An LSTM Method for Predicting CU Splitting in H.264 to HEVC Transcoding

Yanan Wei [#], Zulin Wang [#*], Mai Xu [#], Shuhao Qiao [#]

[#] *School of Electronic and Information Engineering, Beihang University, Beijing, China*
[*] *Collaborative Innovation Center of Geospatial Technology, Wuhan, China*

Corresponding Author: Mai Xu(maixu@buaa.edu.cn)

***Abstract*—For H.264 to high efficiency video coding (HEVC) transcoding, this paper proposes a hierarchical Long Short-Term Memory (LSTM) method to predict coding unit (CU) splitting. Specifically, we first analyze the correlation between CU splitting patterns and H.264 features. Upon our analysis, we further propose a hierarchical LSTM architecture for predicting CU splitting of HEVC, with regard to the explored H.264 features. The features of H.264, including residual, macroblock (MB) partition and bit allocation, are employed as the input to our LSTM method. Experimental results demonstrate that the proposed method outperforms the state-of-the-art H.264 to HEVC transcoding methods, in terms of both complexity reduction and PSNR performance.**

***Index Terms*—H.264, HEVC, Transcoding, LSTM, CU splitting**

I. Introduction

Transcoding is a technique which converts video stream from one encoding into another. Alongside the evolution of video coding standards, compression efficiency has been gradually improved. As a result, several video coding standards (e.g., MPEG-1, MPEG-2, MPEG-4, H.263, H.264 and high efficiency video coding (HEVC)) co-exist in a certain range of applications, which makes transcoding desirable. Video transcoding is a proper solution that bridges the gap in sharing multimedia contents across various types of multimedia devices (e.g., television, computer, laptop, tablet and smart phone). Therefore, transcoding has attracted increasing attention [1].

In the past two decades, many transcoding algorithms have been proposed with promising performance. However, the latest video coding standard HEVC, which achieves outstanding coding efficiency at the cost of large computational complexity, still challenges the existing transcoding algorithms. As the state-of-the-art video coding standard, HEVC offers excellent rate-distortion performance and supports higher resolution video coding. As a result, a large number of videos are encoded by HEVC over the past few years. Meanwhile, more and more terminals tend to adopt this new standard. On the other hand, extensive video streams encoded by previous H.264 standard need to be transcoded into HEVC domain. To this end, efficient transcoding from H.264 to HEVC receives a great deal of research effort.

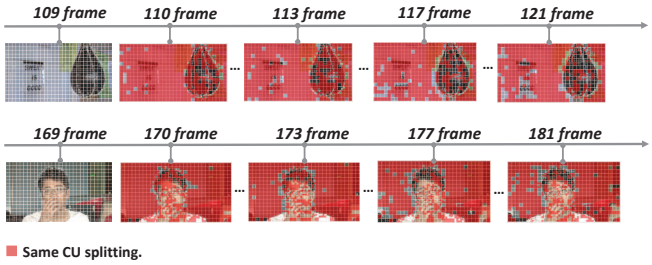

Fig. 1. Two examples of the temporal similarity of CU partition.

In fact, H.264 to HEVC transcoding can be accomplished by a fully H.264 decoding process and then a fully HEVC encoding process. However, such procedures result in inefficiency as HEVC encoding is rather time-consuming. In particular, coding tree unit (CTU) partition of HEVC takes up high computational time [2], as all possible splitting patterns of coding unit (CU) need to be traversed for rate-distortion optimization. Thus, it is important to predict CU partition of HEVC according to H.264 bitstreams, when designing an efficient transcoding method. The methods for H.264 to HEVC transcoding can be divided into two categories: either heuristic or data-driven. Heuristic methods normally leverage or extract some specific knowledge in compressed bitstream, combining with human knowledge, to accomplish the transcoding from H.264 to HEVC. For example, in [3], the variance of motion vectors (MVs) of four H.264 macroblocks (MBs) is used to explore the possibility of merging to form larger CU in HEVC. Mora *et al.* [4] applied motion similarity of H.264 MBs to build a fusion map, which is used to limit the depth of CU in HEVC coded frames. Compared with heuristic methods, data-driven methods make full use of training data to accomplish CU splitting in H.264 to HEVC transcoding, which achieves better performance than heuristic methods. In [2], [5], [6], [7], linear discriminant is applied to map the MB in H.264 to 64×64 or 32×32 CUs in HEVC. Decision tree is utilized in [8] for fast CU splitting decision during H.264 to HEVC transcoding, in light of a mining

978-1-5386-0462-5/17/$31.00 ©2017 IEEE.

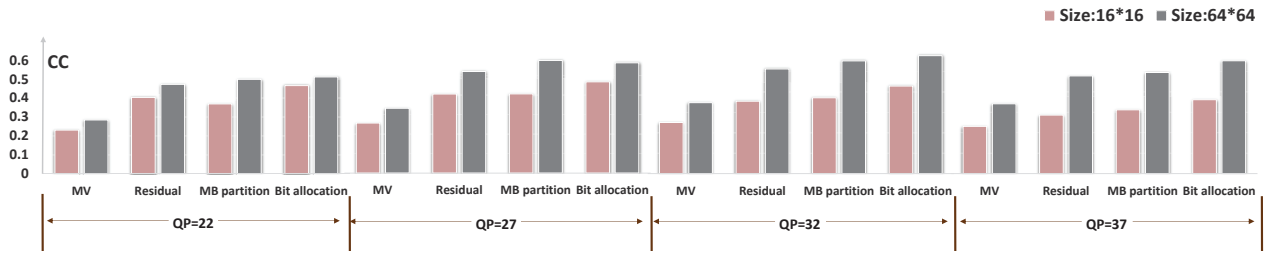

Fig. 2. CC values between features of H.264 bitstream and CU splitting patterns.

process on H.264 decoding attributes. Naive Bayes classifier is applied in [9] and [10] to make decision on CU splitting for H.264 to HEVC transcoding, which avoids exhaustive rate-distortion optimization search on all possible CU patterns. In [11] and [12], features of MB partition types, discrete cosine transformation coefficient and QP are extracted from H.264 bitstreams to train the support vector machine (SVM) classifier, in order to speed up CU splitting in H.264 to HEVC transcoding. MV clustering is utilized in [13] and [14] for CU partition in H.264 to HEVC transcoding.

However, the state-of-the-art transcoding methods does not take into account the correlation of CU partition across frames, as can be seen in Figure 1. Long Short-Term Memory (LSTM) network is an efficient deep learning approach to learn temporal correlation of data. Therefore, in this paper we propose a hierarchical LSTM method for predicting CU partition patterns, in H.264 to HEVC transcoding. Specifically, a hierarchical LSTM architecture which consists of three level LSTM classifiers is built. In this architecture, the input H.264 features are selected, according to the analysis of correlation between features of H.264 and CU splitting of HEVC. The three level LSTM classifiers are adopted in our hierarchical LSTM, which decides whether to split a CU, thus enabling prediction of CU splitting. Therefore, CU splitting can be speeded up for H.264 to HEVC transcoding.

II. Our Method

In section II-A, we first analyze the linear correlation between features in H.264 bitstream and CU patterns, as the preliminary of our method. Then, according to the correlation analysis, we present our hierarchical LSTM method for H.264 to HEVC transcoding.

A. Correlation Analysis

We argue in this section that features in H.264 domain can be employed in fast decision of CU splitting patterns, when transcoding from H.264 to HEVC. In our work, we encode the 18 standard test video sequences of JCT-VC and 93 collected raw video sequences by applying H.264 (JM 19.0), at four QPs (22, 27, 32, 37). Given the encoded data, we explore the correlation between features of H.264 and CU splitting of HEVC. Here, the features of H.264 include MV, residual, MB partition and bit allocation.

The linear correlation can be calculated by correlation coefficient (CC),

$$\mathrm{CC} = \frac{\sum_{i=1}^{N}(f_i - \overline{f})(g_i - \overline{g})}{\sqrt{\sum_{i=1}^{N}(f_i - \overline{f})^2}\sqrt{\sum_{j=1}^{N}(g_j - \overline{g})^2}}. \quad (1)$$

In (1), f_i denotes the feature of H.264 corresponding to the i-th CU in HEVC, and g_i means the ground truth of the i-th CU splitting pattern. Besides, $\overline{f}$ and $\overline{g}$ are the mean values of $\{f_i\}_{i=1}^{N}$ and $\{g_i\}_{i=1}^{N}$. We can see from Figure 2 that CU splitting patterns are highly correlated with bit allocation, MB partition and residual. In particular, CC values of those three features are larger than 0.4 for 64×64 CU. Besides, CC values are above 0.3 for bit allocation, MB partition and residual, when CU is 16×16. Therefore, bit allocation, MB partition and residual of H.264 are selected in our method as features to predict CU splitting of HEVC, for H.264 to HEVC transcoding.

B. Hierarchical LSTM Method

Now, we present the architecture of our hierarchical LSTM method for H.264 to HEVC transcoding. Figure 3 shows the proposed hierarchical LSTM architecture. There are three different LSTM classifiers, corresponding to three levels of CU partition. The inputs to those LSTM classifiers are H.264 features i.e., bit allocation, residual and MB partition. It is worth mentioning that if the time step of LSTM is M, the inputs to an LSTM classifier are a sequence of features coming from M frames, and the outputs are the splitting decision of CUs in those frames. When the current CU is decided to be split, the next level LSTM classifier is activated and makes decision for next level CUs. Figure 4 shows the internal mechanism of LSTM. One LSTM unit consists of one cell and three gates (input, forget and output). The input gate brings new information to the whole network. The forget gate determines whether the information is forgotten or discarded in the network, while the output gate decides which piece of message is sent to the next LSTM unit. The definitions of the three gates are

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, \mathrm{F_t}] + b_i), \\ f_t &= \sigma(W_f \cdot [h_{t-1}, \mathrm{F_t}] + b_f), \\ o_t &= \sigma(W_o \cdot [h_{t-1}, \mathrm{F_t}] + b_o), \end{aligned} \quad (2)$$

where W_i, W_f, W_o are weights of input, forget and output gates, and b_i, b_f and b_o are their corresponding biases. Besides $\sigma(\cdot)$ is a sigmoid function. $\mathrm{F_t}$ is the input H.264 features of

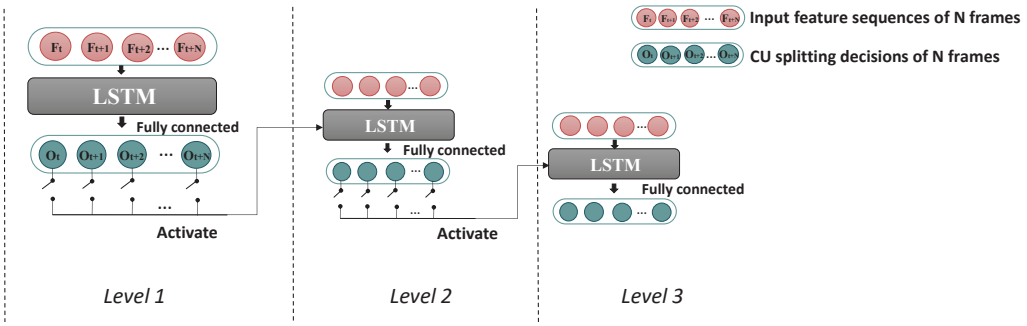

Fig. 3. The architecture of our hierarchical LSTM method for H.264 to HEVC transcoding. Note that all LSTM layers are followed by a fully connected layer.

the current CU in the t-th frame, and h_{t-1} is the output of CU splitting patterns for the $(t-1)$-th LSTM unit.

In our hierarchical LSTM method, three LSTM classifiers are learned by training data (including CU splitting patterns and H.264 features). The loss function of sigmoid cross entropy is employed for training our network:

$$\mathrm{L} = \sum_{i=1}^{K} (y_i \log a_i + (1 - y_i) \log(1 - a_i)). \tag{3}$$

Assume that y_i indicates the ground truth of the i-th CU, in which $y_i = 1$ means that the i-th CU is to be split and $y_i = 0$ is opposite. In addition, a_i is the output modelled by the sigmoid function. In the training phase, those three LSTM classifiers are trained individually. For training the LSTM classifier of *level 1*, 144-dimensional feature vectors, consisting of 16 elements of bit allocation, 64 elements of MB partition and 64 elements of residual, are delivered to each LSTM unit as the inputs. Note that for reducing the dimension of features, residual feature is replaced by taking the sum of the absolute value of residual in each 8×8 CU. Similarly, the LSTM classifiers of *level 2* and *level 3* have 36-dimensional and 9-dimensional input feature vectors. In the test phase, the LSTM classifier of *level 1* is firstly used to make decision on the splitting of 64×64 CU. If the 64×64 CU is decided to be split, the LSTM classifier of the next level (*level 2*) is activated and makes decision for the four CUs at the next level. The operation mechanism between the LSTM classifiers of *level 2* and *level 3* is also activated. Consequently, LSTM classifiers need not to traverse all CUs, and M frames splitting decisions are made one time. Therefore, encoding time is sufficiently saved by applying the proposed hierarchical LSTM.

III. Experiments And Results

In this section, we present the experimental results to evaluate the transcoding performance of our LSTM method. For evaluation, 18 standard test video sequences of JCT-VC and 93 collected raw videos were used in our experiment. In our experiments, 956,555 samples from 93 collected raw video sequences were divided into non-overlap training set (900,000 samples) and validation set (56,555 samples). Note that the ground truth of CU splitting results was extracted and used to learn the proposed hierarchical LSTM model. Then, 18 standard test video sequences were used as the test set. LSTM was implemented in deep learning platform "Tensorflow". For training, the length of LSTM time step was 30 and the learning rate was set to 10^{-3}. Besides, the batch size was set to be 200. Here, all hyperparameters above were tuned to make the validation results appropriate.

The proposed transcoding method was implemented on JM 19.0 and HM 16.0, and video sequences were compressed by HEVC with rate control. Here, we first encoded each video sequence at fixed QP=32. Then, the actual bit-rates used for compressing the video sequences at the fixed QP=32 were set as the target bit-rates for the encoder. Furthermore, the low delay IPPP structure was chosen for implementation, by using the HM default configuration file *encoder lowdelay P main.cfg*. Besides, our experiments were performed on the computer with CPU Intel(R) Core(TM) i7-3770 @ 3.40 GHz, 16 GB memory, and Windows 10 operating system.

We compare our hierarchical LSTM with the state-of-the-art method [12]. The metrics of ΔT and ΔPSNR are measured for comparison. In this paper, ΔT is defined as

$$\Delta T = \frac{T_o - T_p}{T_o} \times 100\%, \tag{4}$$

where T_o and T_p are the time costs of the original transcoder[1] and the proposed transcoder. In addition, ΔPSNR is defined as

$$\Delta PSNR = PSNR_o - PSNR_p. \tag{5}$$

In (5), $PSNR_o$ and $PSNR_p$ denote the PSNR values of the original and proposed transcoders. Table I reports the results of our method and [12]. It can be seen that our method is superior to [12] in both complexity reduction and PSNR performance. In particular, our method has a smaller average PSNR reduction 0.0543 than 0.0614 PSNR reduction of [12]. Note that ΔPSNR can reflect the distortion at the same bit-rate, since all experiments are conducted with rate control. Meanwhile, the transcoding complexity is significantly

[1]The original transcoder refers to encoding the decoded H.264 video stream by the original HEVC encoder of HM 16.0.

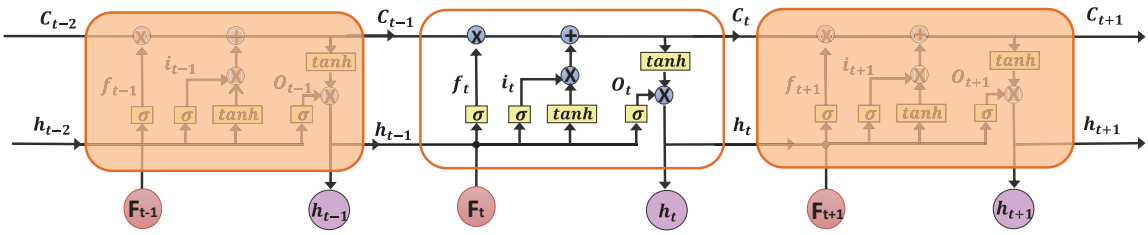

Fig. 4. The mechanism of LSTM units.

TABLE I
RESULTS OF ΔPSNR (dB) AND ΔT (%) FOR VIDEO SEQUENCES.

Class	Sequence	Ours		[12]	
		Δ T	ΔPSNR	Δ T	Δ PSNR
A	PeopleOnStreet	58.34	0.1321	32.59	0.1408
	Traffic	67.84	0.0525	59.41	0.0612
B	BasketballDrive	62.17	0.0330	62.45	0.0498
	BQTerrace	70.90	0.0110	56.81	0.0257
	Cactus	64.70	0.0410	52.87	0.0637
	Kimono	63.06	0.0009	58.24	0.0660
	ParkScene	63.68	0.0052	55.42	0.0461
C	BasketballDrill	61.06	0.0612	50.92	0.0471
	BQMall	60.11	0.1461	47.58	0.0863
	PartyScene	59.15	0.0417	34.99	0.0554
	RaceHorses	57.38	0.1146	44.05	0.0660
D	BasketballPass	59.36	0.0846	41.95	0.1227
	BlowingBubbles	57.60	0.0408	48.48	0.0585
	BQSquare	56.71	0.0178	39.07	0.0392
	RaceHorses	56.13	0.0959	30.32	0.0829
E	FourPeople	67.87	0.0617	63.64	0.0411
	Johnny	69.42	0.0146	56.79	0.0018
	KristenAndSara	67.60	0.0231	65.54	0.0516
Average		**62.39**	**0.0543**	**50.60**	**0.0614**

reduced by applying our method. As we can see in Table I, ΔT of our method is 62.39%[2], which is far better than 50.6% of [12]. In summary, our method performs better than [12], in terms of both quality loss and complexity reduction.

IV. CONCLUSION

This paper has proposed a hierarchical LSTM method for predicting CU splitting in HEVC to H.264 transcoding. First, several H.264 features were chosen to train the three LSTM classifiers, according to the correlation between CU splitting and H.264 features. Then, a new hierarchical LSTM architecture was developed, corresponding to three level classifiers of CU splitting. Finally, the experimental results showed that our method advances the state-of-the-art HEVC to H.264 transcoding.

[2]In spite of the huge amount of time spent on training the proposed LSTM, it averagely takes only 0.15% time of the original transcoder to predict CU splitting by applying the trained LSTM, which has already been taken into account in ΔT.

ACKNOWLEDGMENT

This work was supported by the NSFC projects under Grants 61471022, 61573037, and 61202139, and Fok Ying-Tong education foundation under grant 151061.

REFERENCES

[1] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: an overview," *IEEE Signal processing magazine*, vol. 20, no. 2, pp. 18–29, 2003.
[2] D. Zhang, J. Tong, and D. Zang, "Fast cu partition for h. 264/avc to hevc transcoding based on fisher discriminant analysis," in *IEEE Visual Communications and Image Processing (VCIP)*, 2016, pp. 1–4.
[3] A. Nagaraghatta, Y. Zhao, G. Maxwell, and S. Kannangara, "Fast h. 264/avc to hevc transcoding using mode merging and mode mapping," in *IEEE International Conference on Consumer Electronics (ICCE)*, 2015, pp. 165–169.
[4] E. G. Mora, M. Cagnazzo, and F. Dufaux, "Avc to hevc transcoder based on quadtree limitation," *Multimedia Tools and Applications*, pp. 1–25, 2016.
[5] E. Peixoto, B. Macchiavello, E. M. Hung, A. Zaghetto, T. Shanableh, and E. Izquierdo, "An h. 264/avc to hevc video transcoder based on mode mapping," in *IEEE International Conference on Image Processing (ICIP)*, 2013, pp. 1972–1976.
[6] E. Peixoto, B. Macchiavello, R. L. de Queiroz, and E. M. Hung, "Fast h. 264/avc to hevc transcoding based on machine learning," in *IEEE International Telecommunications Symposium (ITS)*, 2014, pp. 1–4.
[7] E. Peixoto, T. Shanableh, and E. Izquierdo, "H. 264/avc to hevc video transcoder based on dynamic thresholding and content modeling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 1, pp. 99–112, 2014.
[8] G. Correa, L. Agostini, and L. A. D. S. Cruz, "Fast h.264/avc to hevc transcoder based on data mining and decision trees," in *IEEE International Symposium on Circuits and Systems*, 2016, pp. 2539–2542.
[9] A. J. Díaz-Honrubia, J. L. Martínez, J. M. Puerta, J. A. Gámez, J. De Cock, and P. Cuenca, "Fast quadtree level decision algorithm for h. 264/hevc transcoder," in *IEEE International Conference on Image Processing (ICIP)*, 2014, pp. 2497–2501.
[10] A. J. Díaz-Honrubia, J. L. Martínez, P. Cuenca, J. A. Gamez, and J. M. Puerta, "Adaptive fast quadtree level decision algorithm for h. 264 to hevc video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 154–168, 2016.
[11] Q. Huangyuan, L. Song, Y. Ma, R. Xie, and Z. Luo, "Learning based fast h. 264 to h. 265 transcoding," in *IEEE Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 2015, pp. 563–570.
[12] L. Zhu, Y. Zhang, N. Li, G. Jiang, and S. Kwong, "Machine learning based fast h. 264/avc to hevc transcoding exploiting block partition similarity," *Journal of Visual Communication and Image Representation*, vol. 38, pp. 824–837, 2016.
[13] W. Jiang and Y. Chen, "Low-complexity transcoding from h. 264 to hevc based on motion vector clustering," *Electronics Letters*, vol. 49, no. 19, pp. 1224–1226, 2013.
[14] W. Jiang, Y. Chen, and X. Tian, "Fast transcoding from h. 264 to hevc based on region feature analysis," *Multimedia tools and applications*, vol. 73, no. 3, pp. 2179–2200, 2014.