Accelerate CTU Partition to Real Time for HEVC Encoding With Complexity Control

Tianyi Li, *Graduate Student Member, IEEE*, Mai Xu, *Senior Member, IEEE*, Xin Deng, *Member, IEEE*, and Liquan Shen, *Member, IEEE*

***Abstract*— Recently, extensive approaches have been proposed for reducing the encoding complexity of high efficiency video coding, by predicting the coding tree unit partition using deep neural networks. However, these approaches cannot work in real time due to the complexity of the network architectures. In this paper, we propose a network pruning approach to accelerate a state-of-the-art deep neural network model, for real-time coding tree unit partition. Specifically, we first investigate the computational complexity throughout the network, and find that most calculations can be simplified by pruning the weight parameters. Considering that the number of weight parameters drastically differs by network layer and partition level, we design an adaptive pruning scheme by applying a well-suitable retention ratio of weight parameters to each layer at a level. The retention ratio indicates the ratio of weight parameters after and before pruning. By varying the retention ratios, we can obtain several accelerated network models with different levels of complexity. We further propose a complexity control algorithm by applying different accelerated models to different coding tree units, to ensure that the actual encoding complexity is close to a given target. To guarantee the rate-distortion performance, we model the complexity control algorithm as a convex optimization problem, and we can obtain a closed-form solution. Experimental results show that our approach can accelerate the original deep neural network model by 17–20 times, with little expense on the Bjøntegaard delta bit-rate. For complexity control, we achieve high control accuracy with a control error of less than 2% for most video sequences.**

***Index Terms*— High efficiency video coding, coding tree unit partition, complexity control.**

I. Introduction

THE high efficiency video coding (HEVC) standard [1] has been proposed by the Joint Collaborate Team on Video Coding (JCT-VC), to address the challenge on transmitting and storing the explosive amount of high definition videos. Compared with its ancestor advanced video coding (AVC) standard [2], HEVC can save bit-rates by approximately 50% on average. The remarkable coding efficiency benefits from a set of advanced techniques, including the flexible partition of coding tree unit (CTU), the precise intra-/inter- prediction modes, and the in-loop filter. However, these techniques lead to heavy computational complexity,[1] which is averagely 253% higher than AVC [3]. To alleviate this problem and further facilitate real-time encoding, it is essential to reduce the complexity of HEVC while keeping its rate-distortion (RD) performance.

Recent years have witnessed a variety of approaches for HEVC complexity reduction. As measured in the reference software HM [4], the recursive checking on coding unit (CU) partition accounts for more than 80% of the encoding time, similar to that reported in [5]. Therefore, most approaches focus on early deciding the reasonable CU partition patterns and skipping the other patterns. In earlier times, numerous heuristic [6]–[11] and learning-based [12]–[18] approaches were developed, which utilize some intermediate features in encoding to determine the CU partition before exhaustively checking all partition patterns. Despite the considerable reduction of coding complexity, the hand-crafted features rely heavily on the prior knowledge, which may be inefficient and lead to undesirable RD performance. To overcome such drawback, some deep-learning-based approaches were proposed [19]–[22]. They can automatically extract features from video content using deep neural networks (DNNs). For example, Liu *et al.* [19] and Li *et al.* [20] proposed using convolutional neural network (CNN) to determine the CU partition for intra- and inter-modes, respectively. Xu *et al.* [23] designed a fast CU partition approach with a deep early-terminated hierarchical CNN (ETH-CNN). Although the deep-learning-based approaches work well in both complexity reduction and RD performance, their network architectures are still heavy so that the running time of network cannot be ignored for real-time HEVC encoders. In fact, numerical studies [24]–[31] have shown that a DNN can be accelerated with little degradation on prediction accuracy. However, to the best of our knowledge, there is no existing work proposed to speed up the DNNs for HEVC complexity reduction.

Manuscript received November 22, 2019; revised May 9, 2020; accepted June 16, 2020. Date of publication June 25, 2020; date of current version July 13, 2020. This work was supported by the NSFC under Project 61876013, Project 61922009, and Project 61573037. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Joao M. Ascenso. *(Corresponding author: Mai Xu.)*

Tianyi Li is with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China.

Mai Xu is with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China, and also with the Hangzhou Innovation Institute, Beihang University, Hangzhou 310051, China (e-mail: maixu@buaa.edu.cn).

Xin Deng is with the School of Cyber Science and Technology, Beihang University, Beijing 100191, China.

Liquan Shen is with the School of Communication and Information Engineering, Shanghai University, Shanghai 200444, China.

Digital Object Identifier 10.1109/TIP.2020.3003730

[1] Complexity normally refers to time complexity or storage complexity. In this paper, complexity means time complexity, i.e., the computational cost of running the algorithm.

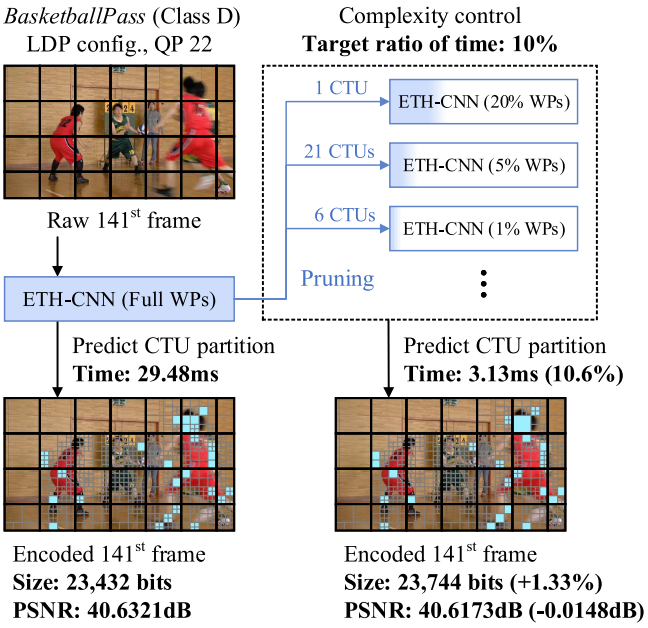

Fig. 1. An example illustrating the CTU partition of a frame at the Low Delay P (LDP) configuration, predicted by the full ETH-CNN model and our pruning approach with 10% of complexity target. The light blue patches represent the CUs with different partition between the two encoded frames. The running time was measured on a computer with Intel (R) Core (TM) i7-8700K CPU @3.2 GHz with only single thread.

This paper proposes accelerating a state-of-the-art DNN model, i.e., ETH-CNN [23], to achieve real-time CTU partition of 1080p video frames with a single CPU thread. First, we investigate the computational complexity of ETH-CNN for all layers, and find that more than 97% of floating-point operations are from the weight parameters (WPs) in trainable layers. Next, we design an adaptive pruning scheme, for determining the retention ratios to prune the WPs at each layer and each level of ETH-CNN. Here, the retention ratio indicates the ratio of WPs after and before pruning. Using the pruning scheme, we obtain seven accelerated ETH-CNN models, corresponding to seven different retention ratios of WPs. Consequently, these models provide varying trade-off between complexity and RD performance, and the partition of each CTU can be predicted by any of them. Furthermore, we develop a complexity control algorithm, enabling the average running time of CTU partition to be continuously adjustable. It is achieved by applying different models to different CTUs, with minimized control error and RD loss. As a result, the CTU partition can be drastically accelerated, and the overall control accuracy can be guaranteed. Figure 1 shows the CTU partition using the full ETH-CNN model and our pruning approach with 10-time acceleration target. As can be seen in Figure 1, the CTU partition predicted by full and pruned models is similar to each other. In addition, the average running time of our approach is shortened to 10.6%, which is quite close to the target 10%. All of these are only at the expense of 1.33% bits overhead and 0.0148dB PSNR degradation. Our acceleration algorithm for ETH-CNN is programmed by the open-source machine learning library TensorFlow 1.12 [32], and then integrated into HM 16.5 [4] with the binary files of the learned ETH-CNN parameters. The source codes are available on https://github.com/tianyili2017/ETH-CNN-Acceleration.

The main contributions of this paper are summarized as follows. (1) We investigate the layer-wise computational complexity of ETH-CNN, such that its most time-consuming components can be accelerated, i.e., WPs in trainable layers. (2) We propose a pruning scheme for ETH-CNN with optimized complexity and RD performance, and obtain various accelerated models for predicting the CTU partition. (3) We develop a complexity control algorithm, enabling the average complexity of CTU partition close to a given target with high control accuracy and small RD loss.

II. Related Works

In recent years, extensive studies have been devoted to HEVC complexity reduction. Considering that the CU partition consumes the most encoding time, most approaches focus on early deciding the reasonable CU partition patterns and skipping the other patterns. Other recursive processes nested in the CU partition can also be accelerated, such as prediction unit (PU) partition, PU mode selection and transform unit (TU) partition. Among the complexity reduction approaches, early works were heuristic ones [6]–[11], which utilize some intermediate features to determine the CTU partition. For example, Shen *et al.* [7] developed a CU partition approach by minimizing the Bayesian risk, based on the RD cost and inter-mode prediction error for each CU. Xiong *et al.* [9] and Kim *et al.* [11] proposed deciding whether each CU is split, according to the pyramid motion divergence and the number of high-frequency key points, respectively. Later, some learning-based approaches [12]–[18] were proposed to learn the optimized criteria on CU partition, benefiting from substantial training data. Specifically, Corrêa *et al.* [14] proposed three early termination schemes via data mining, to simplify the RD optimization process at CU, PU and TU levels. Zhu *et al.* [17] explored several HEVC-domain features related to the CTU partition, such as quantization parameter (QP), RD cost, number of coding bits. Providing these features, a binary and multi-class support vector machine (SVM) algorithm was proposed to predict both CU partition and PU mode with an off-on-line learning mechanism. Recently, Mallikarachchi *et al.* [16] proposed utilizing the RD cost and motion characteristics in inter-mode to decide CU partition, via an online training scheme.

Although the above approaches achieve considerable complexity reduction, the hand-crafted feature extraction can hardly mine sufficient features from the video content. Thus, it may lead to undesirable RD performance for encoding. Aiming at this challenge, some deep-learning-based approaches were proposed [19]–[22], which can automatically extract features from large-scale video data. For example, Liu *et al.* [19] proposed predicting the splitting probability of each CU, through a four-layer CNN. Kuanar *et al.* [21] designed a region-wise deep CNN integrated with textural classification and object detection, for determining the CU partition. Kim and Ro [22] combined a deep CNN with some auxiliary data, containing the PU mode, intra-prediction angle and motion vector. Despite the advantage of deep learning, most of the

above methods are not efficient enough, i.e., the splitting probability of only one CU can be predicted by a CNN in [19], [20], [22]. To overcome this drawback, Xu *et al.* [23] proposed a fast CU partition approach based on the ETH-CNN model, which can be invoked only once to determine the partition of all CUs in a whole CTU. Thus, both the computational overhead of the approach and the overall encoding complexity can be drastically reduced.

Although deep learning is beneficial to both complexity reduction and RD performance, the running time of DNNs cannot be ignored for real-time HEVC encoders, e.g., x265 [33]. Fortunately, numerous studies have shown that the DNNs can be accelerated with little influence on prediction accuracy. These studies can be broadly classified into three categories: network pruning, low-rank decomposition and low-bit quantization. In network pruning approaches [24]–[26], some trivial parameters are set to zero, so that a large proportion of connections and nodes can be removed. Specifically, Han *et al.* [24] proposed a random network pruning approach, where the parameters with magnitude below a threshold are all pruned and then the remaining parameters are fine-tuned to recover network performance. In addition to the random pruning, a DNN can also be accelerated in a more structured manner. For example, Lebedev and Lempitsky [25] attempted to simplify a typical convolutional layer into multiplications of thinner dense matrices, through group-wise regularizers. Later, Wen *et al.* [26] proposed to regularize the structure of DNNs at various levels, e.g., filter, channel and layer depth. Compared with the fine-grained random pruning [24], the coarse-grained structured pruning approaches [25], [26] are more hardware-friendly, while suffering lower performance. For low-rank decomposition, the basic idea is to apply tensor decomposition to a fully connected (FC) layer or a convolutional layer. Some representative techniques include singular value decomposition [27], Parafac/Candecomp decomposition [28] and Tucker decomposition [29]. However, the low-rank decomposition is not effective for convolutional kernel size smaller than 5×5. For parameter storage, low-bit quantization approaches [30], [31] can also be used to compress DNNs and reduce their running time. However, these approaches do not change the number of trainable parameters, and thus the potential for time reduction is limited. Among the above three categories of studies, we choose network pruning in this paper, because it has a large potential to reduce the running time of a DNN and is suitable for both convolutional and FC layers.

Our approach differs from the existing deep-learning-based HEVC complexity reduction approaches in two main aspects. (1) We propose to accelerate a state-of-the-art deep neural network with a pruning algorithm, while no acceleration is considered for a trained DNN in the existing HEVC complexity reduction approaches. Compared with [19]–[22], [34], [35], our approach can drastically reduce the complexity of the DNN with better rate-distortion performance. It is in accord with literatures [24]–[26], indicating that a properly pruned DNN typically outperforms an un-pruned shallower network with similar complexity. (2) We also design a control algorithm, enabling the complexity of ETH-CNN to be continuously adjustable according to the computation resources

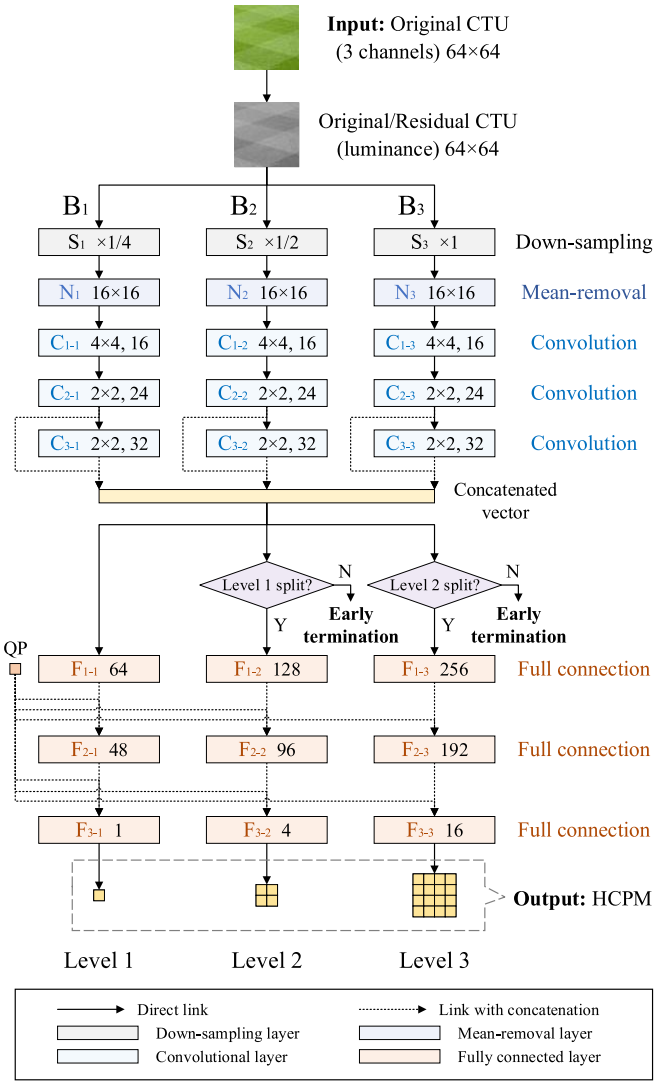

Fig. 2. Structure of ETH-CNN. All layers are illustrated by different colors of boxes, each containing the layer name followed by its configuration. First, for down-sampling layers, the value "$\times s$" indicates that the luminance CTU is down-sampled with the scale of s along both width and height dimensions, via average pooling. Next, mean-removal is applied to the down-sized CTUs in three branches, all conducted in 16×16 non-overlapping blocks. Then, "$p \times p$, q" for each convolutional layer represents q output channels with non-overlapping $p \times p$ sized kernels. Finally, the value after the name of each FC layer means the number of its output features.

in practice. As a result, the average complexity for predicting the CTU partition can approach to any given target, with high control accuracy and minimal RD loss.

III. Acceleration of CTU Partition

A. Preliminary

The proposed approach for CTU partition acceleration is based on the previous work [23]. In [23], a deep ETH-CNN model has been proposed for predicting the CU partition of each CTU, as illustrated in Figure 2. For intra-mode, the input of ETH-CNN is a luminance CTU from its raw frame. For inter-mode, the input luminance CTU is extracted from its residual frame generated by a quick pre-encoding process,

TABLE I
SUMMARY OF NOTATIONS

Notation	Description	Notation	Description
ETH-CNN structure			
l	Level of CU partition	N_l	Mean-removal layer
k	Index of convolutional or fully connected layer	C_{k-l}	Convolutional layer
B_l	Branch	F_{k-l}	Fully connected layer
S_l	Down-sampling layer		
Pre-setting of pruning algorithm			
L	A trainable layer	$n_{\mathrm{P}}(\mathrm{L})$	Number of WPs at layer L with pruning
$\mathcal{L}$	Set of all trainable layers in ETH-CNN	N_{P}	Total number of WPs in ETH-CNN with pruning
$n(\mathrm{L})$	Number of WPs at layer L without pruning	$r(\mathrm{L})$	Retention ratio of WPs for layer L
N	Total number of WPs in ETH-CNN without pruning	r_{P}	Global retention ratio of WPs throughout ETH-CNN
α	An intermediate parameter to determine $\mathcal{R}$	$\mathcal{R}$	Set of retention ratios for all trainable layers
Implementation of pruning algorithm			
$\mathcal{R}_0$	Set of initial retention ratios of WPs for all trainable layers.	f_{P}	Pruning frequency
$r_0(\mathrm{L})$	Initial retention ratio of WPs for layer L	$\mathbf{\Theta}_0(\mathrm{L})$	Initial WPs for layer L
$r_{\mathrm{C}}(\mathrm{L})$	Current retention ratio of WPs for layer L	$\mathbf{\Theta}(\mathrm{L})$	Optimized WPs for layer L
H	Total number of iterations	h	Current number of iterations
H_{P}	Number of iterations to achieve target retention ratio of WPs		
Evaluation of pruning algorithm			
$t(\mathcal{R})$	Running time of ETH-CNN under $\mathcal{R}$	i	Index of global retention ratio
$b(\mathcal{R})$	BD-BR value under $\mathcal{R}$	$\tilde{t}_i$	Normalized running time (NRT) of ETH-CNN model
t_i	Absolute running time of ETH-CNN model	$\tilde{b}_i$	Normalized BD-BR value (NBV)
b_i	Absolute BD-BR value		
Complexity control algorithm			
$\tilde{T}_{\mathrm{C}}$	Target NRT of ETH-CNN	M	Number of CTUs in a frame
$\tilde{T}_{\mathrm{E}}$	Estimated NRT of ETH-CNN averaged over all CTUs for a frame	u	Index of frame
$\tilde{B}_{\mathrm{E}}$	Estimated NBV for a frame	U	Total number of frames
$\boldsymbol{b}$	NBV for different ETH-CNN models	λ, μ	Regulation parameters in CTU-level control
$\boldsymbol{t}$	NRT of ETH-CNN for different ETH-CNN models	β	Decay rate for moving average in frame-level control
$\boldsymbol{m}$	Proportion of CTUs for different ETH-CNN models	$\tilde{T}_{\mathrm{C}}^{*}$	Pre-adjusted value of $\tilde{T}_{\mathrm{C}}$ in frame-level control
$\boldsymbol{m}_{\mathrm{Q}}$	Quantized values of $\boldsymbol{m}$	$\tilde{T}_{\mathrm{S}}$	Summed value of $\tilde{T}_{\mathrm{E}}$ in frame-level control
$\boldsymbol{m}_{\mathrm{A}}$	Further adjusted values of $\boldsymbol{m}$		

considering that the residue itself contains temporal correlation across frames. At the beginning of ETH-CNN, the input CTU is preprocessed with down-sampling and mean removal,[2] to reduce the computational complexity. Next, the preprocessed data flow through three convolutional layers to extract low-level textural features. The preprocessing and convolution operations are conducted in three parallel branches $\{\mathrm{B}_l\}_{l=1}^{3}$, enabling feature extraction at various spatial scales. Here, $l \in \{1, 2, 3\}$ represents the level of CU partition which decides whether to split 64×64, 32×32 or 16×16 CUs, respectively. Then, the feature maps from these three branches are concatenated into a vector, synthesizing the features from the whole CTU. Afterwards, the concatenated vector is processed with FC layers in the three branches $\{\mathrm{B}_l\}_{l=1}^{3}$, to extract the high-level features. Because the QP also has a significant influence on the CU partition, it is added as an external feature at the last two FC layers, which enables ETH-CNN to adapt to various QP values. Finally, the structured results in the form of the proposed hierarchical CU partition map (HCPM) can be obtained. Note that $\{\mathrm{B}_l\}_{l=1}^{3}$ correspond to levels 1, 2 and 3 of HCPM, where each binary label represents the probability for splitting a 64×64, 32×32 or 16×16 CU, respectively.

Reference [23] has made considerable efforts to reduce the computational complexity, including: (1) the efficient HCPM can run the trained model only once to predict the CU partition in the whole CTU, (2) all convolutional operations are non-overlapping, indicating that convolution is performed only once for each pixel in the input feature map, and (3) the early termination mechanism may skip the calculation of certain FC layers when the CUs at level 1 or 2 are non-split. However, the ETH-CNN model in [23] can hardly be implemented in real time, especially for high-resolution videos. To solve this problem, we propose a network pruning approach for accelerating ETH-CNN, which can drastically simplify the ETH-CNN model with negligible RD loss.

[2]Compared with the exact descriptions in [23], the order of two processes (i.e., mean removal and down-sampling) is reversed in this paper. Such reversion is able to reduce the computational complexity of preprocessing layers, while keeping the results unchanged.

B. Problem Statement

Before introducing our network pruning approach, we first investigate the computational complexity of ETH-CNN. Table II shows the configuration of ETH-CNN, including the number of floating-point operations at all layers. Considering that the performance of ETH-CNN mainly depends on its trainable WPs, the operations in this table are tabulated into the following two categories.

- **Weight-dependent operations (WDOs)**: the operations that are in proportion to WPs, i.e., each redundant WP being pruned can result in the reduction of network complexity. WDOs contain the receptive fields in feature maps convolved by their corresponding kernels at convolutional layers, and the feature vectors multiplied by weight matrices at FC layers.
- **Weight-independent operations (WIOs)**: the operations that are not in proportion to WPs. The complexity of the network may not be reduced when an individual WP is pruned. WIOs include the input feature maps added together to generate each output feature map at convolutional layers, and also contain the bias parameters and activation functions applied to the output

TABLE II
COMPUTATIONAL COMPLEXITY OF ETH-CNN

Layer		Size of input features	Size of output features	Number of WPs	Number of WDOs		Number of WIOs		
					Add.	Mult.	Add.	Mult.	Act.
Down-sampling	S_1	64×64×1	16×16×1	0	0	0	3,840	256	0
	S_2	64×64×1	32×32×1	0	0	0	3,072	1,024	0
	S_3	64×64×1	64×64×1	0	0	0	0	0	0
Mean-removal	N_1	16×16×1	16×16×1	0	0	0	511	1	0
	N_2	32×32×1	32×32×1	0	0	0	1,020	4	0
	N_3	64×64×1	64×64×1	0	0	0	8,176	16	0
Convolution	C_{1-1}	16×16×1	4×4×16	256	3,840	4,096	256	0	256
	C_{1-2}	32×32×1	8×8×16	256	15,360	16,384	1,024	0	1,024
	C_{1-3}	64×64×1	16×16×16	256	61,440	65,536	4,096	0	4,096
	C_{2-1}	4×4×16	2×2×24	1,536	4,608	6,144	1,536	0	96
	C_{2-2}	8×8×16	4×4×24	1,536	18,432	24,576	6,144	0	384
	C_{2-3}	16×16×16	8×8×24	1,536	73,728	98,304	24,576	0	1,536
	C_{3-1}	2×2×24	1×1×32	3,072	2,304	3,072	768	0	32
	C_{3-2}	4×4×24	2×2×32	3,072	9,216	12,288	3,072	0	128
	C_{3-3}	8×8×24	4×4×32	3,072	36,864	49,152	12,288	0	512
Full connection	F_{1-1}	2688	64	172,032	171,968	172,032	64	0	64
	F_{1-2}	2688	128	344,064	343,936	344,064	128	0	128
	F_{1-3}	2688	256	688,128	687,872	688,128	256	0	256
	F_{2-1}	65	48	3,120	3,072	3,120	48	0	48
	F_{2-2}	129	96	12,384	12,288	12,384	96	0	96
	F_{2-3}	257	192	49,344	49,152	49,344	192	0	192
	F_{3-1}	49	1	49	48	49	1	0	1
	F_{3-2}	97	4	388	384	388	4	0	4
	F_{3-3}	193	16	3,088	3,072	3,088	16	0	16
Total		-	-	1,287,189	1,497,584	1,552,149	71,184	1,301	8,869*

All calculation in this table is based on the analysis in [36], [37].
For down-sampling, mean-removal and convolutional layers, the size of input/output features "$H \times W \times C$" represents C feature maps with the size of each feature map being $H \times W$. For fully connected layers, the size of input/output features is the number of features in the input/output vector.
* The activation operations contain 8,848 leaky rectified linear units (LReLU) [38] with similar computational complexity to typical floating-point multiplications, and 21 sigmoid functions adding only marginal complexity overhead to ETH-CNN. For simplicity, the complexity of all activation functions is regarded comparable to other operations.

feature maps/vectors at both convolutional and FC layers. In addition, all operations at preprocessing layers are regarded as WIOs, because there is no WP.

Table II indicates that WDOs at trainable layers (i.e., convolutional and FC layers) account for over 97.4% of floating-point operations, which is far more than those accounted for by WIOs. Therefore, we propose pruning redundant WPs at these trainable layers to significantly reduce the overall complexity of ETH-CNN. As shown in Table II, each convolutional or FC layer is named as C_{k-l} or F_{k-l}, respectively, where k is the index of a layer at level l. Here, the set of all trainable layers is denoted as

$$\mathcal{L} = \{\{C_{k-l}\}_{k=1}^{3}\}_{l=1}^{3} \bigcup \{\{F_{k-l}\}_{k=1}^{3}\}_{l=1}^{3}. \tag{1}$$

For each trainable layer $L \in \mathcal{L}$ with $n(L)$ WPs, a preset retention ratio $r(L)$ is applied to its WPs. Compared with only one global retention ratio, presetting $r(L)$ for all trainable layers helps to accurately adjust the WP density for different parts of the network, which reduces the randomness during the pruning process. As a result, $n_P(L) = \lceil n(L)r(L) \rceil$ WPs remain at layer L ("$\lceil \cdot \rceil$" represents the top integral function). Providing a combination of retention ratios $\mathcal{R} = \{r(L)|L \in \mathcal{L}\}$ for all layers, the average complexity-RD performance can be measured on a sufficient number of validation video sequences, i.e., the running time of ETH-CNN $t(\mathcal{R})$ and the Bjøntegaard delta bit-rate (BD-BR) $b(\mathcal{R})$. Therefore, the problem of ETH-CNN acceleration can be modeled as follows,

$$\min_{R} \; t(\mathcal{R}) \quad \text{s.t. } b(\mathcal{R}) \leq b_{\text{THR}}, \tag{2}$$

where the threshold b_{THR} represents the largest acceptable BD-BR for video compression. In the following subsection, we focus on the scheme of network pruning, for solving the problem of (2), such that the running time of ETH-CNN can be reduced with desirable RD performance. Compared with other network acceleration techniques [26]–[29], [39]–[41], we choose to prune the ETH-CNN model on weight parameters, so that the computational complexity of ETH-CNN is flexibly scalable without changing the network structure. Moreover, the scalability is essential for our control algorithm (to be presented in Section IV), which enables the average complexity of ETH-CNN to be continuously adjustable in practice.

C. Network Pruning Scheme

Providing the configuration of ETH-CNN, the total number of WPs without pruning is

$$N = \sum_{L \in \mathcal{L}} n(L). \tag{3}$$

After network pruning with retention ratios $\mathcal{R}$, the total remaining number of WPs N_P can be calculated by

$$N_P = \sum_{L \in \mathcal{L}} n_P(L). \tag{4}$$

Recall that $n(\mathrm{L})$ and $n_{\mathrm{P}}(\mathrm{L})$ denote the numbers of WPs at layer L before and after pruning, respectively. Thus, the global retention ratio of WPs is

$$r_{\mathrm{P}} = \frac{N_{\mathrm{P}}}{N}. \tag{5}$$

For a specific global retention ratio r_{P}, the following task is to determine the whole ratio set $\mathcal{R}$ that contains the values of the retention ratios $r(\mathrm{L})$ of layer L.

As shown in Table II, the number of WPs $n(\mathrm{L})$ varies greatly by layer and level, which should be adaptive to the specific components of ETH-CNN. We choose to set a larger $r(\mathrm{L})$ to a layer with fewer WPs. It is because fewer WPs tend to be more sensitive to pruning, in case that the network structure is broken without sufficient connections among adjacent layers. Additionally, in the proposed ETH-CNN, the prediction accuracy at level 1 is the most important, for alleviating the error propagation downward across levels. Among all three levels, the number of WPs for level 1 is fewer than that for level 2 or 3. Thus, setting larger $r(\mathrm{L})$ to layers with fewer WPs, also helps to maintain the prediction accuracy of level 1. The above observations indicate that $r(\mathrm{L})$ should have a negative relationship to $n(\mathrm{L})$. To this end, we introduce $\alpha \in (0, 1]$ to adjust the $r(\mathrm{L})$ for all layers at all three levels, formulated as

$$n_{\mathrm{P}}(\mathrm{L}) = [n(\mathrm{L})]^{\alpha} \quad \text{s.t. } \frac{1}{N}\sum_{\mathrm{L}\in\mathcal{L}} n_{\mathrm{P}}(\mathrm{L}) = r_{\mathrm{P}}. \tag{6}$$

In (6), $\alpha = 1$ indicates all remaining WPs without pruning, while $\alpha \to 0$ means that the network is sparse with few WPs remaining. Moreover, for an arbitrary $\alpha \in (0, 1]$

$$r(\mathrm{L}) = \frac{[n(\mathrm{L})]^{\alpha}}{n(\mathrm{L})} \tag{7}$$

decreases with larger $n(\mathrm{L})$, satisfying the above negative relationship. Considering that r_{P} is an increasing function of α, the value of α in (6) can be calculated with the bi-section method [42] when providing a specific r_{P}. Then, the retention ratios $\mathcal{R}$ for all individual layers can be obtained.

After $\mathcal{R}$ is determined, a corresponding model of ETH-CNN can be generated. As a precondition, the initial retention ratios of WPs are all 1 (training from scratch) or inherited from those used in another model (fine-tuning). Then, the WPs at all layers $\mathcal{L}$ are pruned during the iterations to update the WPs in ETH-CNN. The procedure of network pruning is detailed in Algorithm 1. During this process, the retention ratio $r_{\mathrm{C}}(\mathrm{L})$ for each layer exponentially decreases from its initial value $r_0(\mathrm{L})$ to the target $r(\mathrm{L})$. Without the abrupt change of $r_{\mathrm{C}}(\mathrm{L})$, such a gradual decrease helps to recover the performance of ETH-CNN. As a result, a pruned ETH-CNN model is obtained with an accurate ratio of WPs remaining.

Algorithm 1 Network Pruning for ETH-CNN

Input:
$\mathcal{X}$: Training data.
$\mathcal{R}_0 = \{r_0(\mathrm{L})\}_{L\in\mathcal{L}}$: Initial retention ratios for all trainable layers.
$\mathcal{R} = \{r(\mathrm{L})\}_{L\in\mathcal{L}}$: Target retention ratios for all trainable layers.
H: Total number of iterations.
H_{P}: Number of iterations to achieve target $\mathcal{R}$ ($1 \le H_{\mathrm{P}} \le H$).
f_{P}: Pruning frequency, i.e., the interval number of iterations between two adjacent pruning processes.
$\{\mathbf{\Theta}_0(\mathrm{L})\}_{L\in\mathcal{L}}$: Initial WPs for all trainable layers (optional).
Output:
$\{\mathbf{\Theta}(\mathrm{L})\}_{L\in\mathcal{L}}$: Optimized WPs for all trainable layers.

```
if training from scratch then
    Assign {Θ(L)}_{L∈L} with Xavier initialization [43].
else
    Assign {Θ(L)}_{L∈L} ← {Θ_0(L)}_{L∈L}.
end if
for 1 ≤ h ≤ H do
    Update non-zero values in {Θ(L)}_{L∈L} according to the learning rate and the gradients derived from one training batch in X.
    if mod(h, f_P) == 0 then
        for L ∈ L do
            Calculate the current retention ratio for this layer:
                r_C(L) = { r_0(L)[r(L)/r_0(L)]^{h/H_P},   1 ≤ h ≤ H_P
                         { r(L),                          H_P < h ≤ H        (8)
            Keep ⌈n(L) r_C(L)⌉ WPs in Θ(L) with maximum absolute values.
            Set other WPs in Θ(L) to 0.
        end for
    end if
end for
Return all optimized WPs {Θ(L)}_{L∈L}.
```

D. Scheme Evaluation

To evaluate the complexity and RD performance of the pruning scheme, we first generate both un-pruned and pruned ETH-CNN models, using the database [23] for CU partition of HEVC (named CPH database). The CPH database contains 2,000 raw images and 200 raw video sequences,[3] with the corresponding labels of CU partition encoded by HM 16.5 [4] at four QP values $\{22, 27, 32, 37\}$. Among them, 1,700 images and 164 video sequences are used for training ETH-CNN models, while other 100 images and 18 video sequences are validation data for evaluation in this section. The rest 200 images and 18 sequences are the test data, to be used in Section V. Note that the training, validation and test sets in this database are all non-overlapping.

With the sufficient data, we train ETH-CNN models with seven global retention ratios $r_{\mathrm{P}} \in \{100\%, 20\%, 5\%, 1\%, 0.5\%, 0.2\%, 0.1\%\}$ at both intra- and inter-modes, using the All Intra (AI) (file *encoder_intra_main.cfg*) and the LDP (file *encoder_lowdelay_P_main.cfg*) configurations [44], respectively. For either intra- or inter-mode, one un-pruned ETH-CNN model with $r_{\mathrm{P}} = 100\%$ (i.e., all WPs remaining) is first trained from scratch. The learning rate is initially set to 0.01 and then decreased by 1% exponentially every $2,000$ iterations. There are in total $1,000,000$ iterations. Then, the other six models are generated sequentially, according to the descending order of r_{P}, i.e., the model for the i-th r_{P} value (except $i = 1$) is fine-tuned from the model for the $(i-1)$-th r_{P} value. Here, $i \in \{1, 2, \ldots, 7\}$ represents the index of a global retention ratio, among the seven different values of r_{P}. When fine-tuning for each model, the initial learning rate is set to be 0.01 and then decreases by 1% exponentially every $1,000$ iterations. Here, the total number of iterations is $H = 500,000$, in which $H_{\mathrm{P}} = 100,000$ iterations are required

[3]The database for inter-mode has been enlarged from 111 sequences in [23] to 200 sequences in this paper.

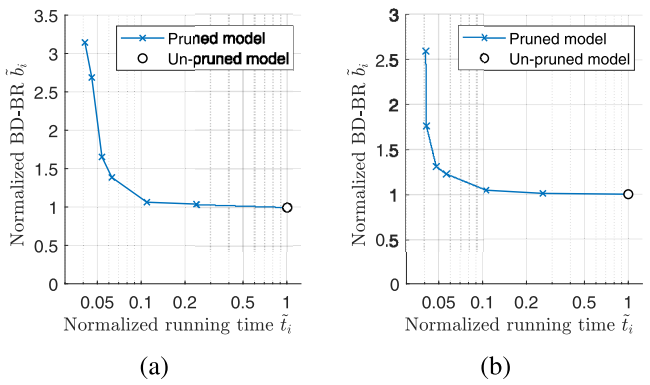

Fig. 3. Complexity-RD performance for ETH-CNN. (a) AI configuration. (b) LDP configuration. The average BD-BR value for encoding with un-pruned ETH-CNN model b_1 is 1.549% and 1.487% at the AI and LDP configurations, respectively. The absolute running time of un-pruned ETH-CNN model t_1 is 1.053ms at either configuration.

to achieve the target retention ratios. The pruning frequency f_{P} is set to be 10. Other settings follow those in [23].

After obtaining the models with all seven r_{P} values, we evaluate the complexity-RD performance by integrating each model into the accelerated HM encoder in [23]. Finally, we record the normalized running time (NRT) of ETH-CNN $\tilde{t}_i$ and the normalized BD-BR value (NBV) $\tilde{b}_i$. These two metrics reflect the complexity and RD performance of the pruning scheme, respectively, and they are calculated as follows

$$\begin{cases} \tilde{t}_i = \dfrac{t_i}{t_1} \\ \tilde{b}_i = \dfrac{b_i}{b_1}. \end{cases} \tag{9}$$

In (9), t_i represents the absolute running time of the ETH-CNN model for one CTU, with the i-th r_{P} value, averaged over all CTUs of 100 validation images or 18 validation sequences. Correspondingly, b_i is the average BD-BR value over these validation data. Note that the BD-BR and running time are both normalized for evaluating the relative performance of each pruned ETH-CNN model, compared over the un-pruned ETH-CNN model. Moreover, both the normalized metrics are with similar magnitudes in terms of their values, necessary for optimization process in our complexity control algorithm (to be presented in Section IV). Figure 3 illustrates the changing tendency of $\tilde{t}_i$ and $\tilde{b}_i$ at various global retention ratios. For the intra-mode statistics, the un-pruned ETH-CNN model requires 1.053ms per CTU with negligible 1.549% of BD-BR on average. When an accelerated setting is applied (e.g., $r_{\mathrm{P}} = 0.5\%$), the running time of ETH-CNN is shortened to only 43.3μs per CTU, meaning that the CTU partition of HEVC can be predicted in real time for 1080p video at 45 frames per second. Meanwhile, the average 2.559% BD-BR is still acceptable during encoding. For inter-mode, similar statistical results are achieved. Given the above analysis, the proposed pruning scheme can accelerate the CTU partition in HEVC with a significantly simplified ETH-CNN model, while maintaining desirable RD performance.

IV. Complexity Control for CTU Partition

In the previous section, we proposed seven accelerated ETH-CNN models for accelerating CTU partition in HEVC. Based on such acceleration, a control algorithm is proposed in this section for more practical use, which enables the computational complexity of ETH-CNN to be continuously adjustable while still maintaining the RD performance. To this end, the NRT of ETH-CNN needs to be controlled at the CTU level within each individual frame. Moreover, a frame-level control algorithm is also designed, guaranteeing the overall control accuracy for a whole video sequence.

A. CTU-Level Control

As analyzed in Section III-D, the NRT of ETH-CNN ranges from $\tilde{t}_7$ to $\tilde{t}_1$ under the 7 retention ratios. By applying different retention ratios of ETH-CNN to the CTUs across a frame, the average NRT of ETH-CNN can be more flexibly controlled within this range. Assuming that the target NRT of ETH-CNN is $\tilde{T}_{\mathrm{C}}$ (satisfying $\tilde{T}_{\mathrm{C}} \in [\tilde{t}_7, \tilde{t}_1]$) for a frame, the goal is to ensure the average NRT of ETH-CNN close to $\tilde{T}_{\mathrm{C}}$ with negligible RD loss, formulated as

$$\min_{\boldsymbol{m}} \ \tilde{B}_{\mathrm{E}} \quad \text{s.t. } \tilde{T}_{\mathrm{E}} = \tilde{T}_{\mathrm{C}}. \tag{10}$$

In (10), vector $\boldsymbol{m} = [m_1, m_2, \ldots, m_7]^{\mathrm{T}}$ represents the quantitative proportion of CTUs at 7 retention ratios, satisfying $\sum_{i=1}^{7} m_i = 1$ and $m_i \geq 0, \forall i$. Along with the change of $\boldsymbol{m}$, $\tilde{B}_{\mathrm{E}}$ denotes the estimated NBV for this frame, and $\tilde{T}_{\mathrm{E}}$ denotes the estimated NRT of ETH-CNN averaged over all CTUs of this frame. Let vectors $\boldsymbol{b} = [\tilde{b}_1, \tilde{b}_2, \ldots, \tilde{b}_7]^{\mathrm{T}}$ and $\boldsymbol{t} = [\tilde{t}_1, \tilde{t}_2, \ldots, \tilde{t}_7]^{\mathrm{T}}$ be the NBV and NRT at 7 retention ratios of ETH-CNN. Then, $\tilde{B}_{\mathrm{E}}$ and $\tilde{T}_{\mathrm{E}}$ can be calculated as

$$\begin{cases} \tilde{B}_{\mathrm{E}} = \boldsymbol{b}^{\mathrm{T}}\boldsymbol{m} \\ \tilde{T}_{\mathrm{E}} = \boldsymbol{t}^{\mathrm{T}}\boldsymbol{m}. \end{cases} \tag{11}$$

By substituting (11) into (10), we have the following formulation

$$\min_{\boldsymbol{m}} \ \|\boldsymbol{b}^{\mathrm{T}}\boldsymbol{m}\|_2^2 \quad \text{s.t. } \boldsymbol{t}^{\mathrm{T}}\boldsymbol{m} = \tilde{T}_{\mathrm{C}}. \tag{12}$$

Using the Lagrange multiplier [45], the optimization problem (12) can be reformulated as

$$\min_{\boldsymbol{m}} \ \|\boldsymbol{t}^{\mathrm{T}}\boldsymbol{m} - \tilde{T}_{\mathrm{C}}\|_2^2 + \lambda\|\boldsymbol{b}^{\mathrm{T}}\boldsymbol{m}\|_2^2, \tag{13}$$

where λ is the multiplier for adjusting the importance of RD performance over the complexity control accuracy. Equation (13) is a least square optimization problem, and it can be solved using the ridge regression [46]. The closed-form solution to (13) is

$$\boldsymbol{m} = (\boldsymbol{t}\boldsymbol{t}^{\mathrm{T}} + \lambda\boldsymbol{b}\boldsymbol{b}^{\mathrm{T}} + \mu\boldsymbol{I})^{-1}\boldsymbol{t} \cdot \tilde{T}_{\mathrm{C}}, \tag{14}$$

where μ is the regularization coefficient. Here, vector $\boldsymbol{m}$ is continuously valued, representing the quantitative proportion of CTUs.

Considering a frame containing a finite number of CTUs, the elements in $\boldsymbol{m}$ should be discretely quantized in practice. Thus, for a frame containing M CTUs, another vector $\boldsymbol{m}_{\mathrm{Q}}$

with the same length as $\boldsymbol{m}$ is calculated, which represents the quantized proportion of CTUs under 7 retention ratios

$$\boldsymbol{m}_{\mathrm{Q}} = \max\{0, \frac{1}{M}\lceil M\boldsymbol{m} - \frac{1}{2}\rceil\}. \quad (15)$$

In the above equation, each element of $\boldsymbol{m}_{\mathrm{Q}}$ is assigned a positive multiple of $\frac{1}{M}$ nearest to the corresponding element in $\boldsymbol{m}$. After discrete quantization, another constraint is still to be considered, i.e., the summation of elements in $\boldsymbol{m}_{\mathrm{Q}} = [m_{\mathrm{Q},1}, m_{\mathrm{Q},2}, \ldots, m_{\mathrm{Q},7}]^{\mathrm{T}}$ may be not exactly equivalent to 1. Thus, we adjust the last one or several elements in $\boldsymbol{m}_{\mathrm{Q}}$ to satisfy the one-sum constraint $\sum_{i=1}^{7} m_{\mathrm{Q},i} = 1$, because the last elements in $\boldsymbol{m}_{\mathrm{Q}}$ represent the proportion of CTUs with the shortest NRT of ETH-CNN and have the least impact on the estimated $\tilde{T}_{\mathrm{E}}$. With this adjustment, the proportion of CTUs can be formulated as

$$\boldsymbol{m}_{\mathrm{A}} = [m_{\mathrm{Q},1}, \cdots, m_{\mathrm{Q},i'}, (1 - \sum_{i=1}^{i'} m_{\mathrm{Q},i}), \overbrace{0, \cdots, 0}^{(6-i')\ \text{zeros}}]^{\mathrm{T}}, \quad (16)$$

where i' denotes the largest index satisfying $\sum_{i=1}^{i'} m_{\mathrm{Q},i} < 1$. Note that $\boldsymbol{m}_{\mathrm{A}}$ is appended with $(6 - i')$ zeros to ensure that $\boldsymbol{m}_{\mathrm{A}}$ and $\boldsymbol{m}_{\mathrm{Q}}$ have the same length (=7). As such, all elements in one-sum vector $\boldsymbol{m}_{\mathrm{A}}$ are positive multiples of $\frac{1}{M}$. Therefore, $\boldsymbol{m}_{\mathrm{A}}$ is practically used as an available solution of $\boldsymbol{m}$ in (10), which represents the proportion of CTUs under all 7 retention ratios of ETH-CNN. Correspondingly, the numbers of CTUs under 7 retention ratios can be solved within a frame.

B. Frame-Level Control

An estimation error of NRT for the ETH-CNN model always exists between $\tilde{T}_{\mathrm{E}}$ and $\tilde{T}_{\mathrm{C}}$, which is accumulative across multiple frames. Therefore, we further propose a frame-level control algorithm for alleviating the error accumulation, thereby enhancing the stability of complexity control. Assume that the global target NRT of ETH-CNN is $\tilde{T}_{\mathrm{C}}$ for all frames throughout the video sequence. For the 1st frame, i.e., frame index $u = 1$, the target NRT of ETH-CNN $\tilde{T}_{\mathrm{C}}(u)$ is naturally set to be $\tilde{T}_{\mathrm{C}}$, and then the estimated NRT of ETH-CNN for this frame $\tilde{T}_{\mathrm{E}}(u)$ can be correspondingly calculated. Then, both the target and estimated NRT for all subsequent frames can be adaptively adjusted with the exponential moving average, as detailed in Algorithm 2.

In this algorithm, the target NRT $\tilde{T}^{*}_{\mathrm{C}}(u)$ for a frame is pre-adjusted according to the summation of estimated NRT $\tilde{T}_{\mathrm{E}}(u)$ over all previous frames (on Line 5 of Algorithm 2). Such design avoids the estimation error accumulated across frames. Additionally, with the moving average of $\tilde{T}^{*}_{\mathrm{C}}(u)$ over the current frame and previous frames (Lines 6 and 7 of Algorithm 2), the target NRT $\tilde{T}_{\mathrm{C}}(u)$ is more stable. As a result, both the target and estimated NRT $\tilde{T}_{\mathrm{C}}(u)$ and $\tilde{T}_{\mathrm{E}}(u)$ are close to the global target NRT $\tilde{T}_{\mathrm{C}}$ as frame index u increases, and thus the frame-level complexity control for ETH-CNN is achieved. It is worth mentioning that Algorithm 1 aims to reduce the computational complexity of ETH-CNN [23], while Algorithm 2 is used to control the reduced complexity to a target, based on the complexity reduction of Algorithm 1.

Algorithm 2 Frame-Level Complexity Control for ETH-CNN

Input:
$\tilde{T}_{\mathrm{C}}$: Global target NRT of ETH-CNN.
U: Number of frames in the whole video sequence.
$\beta \in (0, 1]$: Exponential decay rate for moving average.
Output:
$\{\tilde{T}_{\mathrm{C}}(u)\}_{u=1}^{U}$ and $\{\tilde{T}_{\mathrm{E}}(u)\}_{u=1}^{U}$: Target and estimated NRT of ETH-CNN for all frames, where u is the frame index.
1: Assign $\tilde{T}_{\mathrm{C}}(1) \leftarrow \tilde{T}_{\mathrm{C}}$.
2: Calculate $\tilde{T}_{\mathrm{E}}(1)$ using the CTU-level control algorithm.
3: Initialize the summation of estimated NRT for ETH-CNN $\tilde{T}_{\mathrm{S}} \leftarrow \tilde{T}_{\mathrm{E}}(1)$.
4: **for** $2 \le u \le U$ **do**
5: Pre-adjust the target NRT for frame u: $\tilde{T}^{*}_{\mathrm{C}}(u) \leftarrow u\tilde{T}_{\mathrm{C}} - \tilde{T}_{\mathrm{S}}$.
6: Calculate the moving average of target NRT: $\tilde{T}^{*}_{\mathrm{C}}(u) \leftarrow \beta\tilde{T}_{\mathrm{C}}(u-1) + (1-\beta)\tilde{T}^{*}_{\mathrm{C}}(u)$.
7: Calculate the bias-corrected target NRT: $\tilde{T}_{\mathrm{C}}(u) \leftarrow \frac{\tilde{T}^{*}_{\mathrm{C}}(u)}{1-\beta^{u-1}}$.
8: Calculate $\tilde{T}_{\mathrm{E}}(u)$ using the CTU-level control algorithm.
9: Update the summation of estimated NRT: $\tilde{T}_{\mathrm{S}} \leftarrow \tilde{T}_{\mathrm{S}} + \tilde{T}_{\mathrm{E}}(u)$.
10: **end for**
11: Return $\{\tilde{T}_{\mathrm{C}}(u)\}_{u=1}^{U}$ and $\{\tilde{T}_{\mathrm{E}}(u)\}_{u=1}^{U}$.

V. Experimental Results

In this section, we evaluate the performance of the proposed approach in terms of both acceleration and complexity control. Section V-A presents the experimental settings. Sections V-B and V-C evaluate the acceleration and complexity-RD performance, by comparing our approach with other four state-of-the-art approaches [16], [19], [22], [23]. Section V-D evaluates the accuracy of complexity control. In Section V-E, we transfer our approach to other configurations and standards. Finally, the ablation study is analyzed in Section V-F.

A. Experimental Settings

For performance evaluation, we implemented our approach in the HEVC reference software HM 16.5 [4]. The experiments were conducted at both inter- and intra-modes of HEVC, using the LDP (file *encoder_lowdelay_P_main.cfg*) and the AI (file *encoder_intra_main.cfg*) configurations [44], respectively. All 18 JCT-VC test sequences from Classes A∼E [47] were encoded. Here, four QP values {22, 27, 32, 37} were chosen to compress the test sequences. For acceleration performance, the running time of all approaches was recorded during encoding, and the BD-BR was calculated to assess the RD performance. For complexity control, four target NRT values $\tilde{T}_{\mathrm{C}} \in \{50\%, 20\%, 10\%, 5\%\}$ were set at each mode and each QP value, and then the gap between measured NRT of ETH-CNN (denoted as $\tilde{T}_{\mathrm{M}}$) and $\tilde{T}_{\mathrm{C}}$ was used to evaluate the control accuracy. In CTU-level control, the Lagrange multiplier λ and the regularization coefficient μ of (14) were set to be 10^{-4} and 10^{-3}, respectively; in frame-level control, the exponential decay rate β (mentioned in Algorithm 2) was set to 0.5. The hyper-parameters L, μ and β were all tuned over the validation set of the CPH database [23] for the desirable performance. All experiments were conducted on a computer with an Intel (R) Core (TM) i7-8700K CPU @3.2 GHz, 64 GB RAM and the Ubuntu 16.04 (64-bit) operating system. Note that only single CPU thread was used to measure the running time of all approaches.

TABLE III
COMPARISON OF COMPLEXITY REDUCTION APPROACHES FOR HEVC (LDP)

Approach		Class A 2560×1600		Class B 1920×1080		Class C 832×480		Class D 416×240		Class E 1280×720		Overall				
		Model time (ms)	BD-BR (%)	Model time (ms)	BD-BR (%)	Model time (ms)	BD-BR (%)	Model time (ms)	BD-BR (%)	Model time (ms)	BD-BR (%)	Model time (ms)	Model speed (fps)	Encoding speed (fps)	Relative model time (%)	BD-BR (%)
[16]		70.36	5.638	100.99	4.026	67.60	3.408	40.32	2.166	63.09	3.797	68.47	14.60	0.0545	0.207	3.807
[22]		2615.30	5.040	1028.02	3.023	271.16	4.739	78.58	4.321	230.87	6.616	844.79	1.18	0.0564	2.558	4.748
[23]		1052.23	1.521	536.89	1.680	109.69	1.438	29.48	1.157	252.36	1.695	396.13	2.52	0.0622	1.199	1.498
Our	20% WPs	273.51	1.415	144.53	1.706	29.89	1.462	8.08	1.580	68.08	1.545	104.82	9.54	0.0640	0.317	1.542
	5% WPs	111.80	1.437	59.07	1.736	12.35	1.527	3.34	1.626	27.47	1.608	42.81	23.36	0.0644	0.130	1.587
	1% WPs	60.63	1.307	31.95	1.863	6.75	1.575	1.83	1.476	14.67	1.877	23.17	43.16	0.0645	0.070	1.620
	0.5% WPs	52.04	1.230	27.37	1.999	5.75	1.732	1.55	1.625	12.35	2.076	19.81	50.48	0.0649	0.060	1.733
	0.2% WPs	45.47	1.441	23.66	2.423	4.90	1.891	1.33	2.097	10.52	2.389	17.18	58.21	0.0651	0.052	2.048
	0.1% WPs	44.18	2.590	22.43	2.709	4.48	2.414	1.23	2.962	9.67	3.913	16.40	60.98	0.0652	0.050	2.918

TABLE IV
COMPARISON OF COMPLEXITY REDUCTION APPROACHES FOR HEVC (AI)

Approach		Class A 2560×1600		Class B 1920×1080		Class C 832×480		Class D 416×240		Class E 1280×720		Overall				
		Model time (ms)	BD-BR (%)	Model time (ms)	BD-BR (%)	Model time (ms)	BD-BR (%)	Model time (ms)	BD-BR (%)	Model time (ms)	BD-BR (%)	Model time (ms)	Model speed (fps)	Encoding speed (fps)	Relative model time (%)	BD-BR (%)
[19]		79.48	4.457	33.26	4.531	8.36	8.538	2.08	6.061	13.57	7.145	27.35	36.56	0.2406	0.259	6.146
[22]		4184.82	4.259	1682.26	3.560	425.85	4.146	111.15	2.721	599.19	6.391	1400.65	0.71	0.1912	13.241	4.215
[23]		1052.76	2.458	537.10	2.584	109.72	1.897	29.49	1.174	252.51	3.464	396.32	2.52	0.2398	3.747	2.315
Our	20% WPs	287.81	3.104	136.62	3.046	30.62	2.263	8.44	1.456	58.95	4.314	104.49	9.57	0.2556	0.988	2.836
	5% WPs	123.58	3.163	61.09	3.172	13.11	2.345	3.59	1.507	26.99	4.320	45.67	21.90	0.2603	0.432	2.901
	1% WPs	68.89	3.513	35.09	3.722	7.27	2.631	1.98	1.809	15.82	4.806	25.81	38.74	0.2631	0.244	3.296
	0.5% WPs	59.94	4.001	30.85	4.300	6.34	3.248	1.72	2.203	13.93	5.446	22.56	44.33	0.2648	0.213	3.840
	0.2% WPs	50.21	3.954	25.80	5.137	5.30	5.607	1.46	4.071	11.64	6.188	18.88	52.97	0.2662	0.178	4.992
	0.1% WPs	46.16	5.185	24.18	5.604	4.86	7.450	1.33	5.983	10.73	8.296	17.45	57.31	0.2677	0.165	6.504

B. Evaluation on Acceleration

First, we evaluate the acceleration performance via comparing the running time of our and other state-of-the-art approaches [16], [19], [22], [23]. Among these approaches, the deep ETH-CNN [23] and the six-layer CNN [22] can be used for both inter- and intra-modes, while the non-deep-learning approach [16] (i.e., content-adaptive CU size prediction with online and offline model training) is only for inter-mode and the four-layer CNN [19] only for intra-mode. Tables III and IV report the complexity of all approaches at inter- and intra-modes, respectively. Note that the results are averaged over all test sequences of each class at four QPs. As shown in Table III, the five pruned ETH-CNN models provide 3.78∼24.15 times acceleration compared with [23] at inter-mode. For all five classes of test sequences, the running time of pruned ETH-CNN models is significantly shorter than that of [22], and there exist at least four settings of our approach (i.e., 0.1%∼1% WPs) quicker than [16]. As can be seen from Table IV, there exist similar results for intra-mode coding, where the ETH-CNN model in [23] can be accelerated by 3.79∼22.71 times. Also, the running time of our approach is always shorter than that of [22] and four settings of our approach are quicker than [19]. In addition, the columns "Encoding speed" and "Model speed" in Tables III and IV indicate frames per second (fps) for the whole encoding process and the model, respectively. These results indicate that the pruned ETH-CNN models with $\leq$1% WPs can be implemented in real time with more than 30 fps, for both the LDP and AI configurations.

Next, we analyze the real encoding time of different approaches, as compared in Table V. This table reports the time overhead of each model (i.e., the DNN models in our approach, [22] and [19], or the CU size decision model in [16]) and the time for other processes. Note that the results in this table are averaged over all 18 JCT-VC test sequences [47] at QP values {22, 27, 32, 37}. For the LDP configuration, the inference time of our model decreases from 0.396 s to 0.016 s per frame, achieving real-time CTU partition. Similarly, the pruning of model at the AI configuration also leads to real-time CTU partition for faster encoding. Compared with all other approaches [16], [19], [22], our approach with a pruned model (i.e., WPs $\leq$ 20%) also consumes less encoding time at both LDP and AI configurations. In summary, our approach achieves desirable acceleration performance, faster than the other approaches.

TABLE V
ENCODING TIME ON HM 16.5

Config.	Approach	Real running time per frame (s)				
		Original HM	Model overhead	Other processes	Total encoding	Reduction
LDP	[16]	33.025	0.069	18.277	18.346	-14.680
	[22]		0.845	16.880	17.725	-15.301
	Our, 100% WPs		0.396	15.684	16.080	-16.945
	Our, 20% WPs		0.105	15.525	15.630	-17.395
	Our, 5% WPs		0.043	15.487	15.530	-17.496
	Our, 1% WPs		0.023	15.471	15.494	-17.531
	Our, 0.5% WPs		0.020	15.391	15.411	-17.614
	Our, 0.2% WPs		0.017	15.334	15.352	-17.674
	Our, 0.1% WPs		**0.016**	**15.319**	**15.336**	**-17.690**
AI	[19]	10.578	0.027	4.128	4.156	-6.422
	[22]		1.401	3.832	5.230	-5.345
	Our, 100% WPs		0.396	3.773	4.170	-6.409
	Our, 20% WPs		0.105	3.807	3.912	-6.667
	Our, 5% WPs		0.046	3.796	3.842	-6.737
	Our, 1% WPs		0.026	3.775	3.801	-6.777
	Our, 0.5% WPs		0.023	3.754	3.776	-6.802
	Our, 0.2% WPs		0.019	3.737	3.756	-6.822
	Our, 0.1% WPs		**0.018**	**3.719**	**3.736**	**-6.842**

C. Evaluation on Complexity-RD Performance

In this section, we evaluate the complexity-RD performance of our and other approaches. In addition to the running time,

TABLE VI

PREDICTION ACCURACY FOR THREE LEVELS OF CU PARTITION

Configuration		Each model in our approach						
		100% WPs	20% WPs	5% WPs	1% WPs	0.5% WPs	0.2% WPs	0.1% WPs
LDP	Accuracy: 64×64 CUs (%)	88.39	88.39	88.36	88.36	88.24	87.88	86.23
	Accuracy: 32×32 CUs (%)	83.49	83.49	83.27	83.04	82.86	82.66	82.39
	Accuracy: 16×16 CUs (%)	81.90	81.87	81.83	81.79	81.65	81.60	80.95
	Accuracy: average (%)	84.59	84.58	84.49	84.40	84.25	84.04	83.19
	Time overhead (ms)*	396.13	104.82	42.81	23.17	19.81	17.18	16.40
AI	Accuracy: 64×64 CUs (%)	93.96	93.95	93.83	93.70	93.32	92.86	92.68
	Accuracy: 32×32 CUs (%)	86.23	85.95	85.51	84.14	81.15	73.46	70.23
	Accuracy: 16×16 CUs (%)	78.64	78.60	78.44	77.36	75.94	72.73	69.46
	Accuracy: average (%)	86.28	86.16	85.93	85.07	83.47	79.69	77.45
	Time overhead (ms)*	396.32	104.49	45.67	25.81	22.56	18.88	17.45

*Average running time of model for each frame.

Tables III and IV also report the RD performance in terms of BD-BR, averaged over all test sequences of each class at four QPs. We can observe from Table III that pruning off 80%∼99.8% of WPs results in at most 2.048% BD-BR increase at inter-mode, i.e., 0.550% increment over the original ETH-CNN approach (1.498% of BD-BR) [23]. This indicates that our acceleration approach has slight degradation on RD performance. For all classes, at least four settings of our approach achieve both better complexity and RD performance than [16] and [22]. For intra-mode, similar results can be found in Table IV. The better performance of our approach benefits from the higher prediction accuracy of the accelerated ETH-CNN model, as shown in Table VI. Here, the results are averaged over all 18 test sequences at QPs {22, 27, 32, 37}. As the WPs are pruned off from 100% to 1% at the LDP configuration, the average drop of accuracy is only 0.19%, i.e., from 84.59% to 84.40%. Meanwhile, the time overhead for each frame is shortened from 396.13 ms to 23.17 ms on average, achieving 17 times of acceleration. For the AI configuration, there exist similar results. Therefore, the model in our approach can almost maintain its classification accuracy with significant time acceleration.

Moreover, Figure 4 illustrates the curves of complexity-RD performance under different acceleration ratios of ETH-CNN. Here, the results are averaged over all 18 test sequences at four QPs. For both inter- and intra-modes, RD performance always degrades with higher acceleration ratio, in accord with Tables III and IV. Nonetheless, such acceleration introduces only little RD loss compared with the un-pruned model [23], when the ETH-CNN model is not radically pruned, e.g., 17∼20 times of acceleration within BD-BR of 2% and 4% at inter- and intra-modes, respectively. For complexity control, all four settings under different target NRT values $\tilde{T}_C \in \{50\%, 20\%, 10\%, 5\%\}$ achieve high overall accuracy. Also, the red squares are always close to the blue curve in Figure 4, indicating that complexity control leads to almost no bit-rate overhead compared with the results without control.

To further show the change of performance across different sequences, Table VII tabulates the standard deviation of model time and BD-BR over all 18 JCT-VC test sequences. As seen in this table, the standard deviation of model time for our approach generally decreases at both LDP and AI configurations, when WPs are pruned off from 20% to 1%. In addition, the standard deviation values of BD-BR in our approach range

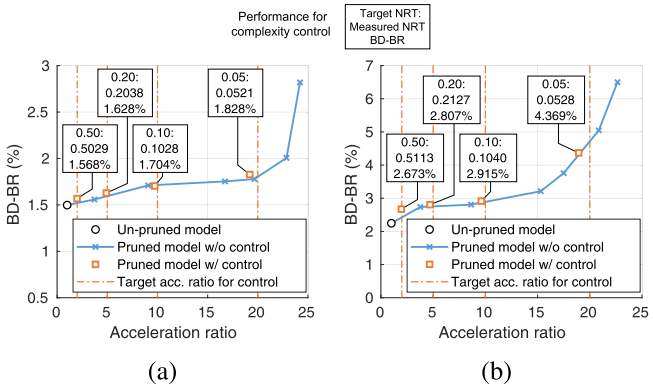

Fig. 4. Complexity-RD curves of our approach. (a) LDP configuration. (b) AI configuration. For complexity control, "a_1: a_2 b%" in each squared tag represents target NRT $\tilde{T}_C = a_1$, measured NRT $\tilde{T}_M = a_2$ and BD-BR of b%.

TABLE VII

STANDARD DEVIATION OF MODEL TIME AND BD-BR FOR COMPLEXITY REDUCTION OF HEVC

Approach		LDP config.		AI config.	
		Std. of model time (ms)	Std. of BD-BR (%)	Std. of model time (ms)	Std. of BD-BR (%)
[16]		21.79	1.246	-	-
[19]		-	-	23.43	2.529
[22]		778.88	1.436	1229.41	2.161
[23]		320.40	0.436	319.83	**0.991**
Our	20% WPs	84.26	**0.330**	86.26	1.227
	5% WPs	34.24	0.343	37.23	1.247
	1% WPs	18.49	0.449	20.88	1.363
	0.5% WPs	15.88	0.502	18.23	1.467
	0.2% WPs	13.85	0.562	15.27	1.381
	0.1% WPs	**13.50**	0.881	**14.13**	1.629

in 0.330%∼0.881% and 1.227%∼1.629% at LDP and AI configurations, respectively, which are smaller than those of all state-of-the-art approaches [16], [19], [22] and comparable to those of the un-pruned model [23]. As an overall analysis, our approach with 0.1%∼1% WPs has less standard deviation in both model time and BD-BR, compared with [16], [19], [22]. Therefore, our approach can drastically accelerate the ETH-CNN model with insignificant performance change across different sequences.

D. Evaluation on Control Accuracy

In this section, we analyze the NRT values of ETH-CNN to evaluate the performance of complexity control.

TABLE VIII
MEASURED NRT OF ETH-CNN (LDP)

Class	Sequence	QP	$\tilde{T}_{\mathrm{M}}$ (%)			
			$T_{\mathrm{C}}=50\%$	$T_{\mathrm{C}}=20\%$	$T_{\mathrm{C}}=10\%$	$T_{\mathrm{C}}=5\%$
A	*PeopleOnStreet*	22	50.18 ± 1.15	20.37 ± 0.62	10.35 ± 0.61	5.35 ± 0.32
		27	50.33 ± 0.94	20.47 ± 0.81	10.20 ± 0.80	5.27 ± 0.56
		32	50.09 ± 2.59	20.46 ± 0.52	10.34 ± 0.89	5.27 ± 0.52
		37	50.25 ± 1.20	20.37 ± 0.62	10.15 ± 1.01	5.17 ± 0.56
	Traffic	22	50.13 ± 1.52	20.21 ± 0.70	9.98 ± 1.05	5.19 ± 0.42
		27	50.07 ± 1.34	20.29 ± 2.17	9.83 ± 1.11	4.97 ± 0.59
		32	50.13 ± 1.43	20.26 ± 2.53	9.84 ± 1.15	5.00 ± 0.55
		37	50.15 ± 1.41	20.25 ± 0.54	**9.61 ± 1.14**	5.12 ± 0.55
B	*BasketballDrive*	22	49.77 ± 0.77	20.24 ± 0.65	10.01 ± 0.70	5.16 ± 0.37
		27	50.01 ± 0.77	20.20 ± 0.85	9.99 ± 0.63	5.18 ± 0.48
		32	49.84 ± 0.65	20.21 ± 0.65	10.27 ± 0.61	5.15 ± 0.44
		37	49.85 ± 0.63	20.10 ± 0.65	10.18 ± 0.68	5.31 ± 0.48
	BQTerrace	22	49.86 ± 0.68	20.31 ± 0.58	10.18 ± 0.66	5.25 ± 0.41
		27	50.16 ± 0.69	20.46 ± 0.69	10.32 ± 0.63	5.26 ± 0.48
		32	50.14 ± 0.67	20.39 ± 0.76	10.10 ± 0.58	5.30 ± 0.52
		37	50.08 ± 0.75	20.35 ± 0.63	10.20 ± 0.67	5.26 ± 0.49
	Cactus	22	49.89 ± 0.79	20.39 ± 0.63	9.98 ± 0.58	5.27 ± 0.44
		27	50.10 ± 0.85	20.42 ± 0.68	10.34 ± 0.63	5.17 ± 0.46
		32	50.13 ± 0.59	20.43 ± 0.63	10.54 ± 0.64	5.04 ± 0.51
		37	50.11 ± 0.69	20.37 ± 0.63	10.52 ± 0.55	5.15 ± 0.49
	Kimono	22	49.99 ± 0.75	20.27 ± 0.68	10.20 ± 0.60	5.22 ± 0.38
		27	50.17 ± 0.97	20.24 ± 0.51	10.33 ± 0.64	5.21 ± 0.51
		32	50.02 ± 0.77	20.22 ± 0.67	10.26 ± 0.58	5.17 ± 0.50
		37	49.81 ± 0.75	**19.97 ± 0.67**	10.06 ± 0.61	5.15 ± 0.55
	ParkScene	22	49.94 ± 0.83	20.32 ± 0.78	10.14 ± 0.58	5.32 ± 0.33
		27	50.12 ± 0.74	20.07 ± 2.38	**10.63 ± 0.52**	5.22 ± 0.51
		32	50.12 ± 0.64	20.31 ± 0.81	10.50 ± 0.72	5.21 ± 0.53
		37	50.05 ± 0.73	20.28 ± 0.80	10.13 ± 0.77	5.12 ± 0.48
C	*BasketballDrill*	22	50.13 ± 1.50	20.42 ± 1.16	10.12 ± 1.01	5.40 ± 0.23
		27	50.29 ± 0.90	20.48 ± 1.14	10.16 ± 1.12	5.10 ± 0.55
		32	50.31 ± 1.48	20.37 ± 1.12	10.01 ± 1.09	5.07 ± 0.53
		37	50.37 ± 1.35	20.53 ± 0.98	10.09 ± 1.20	5.10 ± 0.56
	BQMall	22	50.05 ± 1.72	20.36 ± 1.05	10.24 ± 0.94	5.24 ± 0.40
		27	50.28 ± 1.24	20.42 ± 0.87	10.12 ± 1.11	5.12 ± 0.50
		32	50.36 ± 1.74	20.37 ± 0.93	9.80 ± 0.98	5.07 ± 0.54
		37	50.38 ± 1.76	20.52 ± 1.44	10.07 ± 1.07	5.04 ± 0.57
	PartyScene	22	50.16 ± 2.21	20.62 ± 1.22	10.29 ± 1.07	5.33 ± 0.30
		27	50.40 ± 2.22	20.46 ± 1.07	10.13 ± 1.04	5.20 ± 0.52
		32	50.03 ± 1.62	20.33 ± 2.12	10.08 ± 1.09	5.06 ± 0.48
		37	50.17 ± 1.69	20.36 ± 1.87	10.18 ± 1.08	5.17 ± 0.45
	RaceHorses	22	49.85 ± 1.60	20.23 ± 2.41	10.29 ± 0.81	5.45 ± 0.27
		27	50.21 ± 1.56	20.43 ± 0.99	10.31 ± 1.03	5.11 ± 0.61
		32	49.97 ± 1.21	20.14 ± 0.96	10.08 ± 0.97	5.04 ± 0.50
		37	50.30 ± 1.34	20.44 ± 0.88	10.06 ± 0.95	5.15 ± 0.58
D	*BasketballPass*	22	50.74 ± 1.72	20.41 ± 1.08	10.36 ± 1.25	5.37 ± 0.46
		27	50.93 ± 1.74	20.65 ± 0.89	10.55 ± 1.05	5.30 ± 0.44
		32	51.02 ± 1.77	20.34 ± 0.83	10.59 ± 1.18	5.20 ± 0.65
		37	51.03 ± 2.63	20.55 ± 0.98	10.41 ± 1.05	5.36 ± 0.24
	BlowingBubbles	22	**51.30 ± 3.12**	**20.94 ± 1.44**	10.55 ± 1.01	5.40 ± 0.55
		27	51.27 ± 3.24	20.93 ± 1.22	10.36 ± 1.09	5.19 ± 0.68
		32	50.96 ± 1.78	20.87 ± 0.78	10.43 ± 1.03	5.10 ± 0.67
		37	50.90 ± 2.56	20.56 ± 0.95	10.59 ± 1.05	5.23 ± 0.70
	BQSquare	22	51.00 ± 2.98	20.80 ± 0.97	10.48 ± 0.99	**5.46 ± 0.39**
		27	50.87 ± 2.60	20.46 ± 1.00	10.39 ± 1.04	5.44 ± 0.48
		32	50.75 ± 2.68	20.19 ± 1.61	10.40 ± 1.27	5.01 ± 0.57
		37	50.96 ± 2.88	20.03 ± 0.88	10.52 ± 1.10	5.15 ± 0.57
	RaceHorses	22	50.91 ± 1.98	20.50 ± 1.24	10.31 ± 1.23	5.33 ± 0.40
		27	50.78 ± 1.58	20.75 ± 1.16	10.58 ± 1.21	5.41 ± 0.42
		32	50.90 ± 1.46	20.81 ± 0.91	10.44 ± 1.18	5.22 ± 0.50
		37	51.04 ± 3.16	20.48 ± 1.02	10.33 ± 1.20	5.23 ± 0.61
E	*FourPeople*	22	49.75 ± 1.58	20.16 ± 0.91	10.27 ± 0.57	5.08 ± 0.50
		27	50.05 ± 1.48	20.25 ± 0.87	10.43 ± 0.39	4.98 ± 0.57
		32	50.10 ± 1.15	20.29 ± 1.45	10.28 ± 0.61	4.92 ± 0.57
		37	50.05 ± 0.91	20.19 ± 1.98	10.33 ± 0.77	5.02 ± 0.60
	Johnny	22	**49.64 ± 0.86**	20.11 ± 2.16	10.01 ± 0.78	5.14 ± 0.40
		27	50.10 ± 0.83	20.02 ± 1.48	10.14 ± 0.79	5.04 ± 0.58
		32	50.10 ± 0.98	20.14 ± 1.08	10.20 ± 0.63	4.98 ± 0.63
		37	50.02 ± 1.00	20.14 ± 1.11	10.08 ± 0.64	5.03 ± 0.62
	KristenAndSara	22	49.75 ± 1.02	20.20 ± 0.97	10.13 ± 0.57	5.17 ± 0.43
		27	50.10 ± 1.05	20.37 ± 1.16	9.94 ± 0.67	4.92 ± 0.53
		32	50.03 ± 0.75	20.25 ± 0.94	10.07 ± 0.69	4.99 ± 0.51
		37	50.08 ± 0.72	20.19 ± 0.97	9.94 ± 0.87	**4.91 ± 0.56**
Average $\tilde{T}_{\mathrm{M}}$		22	50.17 ± 1.49	20.38 ± 1.07	10.22 ± 0.83	5.29 ± 0.39
		27	50.35 ± 1.37	20.41 ± 1.11	10.26 ± 0.86	5.17 ± 0.53
		32	50.28 ± 1.33	20.35 ± 1.07	10.23 ± 0.88	5.10 ± 0.54
		37	50.31 ± 1.45	20.32 ± 0.98	10.19 ± 0.91	5.15 ± 0.54
Average BD-BR (%)			1.568	1.628	1.704	1.828

TABLE IX
MEASURED NRT OF ETH-CNN (AI)

Class	QP	$\tilde{T}_{\mathrm{M}}$ (%)			
		$T_{\mathrm{C}}=50\%$	$T_{\mathrm{C}}=20\%$	$T_{\mathrm{C}}=10\%$	$T_{\mathrm{C}}=5\%$
A	22	52.11 ± 0.89	22.25 ± 0.25	10.63 ± 0.23	5.49 ± 0.21
	27	51.81 ± 0.78	21.92 ± 0.32	10.53 ± 0.44	5.21 ± 0.20
	32	51.48 ± 0.84	21.63 ± 0.38	10.25 ± 0.77	5.20 ± 0.10
	37	50.88 ± 0.73	20.97 ± 0.28	10.24 ± 0.49	5.16 ± 0.18
B	22	51.55 ± 0.85	21.79 ± 0.55	10.53 ± 0.32	**5.68 ± 0.25**
	27	50.84 ± 1.02	20.96 ± 0.65	10.32 ± 0.35	5.22 ± 0.21
	32	49.95 ± 1.05	19.96 ± 0.73	10.01 ± 0.53	5.17 ± 0.23
	37	49.06 ± 0.99	18.93 ± 0.76	9.71 ± 0.39	5.10 ± 0.18
C	22	52.43 ± 1.38	22.68 ± 0.66	10.80 ± 0.50	5.56 ± 0.62
	27	52.29 ± 1.19	22.49 ± 0.83	10.78 ± 0.42	5.30 ± 0.25
	32	52.05 ± 1.16	22.19 ± 0.83	10.70 ± 0.40	5.28 ± 0.21
	37	51.56 ± 1.65	21.81 ± 0.90	10.46 ± 0.84	5.25 ± 0.23
D	22	**52.67 ± 2.23**	**23.26 ± 0.69**	**11.11 ± 0.67**	5.40 ± 0.70
	27	52.49 ± 2.12	22.98 ± 0.83	11.10 ± 0.71	5.34 ± 0.75
	32	52.25 ± 1.92	22.64 ± 0.90	10.93 ± 0.66	5.28 ± 0.58
	37	51.94 ± 2.20	22.31 ± 1.05	10.88 ± 0.61	5.31 ± 0.68
E	22	50.47 ± 1.33	20.49 ± 1.09	10.06 ± 0.50	5.45 ± 0.35
	27	49.46 ± 1.54	19.28 ± 0.44	9.72 ± 0.28	5.06 ± 0.17
	32	48.82 ± 1.52	18.54 ± 0.54	9.51 ± 0.28	5.01 ± 0.15
	37	**48.15 ± 1.32**	**17.81 ± 1.17**	**9.29 ± 0.24**	**4.98 ± 0.15**
Average $\tilde{T}_{\mathrm{M}}$	22	51.88 ± 1.36	22.15 ± 0.66	10.65 ± 0.46	5.53 ± 0.44
	27	51.41 ± 1.36	21.58 ± 0.66	10.52 ± 0.44	5.24 ± 0.33
	32	50.91 ± 1.32	21.00 ± 0.72	10.31 ± 0.52	5.20 ± 0.28
	37	50.31 ± 1.43	20.36 ± 0.87	10.12 ± 0.53	5.16 ± 0.30
Average BD-BR (%)		2.673	2.807	2.915	4.369

Tables VIII and IX present the measured NRT $\tilde{T}_{\mathrm{M}}$ of ETH-CNN at various target NRT $\tilde{T}_{\mathrm{C}}$ for complexity control. Here, each tabulated value represents the mean and standard deviation of $\tilde{T}_{\mathrm{M}}$ over all frames of a sequence. As such, the statistics can reflect the fluctuation of NRT, in addition to the overall average NRT during the whole encoding process. Note that the longest and shortest $\tilde{T}_{\mathrm{M}}$ for each $\tilde{T}_{\mathrm{C}}$ are highlighted in red and blue bold fonts, respectively. We can see from Table VIII that the control error of all cases is within 1.30% at inter-mode, with the largest error in *BlowingBubbles*, i.e., $\tilde{T}_{\mathrm{M}} = (51.30 \pm 3.12)\%$ for $\tilde{T}_{\mathrm{C}} = 50\%$ at QP 22. The smallest error in this table is almost zero, where the fluctuation of $\tilde{T}_{\mathrm{M}}$ can be negligible, e.g., $\tilde{T}_{\mathrm{M}} = (50.01 \pm 0.77)\%$ for $\tilde{T}_{\mathrm{C}} = 50\%$ at QP 27. Throughout this table, the control error for most video sequences at most QP values (more than 98% of $\tilde{T}_{\mathrm{M}}$ values in Table VIII) is less than 1%. Also, the standard deviation of $\tilde{T}_{\mathrm{M}}$ is always less than 1%. Such results indicate that our approach can accurately control the NRT of ETH-CNN at inter-mode. The complexity control is at little expense of bit-rate overhead, resulting in BD-BR of 1.568%∼1.828% close to that of 1.542%∼1.733% without control, under almost the same acceleration ratios. For intra-mode, similar results can be found in Table IX, where the control error of most cases is less than 2% and the RD-performance is comparable to that without control.

It is worth mentioning that the frame-level complexity control (Algorithm 2 in Section IV-B) is crucial to enhance the control accuracy across multiple frames. Figure 5 plots the curves of control error $|\tilde{T}_{\mathrm{M}}(u) - \tilde{T}_{\mathrm{C}}|$ for both the LDP and AI configurations, averaged over all test sequences at four QP values $\{22, 27, 32, 37\}$. As shown in this figure, the error is always within 2.5%, 2.5%, 1.0% and 1.0% after encoding 10 frames, with the target NRT of ETH-CNN $\tilde{T}_{\mathrm{C}} \in \{50\%, 20\%, 10\%, 5\%\}$, respectively. Figure 6 illustrates the fluctuation of $\tilde{T}_{\mathrm{M}}$ across frames for the *ParkScene* sequence as an example. As we can see, the measured NRT $\tilde{T}_{\mathrm{M}}$ of most frames is close to the target NRT $\tilde{T}_{\mathrm{C}}$. For each $\tilde{T}_{\mathrm{C}}$ value, although the measured NRT may be away from the target in the initial frames, it quickly approaches to the target in few successive frames. The initial fluctuation is due to the limited precision of CTU-level control, while the frame-level control can compensate it and make the NRT rapidly approach to the target. For other video sequences and QPs, we can observe similar results. In a word, the accuracy and convergence of our complexity control algorithm have been verified.

E. Transfer to Other Configurations and Standards

1) Transfer to the Random Access (RA) Configuration: We first transfer our approach to the RA configuration, which is a widely used temporal configuration for HEVC. The complexity-RD performance at the RA configuration is shown in Table X. Here, the seven un-pruned and

TABLE X
COMPARISON OF COMPLEXITY REDUCTION APPROACHES FOR HEVC (RA)

Approach		Class A 2560×1600		Class B 1920×1080		Class C 832×480		Class D 416×240		Class E 1280×720		Overall				
		Model time (ms)	BD-BR (%)	Model time (ms)	BD-BR (%)	Model time (ms)	BD-BR (%)	Model time (ms)	BD-BR (%)	Model time (ms)	BD-BR (%)	Model time (ms)	Model speed (fps)	Encoding speed (fps)	Relative model time (%)	BD-BR (%)
[16]		114.64	7.746	113.10	5.185	110.72	7.124	86.45	7.395	105.15	2.523	106.01	9.43	0.0497	0.296	5.994
[22]		2465.02	6.333	1272.10	4.321	251.88	4.079	60.40	3.237	402.94	3.193	890.47	1.12	0.0466	2.485	4.232
[23]		1053.74	2.024	536.81	1.929	109.88	2.150	29.54	2.008	251.44	1.147	396.28	2.52	0.0658	1.106	1.852
Our	20% WPs	305.90	2.125	139.57	1.934	32.35	2.208	8.82	2.059	50.48	1.154	107.42	9.31	0.0671	0.300	1.896
	5% WPs	121.73	2.130	57.42	2.030	12.71	2.216	3.49	2.132	23.76	1.198	43.82	22.82	0.0680	0.122	1.941
	1% WPs	66.61	2.318	31.94	2.362	6.78	2.454	1.81	2.259	13.30	1.414	24.09	41.51	0.0682	0.067	2.161
	0.5% WPs	56.56	2.584	27.23	2.515	5.73	2.644	1.54	2.383	11.49	1.480	20.51	48.76	0.0695	0.057	2.321
	0.2% WPs	48.98	2.631	23.19	2.557	4.92	2.700	1.32	2.530	9.92	1.305	17.67	56.59	0.0702	0.049	2.344
	0.1% WPs	45.31	2.721	22.07	2.524	4.57	2.831	1.22	2.810	9.47	1.459	16.53	60.50	0.0703	0.046	2.469

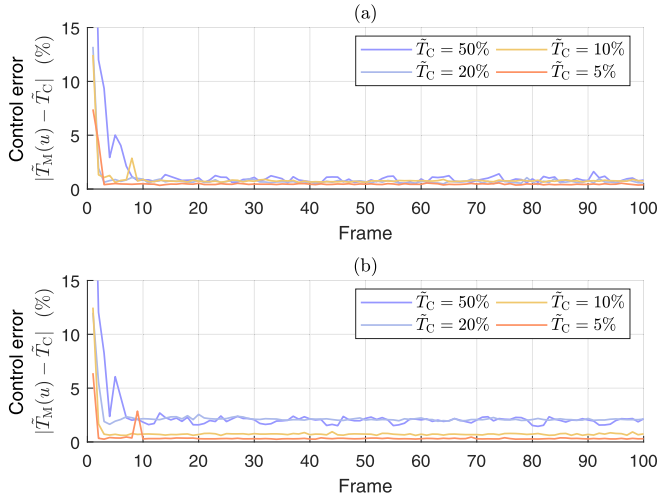

Fig. 5. Control error in the frame-level control algorithm. Here, $\tilde{T}_C$ represents the target NRT of ETH-CNN for the whole video sequence, while $\tilde{T}_M(u)$ is the measured NRT for each frame. Thus, the control error is reflected by $|\tilde{T}_M(u) - \tilde{T}_C|$. (a) LDP configuration. (b) AI configuration.

TABLE XI
MEASURED NRT OF ETH-CNN (RA)

Class	QP	$\tilde{T}_M$ (%)			
		$T_C = 50\%$	$T_C = 20\%$	$T_C = 10\%$	$T_C = 5\%$
A	22	50.89 ± 0.82	**20.51 ± 0.75**	**10.27 ± 0.54**	**5.92 ± 0.62**
	27	50.83 ± 0.80	19.96 ± 0.64	10.20 ± 0.42	5.74 ± 0.38
	32	50.65 ± 0.69	19.54 ± 0.16	9.85 ± 0.07	5.73 ± 0.38
	37	50.58 ± 0.68	19.56 ± 0.15	**9.84 ± 0.07**	5.76 ± 0.37
B	22	50.68 ± 1.14	20.05 ± 0.89	9.99 ± 0.39	5.42 ± 0.54
	27	50.57 ± 0.98	19.74 ± 0.65	10.01 ± 0.43	5.37 ± 0.35
	32	**50.49 ± 0.64**	19.54 ± 0.29	9.96 ± 0.32	5.38 ± 0.40
	37	50.56 ± 0.64	**19.41 ± 0.12**	10.07 ± 0.47	5.36 ± 0.32
C	22	**51.15 ± 1.92**	20.24 ± 1.14	9.99 ± 0.45	5.29 ± 0.36
	27	51.04 ± 1.68	20.26 ± 1.14	9.85 ± 0.12	5.24 ± 0.05
	32	50.72 ± 1.06	19.98 ± 0.84	9.89 ± 0.16	5.23 ± 0.04
	37	50.81 ± 1.15	20.12 ± 0.98	9.87 ± 0.15	5.23 ± 0.03
D	22	50.77 ± 1.93	20.06 ± 1.17	9.97 ± 0.58	5.27 ± 0.27
	27	50.92 ± 1.91	19.99 ± 1.11	9.84 ± 0.36	5.21 ± 0.08
	32	50.80 ± 1.91	20.04 ± 1.14	9.86 ± 0.37	5.20 ± 0.07
	37	50.78 ± 1.80	19.99 ± 1.16	9.87 ± 0.36	5.21 ± 0.10
E	22	50.71 ± 1.37	19.82 ± 0.56	9.96 ± 0.64	5.29 ± 0.68
	27	50.74 ± 1.36	19.67 ± 0.40	9.95 ± 0.56	5.19 ± 0.30
	32	50.85 ± 1.29	19.54 ± 0.35	9.92 ± 0.53	5.16 ± 0.23
	37	50.70 ± 1.39	19.57 ± 0.27	9.86 ± 0.40	**5.13 ± 0.16**
Average	22	50.83 ± 1.49	20.11 ± 0.94	10.01 ± 0.50	5.39 ± 0.47
	27	50.81 ± 1.38	19.93 ± 0.82	9.95 ± 0.37	5.32 ± 0.22
	32	50.69 ± 1.13	19.75 ± 0.60	9.90 ± 0.30	5.31 ± 0.21
	37	50.69 ± 1.14	19.74 ± 0.57	9.92 ± 0.32	5.30 ± 0.18
Average BD-BR (%)		1.883	1.944	2.085	2.355

pruned ETH-CNN models were retrained on the CPH database [23] for the RA configuration. As seen in this table, the average model time of our approach with ≤5% WPs is less than that of all other approaches [16], [22], [23]. For the overall encoding speed, all settings of our approach require less time than other approaches. On the other hand, the BD-BR of pruned ETH-CNN models in our approach is negligible (i.e., 1.896%∼2.469% on average), only 0.044%∼0.617% higher than the un-pruned ETH-CNN model [23], which significantly outperforms [22] and [16]. Therefore, our acceleration approach performs much better than the state-of-the-art approaches, in the complexity-RD performance. Moreover, Table XI reports the performance of our complexity control algorithm at the RA configuration of HEVC. We can see from this table that the control error is within 1.15%, and that the standard deviation of controlled complexity is less than 2%. In terms of the RD performance, the complexity control introduces little bit-rate overhead (1.883%∼2.355%), in particular compared with that of our acceleration approach (1.896%∼2.231%). Figure 6-(c) shows the fluctuation of $\tilde{T}_M$ across frames for the *ParkScene* sequence as an example. As we can see, the measured NRT values ($\tilde{T}_M$) of 50.60%, 19.46%, 9.83% and 5.23% are close to the target NRT values ($\tilde{T}_C$) of 50%, 20%, 10% and 5%, respectively, with the control error less than 1%. For each $\tilde{T}_C$ value, the measured NRT value of $\tilde{T}_M$ can quickly converge to the target alongside encoded frames. As a result, the overall fluctuation of $\tilde{T}_M$ is insignificant, with the standard deviation of only 0.02%∼0.53%. We can further see from Figure 6-(a), (b) and (c) that the results for the RA configuration are similar to those at the LDP and AI configurations. It is mainly because the quick convergence of $\tilde{T}_M$ benefits from the frame-level complexity control for all configurations. From the above results, both the control accuracy and desirable RD performance of our control algorithm can be verified.

2) Transfer to the x265 Encoder: It is common knowledge that the HM encoder is far from the practical use, due to its heavy computational cost. Thus, our approach is transferred to the x265 encoder [33], which is implemented much faster than HM and widely used in practice. The results are tabulated in Table XII, taking the "medium" setting of the original x265 as an anchor. Here, the un-pruned model with 100% WPs is not so efficient, because the inference time of the model accounts for a large proportion of encoding time. In contrast, the pruned model with 1% WPs can reduce the encoding time from 1.838 s to 0.748 s per frame, as the model itself is drastically accelerated from 0.396 s to 0.026 s for one frame. Meanwhile, the average BD-BR of the pruned model is 4.439%, only 1.882% higher than the un-pruned model. Thus,

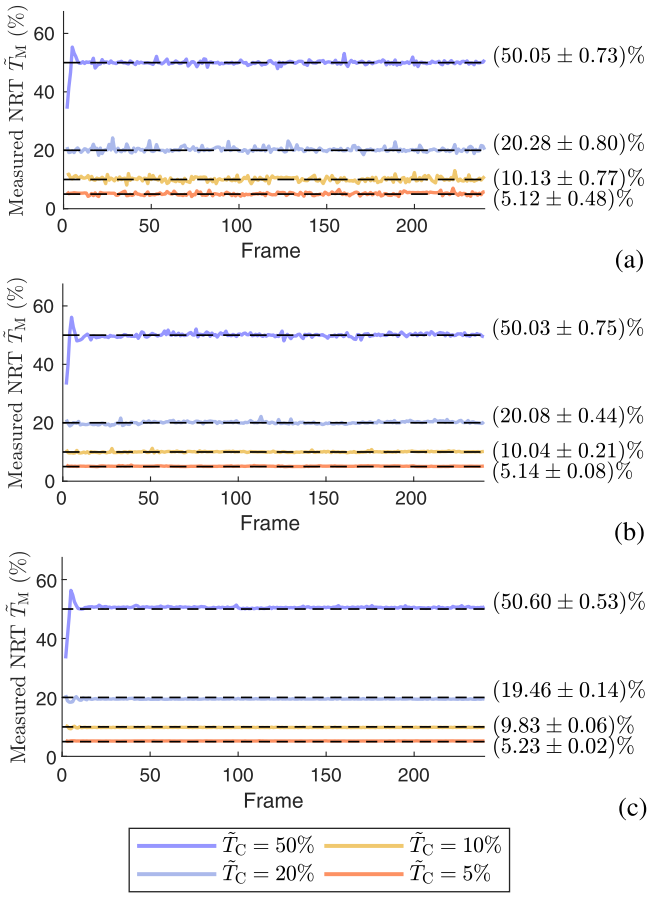

Fig. 6. Fluctuation of measured NRT for ETH-CNN, for *PackScene* at QP 37. (a) LDP configuration. (b) AI configuration. (c) RA configuration. The dashed lines represent four target NRT values, while the solid lines represent the measured NRT values.

TABLE XII
ENCODING TIME ON X265 (AI)

Approach	Real running time per frame (s)					BD-BR (%)
	Baseline*	Model overhead	Other processes	Total encoding	Reduction	
Our, 100% WPs, "medium"	1.838	0.396	0.972	1.368	-0.470	**2.557**
Our, 1% WPs, "medium"		**0.026**	**0.722**	**0.748**	**-1.090**	4.439
"very fast"		-	-	1.199	-0.639	8.811

*Original x265 at "medium" setting

the pruned model introduces little wrong decision of the CTU partition. In addition, we have tested the "very fast" setting of the original x265. Both the encoding time and the BD-BR of our approach are significantly better than the "very fast" setting of x265. Therefore, our acceleration approach integrated into a slower setting of x265 can outperform its built-in faster setting, in terms of both complexity and RD performance.

3) Transfer to Versatile Video Coding (VVC): In our experiments, we implement our approach to the latest VVC standard for verifying its effectiveness. Here, we transfer our acceleration approach from HEVC to VVC, at the AI configuration as an example. In the training phase, both un-pruned and pruned ETH-CNN models are trained according to the CTU partition of the VVC standard. In the test phase, each model is integrated into the VTM-7.0 encoder [48]

TABLE XIII
ENCODING TIME ON VVC (AI)

Setting of our approach	Real running time per frame (s)					BD-BR (%)
	Baseline: original VTM	Model overhead	Other processes	Total encoding	Reduction	
100% WPs	312.461	0.396	181.958	182.354	-130.107	**1.295**
1% WPs		**0.027**	**176.579**	**176.606**	**-135.855**	1.474

for deciding the quad-tree based partition structure of CTU, while the binary- and ternary-tree based partition is kept unchanged. Consequently, the encoding time per frame is shown in Table XIII, taking the original VTM encoder as anchor. From this table, we can see that both un-pruned and pruned ETH-CNN models achieve considerable time reduction with negligible increment on BD-BR. Compared with the un-pruned ETH-CNN model with 100% WPs, the pruned model with 1% WPs can accelerate the ETH-CNN model by 14.7 times, as the inference time of our acceleration approach is shortened from 0.396 s to 0.027 s per frame. This means that our acceleration approach is able to achieve real-time CTU partition for VVC at 30 fps. Meanwhile, the average BD-BR of our pruned model is 1.474%, only 0.179% higher than the un-pruned model. The above results verify the effectiveness of our approach on the VVC standard, in terms of both complexity and RD performance.

F. Ablation Study

We further conduct an ablation study about network acceleration and complexity control to investigate the effectiveness of our approach. For acceleration, the complexity of ETH-CNN is reduced by pruning off its WPs while keeping the number of layers unchanged. It is also possible to remove some entire layers in ETH-CNN and the computation can be simplified. In the ablation experiments, we remove the first FC layer in all three levels, i.e., $\{\mathrm{F}_{1-l}\}_{l=1}^{3}$, which has the highest complexity in the original ETH-CNN. As such, the concatenated vector synthesizing convolutional features directly flows through the second FC layer in all three levels, i.e., $\{\mathrm{F}_{2-l}\}_{l=1}^{3}$. Note that the simplified ETH-CNN can also be accelerated by the proposed pruning approach, as mentioned in Section III-C. Figure 7 compares the complexity-RD performance for both original and simplified ETH-CNN models at inter-mode. Note that the results are averaged over all 18 test sequences at four QPs. At the same acceleration ratio, the original model is always with smaller BD-BR than the simplified model, indicating better complexity-RD performance. It is because the original model is deeper than the simplified model to extract high-level features, i.e., three compared to two FC layers on each level. Therefore, it is reasonable to accelerate ETH-CNN with unchanged number of layers, rather than to directly remove certain layers.

For complexity control, the CTU-level control is a prerequisite while the frame-level control is optional. To analyze the effectiveness of frame-level control, we disable it in our ablation study by setting $\tilde{T}_C(u) = \tilde{T}_C(1), \forall u$, meaning that the target NRT $\tilde{T}_C$ of ETH-CNN is constant for all frames in a sequence. Table XIV shows the results for both settings

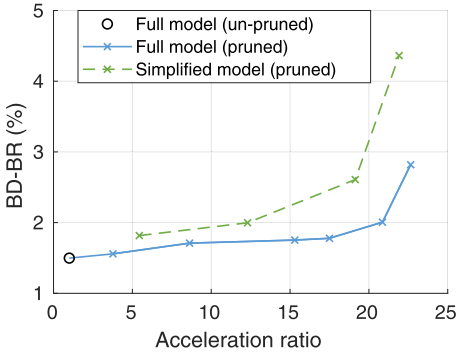

Fig. 7. Ablation results for three most time-consuming layers of ETH-CNN at the LDP configuration.

TABLE XIV
ABLATION RESULTS FOR FRAME-LEVEL CONTROL (LDP)

Setting	Target NRT $\tilde{T}_{\mathrm{C}}$ (%)	Control error $\lvert\tilde{T}_{\mathrm{M}} - \tilde{T}_{\mathrm{C}}\rvert$ (%)	Std. of $\tilde{T}_{\mathrm{M}}$ (%)
Both CTU- and frame-level control	50	**0.33**	**1.41**
	20	**0.37**	**1.06**
	10	**0.26**	0.87
	5	**0.19**	0.50
	Average	**0.26**	**0.96**
Only CTU-level control	50	9.32	2.03
	20	4.48	1.99
	10	2.30	**0.82**
	5	1.49	**0.28**
	Average	4.40	1.28

with and without frame-level control, averaged over all 18 test sequences at four QPs. As we can see, frame-level control can reduce the control error with smaller fluctuation of $\tilde{T}_{\mathrm{M}}$. The above analysis has verified the effectiveness of frame-level control, which can enhance the accuracy of the proposed control algorithm.

VI. CONCLUSION

In this paper, we have proposed accelerating DNNs for CTU partition with complexity control, to facilitate real-time encoding of HEVC. As investigated on the computational complexity throughout a newly-developed DNN model, most calculations can be saved by pruning off its WPs. To this end, we designed an adaptive pruning scheme to accurately configure the retention ratios of pruned WPs. Using this scheme, we obtained seven accelerated DNN models by setting different retention ratios of WPs. As a result, these models provide varying trade-off between complexity and prediction accuracy. Moreover, we developed a control algorithm to adjust the complexity of CTU partition close to a preset target. It was achieved by applying different models to different CTUs during HEVC encoding, enabling the average complexity of CTU partition to be continuously adjustable. Such complexity control is conducted at both the CTU- and frame-levels, ensuring high control accuracy with optimized RD performance. Finally, the experimental results verified that our approach is able to predict the CTU partition for 1080p frames in real time with only single thread of CPU, outperforming other state-of-the-art approaches in terms of both acceleration and RD performance.

For future works, some network pruning approaches with structured sparsity (e.g., pruning WPs at the filter- or channel-level) may be incorporated to accelerate the DNNs for CTU partition of HEVC, facilitating more hardware-friendly implementation of DNNs. For example, the practical performance of our approach on portable hardware, e.g., field programmable gate array (FPGA) and digital signal processor (DSP), is to be further explored as an interesting future work. In addition, our approach also has the potential for next-generation video coding standards, e.g., the VVC standard with further enhanced coding efficiency but extremely high complexity to be accelerated.

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] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[3] 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.

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

[5] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC complexity and implementation analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1685–1696, Dec. 2012.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] G. Correa, 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.

[15] 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.

[16] 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.

[17] 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, Jun. 2017.

[18] S. Ryu and J. Kang, "Machine learning-based fast angular prediction mode decision technique in video coding," *IEEE Trans. Image Process.*, vol. 27, no. 11, pp. 5525–5538, Nov. 2018.
[19] 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.
[20] Y. Li, Z. Liu, X. Ji, and D. Wang, "CNN based CU partition mode decision algorithm for HEVC inter coding," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 993–997.
[21] S. Kuanar, K. R. Rao, M. Bilas, and J. Bredow, "Adaptive CU mode selection in HEVC intra prediction: A deep learning approach," *Circuits, Syst., Signal Process.*, vol. 38, no. 11, pp. 5081–5102, Nov. 2019.
[22] K. Kim and W. W. Ro, "Fast CU depth decision for HEVC using neural networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 5, pp. 1462–1473, May 2019.
[23] M. Xu, T. Li, Z. Wang, X. Deng, R. Yang, and Z. Guan, "Reducing complexity of HEVC: A deep learning approach," *IEEE Trans. Image Process.*, vol. 27, no. 10, pp. 5044–5059, Oct. 2018.
[24] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. Int. Conf. Learn. Representations (ICLR)*, May 2016, pp. 1–14.
[25] V. Lebedev and V. Lempitsky, "Fast ConvNets using group-wise brain damage," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2554–2564.
[26] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Barcelona, Spain: Curran Associates, 2016, pp. 2074–2082.
[27] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 1943–1955, Oct. 2016.
[28] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned CP-decomposition," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2015, pp. 1–11.
[29] Y. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2015, pp. 1–16.
[30] C. Leng, Z. Dou, H. Li, S. Zhu, and R. Jin, "Extremely low bit neural network: Squeeze the last bit out with admm," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 3466–3473.
[31] P. Wang, Q. Hu, Y. Zhang, C. Zhang, Y. Liu, and J. Cheng, "Two-step quantization for low-bit neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 4376–4384.
[32] Google Inc. (2019). *TensorFlow*. Accessed: Jan. 1, 2019. [Online]. Available: http://www.tensorflow.org
[33] MulticoreWare Inc. (2019). *X265*. Accessed: Mar. 16, 2019. [Online]. Available: http://www.x265.org/hevc-h265/
[34] S. Kuanar, K. R. Rao, and C. Conly, "Fast mode decision in HEVC intra prediction, using region wise CNN feature classification," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW)*, Jul. 2018, pp. 1–4.
[35] S. Kuanar, C. Conly, and K. R. Rao, "Deep learning based HEVC in-loop filtering for decoder quality enhancement," in *Proc. Picture Coding Symp. (PCS)*, Jun. 2018, pp. 164–168.
[36] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org
[37] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2017, pp. 1–17.
[38] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 30, 2013, pp. 1–6.
[39] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convNets," in *Int. Conf. Learn. Representations (ICLR)*, 2016, pp. 1–13.
[40] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5058–5066.
[41] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Int. Conf. Neural Inf. Process. Syst. Workshop (NIPSW)*, 2014, pp. 1–9.
[42] R. L. Burden and J. D. Faires, *Numerical Analysis*. Boston, MA, USA: PWS Publishers, 1985.
[43] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2010, pp. 249–256.
[44] F. Bossen, *Common Test Conditions and Software Reference Configurations*, Joint Collaborative Team on Video Coding, document Rec. JCTVC-L1100, Jan. 2013.
[45] H. Everett, "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources," *Oper. Res.*, vol. 11, no. 3, pp. 399–417, Jun. 1963.
[46] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, Feb. 1970.
[47] 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.
[48] Joint Video Experts Team (JVET). (2020). *VTM Software*. Accessed: Feb. 23, 2020. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/

Tianyi Li (Graduate Student Member, IEEE) received the B.S. degree in electronic engineering from Beihang University, Beijing, China, in 2017, where he is currently pursuing the Ph.D. degree with the School of Electronic and Information Engineering. His research interests mainly include video coding and machine learning. He received the National Scholarship for undergraduate and Ph.D. student in 2015 and 2019, respectively, and the Outstanding Graduate Student of Beijing City in 2017.

Mai Xu (Senior Member, IEEE) received the B.S. degree from Beihang University in 2003, the M.S. degree from Tsinghua University in 2006, and the Ph.D. degree from Imperial College London in 2010. From 2010 to 2012, he was a Research Fellow with Electrical Engineering Department, Tsinghua University. Since January 2013, he has been with Beihang University, where he was an Associate Professor and then promoted to a Full Professor. From 2014 to 2015, he was a Visiting Researcher of MSRA. His research interests mainly include image processing and computer vision.

Xin Deng (Member, IEEE) received the B.S. and M.S. degrees in electronic engineering from Beihang University, Beijing, China, in 2013 and 2016, respectively, and the Ph.D. degree from the Department of Electrical and Electronic Engineering, Imperial College London, U.K., in 2020. She is currently an Associate Professor with the School of Cyber Science and Technology, Beihang University. Her research interests include sparse coding with applications in image and video processing, machine learning, and multimodal signal processing.

Liquan Shen (Member, IEEE) received the B.S. degree in automation control from Henan Polytechnic University, Jiaozuo, China, in 2001, and the M.E. and Ph.D. degrees in communication and information systems from Shanghai University, Shanghai, China, in 2005 and 2008, respectively. Since 2008, he has been with the faculty of the School of Communication and Information Engineering, Shanghai University, where he is currently a Professor. His major research interests include high-efficiency video coding, perceptual coding, video codec optimization, 3DTV, and video quality assessment.