Using Deep Learning Neural Network for Block Partitioning in H.265/HEVC

Ming Yang[1], Ying Xie[1], Jian Yu[2], Zhe Wang[3], Tao Wu[1]
{mingyang@kennesaw.edu, yxie2@kennesaw.edu, yujian@tju.edu.cn, snowleoperd@126.com, twu6@students.kennesaw.edu}

College of Computing and Software Engineering, Kennesaw State University, Marietta, GA 30060, USA[1]
School of Computer Science and Technology, Tianjin University, Tianjin 300072, China[2]
College of Science, Tianjin University of Technology, Tianjin 300384, China[3]

Abstract: dividing video frames into Coding Tree Units (CTUs) and Coding Units (CUs) is a critical task of video compression in H.265/HEVC video coding standard. In this paper, we utilize deep learning techniques, especially the deep Convolutional Neural Network (CNN) to speed up the block partitioning process. 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. Block partitioning in video coding is highly dependent on the content of the video frames, and thus it is natural to take advantage of the significant capabilities of deep CNN on image content detection and recognition to perform block partitioning and avoid the time-consuming iterative Rate-Distortion-Optimization (RDO) process. Experimental results have shown that the proposed methodology has largely speed up the coding process and has also achieved coding efficiency comparable to the reference software of H.265/HEVC.

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

1. Introduction

Video coding techniques have been there for decades to enable storage and transmission of digital video contents with limited storage space and transmission bandwidth. The past video coding standards, such as H.263, MPEG-2, H.264, have adopted the hybrid coding architecture which utilized block coding, intra prediction, motion estimation, 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 has made improvements in each of the coding modules and overall it has achieved 50% compression performance gain compared to H.264-AVC.

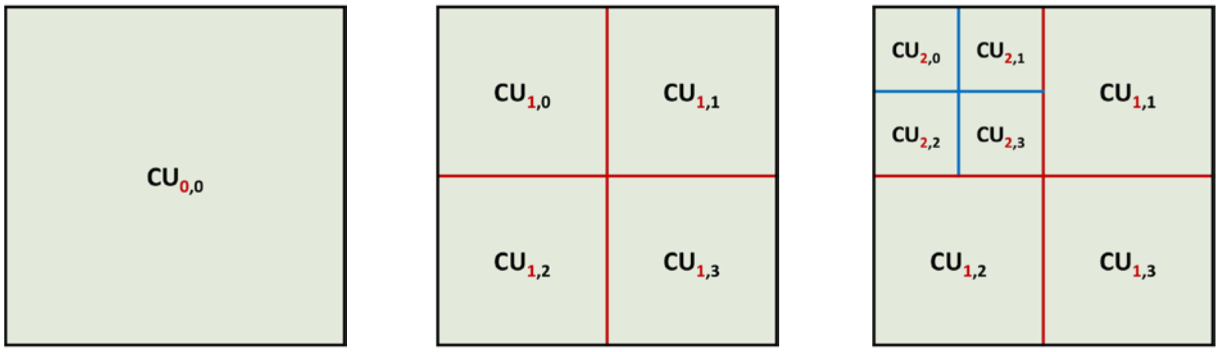
H.265/HEVC has adopted a more flexible blocking strategy, a more sophisticated data structure, more choices on intra-prediction modes, and other advanced techniques to achieve the above performance goal. The tradeoff is more intensive computation, which hinders its penetration to real-time streaming/transmission 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.

MOBIMEDIA 2018, June 21-22, Qingdao, People's Republic of China
Copyright © 2018 ACM
DOI 10.4108/eai.21-6-2018.2276639

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, these decisions need to be made as fast as possible and thus exhaustive search is infeasible. 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 further details.

2. Background and Literature Review

Fig. 1 has illustrated the traditional RDO-based exhaustive search CU partitioning process.



```
1.  Suppose $CU_{0,0}$ is the optimal coding mode
2.  Calculate $J_{0,0}$
3.  Split $CU_{0,0}$ into four smaller CUs of size of $16 \times 16$: $CU_{1,0}$, $CU_{1,1}$ , $CU_{1,2}$
    and $CU_{1,3}$
4.  Calculate $J_{1,0}$
5.  Split $CU_{1,0}$ into four smaller CUs of size of $8 \times 8$: $CU_{2,0}$, $CU_{2,1}$ , $CU_{2,2}$
    and $CU_{2,3}$
6.  Calculate $J_{2,0}$, $J_{2,1}$, $J_{2,2}$ and $J_{2,3}$
7.  Compare $J_{1,0}$ and $(J_{2,0}+J_{2,1}+J_{2,2}+J_{2,3})$
        if $J_{1,0} > J_{2,0}+J_{2,1}+J_{2,2}+J_{2,3}$
                split $CU_{1,0}$
        else
                keep $CU_{1,0}$
8.  Make decision on $CU_{1,1}$ , $CU_{1,2}$ and $CU_{1,3}$
9.  Make decision on $CU_{0,0}$, ( compare $J_{0,0}$ and $J_{1,0}+J_{1,1}+J_{1,2}+J_{1,3}$)
10. Finally we get the optimal coding mode
```

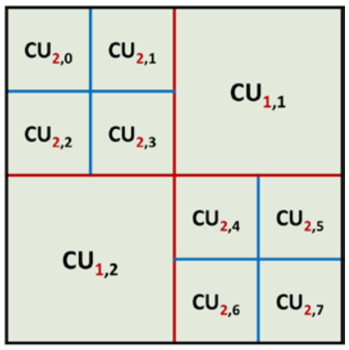


Fig. 1 Iterative Searching Process in CU Partition in H.265/HEVC Reference Software

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 level of adaptivity to the dynamic changes in video contents; it relies 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. Liu et al. ([4] [5]) devised a CNN 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. 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 provides 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 proposed methodology has the following two unique features: (1) it takes advantages of superiority of the-state-of-the-art deep CNN technology on image content detection to enhance content-based video encoding; (2) it utilizes 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 improves video encoding efficiency [7][8], especially for higher-resolution video. 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 remain as one CU or can be 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 Fig. 2. Given a CTU with size of 64x64, instead of recursively partitioning by following the quadtree, we propose designing a CNN to quickly determine the final partition for the CTU.

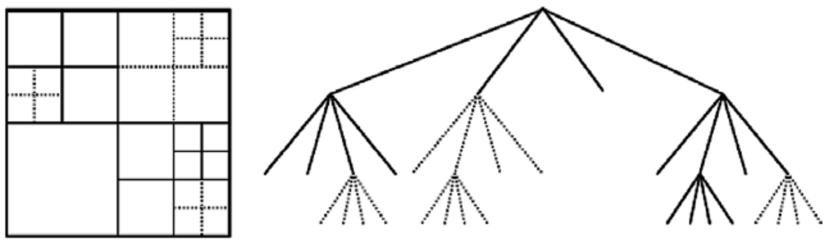


Fig. 2 CU Division Depth Map

3.1 The Architecture of the Deep NN for Partitioning CTUs

The architecture of deep NN for partitioning a CTU can be illustrated in Fig. 3. The input is a 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.

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 infeasible. 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 CNN to determine if the 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 to determine its partitioning mode. 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 Fig. 3, we consider the state-of-the-art CNN designs, including AlexNet [9], ZF Net [10], VGG [11], GoogleNet [12], and ResNet [13]. We determine 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. The depth of ResNet is determined by experimental studies. 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 as well. Therefore the final design of the deep NN for CTU partitioning will be as follows:

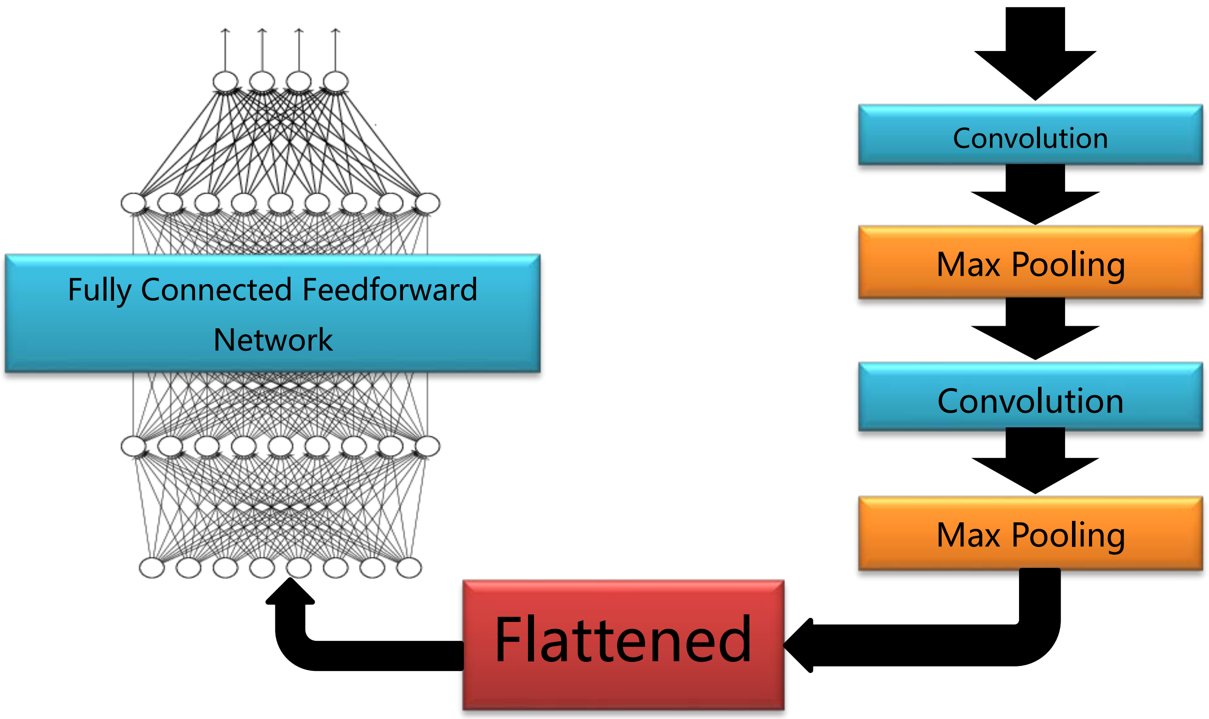


Fig. 3 CNN Architecture

3.2 Generating Training Data Set

Generating a large training data set to train the deep NNs described in Section 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, determine whether it should be partitioned or not. Second, for each 64x64 CTU that is partitioned, extract one of the 17 parturition modes 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. We downloaded 13 videos samples as the training and testing data. Among them, 12 samples were used as training data and the last one is used as testing data. The video samples are in CIF formats with the dimension of 352x288 (as shown in Fig .4). Every frame of the video is divided into CTU with size 32x32. For this sample, the training data includes total of 3150 frames, and each frame has 99 CTUs. So, there are over 300,000 CTUs available for training. The testing data has 260 frames, giving us more than 25,000 CTUs for testing.

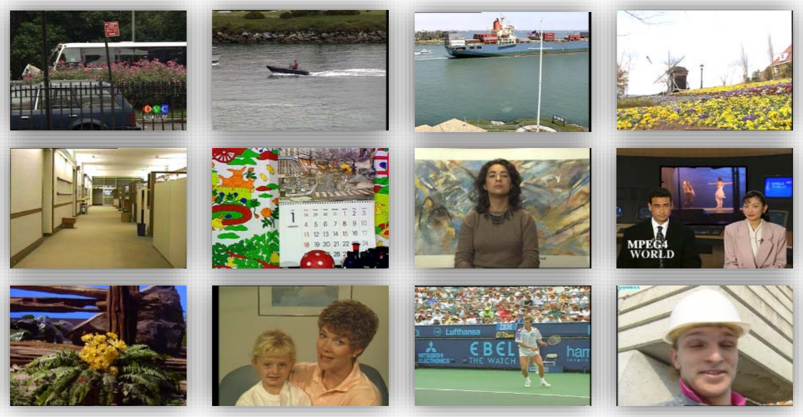


Fig. 4 Video Samples used for Training Set and Testing Set

3.3 Training Data Set Structure

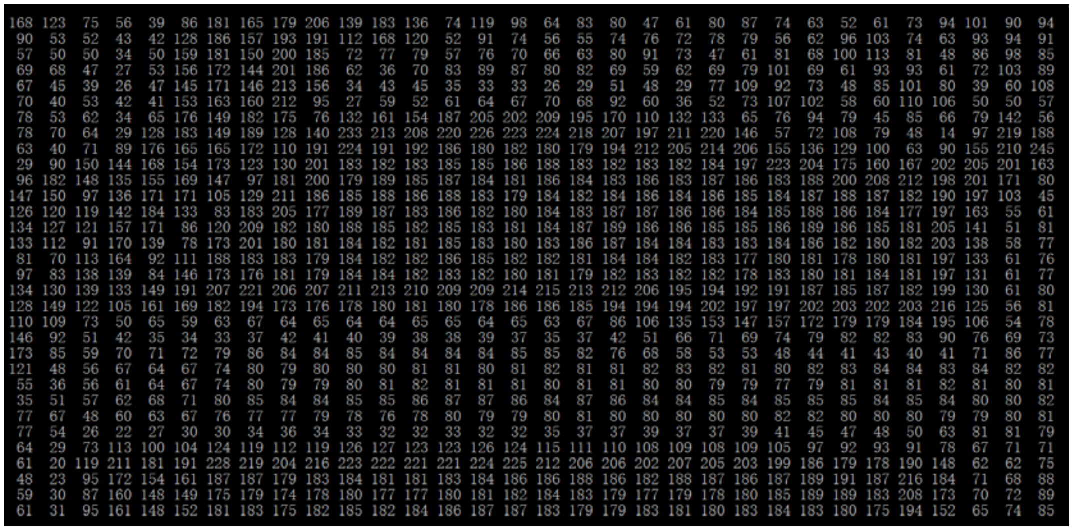
For each training sample, we have two components: the input and desired output. The input is basically the gray scale matrix of the 32x32 CTU, and the desired output is the CU depth map of the CTU. Here is how the depth number is defined:

(1) If a 8x8 block is part of a 32x32 CU, then the depth index corresponding to that 8x8 block is 0;

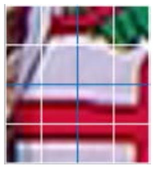
(2) If a 8x8 block is part of a 16x16 CU, then the depth index corresponding to that 8x8 block is 1;

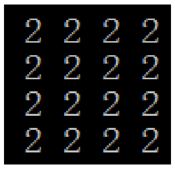
(3) If a 8x8 block is part of a 8x8 CU, then the depth index corresponding to that 8x8 block is 2;

In the following example in Fig. 5, since the 32x32 CTU is divided into 16 8x8 CUs, then the depth index of each 8x8 block is 2; and thus, every cell in the desired output is 2.



168	123	75	56	39	86	181	165	179	206	139	183	136	74	119	98	64	83	80	47	61	80	87	74	63	52	61	73	94	101	90	94
90	53	52	43	42	128	186	157	193	191	112	168	120	52	91	74	56	55	74	76	72	78	79	56	62	96	103	74	63	93	94	91
57	50	50	34	50	159	181	150	200	185	72	77	79	57	76	70	66	63	80	91	73	47	61	81	68	100	113	81	48	86	98	85
69	68	47	27	53	156	172	144	201	186	62	36	70	83	89	87	80	82	69	59	62	69	79	101	69	61	93	93	61	72	103	89
67	45	39	26	47	145	171	146	213	156	34	43	45	35	33	33	26	29	51	48	29	77	109	92	73	48	85	101	80	39	60	108
70	40	53	42	41	153	163	160	212	95	27	59	52	61	64	67	70	68	92	60	36	52	73	107	102	58	60	110	106	50	50	57
78	53	62	34	65	176	149	182	175	76	132	161	154	187	205	202	209	195	170	110	132	133	65	76	94	79	45	85	66	79	142	56
78	70	64	29	128	183	149	189	128	140	233	213	208	220	226	223	224	218	207	197	211	220	146	57	72	108	79	48	14	97	219	188
63	40	71	89	176	165	165	172	110	191	224	191	192	186	180	182	180	179	194	212	205	214	206	155	136	129	100	63	90	155	210	245
29	90	150	144	168	154	173	123	130	201	183	182	183	185	185	186	188	183	182	183	182	184	197	223	204	175	160	167	202	205	201	163
96	182	148	135	155	169	147	97	181	200	179	189	185	187	184	181	186	184	183	186	183	187	186	183	188	200	208	212	198	201	171	80
147	150	97	136	171	171	105	129	211	186	185	188	186	188	183	179	184	182	184	186	184	186	185	184	187	188	187	182	190	197	103	45
126	120	119	142	184	133	83	183	205	177	189	187	183	186	182	180	184	183	187	187	186	186	184	185	188	186	184	177	197	163	55	61
134	127	121	157	171	86	120	209	182	180	188	185	182	185	183	181	184	187	189	186	186	185	185	186	189	186	185	181	205	141	51	81
133	112	91	170	139	78	173	201	180	181	184	182	181	185	183	180	183	186	187	184	184	183	183	184	186	182	180	182	203	138	58	77
81	70	113	164	92	111	188	183	183	179	184	182	182	186	185	182	182	181	184	184	182	183	177	180	181	178	180	181	197	133	61	76
97	83	138	139	84	146	173	176	181	179	184	184	182	183	182	180	181	179	182	183	182	182	178	183	180	181	184	181	197	131	61	77
134	130	139	133	149	191	207	221	206	207	211	213	210	209	209	214	215	213	212	206	195	194	192	191	187	185	187	182	199	130	61	80
128	149	122	105	161	169	182	194	173	176	178	180	181	180	178	186	186	185	194	194	194	202	197	197	202	203	202	203	216	125	56	81
110	109	73	50	65	59	63	67	64	65	64	64	65	65	64	65	63	67	86	106	135	153	147	157	172	179	179	184	195	106	54	78
146	92	51	42	35	34	33	37	42	41	40	39	38	38	39	37	35	37	42	51	66	71	69	74	79	82	82	83	90	76	69	73
173	85	59	70	71	72	79	86	84	84	85	84	84	84	84	85	85	82	76	68	58	53	53	48	44	41	43	40	41	71	86	77
121	48	56	67	64	67	74	80	79	80	80	80	81	81	80	81	82	81	81	82	83	82	81	80	82	83	84	84	83	84	82	82
55	36	56	61	64	67	74	80	79	79	80	81	82	81	81	81	80	81	81	80	80	79	79	77	79	81	81	81	82	81	80	81
35	51	57	62	68	71	80	85	84	84	85	85	86	87	87	86	84	87	86	84	84	85	84	85	85	85	84	85	84	80	80	82
77	67	48	60	63	67	76	77	77	79	78	76	78	80	79	79	80	81	80	80	80	80	80	82	82	80	80	80	79	79	80	81
77	54	26	22	27	30	30	34	36	34	33	32	32	33	32	32	35	37	37	39	37	37	39	41	45	47	48	50	63	81	81	79
64	29	73	113	100	104	124	119	112	119	126	127	123	123	126	124	115	111	110	108	109	108	109	105	97	92	93	91	78	67	71	71
61	20	119	211	181	191	228	219	204	216	223	222	221	221	224	225	212	206	206	202	207	205	203	199	186	179	178	190	148	62	62	75
48	23	95	172	154	161	187	187	179	183	184	181	181	183	184	186	186	188	186	182	188	187	186	187	189	191	187	216	184	71	68	88
59	30	87	160	148	149	175	179	174	178	180	177	177	180	181	182	184	183	179	177	179	178	180	185	189	189	183	208	173	70	72	89
61	31	95	161	148	152	181	183	175	182	185	182	184	186	187	187	183	179	179	183	181	180	183	184	183	180	175	194	152	65	74	85





2	2	2	2
2	2	2	2
2	2	2	2
2	2	2	2

Fig. 5 Input and Desired Output of a Training Sample

4. Test Results

As can be seen from Fig. 6 and Fig. 7, with the CNN we designed, an initial accuracy rate of 60% was achieved. As the training moves forward, the accuracy rate will increase rapidly and will approach 100% at around the 200th iteration. In Fig. 6, we have set the value of batch-size parameter to 32, and then we change the value of "Adam". As can be seen, when the value of "Adam" is too large (lr = le-1) or too small (lr = le-1), the results of training are not satisfactory. When we set the value of "Adam" to "lr = le-3", the learning curve will reach 100% very quickly and will remain stable after that.

In Fig. 7, we have fixed the value of "Adam" to "lr = le-3", and then change "batch-size" to different values. When the value of "batch-size" is smaller, the accuracy rate will reach 100% with less number of iterations. However, with smaller "batch-size" value, each iteration will take longer. Thus, we need to choose an appropriate "batch-size" value. Overall, in order to reach satisfactory training results, we need to adjust the values of the parameters. The ultimate goal is to obtain a near-optimum CNN architecture.

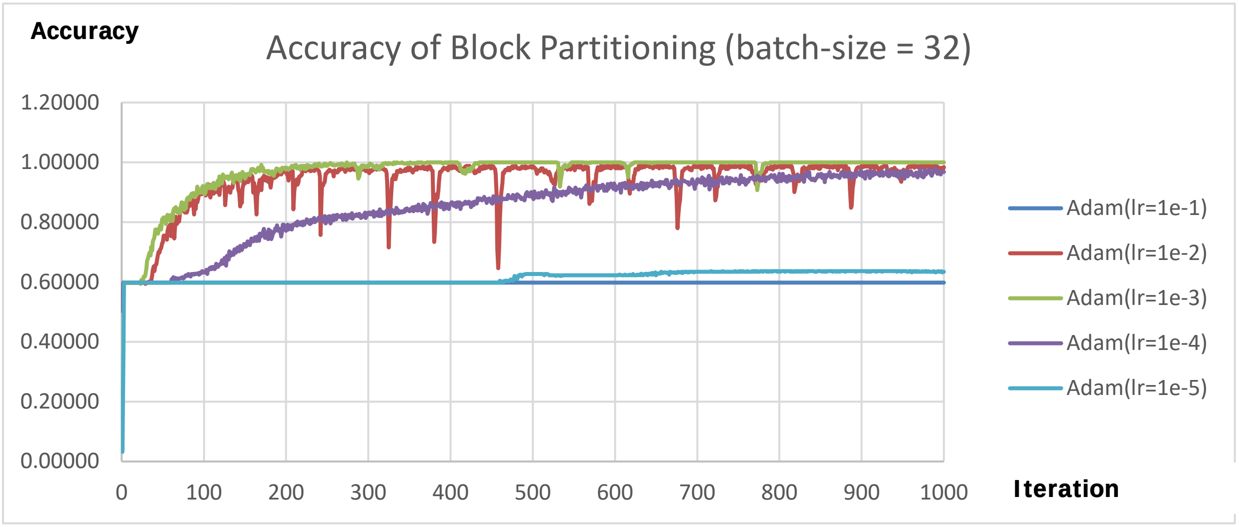


Fig. 6 Accuracy of Blocking (batch-size = 32)

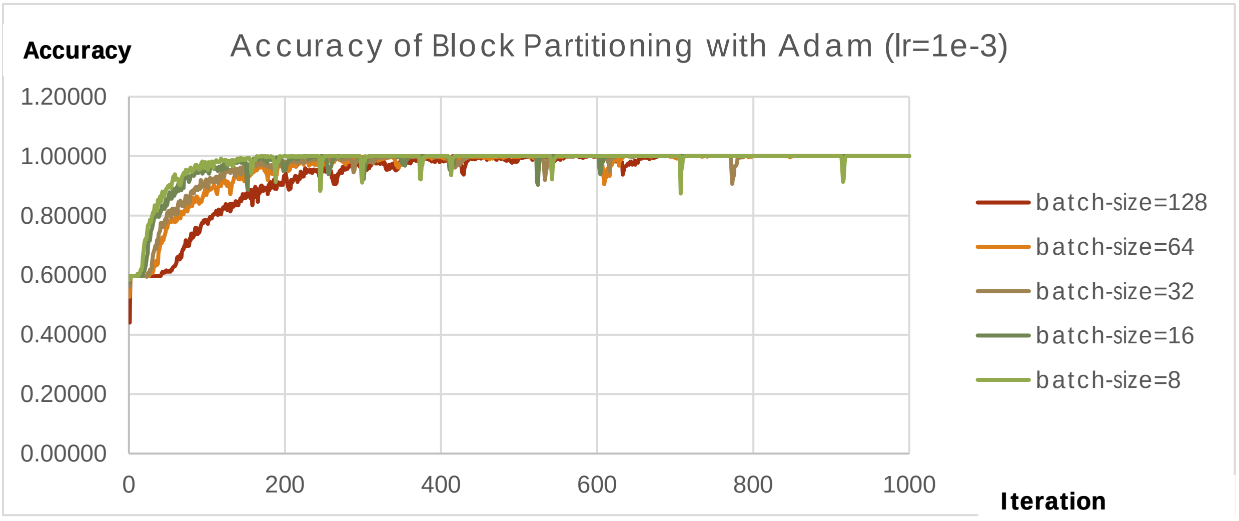


Fig. 7 Accuracy of Blocking with Adam(lr=1e-3)

6. Discussion and Future Work

In this paper, we proposed using deep NN as the primary technique for efficient video encoding in H.265/HEVC. We proposed several deep NN designs for CTU partitioning in video encoding. 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 solution to this challenge is using the H.265 reference software to generate the most optimized outputs for each of the above tasks, based on which training data can be extracted. The proposed CNN based algorithm has largely speed up this process (compared to RDO-based block partitioning), and now it can achieve real-time block partitioning. This has enabled the application of H.265/HEVC to real-time application scenarios. 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. We also plan to apply the proposed algorithm to other modules of H.265/HEVC, such as intra prediction mode decision, intra prediction direction decision, and inter-frame motion estimation.

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