorithm with momentum. Finally, given the trained LSTM cells, ETH-LSTM can be used to generate HCPM for predicting the CU partition at the inter-frames.

C. Computational Complexity of ETH-LSTM

As shown in the last two columns of Table II, there are in total 757,118 additions and 759,273 multiplications for ETH-LSTM, almost half of those for ETH-CNN. With the early termination, the total number of floating-point operations in Table II can be further reduced by 4.3% when running ETH-LSTM for compressing 18 standard JCT-VC sequences at QP = 32 (the inter-mode configuration is the same as that in the section of experimental results). Because ETH-LSTM requires the extracted features of ETH-CNN as the input, our complexity reduction approach consumes in total 4.37×10^6 single-precision floating-point operations for a CTU. In other words, our approach consumes 66.9 G single-precision FLOPS for 1080p@30 Hz sequences, much less than the 287.9 G double-precision FLOPS for an Intel(R) Core(TM) i7-7700K CPU @4.2 GHz. As analyzed in the *Supporting Document*, our approach requires less than 3 ms/frame running time in the FPGA device of Virtex UltraScale+ VU13P.

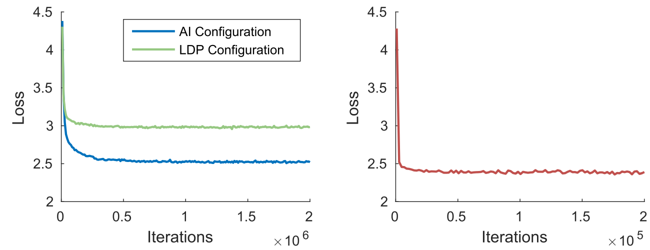
VI. EXPERIMENTAL RESULTS

In this section, we present experimental results to validate the effectiveness of our approach in reducing HEVC complexity at both intra- and inter-modes. For evaluating performance at intra-mode, we compare our approach with two state-of-the-art approaches: the SVM based approach [8] and the CNN based approach [9]. For evaluating performance at inter-mode, we compare our approach to the latest approaches of [8], [7] and [10]. In Section VI-A, we discuss the configuration of the experiments and the settings of our approach. In Sections VI-B and VI-C, we compare the performance of our and other approaches for intra- and inter-mode HEVC, respectively. Finally, Section VI-D analyzes the running time of our complexity reduction approach.

A. Configuration and Settings

1) Configuration of Experiments: In our experiments, all complexity reduction approaches were implemented in the HEVC reference software HM 16.5 [3]. In HM 16.5, the AI configuration was applied with the default configuration file *encoder_intra_main.cfg* [36] for the performance evaluation at intra-mode, and the LDP configuration was used with the default configuration file *encoder_low_delay_P_main.cfg* [36] for the performance evaluation at inter-mode. Here, four QP values, $\{22, 27, 32, 37\}$, were chosen to compress the images and video sequences. At both intra- and inter-modes, our experiments were tested on 18 video sequences of the JCT-VC standard test set [38]. Moreover, all 200 test images from the CPH-Intra database were evaluated for intra-mode HEVC complexity reduction. In our experiments, the Bjøntegaard delta bit-rate (BD-BR) and Bjøntegaard delta PSNR (BD-PSNR) [43] were measured to assess the RD performance. Additionally, ΔT, which denotes the encoding time-saving rate over the original HM, was used to measure the complexity reduction. All experiments were conducted on a computer with an Intel(R) Core (TM) i7-7700K CPU @4.2 GHz, 16 GB RAM and the Windows 10 Enterprise 64-bit operating system. Note that a GeForce GTX 1080 GPU was used to accelerate the training speed, but it was disabled when testing the HEVC complexity reduction.

2) Training Settings: In our complexity reduction approach, two ETH-CNN models were trained from the training sets of the CPH-Intra and CPH-Inter databases, corresponding to intra- and inter-mode HEVC, respectively. In addition, one ETH-LSTM model was trained from the training set of the CPH-Inter database. In training these models, the hyper-parameters were tuned on the validation sets of the CPH-Intra and CPH-Inter databases. Specifically, all trainable parameters in ETH-CNN and ETH-LSTM were randomly initialized, obeying the truncated normal distribution with zero mean and standard deviation of 0.1. The batch size R for training was 64, and the momentum of the stochastic gradient descent algorithm was set to 0.9. To train the ETH-CNN models, the initial learning rate was 0.01 and decreased by 1% exponentially every 2,000 iterations, and there were in total 1,000,000 iterations. To train ETH-LSTM, the initial learning rate was 0.1 and decreased by 1% exponentially every 200 iterations, and the total number of iterations was 200,000. The length of the trained ETH-LSTM model was $T = 20$. Moreover, at the LDP configuration, the overlapping of 10 frames was applied to generate the training samples of the ETH-LSTM model. Such overlapping was introduced to enhance the diversity of the training samples and meanwhile augment the training data.



(a) ETH-CNN (b) ETH-LSTM

Fig. 8. Training loss alongside iterations when learning the models of ETH-CNN at intra-mode, ETH-CNN at inter-mode and ETH-LSTM. Note that the training loss of intra-mode ETH-CNN is over the training set of CPH-Intra database, while the training loss of inter-mode ETH-CNN and ETH-LSTM is over the training set of CPH-Inter database.

3) Test Settings: For the test, T was set equal to the number of P frames in the sequence, where the internal information of ETH-LSTM can propagate throughout the whole sequence to leverage the long- and short-term dependencies. Note that ETH-LSTM is invoked step-wise during the test, i.e., the hidden state and the output of frame t are calculated after the processing of frame $t-1$ is accomplished, instead of yielding the output of all T frames at one time. For the bi-threshold decision scheme, at intra-mode the values of the thresholds were $\bar{\alpha}_l = \alpha_l = 0.5$ where $l \in \{1, 2, 3\}$ is the CU depth. It is because our approach is able to achieve sufficiently small loss of RD performance at intra-mode and the bi-threshold decision is not necessary. At inter-mode, we set $[\bar{\alpha}_1, \alpha_1] = [0.4, 0.6]$, $[\bar{\alpha}_2, \alpha_2] = [0.3, 0.7]$ and $[\bar{\alpha}_3, \alpha_3] = [0.2, 0.8]$, corresponding to the different CU depth of $l \in \{1, 2, 3\}$. Here, we followed [8] to apply this bi-threshold setting, which can efficiently enhance RD performance with desirable rate of complexity reduction. The width of uncertain zone increases along with increased level l of CU depth, since the smaller CU depth tends to early bypass redundant checking of CUs in RD optimization (RDO), which is achieved by a narrower uncertain zone $[\bar{\alpha}_l, \alpha_l]$.

B. Performance Evaluation at Intra-Mode

1) Evaluation on Training Performance: When predicting the CU partition of intra-mode HEVC, our approach relies on the training model of ETH-CNN, which is optimized by (2) over the training set. Therefore, it is necessary to evaluate the training performance of our approach at intra-mode. Fig. 8-(a) shows the training loss along with iterations during the model training of ETH-CNN. Note that the training loss is obtained through calculating (2) at each training iteration. We can see from this figure that the loss converges after 5×10^5 iterations, when training the ETH-CNN model at the AI configuration.

2) Evaluation on Prediction Accuracy: First, we evaluate the accuracy of the CU partition predicted by our ETH-CNN approach, when compressing images/sequences with intra-mode HEVC. In our experiments, the accuracy is averaged over the CU partition results of 200 test images and 18 test sequences, which are compressed by HM 16.5 at the

TABLE III

RESULTS FOR IMAGES FROM OUR CPH-INTRA TEST SET (AI)

Image Source	Resolution	Appr.	BD-BR (%)	BD-PSNR (dB)	ΔT (%) QP=22	QP=27	QP=32	QP=37
CPH-Intra Test Set	768 × 512	[25]	5.113	-0.343	**-59.43**	-54.70	-48.74	-44.83
		[9]	2.885	-0.210	-54.97	**-58.78**	**-61.78**	**-64.41**
		Our	**0.935**	**-0.067**	-51.10	-55.16	-57.31	-62.18
	1536 × 1024	[25]	6.002	-0.374	-58.94	-54.85	-50.57	-50.95
		[9]	3.134	-0.208	-55.84	-59.46	-62.43	-64.17
		Our	**1.220**	**-0.081**	**-59.50**	**-60.78**	**-64.71**	**-66.76**
	2880 × 1920	[25]	4.035	-0.207	-57.03	-52.79	-52.31	-59.51
		[9]	2.130	-0.115	-59.95	-63.14	-68.07	-69.46
		Our	**1.661**	**-0.094**	**-70.91**	**-74.48**	**-76.06**	**-78.47**
	4928 × 3264	[25]	4.630	-0.209	-58.02	-62.74	-65.30	-67.46
		[9]	1.863	-0.086	-61.43	-65.27	-68.70	-71.00
		Our	**1.728**	**-0.082**	**-75.51**	**-76.96**	**-78.34**	**-78.68**
Std. dev.		[25]	0.720	0.076	**0.92**	3.82	6.51	8.57
		[9]	0.604	0.064	3.13	**3.07**	**3.64**	**3.48**
		Our	**0.376**	**0.011**	11.05	10.55	9.87	8.36
Best		[25]	4.035	-0.207	-59.43	-62.74	-65.30	-67.46
		[9]	1.863	-0.086	-61.43	-65.27	-68.70	-71.00
		Our	**0.935**	**-0.067**	**-75.51**	**-76.96**	**-78.34**	**-78.68**
Average		[25]	4.945	-0.284	-58.36	-56.27	-54.23	-55.69
		[9]	2.353	-0.155	-58.05	-61.66	-65.25	-67.26
		Our	**1.386**	**-0.081**	**-64.01**	**-66.09**	**-68.11**	**-70.52**

TABLE IV

RESULTS FOR SEQUENCES OF THE JCT-VC TEST SET (AI)

Class	Sequence	Appr.	BD-BR (%)	BD-PSNR (dB)	ΔT (%) QP=22	QP=27	QP=32	QP=37
A	*PeopleOnStreet*	[25]	9.627	-0.492	-52.12	-50.63	-37.79	-34.81
		[9]	3.969	-0.209	-50.79	-53.87	-56.58	-61.15
		Our	**2.366**	**-0.126**	**-59.16**	**-59.42**	**-61.63**	**-63.82**
	Traffic	[25]	6.411	-0.304	-37.11	-25.36	-19.63	-33.38
		[9]	4.945	-0.240	-53.86	-59.08	-63.54	-66.88
		Our	**2.549**	**-0.125**	**-66.48**	**-70.98**	**-71.85**	**-73.85**
B	*BasketballDrive*	[25]	8.923	-0.244	-48.53	-37.13	-41.05	-46.89
		[9]	6.018	-0.141	-68.50	-68.55	-70.30	-70.70
		Our	**4.265**	**-0.121**	**-74.26**	**-76.18**	**-76.93**	**-77.90**
	BQTerrace	[25]	6.627	-0.295	**-67.32**	-63.09	-59.57	-36.48
		[9]	4.815	-0.267	-53.35	-56.81	-60.26	-61.13
		Our	**1.842**	**-0.086**	-52.02	**-65.57**	**-68.25**	**-73.05**
	Cactus	[25]	7.533	-0.248	-38.37	-40.83	-43.61	-51.23
		[9]	6.021	-0.208	**-58.18**	**-61.01**	-64.94	-67.78
		Our	**2.265**	**-0.077**	-48.95	-53.87	**-66.92**	**-74.09**
	Kimono	[25]	5.212	-0.170	-34.51	-43.21	-49.88	-63.58
		[9]	**2.382**	**-0.082**	-70.66	-72.75	-73.62	-73.86
		Our	2.592	-0.087	**-80.40**	**-84.32**	**-84.45**	**-84.97**
	ParkScene	[25]	3.630	-0.149	-41.69	-44.79	-59.98	-64.92
		[9]	3.417	-0.135	-60.27	-65.10	-68.57	-70.16
		Our	**1.956**	**-0.081**	**-60.42**	**-66.15**	**-70.08**	**-73.47**
C	*BasketballDrill*	[25]	9.818	-0.439	-46.65	-58.86	-47.66	-62.53
		[9]	12.205	-0.538	**-60.29**	**-62.35**	**-64.48**	**-67.20**
		Our	**2.863**	**-0.133**	-42.47	-45.16	-62.11	-62.17
	BQMall	[25]	9.646	-0.486	-52.62	-42.97	-35.52	-37.12
		[9]	8.077	-0.468	-47.08	-51.15	-53.26	-57.05
		Our	**2.089**	**-0.111**	**-52.74**	**-57.05**	**-59.39**	**-64.48**
	PartyScene	[25]	7.383	-0.468	**-63.84**	-49.82	-31.50	-26.88
		[9]	9.448	-0.672	-52.72	**-57.51**	**-59.77**	**-64.98**
		Our	**0.664**	**-0.044**	-37.27	-41.94	-43.78	-55.00
	RaceHorses	[25]	7.220	-0.379	-46.46	-40.13	-41.49	-50.28
		[9]	4.422	-0.264	-50.52	**-59.30**	**-59.81**	**-63.15**
		Our	**1.973**	**-0.108**	**-54.08**	-54.47	-58.59	-61.32
D	*BasketballPass*	[25]	10.054	-0.546	-43.69	-41.03	-37.46	-36.69
		[9]	8.401	-0.457	**-60.24**	**-62.89**	**-64.31**	**-66.67**
		Our	**1.842**	**-0.106**	-54.87	-56.64	-56.93	-57.23
	BlowingBubbles	[25]	6.178	-0.373	**-57.15**	-42.45	-25.73	-22.81
		[9]	8.328	-0.463	-54.62	**-60.45**	**-62.55**	**-65.48**
		Our	**0.622**	**-0.039**	-38.08	-38.66	-39.52	-45.91
	BQSquare	[25]	12.342	-0.876	**-61.45**	**-62.40**	**-58.99**	-46.86
		[9]	2.563	-0.211	-42.55	-46.05	-48.37	**-49.89**
		Our	**0.913**	**-0.067**	-44.13	-45.08	-46.57	-47.48
	RaceHorses	[25]	8.839	-0.487	-43.12	-40.18	-39.54	-38.07
		[9]	4.953	-0.317	**-52.86**	**-57.32**	**-58.80**	**-60.20**
		Our	**1.320**	**-0.075**	-50.91	-54.17	-58.23	-59.72
E	*FourPeople*	[25]	9.077	-0.480	-53.52	-40.88	-26.12	-24.34
		[9]	8.002	-0.439	-54.79	-59.79	-64.39	-67.17
		Our	**3.110**	**-0.171**	**-65.30**	**-72.08**	**-72.99**	**-74.86**
	Johnny	[25]	12.182	-0.474	-58.29	-60.21	-63.98	-70.70
		[9]	7.956	-0.307	-62.92	-65.51	-67.71	-70.05
		Our	**3.822**	**-0.153**	**-70.32**	**-70.57**	**-70.73**	**-71.10**
	KristenAndSara	[25]	13.351	-0.627	-54.44	-56.94	-56.05	-62.61
		[9]	5.478	-0.265	-61.24	-64.61	-65.82	-67.21
		Our	**3.460**	**-0.169**	**-72.76**	**-74.53**	**-75.98**	**-76.17**
Std. dev.		[25]	2.553	0.175	9.42	10.16	13.04	15.01
		[9]	2.603	0.158	**7.10**	**6.24**	**6.09**	**5.63**
		Our	**1.020**	**0.039**	12.67	12.96	11.96	10.75
Best		[25]	3.630	-0.149	-67.32	-63.09	-63.98	-70.70
		[9]	2.382	-0.082	-70.66	-72.75	-73.62	-73.86
		Our	**0.622**	**-0.039**	**-80.40**	**-84.32**	**-84.45**	**-84.97**
Average		[25]	8.559	-0.419	-50.05	-46.72	-43.09	-45.01
		[9]	6.189	-0.316	-56.41	-60.23	-62.62	-65.04
		Our	**2.247**	**-0.104**	**-56.92**	**-60.38**	**-63.61**	**-66.47**

AI configuration and four QP values $\{22, 27, 32, 37\}$. We find from the experimental results that the average accuracy rates for predicting the intra-mode CU partition at levels 1, 2 and 3 iare 90.98%, 86.42% and 80.42%, respectively. These accuracy rates are rather high, thus leading to a slight loss of RD performance, as discussed below.

3) Comparison of Complexity Reduction: Next, we compare our approach with [25] and [9] in complexity reduction, when compressing images/sequences by HEVC at intra-mode. Tables III and IV tabulate the complexity reduction results in ΔT for 200 test images and 18 test sequences, respectively. As observed in Table III, at QP $= 22, 27, 32$ and 37, our approach reduces the encoding complexity by 64.01%, 66.09%, 68.11% and 70.52% on average, outperforming the 58.36%, 56.27%, 54.23% and 55.69% complexity reductions in [25] and the 58.05%, 61.66%, 65.25% and 67.26% complexity reductions in [9]. In addition, for most scenarios with different QPs and sequences, our approach achieves a better complexity reduction than the other two approaches. Note that our ETH-CNN based approach requires less time than the shallow CNN approach [9], because [9] requires an RDO search for the decision on splitting from 64×64 to 32×32. Our approach outperforms [25] and [9] in terms of maximal ΔT, for compressing the test images and video sequences with intra-mode HEVC. More importantly, when applying our approach and [9] in intra-mode compression of the test images and video sequences, the standard deviations of ΔT are less than $\frac{1}{5} \cdot |\Delta T|$, indicating the stability of both approaches in complexity reduction. For video sequences, similar results can be found in Table IV. From Tables III and IV, we can further observe that the gap in the time reduction between our and other approaches increases along with increasing QP. This is most likely because the increased QP leads to more large CUs, such that early termination in our ETH-CNN model works for more CTUs, resulting in decreased running time. In summary, our approach is capable of improving the time efficiency of intra-mode HEVC.

4) Comparison of RD Performance: The reduced complexity of intra-mode HEVC is at the expense of an RD performance loss. Here, we compare the RD performance of our and the other two approaches in terms of BD-BR and BD-PSNR. Tables III and IV report the BD-BR and BD-PSNR under the three approaches, with the original HM as the anchor. We can see from these tables that the BD-BR increase in our ETH-CNN approach is on average 1.386% for images and 2.247% for sequences, which significantly outperforms [25] (4.945% for images and 8.559% for sequences) and [9] (2.353% for images and 6.189% for sequences). In addition, our approach incurs -0.081 dB and -0.104 dB BD-PSNR for images and sequences, respectively, which are far better than the -0.284 dB and -0.419 dB of [25], and the -0.155 dB and -0.316 dB of [9]. Our approach is also superior to [25] and [9], in terms of both standard deviations and best values of BD-BR and BD-PSNR. Thus, our approach performs best among the three approaches in terms of RD performance. The RD improvement of our approach is mainly due to the high prediction accuracy of the CU partition, benefiting from the deep ETH-CNN structure with sufficient parameters learned from our large-scale CPH-Intra database.

5) Comparison of Complexity-RD Performance: As mentioned in Section IV-D, the bi-threshold scheme can control the RD performance and encoding complexity. Therefore, we changed the uncertain zone $[\bar{\alpha}_l, \alpha_l]$ of the bi-threshold scheme to assess the complexity-RD (CRD) performance of our approach. As mentioned in Section IV-D, $[\bar{\alpha}_1, \alpha_1] \subset [\bar{\alpha}_2, \alpha_2] \subset [\bar{\alpha}_3, \alpha_3]$ and $\bar{\alpha}_l = 1 - \alpha_l$ exist for our bi-threshold

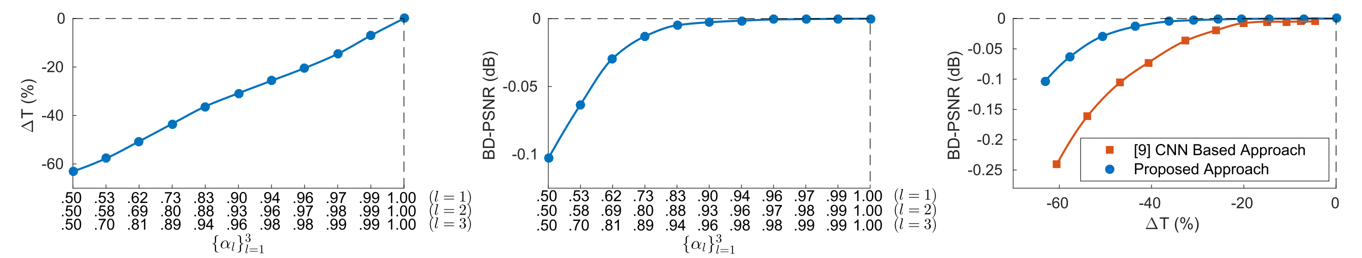


Fig. 9. CRD performance for intra-mode HEVC complexity reduction. Left and middle: Complexity reduction and RD performance at different uncertain zones. No that the labels on horizontal axes are upper thresholds $\{\alpha_l\}_{l=1}^3$, and the corresponding lower thresholds are $\bar{\alpha}_l = 1 - \alpha_l$ for $l \in \{1, 2, 3\}$. Right: Comparison of CRD performance between both approaches.

scheme. Therefore, the specific values of $\{[\bar{\alpha}_l, \alpha_l]\}_{l=1}^3$ at each level l are determined by power functions for simplicity:

$$\begin{cases} \alpha_l = 0.5 + 0.5d^{2-0.5l} \\ \bar{\alpha}_l = 0.5 - 0.5d^{2-0.5l} \end{cases} \quad l \in \{1, 2, 3\}, . \tag{7}$$

where $d \in [0, 1]$ is the width of the uncertain zone at level $l = 2$. Specifically, Fig. 9-left and -middle show the results of the encoding complexity reduction and RD performance degradation at different uncertain zones by adjusting d, for intra-mode HEVC. The results are averaged over the JCT-VC test sequences at QP $= 22, 27, 32,$ and 37. In this figure, the encoding complexity reduction is measured by ΔT (i.e., encoding time saving) and the RD performance degradation is assessed by BD-PSNR. We can see from Fig. 9-left and -middle that the absolute values of both ΔT and BD-PSNR are increased when the uncertain zone of $[\bar{\alpha}_l, \alpha_l]$ becomes narrower. This indicates that the trade-off between the RD performance and encoding complexity can be controlled by changing the uncertain zone $[\bar{\alpha}_l, \alpha_l]$. In Fig. 9-right, we also plot the CRD curves of our approach and the state-of-the-art CNN based approach [9] by plotting the values of ΔT and BD-PSNR, which are obtained at different uncertain zones of $[\bar{\alpha}_l, \alpha_l]$. This figure shows that the RD performance is improved at less reduction of encoding complexity. This figure further shows that our approach achieves a better RD performance than [9] at the same encoding complexity reduction.

C. Performance Evaluation at Inter-Mode

1) Evaluation on Training Performance: First, we evaluate the training loss at different iterations when training the ETH-CNN and ETH-LSTM models for inter-mode HEVC complexity reduction. Fig. 8-(a) and -(b) illustrate the training loss curves for ETH-CNN and ETH-LSTM, respectively. Here, the loss values are obtained from (2) for training the ETH-CNN model and from (6) for training the ETH-LSTM model. We can see that both the ETH-CNN and LSTM models converge at a fast speed. Additionally, at the LDP configuration, the training loss of the ETH-LSTM model converges to a smaller value in predicting the CU partition of HEVC, compared with the ETH-CNN model.

2) Evaluation on Prediction Accuracy: First, we evaluate the accuracy of the CU partition predicted by ETH-LSTM when compressing 18 test sequences using HEVC at the LDP configuration. The experimental results show that the average accuracy rates of the inter-mode CU partition at three levels are 93.89%, 88.01% and 80.91%. These values are much higher than the 90.21%, 82.62% and 79.42% of the three-level joint SVM classifier reported in [8]. Moreover, the accuracy rates of the inter-mode CU partition are higher than those of the intra-mode CU partition in our deep learning approach (reported in Section VI-B) because of the temporal correlation of the CU partition learned in ETH-LSTM.

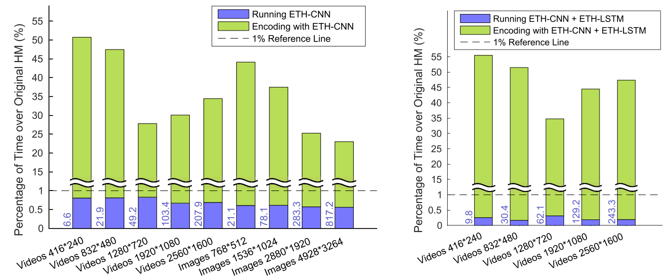
3) Evaluation on Complexity Reduction: Next, we compare our approach with the latest work of [8], [7] and [10] for encoding complexity reduction in inter-mode HEVC. In Table V, ΔT illustrates the complexity reduction rates of 18 test sequences. As seen in this table, at QP $= 22, 27, 32$ and 37, our approach can reduce the encoding complexity by 43.84%, 52.13%, 57.89% and 62.94% on average. Such complexity reduction rates are superior to the 38.31%, 46.10%, 48.95% and 46.50% of [8], the 32.16%, 39.59%, 49.31% and 57.20% of [7], and the 43.67%, 43.10%, 43.54% and 43.98% of [10]. In addition, our approach is able to reduce more complexity than the other three approaches on most sequences under different QPs and resolutions. This is mainly because our approach is able to predict all the CU partition of an entire CTU at a time in the form of HCPM, whereas the other approaches need to run the classifier several times or to check the RDO of some CU partition in splitting a CTU. We can further see from Table V that our approach at the LDP configuration has the smallest standard deviations of ΔT among all four approaches. This implies that our approach is robust in reducing complexity of inter-mode HEVC. However, the maximal complexity reduction of our approach is less than that of [7] for inter-mode HEVC. Again, this implies the robustness of our approach in complexity reduction of inter-mode HEVC, since our approach has more average complexity reduction than [7].

4) Evaluation on RD Performance: In addition to complexity reduction, RD performance is also a critical metric for evaluation. In our experiments, we compare RD performance of all four approaches, in terms of BD-BR and BD-PSNR. Table V tabulates BD-BR and BD-PSNR results of four approaches, with the original HM as anchor. The BD-BR of our ETH-LSTM approach is averagely 1.495%, better than 1.799% of [8], 5.051% of [7] and 3.616% of [10]. On the other hand, the BD-PSNR of our approach is -0.046 dB, which is superior to -0.054dB of [8], -0.163dB of [7] and -0.108dB of [10]. Also, Table V shows that our approach is generally superior to [8], [7] and [10] for both standard deviations and best values of the RD loss (evaluated by BD-BR and BD-PSNR). Therefore, our approach performs best in terms of RD, among all four approaches. It is mainly attributed to the high accuracy of the CU partition predicted by ETH-LSTM as discussed above.

TABLE V

RESULTS FOR SEQUENCES OF THE JCT-VT TEST SET (LDP)

Class	Sequence	Appr.	BD-BR (%)	BD-PSNR (dB)	ΔT (%)			
					QP=22	QP=27	QP=32	QP=37
A	*PeopleOnStreet*	[8]	1.456	-0.062	-35.26	-42.53	-45.40	-46.32
		[7]	9.813	-0.400	-29.05	-26.39	-31.28	-39.18
		[10]	4.292	-0.180	-39.32	-40.46	-41.85	-43.99
		Our	**1.052**	**-0.045**	**-40.19**	**-45.73**	**-50.02**	**-52.07**
	Traffic	[8]	2.077	-0.056	-43.66	-57.25	-62.91	-58.04
		[7]	4.639	-0.119	-38.30	-50.70	-61.39	-68.60
		[10]	6.983	-0.178	**-48.71**	-52.22	-56.96	-60.03
		Our	**1.990**	**-0.052**	-47.43	**-59.39**	**-65.00**	**-70.59**
B	*BasketballDrive*	[8]	**2.246**	-0.055	-33.61	-48.72	-50.70	-47.62
		[7]	6.372	-0.145	-33.90	-45.16	-52.16	-58.47
		[10]	3.991	-0.092	-40.77	-44.36	-46.67	-48.67
		Our	2.268	**-0.052**	**-41.16**	**-55.34**	**-60.18**	**-66.69**
	BQTerrace	[8]	1.733	-0.029	-32.15	-41.84	-51.59	-49.80
		[7]	1.934	-0.028	-15.32	-35.77	-58.41	-71.95
		[10]	2.892	-0.042	**-45.49**	-47.80	-53.76	-56.01
		Our	**1.090**	**-0.017**	-44.39	**-55.72**	**-67.97**	**-71.97**
	Cactus	[8]	**2.005**	-0.044	-33.24	-47.21	-52.77	-50.00
		[7]	5.125	-0.105	-25.09	-42.48	-53.95	-62.03
		[10]	5.516	-0.111	**-43.92**	-49.32	-52.44	-53.60
		Our	2.071	**-0.043**	-41.51	**-55.93**	**-61.56**	**-68.51**
	Kimono	[8]	2.219	-0.066	-41.68	-52.39	-53.98	-53.27
		[7]	4.382	-0.128	-43.06	-51.24	-57.77	**-66.33**
		[10]	3.239	-0.097	**-46.21**	-47.70	-49.76	-50.01
		Our	**1.497**	**-0.048**	-45.58	**-54.99**	**-59.53**	-64.02
	ParkScene	[8]	1.559	-0.044	-38.88	-47.36	-54.17	-49.50
		[7]	3.493	-0.098	-29.65	-39.63	-53.37	-64.09
		[10]	4.493	-0.126	-43.93	-45.53	-49.39	-51.86
		Our	**1.474**	**-0.042**	**-46.48**	**-56.08**	**-63.25**	**-69.10**
C	*BasketballDrill*	[8]	**1.599**	**-0.058**	-36.96	-43.63	-45.48	-42.93
		[7]	7.485	-0.266	-29.66	-34.97	-43.46	-53.26
		[10]	4.055	-0.147	-44.00	-41.93	-41.56	-41.99
		Our	1.953	-0.072	**-49.75**	**-52.75**	**-57.63**	**-60.66**
	BQMall	[8]	**1.604**	**-0.058**	-39.48	-43.32	-47.54	-43.84
		[7]	6.801	-0.239	-34.13	-36.04	-46.16	-55.83
		[10]	4.092	-0.147	-40.73	-39.05	-39.04	-39.92
		Our	1.914	-0.071	**-42.02**	**-48.58**	**-54.50**	**-57.86**
	PartyScene	[8]	1.301	-0.047	-31.34	-34.49	-36.71	-33.40
		[7]	2.988	-0.107	-12.64	-18.14	-28.75	-43.19
		[10]	2.469	-0.094	**-42.75**	-38.35	-37.40	-37.89
		Our	**1.011**	**-0.039**	-38.31	**-41.77**	**-49.00**	**-58.24**
	RaceHorses	[8]	2.085	-0.072	-36.39	-39.58	-39.18	-37.80
		[7]	6.204	-0.215	-20.17	-25.83	-33.28	-40.32
		[10]	3.016	-0.105	-36.14	-33.56	-31.90	-32.97
		Our	**0.872**	**-0.032**	**-39.65**	**-43.25**	**-48.65**	**-53.34**
D	*BasketballPass*	[8]	**1.321**	**-0.059**	-31.74	-33.89	-34.58	-33.59
		[7]	6.995	-0.310	-31.66	-29.44	-33.39	-41.93
		[10]	1.949	-0.086	-36.80	-31.04	-29.01	-25.80
		Our	1.453	-0.066	**-43.69**	**-46.85**	**-50.66**	**-54.97**
	BlowingBubbles	[8]	**0.974**	**-0.034**	-23.81	-27.92	-26.40	-18.11
		[7]	2.180	-0.078	-14.18	-19.91	-31.71	-40.22
		[10]	2.479	-0.086	**-35.52**	-31.21	-29.04	-26.03
		Our	1.292	-0.044	-33.63	**-41.40**	**-50.21**	**-57.26**
	BQSquare	[8]	1.175	-0.036	-24.80	-31.02	-31.84	-27.88
		[7]	2.090	-0.062	-13.96	-20.62	-43.86	-45.72
		[10]	1.998	-0.067	**-45.19**	-34.69	-26.46	-26.09
		Our	**0.770**	**-0.028**	-40.24	**-40.61**	**-49.33**	**-57.47**
	RaceHorses	[8]	1.306	-0.054	-29.48	-29.34	-28.37	-32.78
		[7]	6.867	-0.281	-19.58	-20.26	-25.37	-33.07
		[10]	2.240	-0.095	-34.81	-27.72	-22.92	-20.09
		Our	**1.112**	**-0.047**	**-37.07**	**-39.26**	**-43.14**	**-47.99**
E	*FourPeople*	[8]	2.593	-0.082	-58.11	-67.79	-71.85	-67.62
		[7]	5.492	-0.165	**-59.74**	**-69.08**	**-74.29**	**-79.19**
		[10]	3.974	-0.122	-53.81	-56.60	-57.11	-58.12
		Our	**1.837**	**-0.052**	-53.36	-64.29	-67.89	-71.97
	Johnny	[8]	3.024	-0.058	-59.16	-72.00	-74.29	-73.91
		[7]	3.769	-0.079	**-65.45**	**-75.72**	**-81.45**	**-84.83**
		[10]	3.452	-0.065	-55.45	-58.49	-59.50	-59.82
		Our	**1.691**	**-0.038**	-51.99	-67.68	-71.34	-74.97
	KristenAndSara	[8]	2.098	-0.055	-59.90	-69.46	-73.25	-70.55
		[7]	4.293	-0.113	**-63.34**	**-71.53**	**-77.57**	**-81.40**
		[10]	3.964	-0.104	-52.68	-55.75	-58.98	-58.77
		Our	**1.558**	**-0.045**	-52.69	-68.77	-72.19	-75.27
Std. dev.		[8]	0.534	**0.013**	10.84	13.44	14.82	14.88
		[7]	2.146	0.100	16.62	18.13	17.18	16.08
		[10]	1.282	0.037	6.17	9.34	12.04	13.30
		Our	**0.448**	0.014	**5.60**	**9.27**	**8.84**	**8.45**
Best		[8]	0.974	-0.029	-59.90	-72.00	-74.29	-73.91
		[7]	1.934	-0.028	**-65.45**	**-75.72**	**-81.45**	**-84.83**
		[10]	1.949	-0.042	-55.45	-58.49	-59.50	-60.03
		Our	**0.770**	**-0.017**	-53.36	-68.77	-72.19	-75.27
Average		[8]	1.799	-0.054	-38.31	-46.10	-48.95	-46.50
		[7]	5.051	-0.163	-32.16	-39.59	-49.31	-57.20
		[10]	3.616	-0.108	-43.67	-43.10	-43.54	-43.98
		Our	**1.495**	**-0.046**	**-43.84**	**-52.13**	**-57.89**	**-62.94**



(a) (b)

Fig. 10. Time percentages for running our deep learning models and encoding videos/images by HM with our complexity reduction approach, with the original HM 16.5 as an anchor. Note the number in blue font next to each bar indicates the real running time of ETH-CNN and ETH-CNN + ETH-LSTM (in millisecond) for one frame. (a) AI configuration. (b) LDP configuration.

TABLE VI

RESULTS FOR ABLATION STUDY

Config.	Settings	BD-BR (%)	BD-PSNR (dB)	ΔT (%)			
				QP=22	QP=27	QP=32	QP=37
AI	1. Shallow CNN [9]	6.189	-0.316	-56.41	-60.23	-62.62	-65.04
	2. Deep CNN [12]	2.249	-0.105	-55.44	-58.23	-60.12	-63.79
	3. ETH-CNN	**2.247**	**-0.104**	**-56.92**	**-60.38**	**-63.61**	**-66.47**
LDP	1. ETH-CNN fed with original CTUs	8.911	-0.248	**-47.00**	**-55.10**	**-60.68**	**-65.53**
	2. ETH-CNN fed with residual CTUs	3.450	-0.106	-44.76	-52.87	-58.54	-63.79
	3. ETH-CNN fed with residual CTUs + ETH-LSTM	**1.495**	**-0.046**	-43.84	-52.13	-57.89	-62.94

D. Analysis on Running Time

We analyze the running time of our deep learning approach by comparing with that of the original HM 16.5 [3]. Fig. 10 shows the time percentages of running ETH-CNN and ETH-CNN + ETH-LSTM relative to the original HM. It also plots the time percentages of encoding with ETH-CNN and ETH-CNN + ETH-LSTM, with the original HM as an anchor. The number in blue font next to each bar in Fig.10 indicates the absolute time[3] of running ETH-CNN and ETH-CNN + ETH-LSTM (in millisecond) for one frame. The results of Fig. 10 are obtained by averaging over all test images/sequences with the same resolution at four QP values {22, 27, 32, 37}. We can see from Fig. 10-(a) that ETH-CNN consumes less than 1% of the time required by the original HM. Moreover, ETH-CNN takes up a small portion of time when compressing images/sequences by HM with our complexity reduction approach. Hence, ETH-CNN[4] introduces little time overhead in reducing the complexity of intra-mode HEVC.

As seen in Fig. 10-(b), ETH-CNN and ETH-LSTM achieve shorter running time compared with the encoding time of the original HM at the LDP configuration. Note that the time proportion of ETH-CNN and ETH-LSTM in inter-mode HEVC is far less than that of ETH-CNN in intra-mode HEVC, since inter-mode HEVC has a significantly longer encoding time than intra-mode HEVC. Besides, Fig. 10-(b) shows that the CU partition by our approach consumes relatively little encoding time, when applying our deep learning approach for complexity reduction in inter-mode HEVC.

E. Ablation Study

In this section, we conducted a series of ablation experiments to analyze the impact of major components in the proposed approach. In our ablation experiments, we started from a simple CNN model and then added the components stepwise, finally reaching the proposed approach. Table VI reports the results of ablation experiments. Note that the ablation results are averaged over all 18 test sequences. In the following, we discuss these ablation results in details.

[3]Note that only one thread of CPU is used for the above running time, in accord with the HM platform that is also implemented in one thread. If enabling the multi-thread parallelism of CPU, the running time can be drastically reduced.

[4]Although running ETH-CNN requires little time for each video frame, a huge amount of time is taken for training the ETH-CNN model, as it is trained from large-scale data. This can be done off-line, making our ETH-CNN practical for complexity reduction in HEVC.

1) Deep CNN Structure vs. Shallow CNN Structure: In our approach, a deep structure of CNN is developed, which is significantly deeper than the shallow CNN structure in the state-of-the-art HEVC complexity reduction approach [9]. Such a deep CNN structure has been presented in the conference version [12] of this paper, and it has similar structure and comparable amount of trainable parameters to our ETH-CNN model. Therefore, we analyze the impact of the deep structure in our approach by comparing the results between [12] and [9]. As can be seen in Table VI, the deep CNN structure improves the RD performance for intra-mode HEVC complexity reduction, which has 3.940% BD-BR saving and 0.211dB BD-PSNR increase over the shallow structure of [9].

2) The ETH-CNN Model vs. the General Classification-Oriented CNN Model: In the ETH-CNN model, the early-terminated hierarchical (ETH) decision is proposed to predict the CU partition, which replaces the conventional three-level classification of [12]. In fact, [12] is the conference version of this paper, in which three deep networks of the CNN model are invoked to predict the three-level classification of 64×64, 32×32 and 16×16 CU partition, respectively. Note that the CNN model of [12] has similar structure and comparable amount of trainable parameters to our ETH-CNN model. In our ablation experiments, we therefore compare the results of the ETH-CNN model with those of the classification-oriented CNN model [12], to investigate the impact of the ETH decision. We can see from Table VI that the ETH-CNN model can save $1.48\% - 3.49\%$ encoding time over [12] for intra-mode HEVC. This indicates the effectiveness of the ETH decision proposed in this paper, which replaces the general CNN model based classification in [12].

3) Residual CTUs vs. Original CTUs: For inter-mode HEVC complexity reduction, we propose to feed the ETH-CNN model with residual CTUs, rather than the original CTUs. In our ablation experiments, we thus validate the effectiveness of residual CTUs in our approach. To this end, we compare the performance of the ETH-CNN model fed with residual CTUs and original CTUs. Note that original CTUs are replaced by residual CTUs in both training and test procedures. Table VI shows that the residual CTUs can improve the RD performance with 5.461% BD-BR saving and 0.142dB BD-PSNR increase. The cost is $1.12\% - 2.24\%$ increase in the encoding time.

4) ETH-CNN + LSTM vs. Only ETH-CNN: In our approach, ETH-LSTM is a major contribution for inter-mode HEVC complexity reduction. We thus show the performance improvement of embedding the ETH-LSTM model in our approach, i.e., the combination of the ETH-CNN and ETH-LSTM models. Note that the residual CTUs are as the input to the ETH-CNN model followed by the ETH-LSTM model. As shown in Table VI, the ETH-LSTM model prominently improves the RD performance with less than one half BD-BR and BD-PSNR, and meanwhile has insignificant change of the encoding time. This indicates the effectiveness of the ETH-LSTM model in our approach. Besides, the ETH-LSTM model introduces little computational time in our approach, since Sections IV-E and V-C have investigated that the ETH-LSTM model consumes much less floating-point operations than the ETH-CNN model.

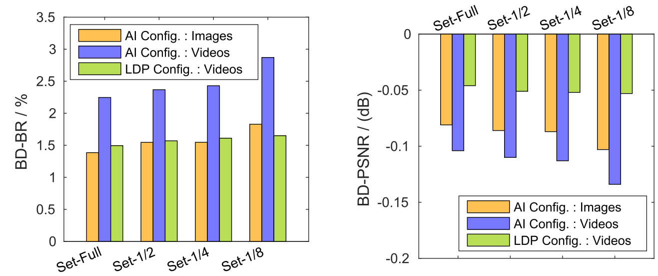


Fig. 11. RD performance with different training sets. At the AI configuration, Set-1/2, Set-1/4 and Set-1/8 contain 850, 425 and 213 images, respectively. At the LDP configuration, those are 42, 21 and 11 video sequences.

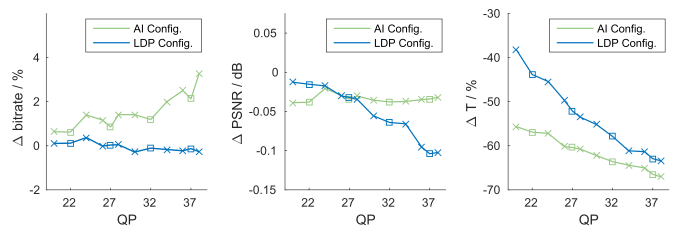


Fig. 12. Curves for Δbitrate, ΔPSNR and ΔT of our approach at different QPs. The square marks are the results at four anchored QPs, and the cross marks represent the results at other QPs.

F. Performance Evaluation With Various Settings

1) Evaluation With Different Amount of Training Data: Since the proposed approach is data-driven, it is necessary to evaluate its performance when changing the amount of training data. For intra- and inter-modes, the 1700 training images and 83 training sequences in our CPH-Intra and CPH-Inter databases are regarded as the full training sets, namely Set-Full. For either intra- or inter-mode, three smaller training sets are generated via randomly selecting $1/2$, $1/4$ and $1/8$ of images or videos from the corresponding Set-Full, namely Set-$1/2$, Set-$1/4$ and Set-$1/8$, respectively. Afterwards, the ETH-CNN and ETH-LSTM models are retrained on Set-$1/2$, Set-$1/4$ and Set-$1/8$, respectively. Fig. 11 shows the RD performance of our approach, when changing the amount of training data (i.e., Set-full, Set-$1/2$, Set-$1/4$ and Set-$1/8$). Note that the results are averaged over all test images or all test sequences at QPs $22, 27, 32$ and 37. We can observe in Fig. 11 that BD-BR decreases and BD-PSNR increases along with the increased amount of training data, for both AI (i.e., intra-mode) and LDP (i.e., inter-mode) configurations. Therefore, the large-scale CPH databases are essential to achieve the desirable RD performance.

2) Generalization Capability at Different QPs: In addition to four QPs evaluated above (QP $= 22, 27, 32, 37$), we further test our approach for reducing complexity of intra- and inter-mode HEVC at other eight QPs, i.e., QP $= 20, 24, 26, 28, 30, 34, 36, 38$. To test the generalization capability of our approach, we directly use the ETH-CNN and ETH-LSTM models trained at four original QPs (i.e., QP $= 22$, $27, 32, 37$) without re-training on the sequences compressed at other QPs. Fig. 12 illustrates the bit-rate difference (Δbitrate), PSNR loss (ΔPSNR) and time reduction (ΔT) of our approach at different QPs, for complexity reduction of HEVC at both AI and LDP configurations. Note that the results are averaged over all test video sequences. In this figure, the square marks

TABLE VII
RESULTS FOR SEQUENCES OF THE JCT-VT TEST SET (LDB & RA)

Config.	Approach	BD-BR (%)	BD-PSNR (dB)	ΔT (%)			
				QP=22	QP=27	QP=32	QP=37
LDB	[8]	1.840	-0.057	-29.57	-45.69	-44.91	-44.12
	[7]	4.312	-0.144	-32.07	-38.68	-48.03	-57.11
	[10]	3.458	-0.109	**-45.76**	-46.98	-47.67	-48.99
	Our	**1.722**	**-0.053**	-39.76	**-47.43**	**-54.72**	**-59.84**
RA	[8]	2.319	-0.074	-42.50	-48.17	-51.89	-51.13
	[7]	6.246	-0.210	-40.96	-49.17	-55.41	-61.10
	[10]	6.101	-0.204	-40.17	-41.98	-43.20	-43.02
	Our	**1.483**	**-0.048**	**-43.14**	**-51.64**	**-59.55**	**-64.07**

denote the test results at four original QPs, whereas the cross marks mean the test results at eight other QPs (the sequences compressed at these QPs are not used for training). As shown in Fig. 12-left and -middle, Δbitrate and ΔPSNR at each of other QPs (i.e., cross marks) are close to those of the adjacent original QPs (i.e., square marks), with only slight fluctuation. This indicates the generalization capability of our approach at different QPs. Besides, we can see from Fig. 12-right that the higher QPs result in larger reduction rate of encoding time.

3) Evaluation at LDB and RA Configurations: We further evaluate the performance of our approach for HEVC complexity reduction at the LDB and RA configurations. In our experiments, the ETH-CNN and ETH-LSTM models for both configurations were retrained from the training sequences at the corresponding configurations in our CPH-Inter database. For the LDB configuration, the experimental settings followed those for the LDP configuration. For the RA configuration, we slightly modified our approach to reduce HEVC complexity. Specifically, the length of ETH-LSTM was 32 for both training and test at the RA configuration, in accord with the period between two adjacent I frames. Here, the training samples were non-overlapping instead of the 10-frame overlapping used in the LDP configuration, ensuring all frames of the training samples locate in the same period within two adjacent random access points. Besides, the order of frames fed into ETH-LSTM followed the order of encoding rather than that of displaying. Table VII shows the performance of our and other state-of-the-art approaches for HEVC complexity reduction at the LDB and RA configurations. Similar to the above evaluation, the results in this table are averaged over all 18 standard video sequences from the JCT-VC test set [2]. As shown in Table VII, our approach at the RA configuration is able to reduce $43.14\% \sim 64.07\%$ encoding time while incurring 1.483% of BD-BR increment and 0.048dB of BD-PSNR loss, similar to those at the LDP configuration ($-43.84\% \sim -62.94\%$ of ΔT, 1.495% of BD-BR and -0.046dB of BD-PSNR). More importantly, our approach outperforms other state-of-the-art approaches [8], [7] and [10] in terms of both encoding time and RD performance. At the LDB configuration, similar results can be found. Therefore, the effectiveness of our approach at the LDB and RA configurations is verified.

4) Evaluation on Sequences With Drastic Scene Change: In practice, drastic scene change may occur in videos. Thus, we further evaluate the performance of our approach on the sequences that have scene change, under the LDP and RA configurations. Among the 18 JCT-VC test sequences [2], only sequence *Kimono* (1920×1080) has drastic scene change. Hence, in addition to *Kimono*, two sequences with drastic scene change, *Tennis* (1920×1080) and *Mobisode* (832×480), were tested in our experiments. Table IV of the *Supporting Document* shows the overall performance of our and other approaches, averaged over all frames of each sequence at QP $= 22, 27, 32,$ and 37. We can see from this table that our approach outperforms other approaches for sequences with scene change. Moreover, in the *Supporting Document* we further compare the results of our and other approaches in the scene changed frames of the above three sequences, and the results show that our approach is robust to scene change.

VII. CONCLUSIONS

In this paper, we have proposed a deep learning approach to reduce the encoding complexity of intra- and inter-mode HEVC, which learns to predict the optimal CU partition instead of using conventional brute-force RDO search. To deepen the networks of our deep learning approach, the CPH-Intra and CPH-Inter database was established, consisting of large-scale data of HEVC CU partition at intra- and inter modes, respectively. Then, two deep learning architectures, i.e., ETH-CNN and ETH-LSTM, were proposed to predict the CU partition for reducing the HEVC complexity at the intra- and inter-modes. The output of these architectures is HCPM, which hierarchically represents the CU partition in a CTU. Upon the representation of HCPM, an early termination mechanism was introduced in ETH-CNN and ETH-LSTM to save the computational time. The experimental results show that our deep learning approach performs much better than other state-of-the-art approaches in terms of both complexity reduction and RD performance.

There are three promising directions for future works. Our work, at the current stage, mainly focuses on predicting the CU partition to reduce the HEVC encoding complexity. Other components, such as PU and TU prediction, can also be replaced by deep learning models to further reduce the encoding complexity of HEVC. This is an interesting future work. In the deep learning area, various techniques have been proposed to accelerate the running speed of deep neural networks. Our deep learning approach may be sped up by applying these acceleration techniques, which is seen as another promising future work. Beyond the CPU implementation of the current work, our approach may be further implemented in the FPGA device in future, for practical applications.

REFERENCES

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

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

[3] JCT-VC. (2014). *HM Software*. Accessed: Nov. 5, 2016. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svnHEVCSoftware/tags/HM-16.5/

[4] J. Leng, L. Sun, T. Ikenaga, and S. Sakaida, "Content based hierarchical fast coding unit decision algorithm for HEVC," in *Proc. Int. Conf. Multimedia Signal Process.*, May 2011, pp. 56–59.

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

[6] X. Shen, L. Yu, and J. Chen, "Fast coding unit size selection for HEVC based on Bayesian decision rule," in *Proc. Picture Coding Symp.*, May 2012, pp. 453–456.

[7] G. Corrêa, P. A. Assuncao, L. V. Agostini, and L. A. da Silva Cruz, "Fast HEVC encoding decisions using data mining," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 4, pp. 660–673, Apr. 2015.

[8] 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 Trans. Image Process.*, vol. 24, no. 7, pp. 2225–2238, Jul. 2015.

[9] 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 Trans. Image Process.*, vol. 25, no. 11, pp. 5088–5103, Nov. 2016.

[10] T. Mallikarachchi, D. S. Talagala, H. K. Arachchi, and A. Fernando, "Content-adaptive feature-based CU size prediction for fast low-delay video encoding in HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 3, pp. 693–705, Mar. 2018.

[11] L. Zhu, Y. Zhang, Z. Pan, R. Wang, S. Kwong, and Z. Peng, "Binary and multi-class learning based low complexity optimization for HEVC encoding," *IEEE Trans. Broadcast.*, vol. 63, no. 3, pp. 547–561, Sep. 2017.

[12] T. Li, M. Xu, and X. Deng, "A deep convolutional neural network approach for complexity reduction on intra-mode HEVC," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2017, pp. 1255–1260.

[13] S. Cho and M. Kim, "Fast CU splitting and pruning for suboptimal CU partitioning in HEVC intra coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 9, pp. 1555–1564, Sep. 2013.

[14] 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 *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, Jun. 2016, pp. 1–3.

[15] M. U. K. Khan, M. Shafique, and J. Henkel, "An adaptive complexity reduction scheme with fast prediction unit decision for HEVC intra encoding," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2013, pp. 1578–1582.

[16] H.-M. Yoo and J.-W. Suh, "Fast coding unit decision algorithm based on inter and intra prediction unit termination for HEVC," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2013, pp. 300–301.

[17] 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, 2012.

[18] J. Cui, S. Wang, S. Wang, X. Zhang, S. Ma, and W. Gao, "Hybrid Laplace distribution-based low complexity rate-distortion optimized quantization," *IEEE Trans. Image Process.*, vol. 26, no. 8, pp. 3802–3816, Aug. 2017.

[19] K. Miyazawa, T. Murakami, A. Minezawa, and H. Sakate, "Complexity reduction of in-loop filtering for compressed image restoration in HEVC," in *Proc. Picture Coding Symp.*, May 2012, pp. 413–416.

[20] H. Zhang and Z. Ma, "Fast intra mode decision for high-efficiency video coding (HEVC)," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 4, pp. 660–668, Apr. 2014.

[21] J. Vanne, M. Viitanen, and T. D. Hamalainen, "Efficient mode decision schemes for HEVC inter prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 9, pp. 1579–1593, Sep. 2014.

[22] L. Shen, Z. Zhang, and Z. Liu, "Effective CU size decision for HEVC intracoding," *IEEE Trans. Image Process.*, vol. 23, no. 10, pp. 4232–4241, Oct. 2014.

[23] C. S. Park, "Edge-based intramode selection for depth-map coding in 3D-HEVC," *IEEE Trans. Image Process.*, vol. 24, no. 1, pp. 155–162, Jan. 2015.

[24] Q. Hu, Z. Shi, X. Zhang, and Z. Gao, "Fast HEVC intra mode decision based on logistic regression classification," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, Jun. 2016, pp. 1–4.

[25] D. Liu, X. Liu, and Y. Li, "Fast CU size decisions for HEVC intra frame coding based on support vector machines," in *Proc. IEEE 14th Intl Conf. Dependable, Auto. Secure Comput. (DASC)*, Aug. 2016, pp. 594–597.

[26] T. Laude and J. Ostermann, "Deep learning-based intra prediction mode decision for HEVC," in *Proc. Picture Coding Symp. (PCS)*, Dec. 2016, pp. 1–5.

[27] H.-S. Kim and R.-H. Park, "Fast CU partitioning algorithm for HEVC using an online-learning-based Bayesian decision rule," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 130–138, Jan. 2016.

[28] H. R. Tohidypour, H. Bashashati, M. T. Pourazad, and P. Nasiopoulos, "Online-learning-based mode prediction method for quality scalable extension of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 10, pp. 2204–2215, Oct. 2017.

[29] J. F. de Oliveira and M. S. Alencar, "Online learning early skip decision method for the HEVC inter process using the SVM-based pegasos algorithm," *Electron. Lett.*, vol. 52, no. 14, pp. 1227–1229, Jul. 2016.

[30] F. Duanmu, Z. Ma, and Y. Wang, "Fast mode and partition decision using machine learning for intra-frame coding in HEVC screen content coding extension," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 6, no. 4, pp. 517–531, Dec. 2016.

[31] S. Momcilovic, N. Roma, L. Sousa, and I. Milentijevic, "Run-time machine learning for HEVC/H.265 fast partitioning decision," in *Proc. IEEE Int. Symp. Multimedia (ISM)*, Dec. 2015, pp. 347–350.

[32] B. Du, W.-C. Siu, and X. Yang, "Fast CU partition strategy for HEVC intra-frame coding using learning approach via random forests," in *Proc. Asia–Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA)*, Dec. 2015, pp. 1085–1090.

[33] Y. Shan and E.-H. Yang, "Fast HEVC intra coding algorithm based on machine learning and Laplacian transparent composite model," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 2642–2646.

[34] R. He *et al.* "Clustering enabled wireless channel modeling using big data algorithms," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 177–183, May 2018.

[35] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato, "RAISE: A raw images dataset for digital image forensics," in *Proc. 6th ACM Multimedia Syst. Conf.*, 2015, pp. 219–224.

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

[37] M. Xu, X. Deng, S. Li, and Z. Wang, "Region-of-interest based conversational HEVC coding with hierarchical perception model of face," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 3, pp. 475–489, Jun. 2014.

[38] J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards—Including high efficiency video coding (HEVC)," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1669–1684, Dec. 2012.

[39] Xiph.org. (2017). *Xiph.org Video Test Media*. [Online]. Available: https://media.xiph.org/video/derf

[40] G. Hinton, N. Srivastava, A. Krizhevsky, R. R. Salakhutdinov, and I. Sutskever, "Improving neural networks by preventing co-adaptation of feature detectors," *Comput. Sci.*, vol. 3, no. 4, pp. 212–223, Jul. 2012.

[41] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. IEEE Int. Workshop Acoust. Signal Enhancement (IWAENC)*, vol. 15, Jun. 2011, pp. 315–323.

[42] LanOC Corporation. (2017). *Intel Core X Series i9-7900X and i7-7740X*. Accessed: Aug. 8, 2017. [Online]. Available: https://lanoc.org/review/cpus/7566-intel-core-x-series-i9-7900x-and-i7-7740x?showall=&start=2

[43] G. Bjøntegaard, *Calculation of Average PSNR Differences Between RD-Curves*, document ITU-T VCEG-M33, Austin, TX, USA, Apr. 2001.

Mai Xu, photograph and biography not available at the time of publication.

Tianyi Li, photograph and biography not available at the time of publication.

Zulin Wang, photograph and biography not available at the time of publication.

Xin Deng, photograph and biography not available at the time of publication.

Ren Yang, photograph and biography not available at the time of publication.

Zhenyu Guan, photograph and biography not available at the time of publication.