REDUCING HEVC INTER-PREDICTION COMPLEXITY:

AN ATTENTION-BASED NETWORK

A Thesis

by

TIANRUI CHEN

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Chao Tian
Committee Members,	Ulisses Braga-Neto
	Anxiao Jiang
	Krishna Narayanan
Head of Department,	Miroslav M. Begovic

December 2021

Major Subject: Electrical and Computer Engineering

ABSTRACT

As an advanced video coding standard, High Efficiency Video Coding (HEVC) can remarkably reduce the bit-rates for equal perceptual video quality. Compared with H.264, the great promotion partly comes from the usage of the deeper coding quad-tree. The deeper coding quad-tree provides HEVC the ability to encode higher resolution video and reduce more bit-rates, however, it significantly increases the time complexity at the meanwhile, since HEVC searches for best coding tree depth by traverse every node exhaustively. Therefore, this thesis proposes a network to bypass the brute-force traversal and obtain the eventual coding unit (CU) partitions. First, a database, which contains each frame and corresponding CUs generated from 111 videos, established by running HEVC official software HM. Then, feed frames into a proposed convolutional neural network (CNN) in [6] to extract features. Third, the extracted features are fed into our proposed attention-based compression (ABC) network. The output of the attention-based network is predicted CUs. Finally, the experimental results shows that we build a trade-off between prediction accuracy rate and running time and can reduce the computational complexity significantly for HEVC.

Index-Terms: **High efficiency video coding, convolutional neural network, attention-based network.**

DEDICATION

To my parents, thank you for your unconditional support.

ACKNONLEDGMENT

Thanks to my parents, Meili Tai and Zhengshang Chen, for their support in life. Thanks to my advisor, Dr. Chao Tian, for his guidance and encouragement. And still thanks to Dr. Ulisses Braga-Neto, Dr. Anxiao Jiang and Dr. Krishna Narayanan for their advice and affirmation for my work and defense.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Chao Tian [advisor], and Professor Ulisses Braga-Neto and Professor Krishna Narayanan of the Department of Electrical and Computer Engineering, and Professor Anxiao Jiang of the Department of Computer Science and Engineering. All the work conducted for the thesis was completed by the student independently under the supervision of Professor Chao Tian.

Funding Sources

No outside funding was received in the research and writing of this document.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

1. INTRODUCTION

1.1 High Efficiency Video Coding

The High Efficiency Video Coding (HEVC) standard, also known as H.265, is first published in June 2013 by the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) standardization organizations, working together in a partnership known as the Joint Collaborative Team on Video Coding (JCT-VC) [1]. This standard was designed to address two key issues: increased video resolution and increased use of parallel processing architectures. HEVC saves about approximately 50% bit-rates for getting equal quality video compressed by H.264/Advanced Video Coding (AVC). Compare these two standards, one of the most pivotal changes is the size of the maximum block that will be encoded and decoded. In H.264/AVC, the compression was performed on each 16×16 size block which called macro block (MB), the size of compressed block is 64×64 to address the first issue and it was called Coding Tree Unit (CTU). Nevertheless, this change causes the increment of the computation complexity (in second) by up to 502.2% [2]. Furthermore, the inter-prediction stage utilizes 60% usage of CPU, which is higher than other stages. Hence, an alternative approach of brute-force inter-prediction can significantly reduce the computational complexity.

1.2 Attention-Based Neural Network

In the past few years, Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM) [3] and Gated Neural Network (GNN) [4] have been state of the art (SOTA) methods in many fields. However, this monopolistic situation was broken by the birth of transformer [5]. The transformer was published in December 2017. Compared with the former approaches' recursive

computations, transformer computes all states parallelly for the usage of attention mechanism. A general attention mechanism in transform is shown in figure 1.

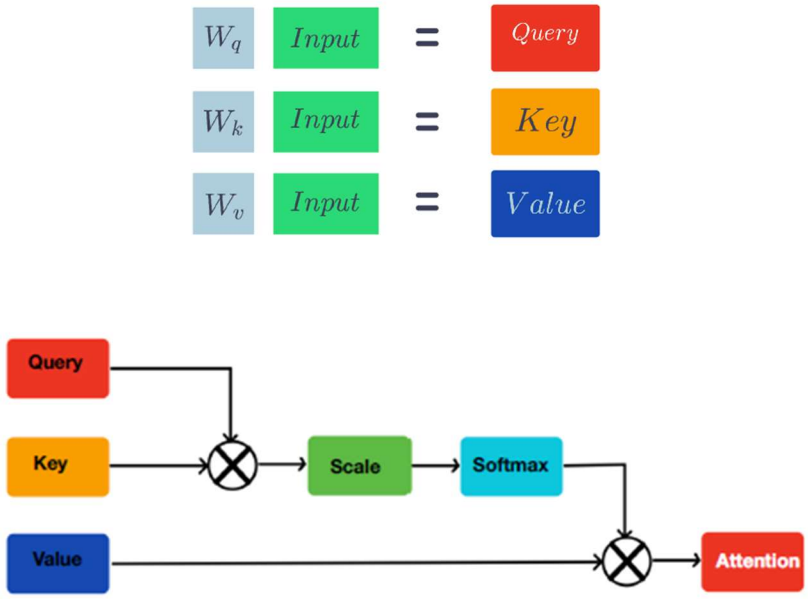


Figure 1: Scaled Dot-Product Attention [5]

As shown in Figure 1, a scaled dot-product attention contains three main inputs: query (Q), key (K) and value (V). These three inputs are calculated from the original input of the system. In practice, Q, K and V are three matrices, and the output of attention are from matrix multiplication as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \times V$$

Q and K has same dimension d_k. V has its own dimension d_v, however, in this thesis and experiment, Q, K and V have same dimension.

Then, why we chose self-attention approach to improve the work in [6]? We found that LSTM used a recursive way to transfer information contained in former states or time points. For a video, a frame F_t can obtain information such as luma or chroma from LSTM cell comes from frame F_{t-1}. However, from the mechanism and result of paper [6], attention-based network can

provide a one-time calculation way for every time point to capture corresponding information from other time point. Furthermore, this mechanism gives every time point an ability to pay more attention to some specific time points. In video compression work, every frame has one or two references. Thus, the usage of attention may help them to find the references. On the other hand, the essence of attention is several matrix multiplication operations, and the spread and upgrading of GPU can make this prediction task faster than LSTM approach.

2. RELATED WORK

To accurate the inter-prediction of the HEVC, Li et al. [6] proposed a mixed network named Early-Terminated Hierarchical LSTM (ETH-LSTM) to address this issue. In their research, the dataset contains two parts: hierarchical CU partition map (HCPM) and video pictures. All the videos will be split into 64×64 size CTU which accords with the maximum size processed by HEVC. For each video segment, it has corresponding HCPM. HCPM is made up by the splitting flag for every 8×8 non-overleaped CU in each CTU. One HCPM example is shown in Figure 2.

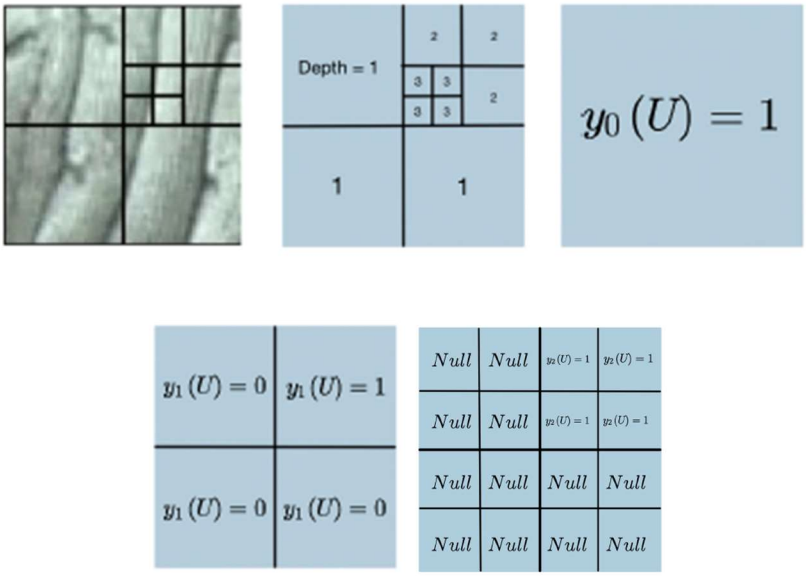


Figure 2: An example of HCPM.

A HCPM contains three levels, level 1, level 2 and level 3 represent the size of CU from 64×64 to 16×16 respectively, and they noted them with $y_0(U), y_1(U)$ and $y_2(U)$

After establishing the database, the dataset will be fed into ETH-CNN to train the CNN part of the system. Then extract the output from the first fully connected layer as the input of the LSTM part of the system (ETH-LSTM). Figure 3 is the schematic diagram.

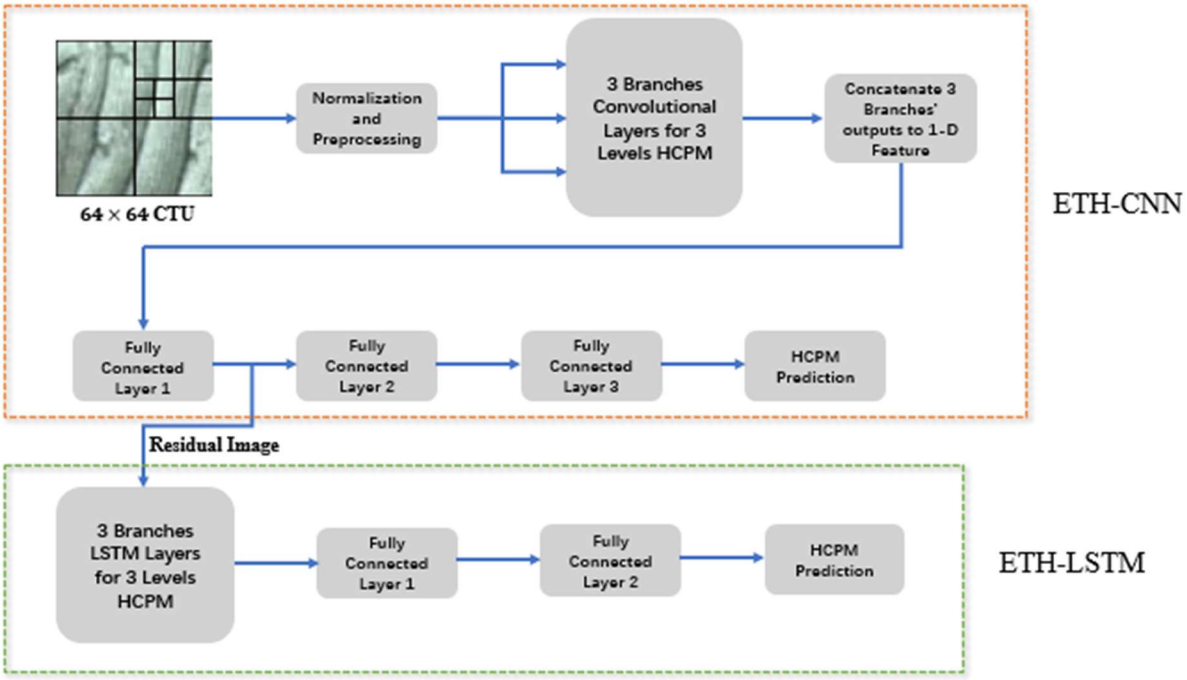


Figure 3: Schematic diagram for ETH-CNN/LSTM

This thesis and research will inherit the HCMP concept and part of the network from [6]. More details of dataset and network will be shown in the following.

3. CTU DATABASE

3.1 Overview of HEVC and HM software

As we mentioned, the core of encoding is coding tree units and coding tree block (CTB) structure, luma CTB and chroma CTB. By default, the size of each CTU is 64×64 pixels and the color sampling standard is $4{:}\,2{:}\,0$. As the name, coding tree, shows, the picture is processed by a quadtree structure and CTU is root for each quadtree. A CTU can be split recursively into different CUs as nodes. Thus, the sizes of CUs can be $64 \times 64, 32 \times 32, 16 \times 16, 8 \times 8$. HEVC picks the optimal combination of CUs in a brute-force way: RDO search. The software will compute RD cost for every combination of CUs from top (64×64) to down (8×8). Figure 4 demonstrates this process between one CU and its sub-CUs. If the sum of 4 sub-CU's RD cost larger than the parent CU, then this parent CU will be split, otherwise it will not be split.

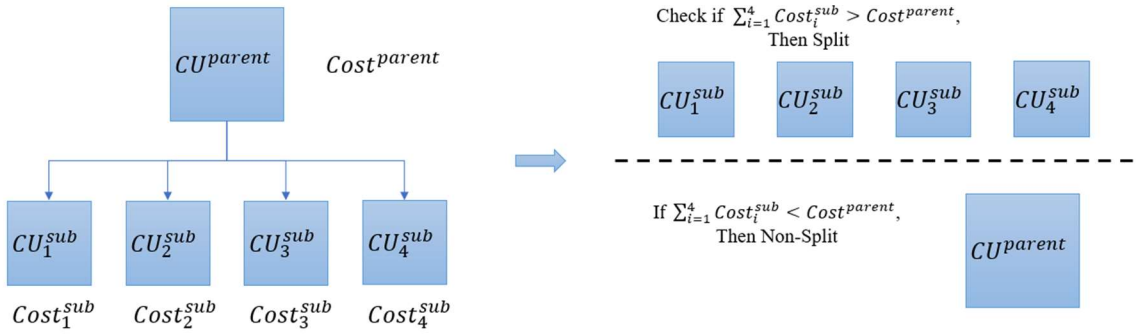


Figure 4: The order of checking and comparing the RD-Cost

Prediction unit (PU) is the syntax that determines how to predict a CU, which means whether to encode a CU by intra-prediction mode or inter-prediction mode. Furthermore, a CU can be split into 1 (same size as current CU) or 2 PUs with 7 different modes. Thus, the sizes of

PUs are ranging from 64×64 to 8×8. Figure 5 shows these different cases. For each PU, HEVC will find it an optimal encoded reference PU block to pair, with lowest RD cost. If current frame is an P slice (only use inter-prediction mode), HEVC will establish a motion vector (MV) candidates list. MV is the vector point from current PU to reference PU. After checking RD costs and pick the best paired reference PU, the software will keep searching with 2 searching algorithms, which are full search or TZsearch. The searching origin is the paired reference PU. In truth, it will consume substantial time to search for each PU and corresponding CU, no matter which algorithm is utilized. In an entire CTU, it contains 1 CU with 64×64 size, 4 CUs with 32×32 size, 16 CUs with 16×16 size and 64 CUs with 8×8 size. Thus, we plan to find a way to detour this exhaustive method.

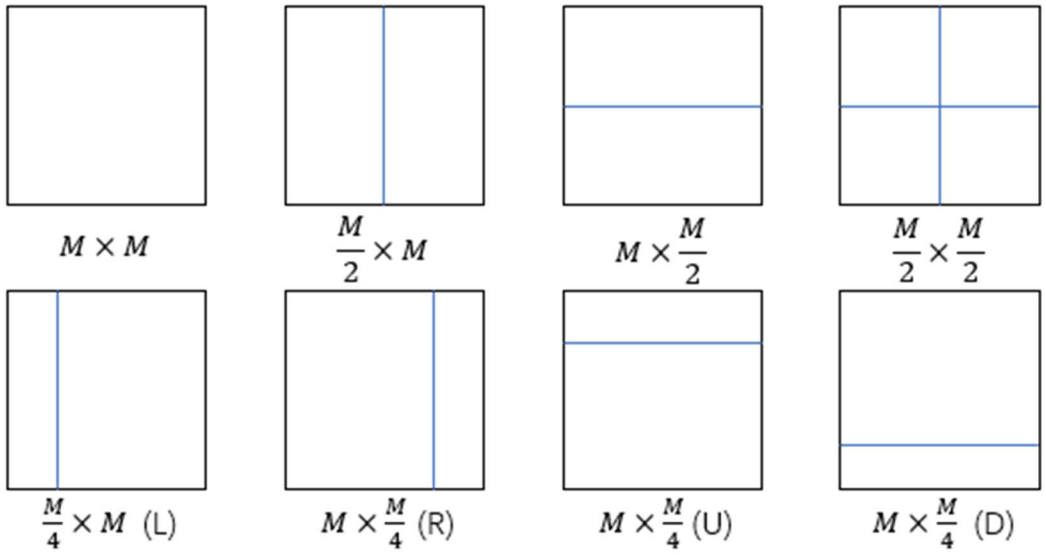


Figure 5: 8 PU modes

3.2 Residual Picture

To compress a picture or a video, current standards always choose the way to only transfer the difference between one block and its reference. Residual data is the name of the difference, and it is residual CTU in HEVC. In truth, residual is obtained on the PU level, and RD cost here can be simplified as SAD, MSE or MPC:

- SAD: $SAD(x,y) = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}|f_i(m,n) - f_{i-1}(m+x,n+y)|$
- MSE: $MSE(x,y) = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}[f_i(m,n) - f_{i-1}(m+x,n+y)]^2$
- MPC:

$$MPC(x,y) = \sum_{m=1}^{M}\sum_{n=1}^{N} T(f_i(m,n), f_{i-1}(m+x,n+y))$$

$$T(a,b) = \begin{cases} 1, & |a-b| \leqslant t \\ 0, & otherwise \end{cases}$$

x, y are the coordinates of MV. M and N are numbers of CTUs in column and row. f_i means i_{th} frame.

After finding best MV and corresponding reference PU, the algorithm of obtaining residual is:

$$Residual = Original\ Current\ PU - Reconstructed\ Reference\ PU.$$

Reconstructed reference is the picture after encoded:

$$Reconstructed\ PU = Reconstructed\ Reference\ PU + Residual.$$

Thus, a reconstructed reference PU comes from its own reference. It should be noticed that reconstructed PU is not equal to the original one due to the following steps, i.e., transform and quantization. These are the reason of distortion.

The reason that choosing residual pictures as inputs is residual pictures contain temporal between CTU and its reference. We also can break one entire frame to 64×64 size blocks as inputs. Another significant point should be emphasized is the residual pictures used in this research

is substituted residual pictures.Transform and quantization also consume time to compute, thus, we omit these two steps can save some computation resources and time.

$$Substituted\ residual = Original\ Current\ \ PU - Original\ Reference\ PU$$

3.3 Partition Information

The goal of this research is to bypass the RDO search, thus, we want to get the result of the CU partition from neural network directly. CU partition is the depth that one CTU was split. For example, if the depth is 0, then this CTU will not be split and keep $64 * 64$ size. The depth equals to 1 means this this is a 32×32 size. Generally speaking, a CTU partition map is 16×16 map, each value represents a 4×4 size square on image. And the layout and storage order of partition depths is Z-scan. Figure 6 demonstrates an example of Z-scan.

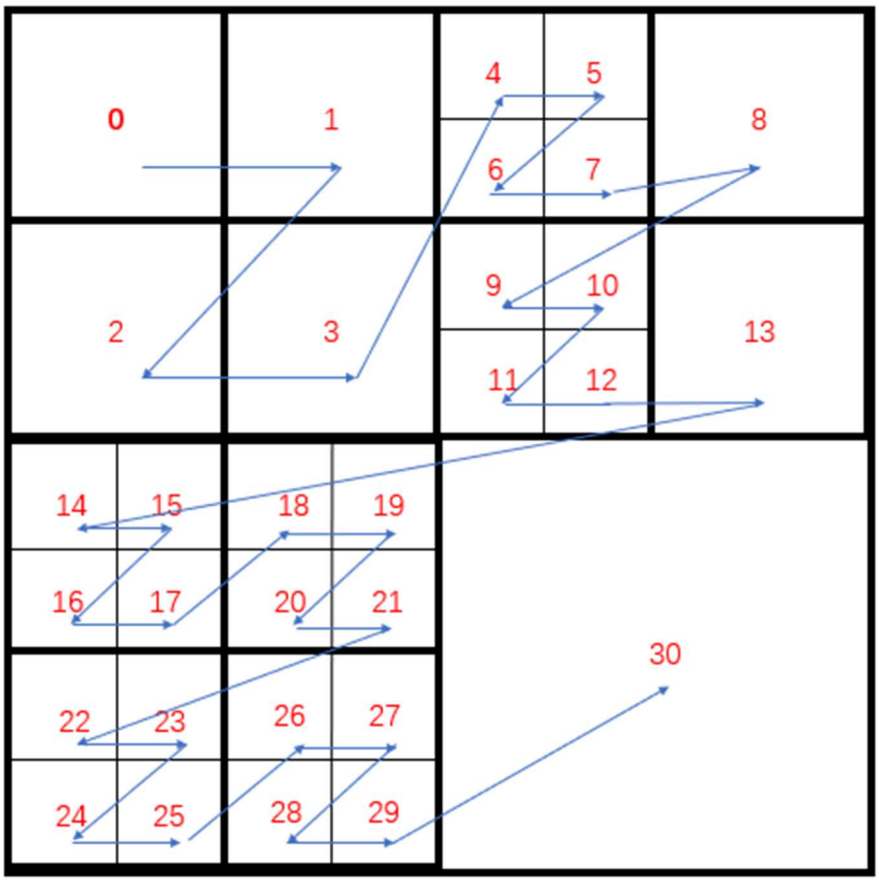


Figure 6: Z-scan

3.4 Configurations

To generate the database, we use HM 16.0 as official software of HEVC. The following are some configurations and modifications.

- Choose low_delay_P.cfg as basic config file. Only the first frame will be compressed as I slice (intra-prediction only), all the following frames will be compressed as P slices (inter-prediction only).
- The specific QP of data will be set as 22. If QP is smaller, the CTU will tend to be split. On the contrast, the tendency is not to be split.
- To generate CTU partition information, the HM software will not be modified. Thus, the CTUs are split optimally.
- To generate substituted residual pictures, the maximum CU partition depth will be set to 64×64 size, which means all the CTU will not be split. The PU mode is $2N \times 2N$, the size of PU is 64×64 therefore. Then, the pointer of calculating residual will be re-pointed to original reference PU instead of the reconstructed one.

4. PROPOSED NETWORK

4.1 Backbone CNN Network

A backbone network is always utilized to extract features that will be fed into the following part for specific task. In this research, ETH-CNN [6] will be the backbone CNN. Figure 7 illustrates the main hierarchy of this backbone network.

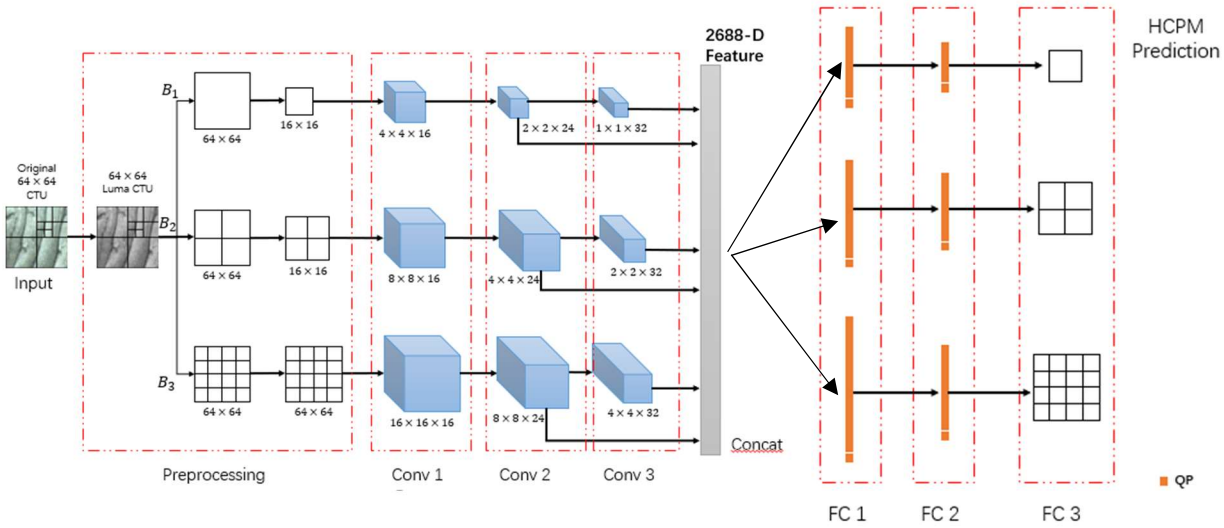


Figure 7: Backbone CNN. For preprocessing and convolution, the size of the output at each layer is shown in black font.

This backbone network contains 4 parts: preprocessing, convolutional layers, concatenating layers, and fully connected layers. There are three branches in the network, B_1, B_2 and B_3. They correspond three levels in HCPM. Here are some details:

- By considering the computational complexity and memory constrains, for a YUV video, only Y (luminance) data will be fed into network. After subtracted by mean value of luminance, the difference between CTU samples. After down-sampling, the feature sizes can be adjusted to fit the output of different levels. On the other hand, for non-split CTU, the contents and textures tend to change smoothly. Therefore, some details can be ignored by down-sampling on size 64×64 and 32×32.

- Convolutional parts contain three layers. Each of branches has 16, 24, 32 kernels in three layers respectively.
- Concatenating layer has 2688×1 dimension by flatting layer 2 and layer 3 from three branches.
- Fully connected layers will be concatenated with QP before being fed into next layers. The outputs of the last layers have same shape with HCPM.

4.2 Attention-based Compression Network

The reason that we need more than CNN is that normal CNNs do not have the ability to process temporal information. Thus, to handle temporal information, this research introduces attention mechanism to address this issue parallelly. Figure 8 illustrates the hierarchy of network. This attention-based network contains three parts: layer normalization, multi-head attention layer and fully connected layers. The layers and corresponding configurations are presented in table I. Here are some details:

- The inputs of this part come from the output of first fully connected layers. Thus, the dimension is 448.
- There are two general normalization layers: batch normalization (BN) and layer normalization (LN). For RNN and LSTM, we also use LN for avoiding vanishing gradient problem. Thus, I choose LN for my ABC network.
- In multi-attention layer, the input will be learned to generate three inputs: Q, K and V. Then these data will be fed to do several scaled dot-productions and produce attentions. Attentions are outputs of this layer.

- There are two fully connected layers. Before feeding data, the input will be concatenated with QP and position information.

TABLE I

NETWROK LAYERS AND CONFIGURATIONS

Network	Layer	Input	Output	Parameters
ETH-CNN	Conv 1 - B1	$16 \times 16 \times 1$	$4 \times 4 \times 16$	256
	Conv 1 - B2	$32 \times 32 \times 1$	$8 \times 8 \times 16$	256
	Conv 1 - B3	$64 \times 64 \times 1$	$16 \times 16 \times 16$	256
	Conv 2 - B1	$4 \times 4 \times 16$	$2 \times 2 \times 24$	1,536
	Conv 2 - B2	$8 \times 8 \times 16$	$4 \times 4 \times 24$	1,536
	Conv 2 - B3	$16 \times 16 \times 16$	$8 \times 8 \times 24$	1,536
	Conv 3 - B1	$2 \times 2 \times 24$	$1 \times 1 \times 32$	3,072
	Conv 3 - B2	$4 \times 4 \times 24$	$2 \times 2 \times 32$	3,072
	Conv 3 - B3	$8 \times 8 \times 24$	$4 \times 4 \times 32$	3,072
	FC 1 – B1	2688×1	64×1	172,032
	FC 1 – B2	2688×1	128×1	344,064
	FC 1 – B3	2688×1	256×1	688,128
	FC 2 – B1	65×1	48×1	3,120
	FC 2 – B2	129×1	96×1	12,384
	FC 2 – B3	257×1	192×1	49,344
	FC 3 – B1	49×1	1×1	49
	FC 3 – B2	97×1	4×1	388
	FC 3 – B3	193×1	16×1	3,088
ABC Network	Attention - B1 (t)	20×68	20×68	256
	Attention – B2 (t)	20×132	20×132	256
	Attention – B3 (t)	20×260	20×260	256
	FC 1 – B1	20×73	20×48	172,032

TABLE I Continued

Network	Layer	Input	Output	Parameters
ABC Network	FC 1 – B2	20×137	20×96	344,064
	FC 1 – B3	20×265	20×192	688,128
	FC 2 – B1	20×53	20×1	3,120
	FC 2 – B2	20×101	20×4	12,384
	FC 2 – B3	20×197	20×16	49,344

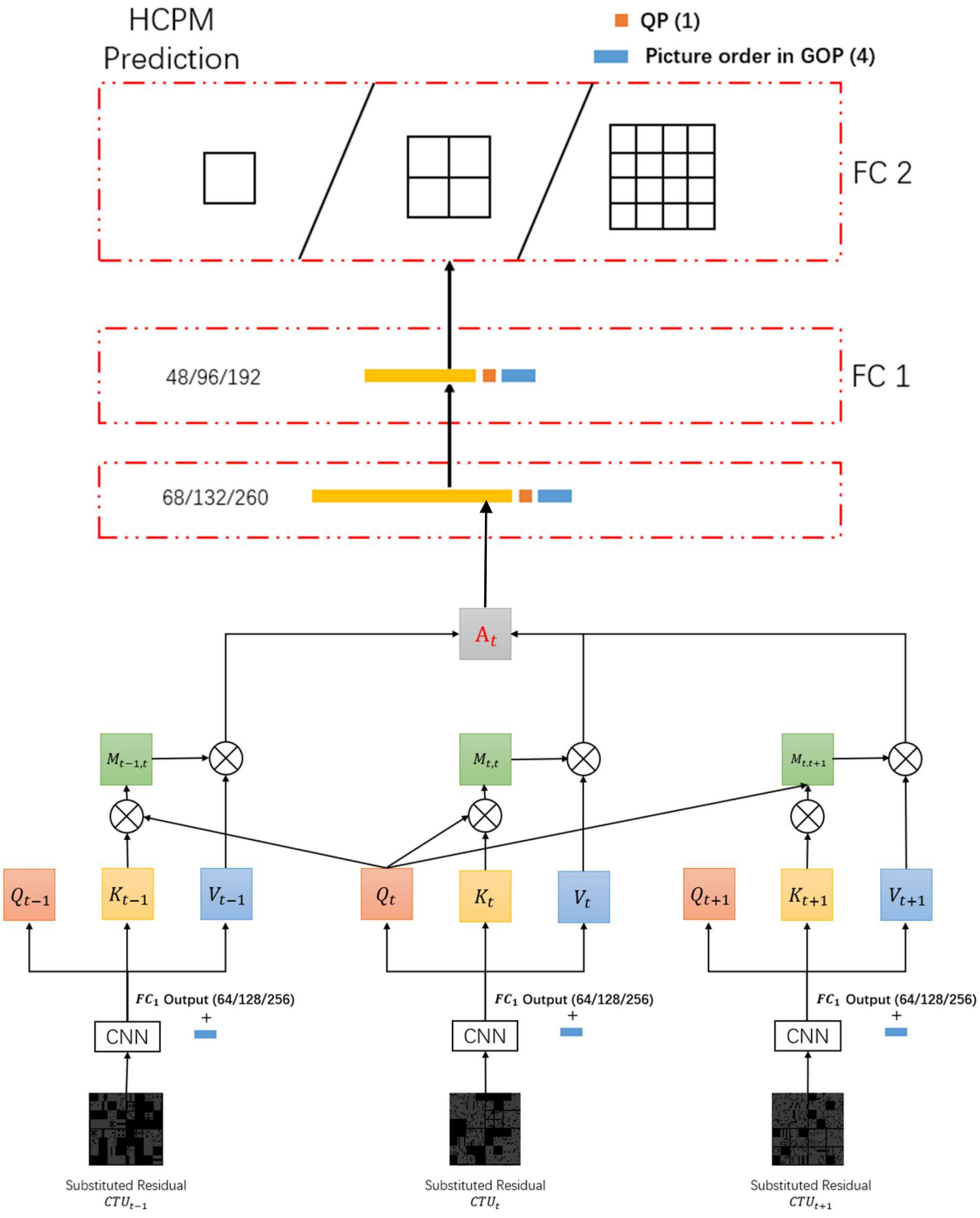


Figure 8: ABC Network.

5. ESTABILISHED DATABASE

5.1 Video Sets

The database CPH-inter [10] was made up of 111 raw videos with YUV format. 6 of them were picked from M. Xu's former work [7], 18 video sequences were from Collaborative Team on Video Coding (JCT-VC) standard test set [8] and 87 were from Xiph.org Video Test Media [9]. All these videos were cropped to 10 seconds. Various resolutions were comprised, SIF (352×240), CIF (352×288), NTSC (720×486), 4CIF (704×576), 240p (416×240), 480p (832×480, 720×480), 720p (1280×720), 1080p, WQXGA (2560×1600) and 2K (2048×1080). Then, all videos were split into 3 sets, training set (83), validation set (10) and test set (18).

5.2 Substituted Residual

As we mentioned before, substituted residuals will be treated as inputs. We can forecast intuitively that the performance will decrease since the ground-truth labels come from the standard program of HEVC, but the substituted residuals were generated from modified HEVC. Modified HEVC jumped the encoding phase, so it can save time at the meanwhile. Here is a comparison between real residual and the substituted one. The measurement of the difference is MSE. Figure 9 shows an example contains original image, corresponding real residual image and substituted residual image and the difference between real and substituted residual image from the video *BasketballPass_416x240, frame 285.* The total MSE-Y of *BasketballPass_416x240, frame 285* is 771.132, which is the largest one in all 500 frames. The top left is one real frame with three components Y, U and Y. However, due to the inputs are just Y information, the real residual (bottom left) and substituted residual (bottom right) contain Y only. By generally comparing real and substituted residual with eyes, we can find that they don't show much difference. After

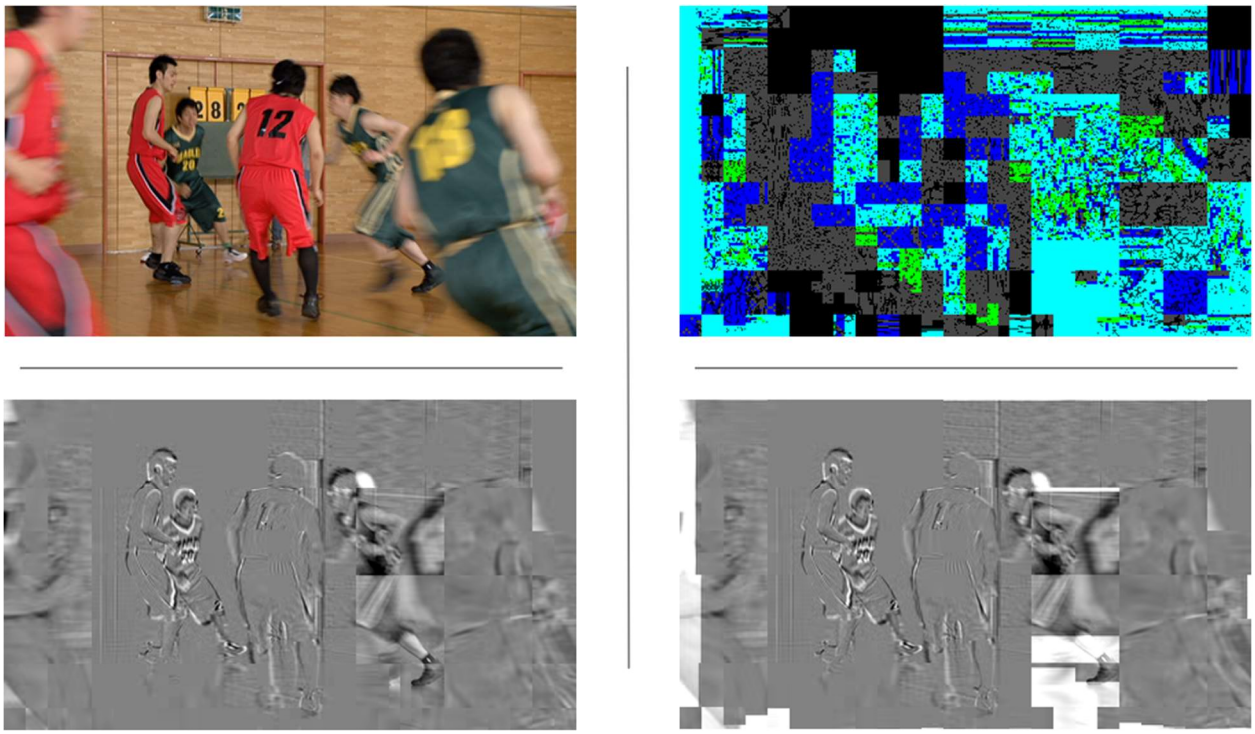


Figure 9. These four pictures are original image, residual difference, real residual, and substituted residual.

checking difference picture and corresponding values, a conclusion can be found that most of the regions' differences are small. There are three different regions in the difference image, and I named them mixed region, black region, and pure region. As shown in figure 9 and figure 10, mixed regions contain several colors in one block, pure regions just have one color except black, and the last one, black regions just contain black. Go into the specific value, different color comes from different combination of Y, U, V. In mixed region, the differences of Y, U, V are quite small, especially in black one, the differences are 0. In pure region, the differences are greater and most of the error in MSE comes from here. Luckily, this main contributor of MSE just occupies small area. The reasons that errors occur are different. On the one hand, for real and substituted residuals whose MVs are same, the errors come from the difference between original PU and homologous reconstructed PU (references have same coordinate). On the other hand, errors come from different

MVs. In this case, the MSE is always greater than the former one. Even though the MSE can reach 1000, it always shows characteristics that rare and concentrated based on the above analysis. Thus, the usage of substituted residual is analysis and won't reduce the performance too much.

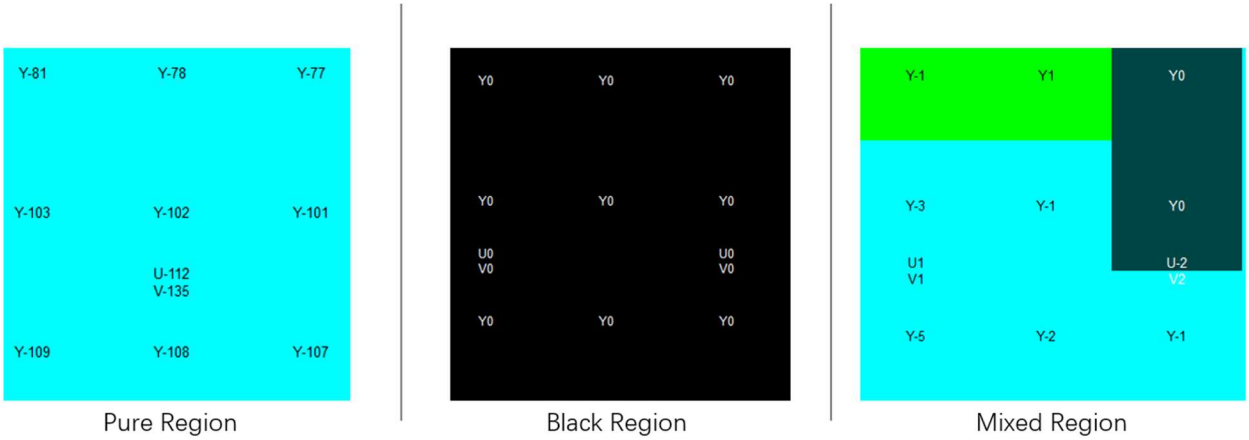
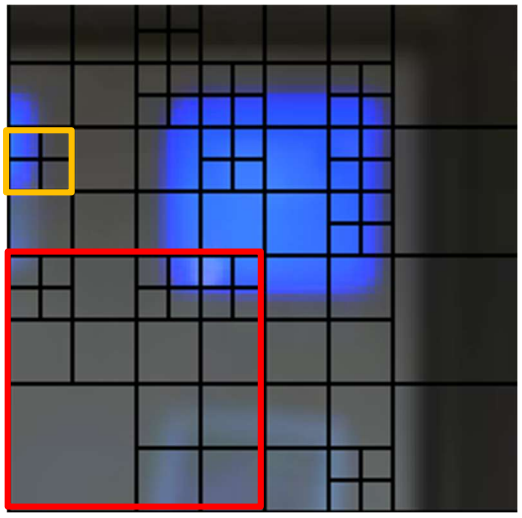


Figure 10. Three different regions

5.3 CU Partition and HCPM

After running HEVC and generate CU partition file, it can be found that the finest unit is 8×8 size, which means each number represents the depth of 8×8 picture. Figure 11 is an example. The top left (TL) picture is a cropped frame from video $akiyo\_cif.yuv$ with size 128×128 (4 CTUs). The background is original image of video and black solid lines are visualized partition mode. The top right (TR) figure is corresponding CU partition file. For each 4×4 image, this figure shows partition depth at the same location. For example, for the part framed by yellow rectangle in TL figure, the relating depth is 3 shown in TR figure. After obtaining CU partition, we can transform it to HCPM. The HCPM shown in the bottom belongs to image and CU partition framed by red rectangle. The whole CTU is split, thus for HCPM level 1, all flags are 1 which means split. For bottom left CU (32×32), the depth is 1, thus the flags of its four sub-CUs (16×16) are 0 in level 2. In level 3, theses 4 sub-CUs' flags are neither split (1) nor

non-split (0), they are marked as *null* or *ff*. Figure 11 is the real format that CU partition information was stored and loaded, the basic unit is 8 × 8. Figure 2 is an abstract expression in different levels.



Address	0	1	2	3	4	5	6	7
00000000	02	02	03	02	02	02	01	01
00000016	02	02	03	03	02	03	01	01
0000002c	03	02	02	03	02	03	01	01
00000042	02	02	02	02	02	03	01	01
00000058	03	02	03	03	02	02	01	01
0000006e	02	02	02	02	02	02	01	01
00000084	01	01	02	02	02	02	01	01
0000009a	01	01	02	02	02	03	01	01

L1				L2				L3			
01	01	01	01	01	01	01	01	01	00	01	01
01	01	01	01	01	01	01	01	00	00	00	00
01	01	01	01	00	00	01	01	ff	ff	00	00
01	01	01	01	00	00	01	01	ff	ff	00	00

Figure 11. The top left is a crop of a picture with size 128 × 128 (4 CTUs). The top right one is corresponding CU partition information. The bottom is HCPM after being transformed from CU partition.

6. EXPERIMENT RESULTS

In this section, the configuration will be set, and the model is trained and tested to show the efficiency of our approach. To set up comparison, ETH-CNN/LSTM approach in [6] is run on real residual database, then our approach, ABC, will be run on substituted residual. Original HEVC is also run on the test set.

6.1 Configuration and Setting

General Setting. In our experiments, all approaches were established on standard HEVC software with version HM 16.5. For inter mode, the basic configuration file and source file is *encoder_lowdelay_P_main.cfg* and *encoder_yuv_source.cfg*. The chosen QP value for compression is 22. As we mentioned before, we picked 18 videos for testing, moreover, these videos are from JCT-VC standard test set [8]. For evaluating the performance, we choose the accuracy of CU partition and time consumption as measurements. The platform software and hardware information is: Linux (CentOS 7), Intel Xeon E5-2680 v4 2.40GHz 14-core, 2 NVIDIA K80 GPU Accelerator 12206MB.

Training Settings. To train CNN part, the total iterations was 1,000,000 with batch size 64. The initial learning rate was 0.01 and decreased by 1% exponentially every 2,000 iterations. The momentum of the stochastic gradient descent (SDG) algorithm was set to 0.9. To train ABC part, the total iterations was 200,0000 with batch size 64. The length of time step was 20 (one sample contains 20 frames). The initial learning rate was 0.01 and decreased by 1% exponentially every 2,000 iterations. The momentum of the stochastic gradient descent (SDG) algorithm was set

to 0.9. Furthermore, to increase the number and diversity of samples, the overleap between two sample groups is 10 frames.

6.2 Time Consumption on Residual Extraction

After proofing the feasibility of the usage of substituted residual pictures, some more modification can be applied to simplify the processes. Due to modified HEVC can bypass the reconstructing process since the residual comes from original frames. Then, the encoding process can be bypassed, too. Such changes can save some computation complexity, table || shows the time consumption of extracting real residual and substituted residual respectively, and comparison between them. By computing the ΔT and $\Delta T(\%)$, the time can be saved significantly after being modified. From the statistics, we found that for most of the videos, time consumed on extracting substituted residuals was less than extracting real residuals, even nearly reached 20% reduction. Only two videos consumed more time on substituted one than the real. The reason might be that modified software searched more reference frames than the standard one.

TABLE II

TIME CONSUMPTION ON RESIDUAL EXTRACTION

RESOLUTION	YUV file	Time for Real (sec)	Time for Substituted (sec)	ΔT and $\Delta T(\%)$
416x240	BasketballPass	62.62	52.22	-10.4, -16.6%
	BlowingBubbles	50.18	39.62	-10.56, -21%
	BQSquare	58.3	47.1	-11.2, -19.2%
	RaceHorses	40.24	36.09	-4.15, -10.3%
832x480	BasketballDrill	191.61	163.08	-28.53, -14.9%
	BQMall	209.26	189.16	-20.1, -9.6%

TABLE II Continued

RESOLUTION	YUV file	Time for Real (sec)	Time for Substituted (sec)	ΔT and $\Delta T(\%)$
832x480	PartyScene	188.1	154.14	-33.96, -18%
	RaceHorses	171.98	153.72	-18.26, -10.6%
1280x720	FourPeople	377.65	389.88	4.6, 1.3%
	Johnny	394.56	353.31	-41.25, -10.5%
	KristenAndSara	370.92	384.45	13.53, 3.6%
1920x1080	BQTerrace	1076.1	1039.54	-36.56, -3.4%
	Cactus	952.17	872.24	-79.93, -8.4%
	Kimono	506.6	469.02	-37.58, -7.4%
	ParkScene	400.23	391.31	-8.92, -2.2%
	BasketballDrive	1257.01	1130.99	-126.02, -10%
2560x1600	PeopleOnStreet	671.17	580.77	-90.4, -13.5%
	Traffic	424.16	406.31	-17.85, -4.2%

6.3 Performance Evaluation

Training Loss. Training loss is one pivotal measurement to adjust hyper-paraments and evaluate training performance. The left part of figure 12 presents the training loss changes of CNN part as the iteration grows, and the right figure presents loss for ABC part. In figure 12, dash line shows the tendency of training, and solid line for validation. From the loss tendency, we found that the model didn't fall into overfitting and underfitting. Moreover, level 1 (64×64) had the smallest loss, while level 3 (16×16) had the greatest loss. However, for these three levels and two parts, the loss converged rapidly, and reached convergence point in the first 1,000 iterations. The following iterations provided generality and diversity for models.

Prediction Accuracy. Accuracy is an important measurement to evaluate the performance of a network. 18 test videos were compressed and extracted to get real and substituted residual

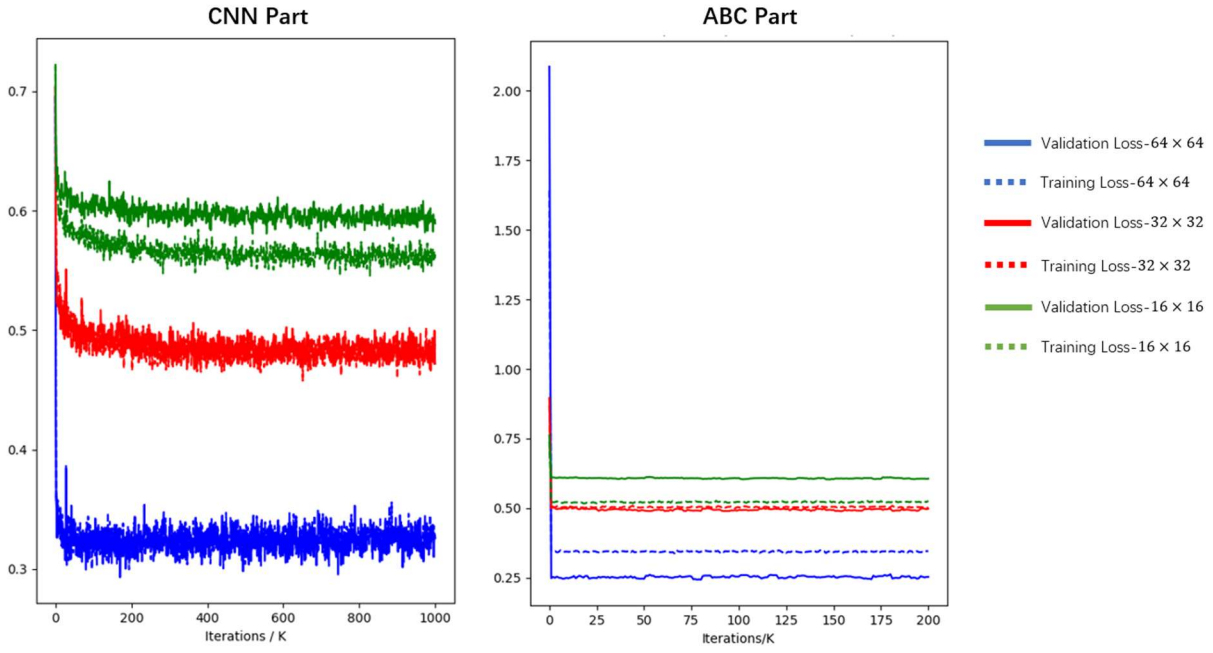


Figure 12. Training and validation loss for training

files, then fed them into [6] approach and ABC to get accuracy, respectively. Figure 13 shows the changes in accuracy during training. For three levels, the accuracy rate converged to 0.85, 0.76 and 0.74 respectively. And table III presents the accuracy rates of all 18 test videos. The average testing accuracy rates of [6] are 93.1%, 82.6% and 72.8%, the average testing accuracy rates of our approach are 89.1%, 79.3% and 71.6%. The corresponding decrease are 4%, 3.3% and 1.2%.

Time Consumption/Computation Complexity. At last, we would like to evaluate the running time of each approach. The baseline was standard HEVC software's searching time. All running time are presented in table IV. Every approach was run three times and then calculated the average running time. Compared to the approach in [6], our approach can save more than 20% time for 10 videos, save 10%~20% time for 7 videos, and only one less than 10%. Furthermore,

table V presents searching time per frame for standard HEVC software's searching time and our approach. It is obviously that our attention-based network can significantly reduce the computational complexity and running time.

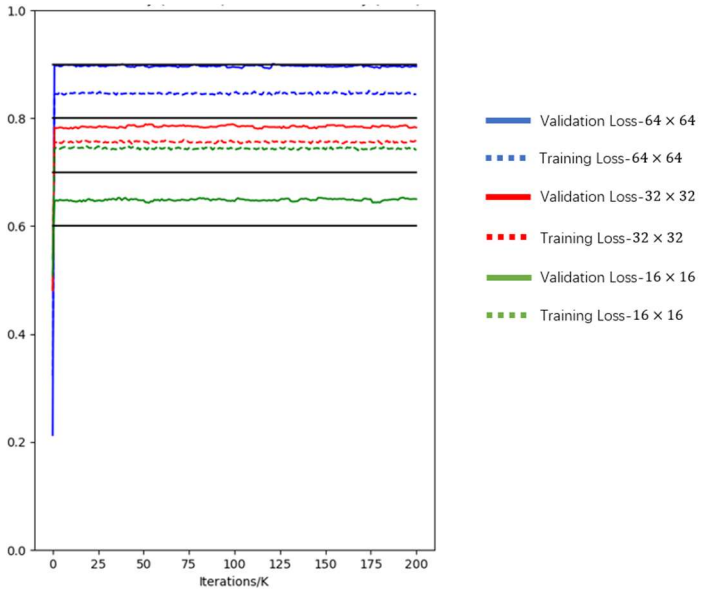


Figure 13. Training accuracy. 4 black lines refer to 0.9. 0.8, 0.7, 0.6 respectively.

TABLE III PREDICTION ACCURACY FOR TEST SET

RESOLUTION	YUV file	ETH-LSTM	OUR	ΔACC
416x240	BasketballPass	94.2%, 86.2%, 75.8%	91.4%, 86.2%, 75.1%	-2.8%, 0, -0.7%
	BlowingBubbles	99.8%, 90.0%, 68.5%	95.7%, 81.3%, 63.7%	-4.1%, -8.7%, -4.8%
	BQSquare	99.3%, 92.1%, 72.2%	95.7%, 87.0%, 67.9%	-3.6%, -5.1%, -4.3%
	RaceHorses	98.8%, 92.4%, 72.9%	97.1%, 90.3%, 72.1%	-1.7%, -2.1%, -0.8%
832x480	BasketballDrill	85.7%, 83.3%, 73.0%	82.5%, 82.9%, 72.1%	-3.2%, -0.4%, -0.9%
	BQMall	91.5%, 82.1%, 72.7%	85.7%, 77.2%, 71.8%	-5.8%, -4.9%, -0.9%
	PartyScene	99.3%, 87.2%, 72.0%	98.0%, 82.9%, 69.3%	--1.3%, -4.3%, -2.7%
	RaceHorses	97.2%, 85.6%, 71.3%	96.4%, 84.1%, 70.7%	-0.8%, -1.5%, -0.6%
1280x720	FourPeople	87.6%, 77.9%, 75.7%	82.3%, 72.4%, 75.4%	-5.3%, -5.5%, -0.3%
	Johnny	90.9%, 70.7%, 80.7%	81.0%, 68.4%, 80.2%	-9.9%, -2.3%, -0.5%
	KristenAndSara	89.2%, 75.1%, 80.7%	81.8%, 72.2%, 80.1%	-7.4%, -2.9%, -0.6%
1920x1080	BQTerrace	95.5%, 89.2%, 65.8%	94.7%, 87.6%, 64.8%	-0.8%, -1,6%, -1.0%
	Cactus	90.1%, 79.6%, 65.4%	85.3%, 75.5%, 64.4%	-4.8%, -4.1%, -1.0%
	Kimono	90.2%, 71.6%, 81.0%	86.7%, 69.5%, 81.1%	-3.6%, -2.1%, +0.1%
	ParkScene	93.5%, 82.7%, 69.3%	84.3%, 75.7%, 68.5%	-9.2%, -7.0%, -0.9%
	BasketballDrive	86.3%, 78.2%, 72.9%	84.4%, 74.3%, 72.3%	-1.9%, -3.9%, -0.6%
2560x1600	PeopleOnStreet	97.3%, 84.5%, 70.1%	95.8%, 83.8%, 69.8%	-1.5%, -0.7%, -0.3%
	Traffic	89.8%, 78.7%, 70.3%	84.2%, 76.6%, 69.6%	-5.6%, -2.1%, -0.7%

TABLE IV

RUNNING TIME ON PREDICTION

RESOLUTION	YUV file	FRAMES #	Eth-LSTM (sec)	OUR (sec)	$\Delta T(\%)$
416x240	BasketballPass	480	7	4.67	-33%
	BlowingBubbles	480	7	5	-29%
	BQSquare	580	7	5	-29%
	RaceHorses	280	6	4.33	-28%
832x480	BasketballDrill	480	12	10	-17%
	BQMall	580	15.33	12.33	-20%
	PartyScene	480	14.33	12	-16%
	RaceHorses	280	9.67	7.33	-24%
1280x720	FourPeople	580	18.33	16.33	-11%
	Johnny	580	17.33	14.33	-17%
	KristenAndSara	580	19	14.67	-23%
1920x1080	BQTerrace	580	72	56.33	-22%
	Cactus	480	50.67	40.67	-20%
	Kimono	220	20.33	17.67	-13%
	ParkScene	220	24.33	22.33	-8%
	BasketballDrive	480	52.33	39.33	-25%
2560x1600	PeopleOnStreet	140	33.67	30	-11%
	Traffic	140	29.33	24.33	-17%

TABLE V

AVERAGE SEARCHING TIME OF HEVC

RESOLUTION	AVERAGE SEARCHING TIME BY HEVC	AVERAGE PREDICTIONG TIME BY ABC NETWORK
416x240	3.04 sec/frame	0.01 sec/frame
832x480	12.02 sec/frame	0.023 sec/frame
1280x720	12.53 sec/frame	0.026 sec/frame
1920x1080	51.65 sec/frame	0.089 sec/frame
2560x1600	115.08 sec/frame	0.194 sec/frame

6.4 Analysis

Foreseeably, the usage of substituted residual will reduce the accuracy rate. However, if the other aspect can compensate the loss, they can build up a trade-off solution. By comparing the accuracy rate reduction and the time saved, we found that the improvement on time consumption was greater than the drop on accuracy rate for most videos in test set. Only one video, which is *ParkScene.yuv,* the 8% reduction on time was smaller than the 9.2% accuracy rate reduction on level 1 (64×64). The gap was not massive, it can be treated as a glitch, therefore. Furthermore, the ΔT (sec) in table III didn't look large. Since all the video's duration was cropped to 10s in length, the longer the video is, the more time can be saved. Furthermore, since the attention calculation can be abstracted as several matrix multiplications which is the advantage to GPU, as the growing of GPU's ability and improvement of CUDA Toolkit, our attention-based network can gain more reduction on consumed time than HEVC run on CPU only and LSTM run on GPU sequentially.

7. FUTURE WORK

A large part of how good an idea is depending on how it serves reality. Our work detours the searching process by using deep neural network and reduce the computational complexity remarkably. Nevertheless, the prediction accuracy rates were not good enough, especially on level 2 (32×32) and level 3 (16×16). Thus, to guarantee the compression quality, we need some method to compensate and raise the accuracy rates of CU partition.

Set thresholds and call standard HEVC software to search is a way. Setting thresholds means give some specific intervals, then discriminant whether the prediction outputs (0~1) of network fall into the intervals. For example, the threshold of level 3 can be set as 0.7. Then the intervals are $[0,0.3)$ and $(0.7,1]$. If the prediction is 0.68, then the partition flag is undecided. Only predict to split when prediction is over 0.7 and predict to nun-split when prediction is blow 0.3. The default threshold for every level in this research is 0.5. After obtaining all the CU partition predictions and being read by HEVC, decided CU can be used by HEVC directly, then do a normal searching for undecided CU. The whole process is presented in figure 14.

Furthermore, the method of extracting residual can be upgraded, too. From table I, we found that extracting residual costed lots of time, thus we can use deep learning approach to replace this part of work.

After defense, we will keep on this research to complete a system in figure 14 that can be released.

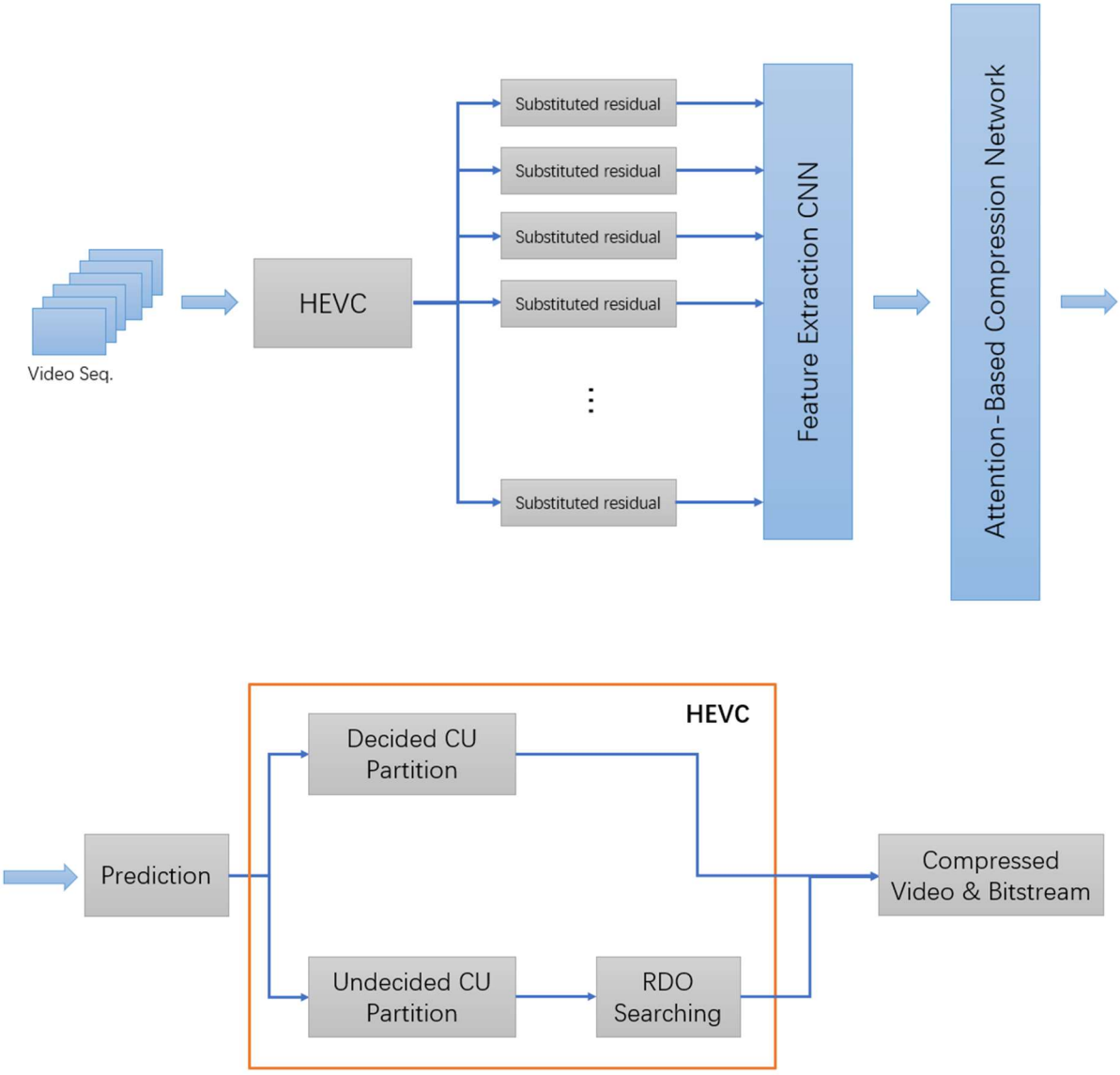


Figure 14. The workflow of our HEVC + ABC Network system.

REFERENCES

[1] G. J. Sullivan, J. R. Ohm, W. J. Han and T. Wiegand, “Overview of the high efficiency video coding (HEVC) standard,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[2] G. Corrˆea, P. Assuncao, L. Agostini, and L. A. da Silva Cruz, “Performance and computational complexity assessment of high-efficiency video encoders,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1899–1909, Dec. 2012.

[3] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. Neural computation, 9(8):1735–1780, 1997.

[4] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. CoRR, abs/1412.3555, 2014.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser and Illia Polosukhin. “Attention is all you need”. arXiv:1706.03762 [cs.CL].

[6] Xu, Mai, et al. "Reducing complexity of HEVC: A deep learning approach," IEEE Transactions on Image Processing, Vol. 7, No. 10, 2018, pp. 5044-5059.

[7] T. Li, M. Xu and X. Deng, "A deep convolutional neural network approach for complexity reduction on intra-mode HEVC," 2017 IEEE International Conference on Multimedia and Expo (ICME), Hong Kong, Hong Kong, 2017, pp. 1255-1260.

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

[9] Xiph.org, Xiph.org Video Test Media, https://media.xiph.org/video/derf/ (2017).

[10] A large-scale database for CU partition of HEVC (CPH), https://github.com/tianyili2017/CPH