Using CNN for Encoder Optimization in H.265/HEVC

[1]Ying Xie, [1]Ming Yang, [2]Jian Yu, [1]Wenchan Jiang, [3]Luguo Hao

[1]College of Computing and Software Engineering, Kennesaw State University, Marietta, GA 30060, USA
{yxie2, mingyang, wjiang6}@kennesaw.edu

[2]School of Computer Science and Technology, Tianjin University, Tianjin 300072, China
yujian@tju.edu.cn

[3]College of Information Engineering, Guangdong University of Technology, Guangzhou, China
haoluguo@gmail.com

Abstract

In this work-in-progress paper, we proposed using deep learning techniques, especially the deep Convolutional Neural Network (CNN) to perform critical tasks of video ending within the framework of H.265/HEVC. Deep CNNs have achieved break-through improvements on image recognition tasks such as image classifications, object identifications, and image annotations. However, very few work has been done in applying deep CNN to video encoding. In order to take advantage of the significant capabilities of deep CNN on image content detection, we proposed using deep CNN as the primary technique to perform critical tasks in video encoding that are relevant to the contents of one or multiple video frames. More specifically, we designed deep CNNs for the following tasks in H.265/HEVC encoder: partitioning CTU to CUs; partitioning CU to PUs; performing intra prediction; and performing inter predictions.

Keywords

Deep Learning, Deep CNN, H.265, HEVC, Video Encoding.

MOBIMEDIA 2017, July 13-14, Chongqing, People's Republic of China

transformation, and entropy coding to achieve high level of compression efficiency. The latest video coding standard, H.265/HEVC, has inherited such type of hybrid coding architecture. It had made improvements in each of the coding modules and overall it has achieved 50% compression performance gain, compared to H.264-AVC.

H.265/HEVC has adopted a more flexible blocking strategy, a more sophisticated data structure, more choices on intra-prediction modes, and other advanced techniques to achieve the above performance goal. The tradeoff is more intensive computation, which hinders its penetration to real-time streaming/transmission applications scenarios at current stage. Many coding decisions have to be made real-time during coding process, such as blocking (CTU-CU, CU-PU, CU-TU), prediction mode decision (intra-mode vs. inter mode), prediction direction decision in intra-prediction.

All these decisions are dependent on the contents of the video frames and making such decisions often times require exhaustive search if Rate-Distortion Optimization (RDO) is needed. In real-time streaming, exhaustive search is impossible and these decisions need to be made as fast as possible. In recent years, Convolutional Neural Network (CNN) has made great advances in the analysis and recognition of image/video contents. Thus, it is natural to apply trained CNNs to perform the above mentioned coding decision making process to largely speed up the coding process of H.265/HEVC and make it feasible for real-time coding and streaming applications. In the following sections, the proposed ideas will be discussed in greater details.

2. Literature Review

A novel fast Coding Tree Unit partitioning for HEVC/H.265 encoder was proposed in [1]. This method does not require any pre-training and provides a high adaptivity to the dynamic changes in video contents relied on run-time trained neural networks for fast Coding Units splitting decisions.

Paper [2] proposed a machine learning based approach for fast CU partition decision using features that describe CU statistics and sub-CU homogeneity. The proposed scheme was implemented as a "preprocessing" module on top of the Screen Content Coding reference software.

In [3], a fast convolutional-neural-network based quantization strategy for HEVC was proposed. Local artifact visibility is predicted via a network trained on data derived from an improved contrast gain control model. Further-more, a structural facilitation model was proposed to capture effects of recognizable structures on distortion visibility via the contrast gain control model.

Liu et al. ([4] [5]) devised a convolution neural network based fast algorithm to decrease no less than two CU partition modes in each CTU for full rate-distortion optimization (RDO) processing, thereby reducing the encoder's hardware complexity. As their algorithm does not depend on the correlations among CU depths or spatially nearby CUs, it was friendly to the parallel processing and did not deteriorate the rhythm of RDO pipelining.

In another study, Chen et al. [6] proposed a fast coding unit (CU) depth decision algorithm for intra coding of HEVC using an artificial neural network (ANN) and a support vector machine (SVM). Machine learning provided a systematic approach for developing a fast algorithm for early CU splitting or termination to reduce intra coding computational complexity.

Compared with existing efforts that applied machine learning in video encoding, our proposal has the following two unique features: 1) trying to take advantages of superiority of the-state-of-the-art deep CNN technology on image content detection to enhance content-based video encoding; 2) trying to use deep CNN as the primary technique for multiple content-relevant tasks in video encoding within the framework of H.265/HEVC.

3. Using CNN to Divide CTU into CUs

Coding Tree Unit (CTU) is the basic logic unit of the H. 265/HEVC standard and replaces macroblocks that were used in the previous standards. CTUs can be 16x16, 32x32, or 64x64 pixels in size. Larger size of CTU typically increase video encoding efficiency [7][8], especially for higher-resolution pictures. Each CTU can be partitioned recursively into coding unit (CU). The smallest CU can be 8X8. A CTU can be one CU or partitioned into 4 equal-size CU. Each CU that is larger than 8x8 can be remaining as one or further partitioned into 4 equal size CU. A quadtree structure can be used to represent the partition of a CTU into CUs, as shown in figure 1. Given a CTU with size of 64x64, instead of recursively determine the partition by following the quadtree, we propose designing a CNN to quickly determine the final partition for the CTU.

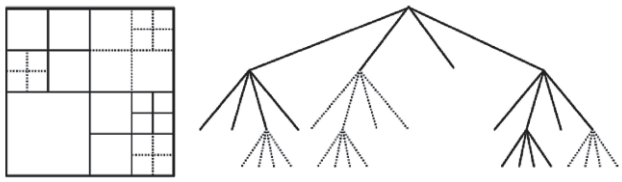


Figure 1. A Quadtree Structure Representing the Partition of A CTU into CUs [7]

3.1 The Architecture of the Deep NN for Partitioning CTUs

The abstract architecture of deep NN for partitioning a CTU can be illustrated in Figure 2. The input is each 64x64 CTU. Each CTU is fed into a CNN with multiple layers. On top of CNN is the full connected layers with softmax outputting the probability that the input CTU belong to each of partitioning types.

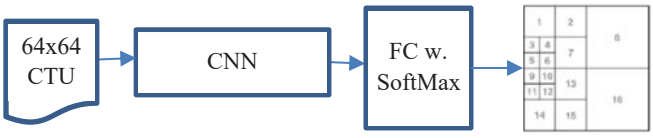


Figure 2. The abstract structure of the deep NN for partitioning CTUs

Based on the quadtree, there are totally 174+1=83522 different possible partitions for a 64x64 CTU. This large number of possibilities leads to the same number of outputs at the Softmax layer, which makes training this deep NN inefficient. One possible solution is that configure each CTU to be the size of 32x32 instead of 64x64. However, this simplified configuration compromises the merit that H.265/HEVC allows larger size of CTU for more efficient encoding. Therefore, our solution is that separating the participating into two steps. The first step uses a deep NN to determine a 64x64 CTU needs to be split or not. If so, then the second step is to split the 64x64 CTU into four 32x32 CTUs, and feed each one of them into another deep NN with the same structure as shown in figure 2 to determine its partitioning type. Not only the two-step approach is consistent with H.265/HEVC on the maximum size of CTU, but also reduce the number of 32x32 CTUs that need to be fed into the deep NN by the filtering process of the first step.

With respect to the design of the CNN component in the deep NN as shown in Figure 2, we consider the state-of-the-art CNN designs, including AlexNet[9], ZF Net[10], VGG [11], GoogleNet [12], and ResNet [13]. We feel that ResNet would be the one that fits our purpose well for the following reasons. First of all, it won ILSVRC 2015 with an incredible error rate of 3.6% using a revolution of depth of 152 layers. Secondly, it incorporated the effective deep residual learning strategy in its design. Thirdly, all filters that ResNet uses have the fixed small size 3x3, which fit the size of the input CTU (64x64 or 32x32) very well. Therefore, the CNN component in figure 2 will be replaced by a ResNet as shown in Figure 3. The depth of ResNet will be determined by experimental studies.

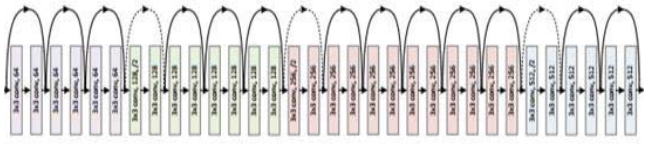
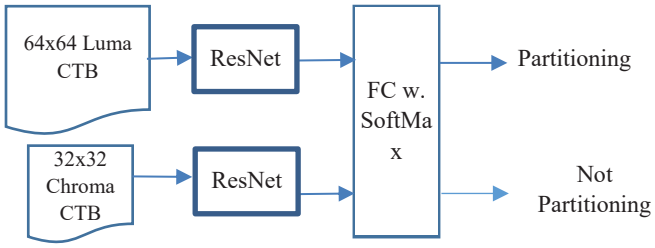
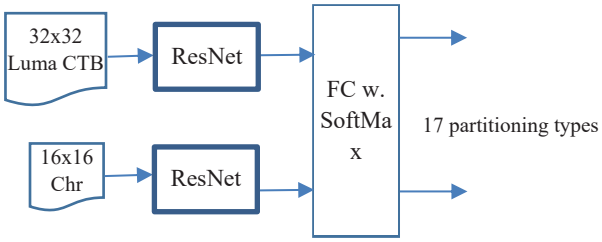


Figure 3. A Sample ResNet [13]

Since a 64X64 conceptual CTU maps to one 64x64 Luma CTB and two 32x32 Chroma CTB, the real inputs of the deep NN for CTU partitioning should be one 64x64 Luma CTB and two 32x32 Chroma CTB. Therefore the final design of the deep NN for CTU partitioning will be as follows:



a) Stage 1: determining if a 64x64 CTU needs further partitioning



b) Stage 2: if a 64x64 CTU is determined to be partitioned in stage 1, then stage 2 determines how each of its 32x32 subarea is partitioned.

Figure 4. The architecture of deep NN for partitioning a 64x64 CTU

3.2 Generating Training Data

Generating a large number of training data to train the deep NNs described in 3.1 is a big challenge. We propose a method that uses H.265 reference software to process a large number of different videos offline in order to generate training data sets. More specifically, we configure the reference software by setting the CTU size to be 64x64, and then output the partition information for each CTU. From the outputs, we extract information to form two training sets. First, for each 64x64 CTU, identify it is partitioned or not. Second, for each 64x64 CTU that is partitioned, extract one of the 17 parturition types for each of its four 32x32 sub-regions. Using the H.265 reference software to partition CTUs is a time consuming process, which is also one of the primary motivations of our proposal of using deep NN for online CTU partitions.

4. Using CNN to Divide CU into PUs

Through the partitions of CTU, we obtain CUs with the following possible sizes: 64x64, 32x32, 16x16, or 8x8. Now for each CU, we need to determine its prediction type (inter prediction or intra prediction) and further partition it to prediction units (PU) based on its prediction type. By following the similar idea that is described in section 3, we apply deep NN to output both the prediction type of CU and its partition to PUs. Figure 5 shows our design of deep NN to partition a CU with size of 2Nx2N into PUs. N can be one of the following values: 32, 16, 8, and 4. We need to have a deep NN as follows for each N value.

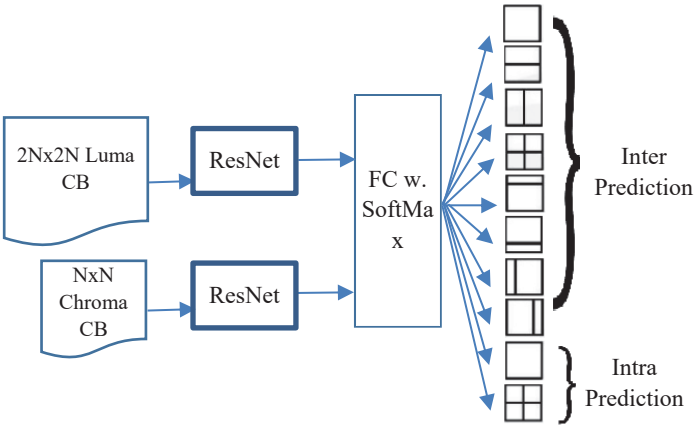


Figure 5. The architecture of deep NN for partitioning a 2Nx2N CU into Pus

5. Using CNN to Perform Intra Prediction

Given a PU that is determined to be of intra prediction by the deep NN described in section 4, we need to further determine its prediction mode. H.265/HEVC allows 35 intra prediction modes as shown in figure 6 (a). The decision on which prediction mode to use for the given PU depends on not only the block that this PU refers to, but also its neighboring blocks as the A, B, C, D and E blocks shown in figure 6(b). We design the following NN as shown in figure 7 to make the decision on prediction model for a given PU.

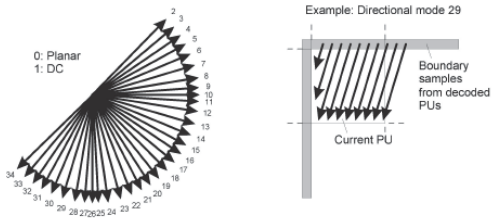


Figure 6. Illustration of Intra Prediction Modes [7]

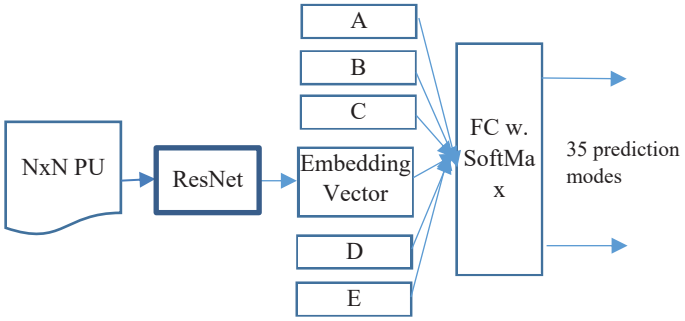


Figure 7. The architecture of deep NN for determine the intra prediction mode for a PU

The input of the deep NN is an NxN PU and the output is the probabilities that this PU belongs to each of the 35 partitioning modes. An embedded vector is first generated for the input PU by the deep CNN. Then the embedded vector and the PU's neighbor blocks A, B, C, D, and E are fed into a fully connected NN with softmax as the output. For each possible N value, including 4, 8, 16, 32, and 64, we need to have such an NN as illustrated in Figure 7.

6. Using CNN to Perform Inter Prediction

Given a PU that is determined to be of inter prediction, the in5er prediction process needs to find a block that is most similar to the block that this PU refers to in reference frames. The most accurate approach is to perform a full search in all reference frames. However, full search is obviously very time consuming. Therefore, different fast search strategies have been used in real applications. In this section, we propose a deep NN approach to directly output the candidate block in a given reference frame. As illustrated in Figure 8, the input of this NN is the PU and a reference frame. The top layers of this NN is a fully connected regression network that output two values

representing the x coordinate and y coordinate of the candidate block in the reference frame. Once obtaining the candidate block from each reference frame through the NN, we simply compare each of the candidate blocks with the block referred by the PU and choose the most similar one for further motion estimation.

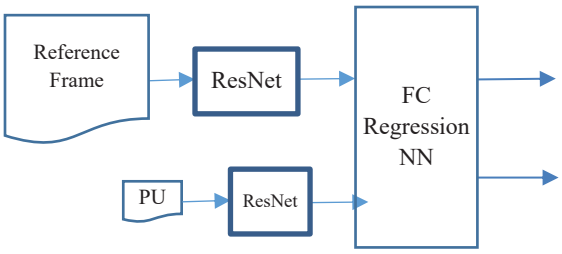


Figure 8. The architecture of deep NN for inter prediction

7. Discussion and Future Work

In this paper, we investigated the possibilities of using deep NN as the primary technique for efficient video encoding in H.265/HEVC. We proposed several deep NN designs for the following tasks in video encoding: partitioning CTU to CUs; partitioning CU to PUs; performing intra prediction; and performing inter predictions. One of the biggest challenges for using deep NN for video encoding is to generate large enough training data for training the designed deep NN models. Our suggested solution to this challenge is using the H.265 reference software to create the most optimized outputs for each of the above tasks, based on which training data can be extracted. Our next step is to conduct large-scale experimental studies on the proposed deep NN designs and refine the designs based on the experimental results.

REFERENCES

[1] Svetislav Momcilovic, Nuno Roma, Leonel Sousa, Run-time machine learning for H.265/HEVC fast partitioning decision, IEEE International Symposium on Multimedia, 2015, pp347-350

[2] Fanyi Duanmu, Zhan Ma, and Yao Wang, Fast CU partition decision using machine learning for screen content compression, IEEE International Conference on Image Processing (ICIP), 2015, pp4972-4976

[3] Md Mushfiqul Alam, Tuan D. Nguyen, Martin T. Hagan, and Damon M. Chandler, A perceptual quantization strategy for HEVC based on a convolutional neural network trained on natural images, SPIE Applications of Digital Image Processing XXXVIII, Sept. 2015, doi: 10.1117/12.2188913

[4] Zhenyu Liu, Xiaoyu Yu, Yuan Gao, Shaolin Chen, Xiangyang Ji, and Dongsheng Wang, CU partition mode decision for HEVC hardwired intra encoder using convolution neural network, IEEE Transactions on Image Processing, Vol. 25, No. 11, Nov. 2016, pp5088-5103

[5] Zhenyu Liu, Xianyu Yu, Shaolin Chen, Dongsheng Wang, CNN oriented fast HEVC intra CU mode decision, IEEE International Symposium on Circuits and Systems (ISCAS), 2016, pp2270-2273

[6] Zong-Yi Chen, Jiunn-Tsair Fang, Yen-Chun Liu, and Pao-Chi Chang, Machine learning-based fast intra coding unit depth decision for High Efficiency Video Coding, Journal of Information Science and Engineering 32, 2016, pp1289-1299

[7] Gary J. Sullivan; Jens-Rainer Ohm; Woo-Jin Han; Thomas Wiegand, Overview of the high efficiency video coding (HEVC) standard, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 22(12), 2012, pp. 1649-1668.

[8] Jens-Rainer Ohm, Gary J. Sullivan; Heiko Schwarz; Thiow Keng Tan; Thomas Wiegand, Comparison of the coding efficiency of video coding standards – including high efficiency video coding (HEVC), IEEE Transactions on Circuits and Systems for Video Technology, Vol. 22(12), 2012, pp. 1669-1684.

[9] Alex Krizhevsky, Ilya Sutskever, Geoffery E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in Nerual Information Processing Systems (NIPS), 2012

[10] Karen Simonyan, Andrew Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv: 1409.1556, 2014.

[11] Matthew D Zeiler, Rob Fergus, Visualizing and understanding convolutional networks, arXiv:1311.2901, 2013.

[12] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, Going deeper with convolutions, arXiv:1409.4842, 2014

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep residual learning for image recognition, arXiv:1512.03385, 2015