HEVC FAST CU DECISION FOR INTRA-PREDICTION BY CNN

A Thesis

by

JIANFENG SONG

Submitted to the Office of Graduate and Professional Studies 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
	Tie Liu
	Zhangyang Wang
Head of Department,	Miroslav M. Begovic

December 2019

Major Subject: Electrical Engineering

ABSTRACT

High Efficiency Video Coding (HEVC) is also know as H.265 was first official introduced in 2013, it is one of the video coding standard of the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group. In the original algorithm, the coding unit depth decision was made by applying a recursive search from top depth 0 to bottom depth 3 on all possible quad-tree structures. Therefore this algorithm is considered to be very time consuming and computational expensive. In my research, I have compared two different methods of reducing the computational complexity for HEVC by using Neural Networks, and propose a new structure of neural network and provide the corresponding result.

DEDICATION

To my mother, my father, my grandfather, and my grandmother. To who is working on optimization of video compression by using deep neural networks, without you this work would not has been done by now.

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Chao Tian, and my committee members, Dr. Zhangyang Wang, Dr. Tie Liu, and Dr. Ulisses Braga-Neto, for their guidance and support throughout the course of this research.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at Texas AM University a great experience.

Finally, thanks to my mother and father for their encouragement and love.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Tian, Chao and Professors Braga-Neto, Ulisses and Liu, Tie of the Department of ELEN and Professor Wang, Zhangyang of the Department of CS.

The ideas in Chapter 4 and the comparison depicted in Chapter 5 were guided by Professor Tian, Chao. The network structures of I and II in Chapter 4 are reproduced from K. Kim and W. W. Ro's work [1], and T. Li et al.'s work [2].

All other work conducted for the thesis was completed by the student independently.

Funding Sources

There are no outside funding contributions to acknowledge related to the research and compilation of this document.

NOMENCLATURE

HEVC	High Efficiency Video Coding
QP	Quantization Parameter
RD	Rate-Distortion

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

As increasing of high-resolution videos such as 2k, 4k, and 8k video, the efficiency and quality of the older algorithm H.264 (Advanced Video Coding) is out of date, since the basic coding unit (macro block) only support the size of $16 * 16$ which limited the performance of H.264. Therefore the size of the coding unit should be reconsidered, and the next generation of video coding–HEVC has a flexible size of coding units from $64*64$ to $8*8$ and that important feature made HEVC much better than H.264. For each coding unit in HEVC, there will be a quad-tree structure which can be calculated using recursive search, and this coding unit is called CTU(coding tree unit) , the size of a CTU can be represented by its depth, there are 4 depths in HEVC which are $0, 1, 2, 3$ represent size of $64, 32, 16, 8$ respectively. The information which is mentioned above is contained in[3].

For each CTU, the optimal splitting quad-tree is calculated by applying a recursive search from largest CTU with the size 64 by 64 to the minimum supported CTU with size 8 by 8. The decision is made by comparing the RD (Rate Distortion ratio) cost. For example, to determine whether a CTU with the size $64 * 64$ need to be split into smaller CTUs, we need to calculate the RD cost for the CTU itself, and also the RD cost for its four sub-part CTUs with the size of $32 * 32$, and for each sub-part CTU we need to compare the RD cost for itself and its sub-part CTUs, in order to get the optimal quad-tree, we need to apply this method recursively until depth 3, then choose the best depth for each sub-part CTU and form the optimal splitting quad-tree.

In order to optimize the compression efficiency, in intra-mode, the RD cost is exhaustively calculated by finding all PU modes for all depths, but the computational complexity is high which made the encoding process highly time consuming. In order to address this problem, some improvement on HEVC algorithm has been made on [4, 5].

Some of those algorithms are accepted by HEVC standard, but they all focus on early termination on the recursive search if the current CU satisfies some conditions. Therefore, the problem of computational complexity remain unsolved. On the other hand, machine learning based CU depth prediction is also considered in [6, 7, 8, 9]. [9] has the smallest time saving which is about 30% on

average with 0.9% penalty on RD-cost, and [6, 7, 8] are having an average time saving about 50% with 2% penalty on RD-cost.

In this paper, we present a comparison of three fast CU depth decision methods which aim to solve computational complexity by using neural networks, different than early termination, neural networks are used to replace recursive search in HEVC algorithm. We will first introduce [1] and [2] which are two neural networks proposed by K. Kim and W. W. Ro, and T. Li et al., then we will propose a new network structure which is inspirit by them and give the overall comparison.

This paper is organized as follows. Section II is the background of HEVC. The detail of database construction will be discussed in section III. Section IV contain the detail of neural networks, including structure and hyper-parameter setting. Section V is comparison of performance of neural networks based on training and testing results. Finally, the summary of this paper is in section VI.

2. BACKGROUND OF HEVC

The HEVC standard was finalized in 2013 which designed to achieve maximize compression efficiency for high resolution video. Different than H.264, HEVC has added a flexible size of coding unit. For the most of cases, the maximum coding efficiency is achieved by dividing the frame into the size of $64 * 64$ CTUs and apply exhaustive search on it for all possible combination of all depths and prediction modes. In the encoding process, one $64 * 64$ CTU can be split into 4 equal size CUs, and each smaller CU can be further split into even smaller CUs until the size $8 * 8$ CU which is the minimum supported size of in HEVC. Figure 2.1 shows an example of CTU with the partition.

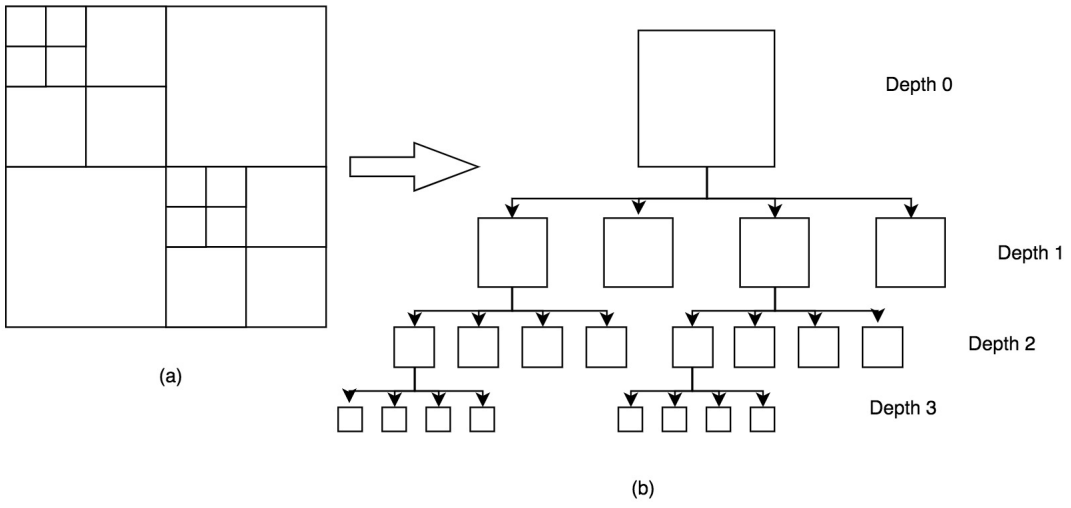


Figure 2.1: (a) is a $64 * 64$ CTU with partition. (b) is the splitting quard tree for (a).

In order to calculate the optimal splitting quad-tree for a $64 * 64$ CTU, We shall define the prediction mode first. In this paper, we only consider intra-mode as our prediction mode, since for inter-mode the prediction of motion compensation is hard to learn by neural networks.

Intra-mode takes each frame from video independently, and use spatial information for data

compression, because a CU and its neighbors always contain similar information. Then a function that can calculate the RD cost will be applied to the current CU and compare it with the lower depth CUs, in another word, this function will be called recursively to find the RD cost for $32 * 32$, $16 * 16$, and $8 * 8$ CUs, after that, the optimal splitting quad-tree can be determined by compare the total RD cost for each depth.

In intra-mode, only two PU mode can be considered which are $2N * 2N$ and $N * N$, $2N * 2N$ can be only used in CU size larger than $8 * 8$, and for size $8 * 8$, PU mode $N * N$ and $2N * 2N$ is used. The PU mode is showing in Figure 2.2. The luminance and chrominance angles are

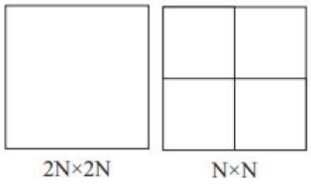


Figure 2.2: Available PU modes for intra-mode. This figure is reprinted from [3]

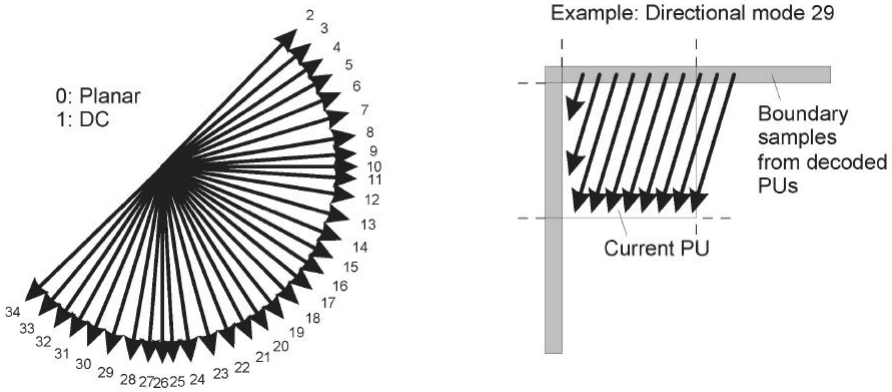


Figure 2.3: Available angles for intra-mode. This figure is reprinted from [3]

used to reconstruct a block by copying the information of boundary pixels and filling the block

follow the angular direction. They can be found for current PU by applying exhaustive search on all possible angles. Then the combination of PU mode and angle which has the lowest RD cost will be selected, and that RD cost will be the RD cost for the current CU. The available angles for intra-mode is shown in Figure 2.3.

Based on the HEVC algorithm, it is fair to conclude that the maximum coding efficiency can be reached by using exhaustive search on CTU structure and prediction mode, but the computational complexity is increased accordingly.

3. DATABASE CONSTRUCTION

In order to train and test three neural networks that designed to reduce computational overhead in HEVC, we need to choose our database carefully. Since we are going to compare three networks we need to make sure they all train and test on the same dataset,with same QP. Due to the limitation of computational power, for now, we only consider using QP 37, 32, 27,22 with resolution $320 * 384$. Even though the resolution of our train and test set is relatively low, with properly designed network structure, this dataset should be able to give a reasonable estimation of performance of three networks, and the estimation results should be able to extend to a higher resolution with different QP. Figure 3.1 is howing the information of our training database

Training Database			
Video Name	Frame Number	Video Name	Frame Number
akiyo-cif.yuv	300	bowing-cif.yuv	300
bridge-close-cif.yuv	2000	beidge-far-cif.yuv	2101
bus-cif.yuv	150	carphone-cif.yuv	382
coastguard-cif.yuv	300	container-cif.yuv	300
flower-cif.yuv	250	foreman-cif.yuv	300
hall-cif.yuv	300	highway-cif.yuv	2000
news-cif.yuv	300	calendar-cif.yuv	300
slient-cif.yuv	300	Total number	9583

Table 3.1: Training Database

The database was constructed based on the HEVC encoding results. For each frame in YUV video, there is a depth matrix which is formed during HEVC encoding process, this depth matrix is the ground truth for the database. All ground truth data is saved as Plain Text Files. Each frame is divided into multiple $64 * 64$ blocks, each block will be marked as NON-SPLIT if the corresponding location in depth matrix is 0 and SPLIT otherwise. Then the frame is divided into multiple $32 * 32$ blocks, and each block will also be marked as NON-SPLIT if the corresponding

location in depth matrix is less or equal to 1 and split otherwise. Then the frame is divided into even smaller blocks with size $16 * 16$, those blocks is marked as NON-SPLIT if the corresponding location in depth matrix is less than or equal to 2, and split otherwise. Figure 3.1 is an example of how to mark SPLIT or NON-SPLIT for different size of blocks.

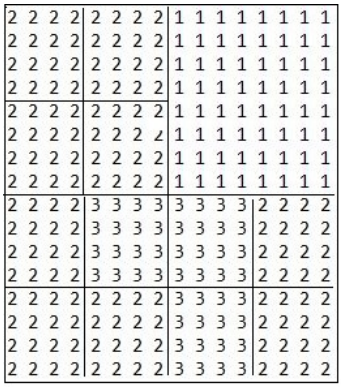


Figure 3.1: This is a depth matrix of a $64 * 64$ sized CU which is marked SPLIT for depth 0, because the elements in this matrix is not all 0. The top right is $32 * 32$ sized block, it is marked as NON-SPLIT for depth 1 since all elements is less than or equal to 1. The top left block is $16 * 16$ sized block which is marked as NON-SPLIT for depth 2 since all elements is less or equal to 2. For the block in third column, third row is marked as SPLIT for depth 2 since all element is greater than 2

This database is different than the database in [1], in [1] if a CU has depth 0 then this CU will not be used in training depth 1 and 2, this method will require more frames to train the network. Therefore, our database has a higher data usage compares to theirs. The splitting equations for

different size is showing below

$$64sizedCU$$

$$SPLIT : a_{ij} > 0 \qquad NON - SPLIT : a_{ij} = 0$$

$$32sizedCU$$

$$SPLIT : a_{ij} > 1 \qquad NON - SPLIT : a_{ij} <= 1$$

$$16sizedCU$$

$$SPLIT : a_{ij} > 2 \qquad NON - SPLIT : a_{ij} <= 2$$

In order to train the network properly, we need to make the number of samples for SPLIT and NON-SPLIT evenly. There are two possible way to do it, one is set a decent weight for the loss function, another way is re-sampling the data. Since the weight in the loss function is very sensitivity, and it requests many tests to set it correctly, we will use re-sampling to make samples evenly. The distribution of our training database before and after re-sampling is showing in talbe 3.2.

	Before Re-sampling		After Re-sampling	
Size of CU	Split	NON Split	Split	NON Split
64	0.8146	0.1854	0.4921	0.5079
32	0.4854	0.5146	0.4854	0.5146
16	0.2144	0.7856	0.5219	0.4781

Table 3.2: Training Database Distribution

Since we are going to compare three networks, there is no need to bring those networks back to HEVC. The reason is that we only need to know their processing time and RD cost for each frame based on the predicted depth, the processing time is neutrally available by applying neural networks, and RD cost information is available during encoding process in original HEVC algorithm. Therefore we extract all RD cost information from HEVC and record them in Plain Text

Files. For each $64 * 64$ sized CU, there are 1 RD cost for depth 0, 4 RD costs for depth 1, 16 RD costs for depth 2 and 64 RD costs for depth 3 correspondingly. The best RD cost for a $64 * 64$ sized CU can be calculated by sum the recorded RD cost corresponding to its depths.

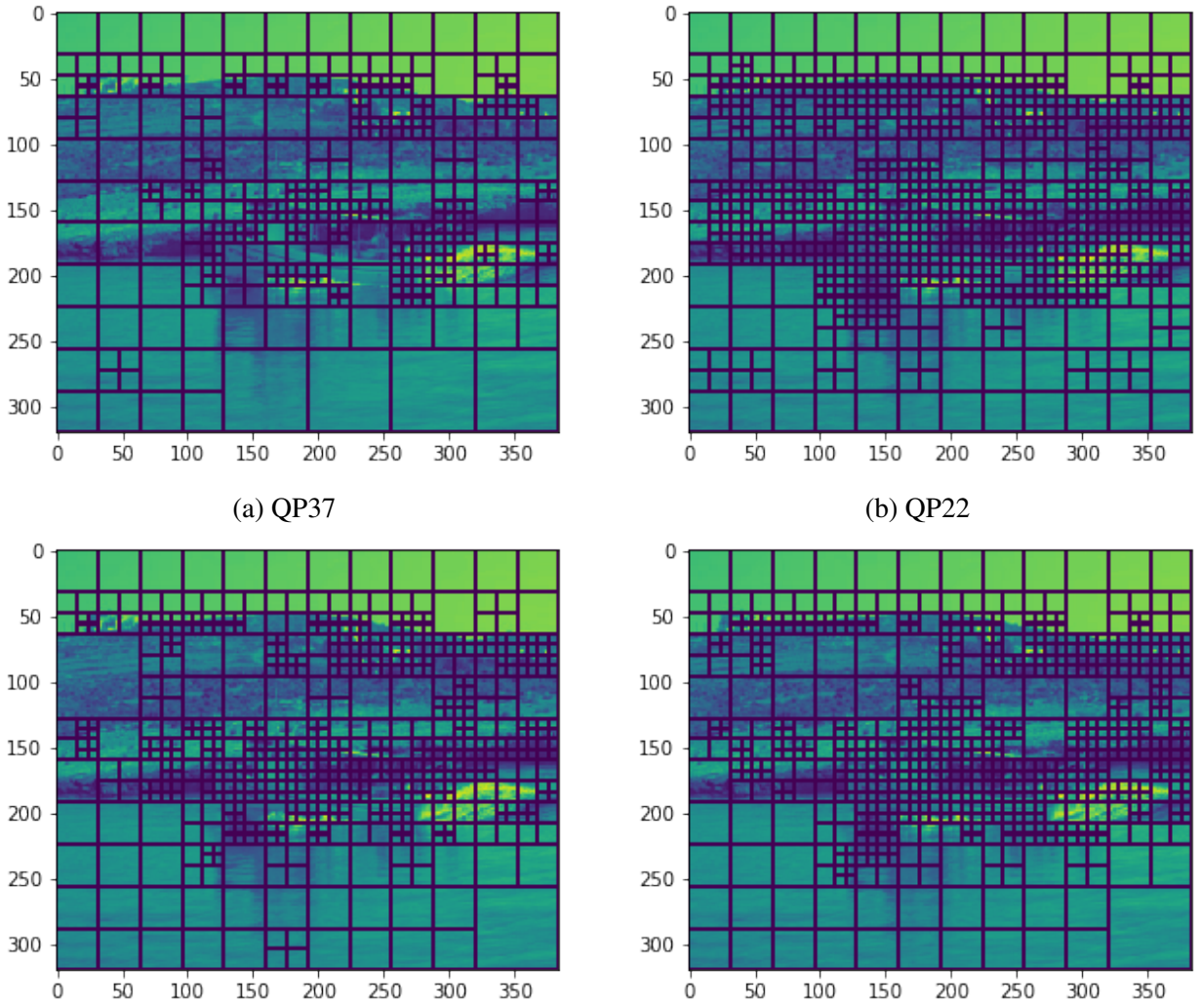
Our test database is completely separated from the training database, we picked some video from video segmentation database, then down-sample them into $320 * 384$ resolution and feed them into HEVC to get the ground truth and RD cost information. Those videos along with their depths and Rd cost information will be used in our evaluation step for comparing three networks. Since it is test database, there are not going to be re-sampled. The information of our test database is showing in table 3.3.

Test Database			
Video Name	Frame Number	Video Name	Frame Number
bear.yuv	82	bike packing.yuv	69
blackswan.yuv	50	bmx bumps.yuv	90
bmx trees.yuv	80	boat.yuv	73
boxing fisheye.yuv	84	breakdance.yuv	84
breakdance flare.yuv	71		
Total Number		683	

Table 3.3: Test Database

Different QPs will lead to different depth information for a same frame. A higher QP means that there will be a higher quantization, more compression and lower quality of reconstructed frame, and the frame tend to be split into larger blocks in encoding processing. The example of a frame that is encoded by using different QPs in HEVC is showing in figure 3.2.

By comparing the encoding result from QP37 to the result from QP22, we can easily point out that QP37 tend to have more larger blocks than QP22. During our training and testing, we also find out two very interesting phenomenons. If a block is tend not to be split in a higher QP but to be split in a lower QP, specially for some background blocks or blocks that contain less information of a frame, then the decision penalty of making a mistake on its depth under same QP is low. The



(a) QP37

(b) QP22

(c) QP32

(d) QP27

Figure 3.2: Frame Splitting Result by Different QPs

second phenomenons is that under same QP, two continuous frame may have a huge difference on their splitting tree, even though the two frame are very similar to each other. This is due to some other processes in HEVC, for example entropy coding, since those processes are only play some minor rules in CU depth decision in HEVC, they will not affect the result of the CU depth if the RD cost of two splitting trees are huge, but if the difference of two splitting trees are small, then they will star to affect the final result, and this is why two frames are similar but having different splitting tree. Since this difference is caused by some minor rules, therefore the penalty of mislabel the depth is also small.

During the training phase, since we are using the picture's information only, features of secondary factors are hard to be captured by our network structure, but such mistakes on predicted depth are not affecting the RD cost much as we have discussed above. Intuitively, we can think such mistake as difference between first and second best result, HEVC is always picking the best result through all possible solutions, but due to the effect of secondary factors our method is only be able to pick the second or third best result.

4.1 Network I

Figure 4.1 represents the layer structure of neural networks in K. Kim and W. W. Ro's work [1]. The network consists of reflection padding layers, convolution layers, batch norm layers,

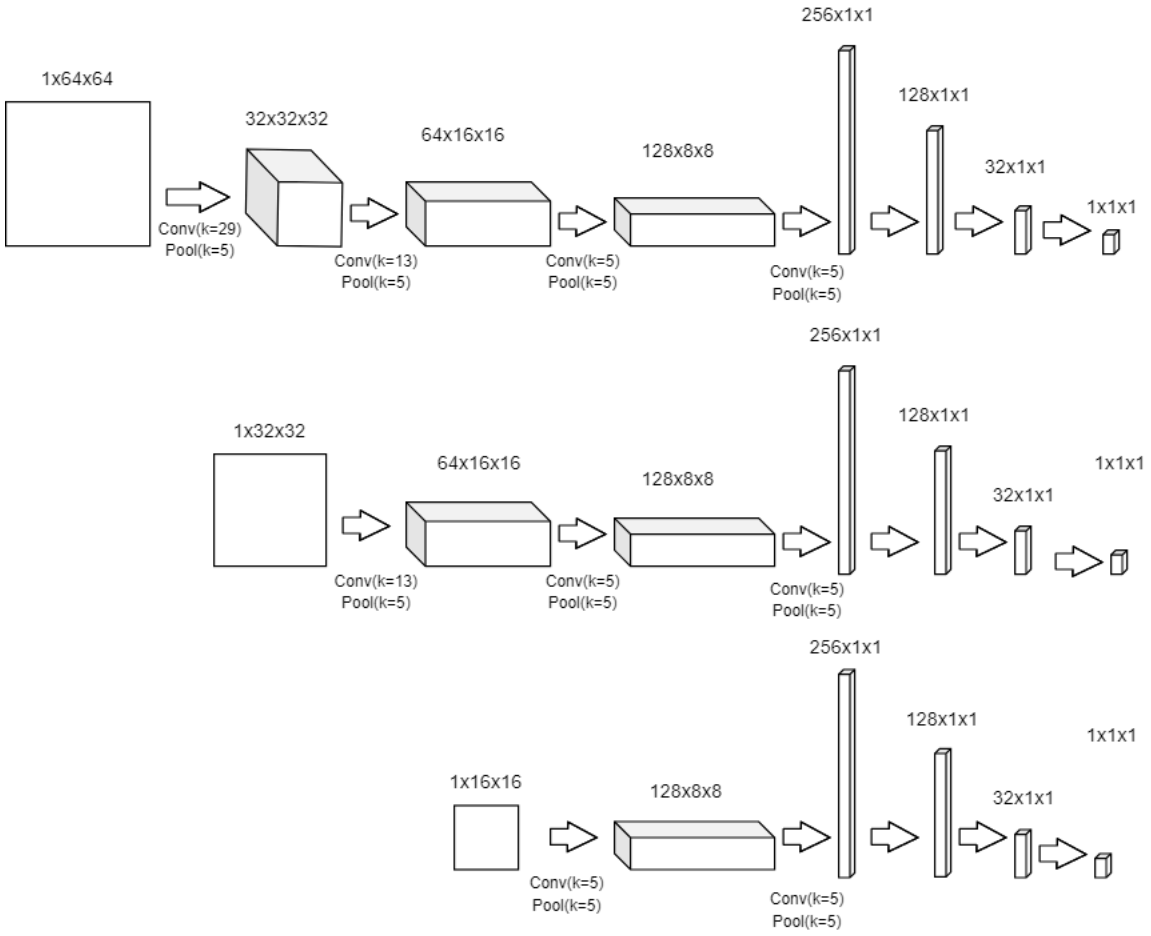


Figure 4.1: Network Structure of Network I. This figure is reproduced from the work of K. Kim and W. W. Ro [1]

Relu activation layers, and max pooling layers. The input image information will first go through a reflection padding layer which can make the size of input and output of convolution layer remain

unchanged, after convolution layer, batch norm layer can make the training more efficiency and Relu activation layer can introduce non-linearity into the output of the networks, then the max pooling layer will extract the key features for further processing.The feature maps from last convolution layer will be feed through three fully connected layers, each layer will randomly drop out 50% of its feature to avoid over-fitting problem and increase training speed. We have changed

	Input Size 64		
Layer	Filter Size	Number Of Filters	Number of Trainable Parameters
Conv1	29x29	32	26944
Conv2	13x13	64	346176
Conv3	5x5	128	204928
Conv4	5x5	256	819456
Fully 1	1x1	128	32896
Fully 2	1x1	32	4128
Fully 3	1x1	1	33
Total Parameters			1434561
	Input Size 32		
Conv1	13x13	64	10880
Conv2	5x5	128	204928
Conv3	5x5	256	819456
Fully 1	1x1	128	32896
Fully 2	1x1	32	4128
Fully 3	1x1	1	33
Total Parameters			1072321
	Input Size 16		
Conv1	5x5	128	3328
Conv2	5x5	256	819456
Fully 1	1x1	128	32896
Fully 2	1x1	32	4128
Fully 3	1x1	1	33
Total Parameters			859841

Table 4.1: Network Configuration

it a little bit based on our understanding of HEVC, the original structure of [1] also take a vector data as input, but this vector data is generated randomly based on a distribution from data mining which represent the PU model and angle information in the prediction process of HEVC algorithm

for current Cu. Since it is randomly generated, each frame may have multiple vector data during the training process which can reduce the training speed. More importantly, it will also affect the testing result since during testing we also need to give a random vector data as input. The reason is that adding such vector data will force the network to remember the pattern of the vector data itself, which will make the predicted result depend mostly on the vector data and this is against the neural of HEVC algorithm. Therefore we removed the vector data from their structure for intra-prediction. The second change is that we are using luma information of a frame in our training and testing process only, because for most of cases, CU partition is the same as luma partition. During the testing, we found out that those two changes does not affect the performance of the network very much and we will discuss the result in the later section.

The networks are designed for different input size, there are totally 3 networks for depth 0, 1, and 2. The output of the network is the probability of splitting the current CU, if the probability is greater than 0.5 then the current CU will be marked as SPLIT. Table 4.1 shows the configuration of Network I.

4.2 Network II

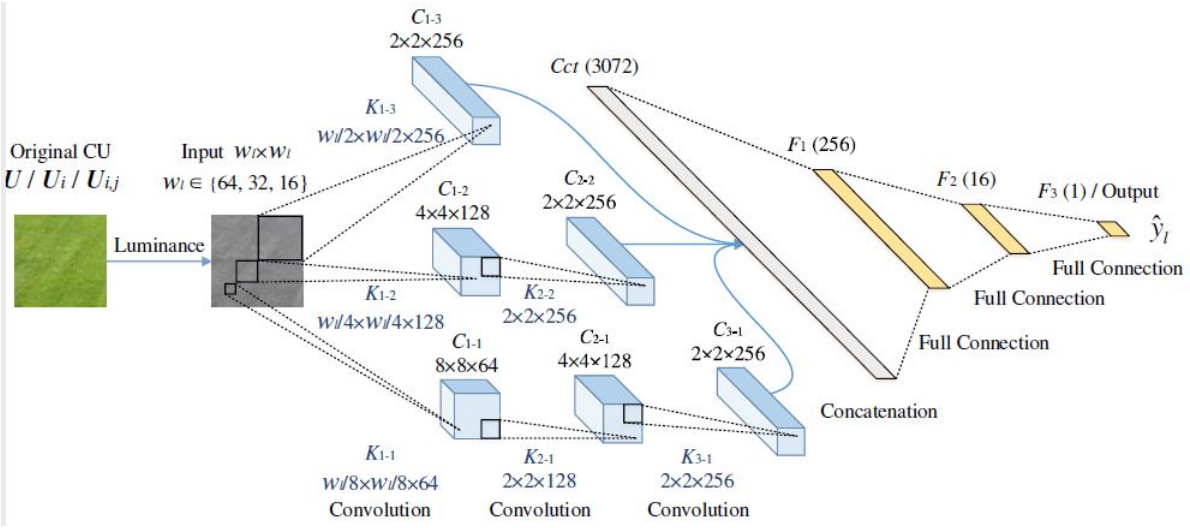


Figure 4.2: Network Structure of Network II. This figure is reprinted from the work of T. Li et al. [2].

The network structure of network II by T. Li et al.[2] is showing in figure 4.2. Same as the network I, according to CTU partition structure in HEVC, they also provide three networks corresponding to different input size, but the structure of their networks are the same, only the hyper-parameter of the first convolution layer can be changed corresponding to different input size. The input layer is luma-infomation of the frame, and the input will be feed into three sub-networks, flowed by convolution layers, batch norm layers, and Relu activation layers, all filters in all convolution layers are non-overlapping, which means the output size of convolution layer decrease very quickly, and the processing time for feed forward process is short. After the last convolution layer, all feature maps are concatenated by a concatenation layer and then flow through classifier which is consisted by three fully-connected layers and three drop out layers with probability 50%. The output of this classifier is also the probability of splitting the current CU.

For the first convolutional layer in three sub-networks of filters $C_{1-1}, C_{1-2}, C_{1-3}$ have kernel size of $\frac{w_l}{8}, \frac{w_l}{4}, \frac{w_l}{2}$, where w_l is the size of input luma-CU. Table 4.2 is the network configuration in [2].

Layer	Filter Size	Number Of Filters	Number of Trainable Parameters
C_{1-1}	$\frac{w_l}{8} * \frac{w_l}{8}$	64	w_l^2
C_{1-2}	$\frac{w_l}{4} * \frac{w_l}{4}$	128	$8w_l^2$
C_{1-3}	$\frac{w_l}{2} * \frac{w_l}{2}$	256	$64w_l^2$
C_{2-1}	2x2	128	32896
C_{2-2}	2x2	256	131328
C_{3-1}	2x2	256	131328
F_1	1x1	256	787200
F_2	1x1	16	4112
F_3	1x1	1	17
Total Parameters for 64 sized CU			1385889
Total Parameters for 32 sized Cu			1161633
Total Parameters for 16 sized CU			1105569

Table 4.2: Database Distribution

4.3 Network III

This is the network that we proposed to solve the computational complexity in HEVC. The network structure is shown in figure 4.3. The proposed method has also three neural networks corresponding to different input size. In HEVC algorithm, a CU is compared to its sub-CU during the recursive search which can determine the partition of the CU, so our idea is to compare the features, which are captured by large filters, to the features, which are captured by small filters.

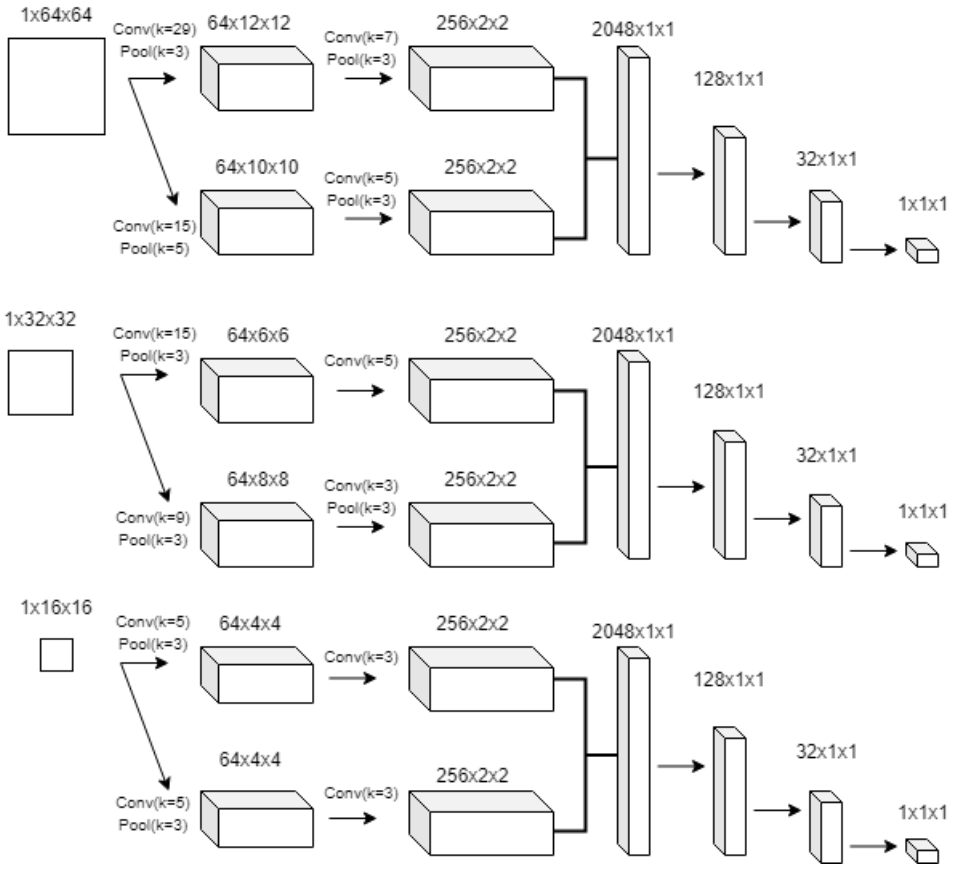


Figure 4.3: Network Structure of Proposed Method

Intuitively, this comparison can be trade as features from a lower depth and features from a higher depth. Since if a CU is going to be divided into smaller CU, then the features, which is contained in the current depth, must be different than at least one set of features, which is

contained in the higher depth. In the original algorithm, this comparison will compare all the depth recursively, but this is not necessary in our proposed method because we are not going to find the optimal splitting quad-tree, we just need to know whether the current CU is going to be split or not.

	Input Size 64		
Layer	Filter Size	Number Of Filters	Number of Trainable Parameters
Conv1	29x29	64	53888
Conv2	7x7	256	803072
Conv3	15x15	64	14464
Conv4	5x5	256	409856
Fully 1	1x1	128	262272
Fully 2	1x1	32	4128
Fully 3	1x1	1	33
Total Parameters			1547713
	Input Size 32		
Conv1	15x15	64	14464
Conv2	5x5	256	409856
Conv3	9x9	64	5248
Conv4	3x3	256	147712
Fully 1	1x1	128	262272
Fully 2	1x1	32	4128
Fully 3	1x1	1	33
Total Parameters			843713
	Input Size 16		
Conv1	5x5	64	1664
Conv2	3x3	256	147712
Conv3	5x5	64	1664
Conv4	3x3	256	147712
Fully 1	1x1	128	262272
Fully 2	1x1	32	4128
Fully 3	1x1	1	33
Total Parameters			565185

Table 4.3: Network Configuration for Proposed Method

For the proposed method, there are two sub-networks for each network. The input will go through a convolution layer which is followed by a batch-norm layer and Relu activation layer,

then a max pooling layer will be applied to extract the key feature from the input. After the two sub-networks, the output will be concatenated into a linear vector and feed into three fully connected layers with three drop out layers. The output is also the probability of splitting the current CU. Table 4.3 is the network configuration for proposed method.

4.4 Overall Structure for Networks in HEVC

The CU-decision algorithm for three networks are designed by the top-down method as shown in figure 4.4 For each frame in test set which will be first divided into $64 * 64$ sized blocks then

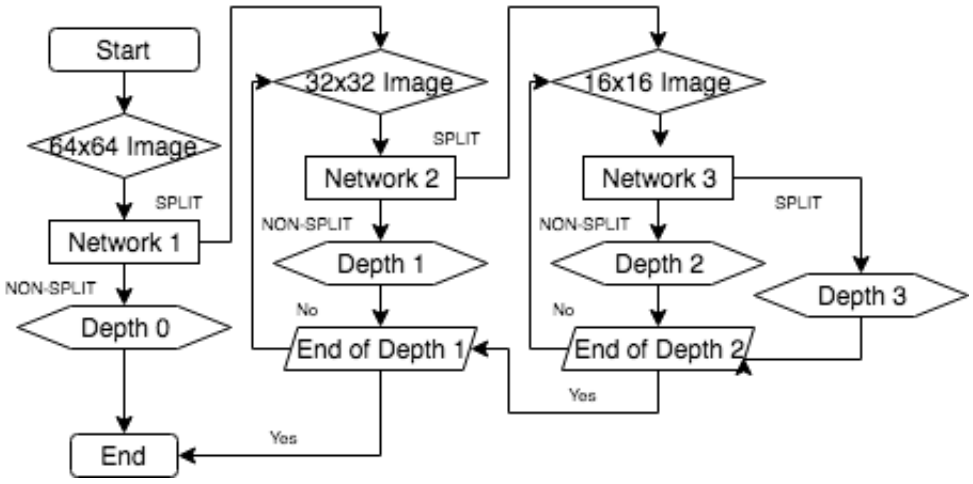


Figure 4.4: Fast Cu Depth Decision Algorithm

feed into the network, starting from depth 0. If depth 0 is marked as split then the current $64 * 64$ sized block will be divided into $32 * 32$ sized blocks and feed into the network which is designed for depth 1, we will apply this method recursively for depth 1 and depth 2, then return the final decision tree to the $64 * 64$ sized block, and this decision tree is our predicted splitting quad-tree for the current CU.

All three networks that we have discussed in this paper are using binary cross entropy as their loss function, since we have re-sampled our data, there is no need to add weight inside. The loss

function is showing below:

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top, \quad l_n = \left[y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n))\right], \tag{4.1}$$

$$\text{where, } \sigma(x_n) = \frac{\exp x_n}{\sum_{i=1}^{N} \exp x_i} \tag{4.2}$$

Additionally, the structure of those three neural networks provide a excellent ability of learning, since they all have huge numbers of trainable parameters, and this is also useful to avoid under fitting problems.

5. TRAINING AND TESTING

Training and testing a deep neural network requires hardware, different hardware will perform different training and testing efficiency. In this paper, all three networks are training and testing on Terra High Performance Research Computing provided by Texas A&M, we are using one CPU with two GPUs, the hardware information can be found on `https://hprc.tamu.edu/wiki/Terra:Intro`.

5.1 Training

The pseudo code of our training process is showing below. The pre-process of our data set has been discussed in section III. After pre-process, we will define our loss function and optimizer, the loss function is Binary Cross Entropy Loss, and we use Adam optimizer for our back propagation. The Adam optimizer converge faster SGD optimizer, which can save a lot of training time, and the result compare to SGD optimizer is still acceptable. The average training time for one epoch of three networks based on different QPs is showing in table 5.1.

[H] Trained Neural Network Get training data loader ready Loss function Optimizer i $<$ Total number of Iteration Load training data from data loader with batch size Feed forward for network Calculate Loss by Loss Function Back Propagation by Optimizer Do for loop for different depth

	QPs			
Networks	37	32	27	22
Network I	2097.83(sec)	2040.56(sec)	1958.17(sec)	1930.52(sec)
Network II	575.58(sec)	490.65(sec)	468.17(sec)	463.68(sec)
Network III	628.04(sec)	637.07(sec)	646.70(sec)	629.85(sec)

Table 5.1: Average Time Cost for Training

Based on the result, it is easily to point out that the network I takes a lot of time to finish one

epoch, Network II are the fastest network out of three in training phase. The training loss and accuracy for three networks in difference QPs will be discussed in section VI.

5.2 Testing

The testing is performed on our testing database using top-down structure, we predict each video frame by frame, and record their processing time and RD cost. The detail of our testing results are shown in section VI.

6. EXPERIMENTAL RESULT

In this section we are going to talk about our training and testing results for three networks and give evaluation of their performance based on results. All three neural networks are trained and tested based on the ground truth from HEVC test model version 16.3. In our results, time means the average time cost to process one frame, and WRD means how much worst compare to HEVC algorithm in the term of rate-distortion cost. The function to calculate WRD is showing below:

$$[h]\frac{PredictedResult - HEVCResult}{HEVCResult} \tag{6.1}$$

Table 6.1, 6.2, 6.3, and 6.4 are test results for three neural networks in different QPs and table 6.5 is the average processing time and WRD on our test database for different QPs.

	Network I		Network II		Network III	
Video's Name	Time	WRD	Time	WRD	Time	WRD
Bear	1.4829	0.0243	0.7679	0.0293	0.7592	0.0283
Bike Packing	1.9619	0.0512	1.5799	0.0549	1.3816	0.0552
Blackswan	1.6320	0.0261	0.9744	0.0307	0.9298	0.0323
Bmx Bumps	1.6047	0.0337	1.1147	0.0333	0.9616	0.0370
Bmx Tress	1.6922	0.0281	1.0092	0.0266	0.9793	0.0276
Boat	1.3406	0.0242	0.8391	0.0284	0.7514	0.0283
Boxing Fisheye	1.8837	0.0427	1.4862	0.0456	1.3049	0.0474
Breakdance	2.0204	0.0566	1.6257	0.0624	1.4015	0.0640
Break Flare	1.7125	0.0365	1.1712	0.0396	1.0517	0.0409

Table 6.1: Test Rsult for QP 22

Through the comparison of all results, we can conclude that our proposed method is faster than network I in all QPs for more than 30% with 1.3% of degradation on quality. Compare to network II, our proposed method is faster by 7% and 0.7% better on the quality. Over all different QPs, our proposed method is the fastest structure, for QP 27 and 32 it also has the best reconstructed quality.

	Network I		Network II		Network III	
Video's Name	Time	WRD	Time	WRD	Time	WRD
Bear	1.2246	0.0233	0.8767	0.0240	0.8284	0.0225
Bike Packing	1.9608	0.0535	1.5542	0.0541	1.3978	0.0536
Blackswan	1.3700	0.0246	1.1429	0.0231	0.9877	0.0225
Bmx Bumps	1.5734	0.0354	1.1200	0.0358	1.0075	0.0312
Bmx Tress	1.5475	0.0304	1.2096	0.0278	1.0270	0.0260
Boat	1.1427	0.0255	0.9341	0.0241	0.8166	0.0229
Boxing Fisheye	1.8464	0.0460	1.4540	0.0464	1.3286	0.0458
Breakdance	1.9363	0.0629	1.5973	0.0610	1.4346	0.0604
Break Flare	1.5164	0.0375	1.2542	0.0369	1.0835	0.0358

Table 6.2: Test Rsult for QP 27

	Network I		Network II		Network III	
Video's Name	Time	WRD	Time	WRD	Time	WRD
Bear	1.5297	0.0175	0.9208	0.0174	0.9228	0.0169
Bike Packing	1.8639	0.0513	1.5040	0.0528	1.3919	0.0486
Blackswan	1.6315	0.0202	1.1342	0.0183	1.1618	0.0192
Bmx Bumps	1.4572	0.0301	1.0552	0.0337	1.0474	0.0291
Bmx Tress	1.666	0.0269	1.1940	0.0259	1.1460	0.0251
Boat	1.2123	0.0213	0.8400	0.0210	0.8786	0.0196
Boxing Fisheye	1.8050	0.0427	1.4270	0.0416	1.3192	0.0418
Breakdance	1.9869	0.0552	1.6196	0.0529	1.4598	0.0523
Break Flare	1.5720	0.0358	1.1753	0.0350	1.1434	0.0334

Table 6.3: Test Rsult for QP 32

In more detail, for video test sets like Bike packing, which has a lot of details in each frame, will be divided into even smaller blocks compare to other videos in the original HEVC algorithm. The test results show that the output quality of such videos from neural networks is typically worse than other videos, which means for lower depth, like 0 and 1, all three networks can predict the correct labels based on the input, but for higher depth, like 2 and 3, they tend to have a higher error rate, which is easy to understand, since for depth 2 and 3 they are having the same area in HEVC algorithm, which made them hard to be distinguished from each other and specially the detail contained in depth 2 and 3 are not always having big differences, so in our test results, we

	Network I		Network II		Network III	
Video's Name	Time	WRD	Time	WRD	Time	WRD
Bear	1.2334	0.0123	0.7917	0.0143	0.9060	0.0126
Bike Packing	1.8038	0.0385	1.4748	0.0401	1.3026	0.0419
Blackswan	1.2333	0.0135	0.09776	0.0123	0.9621	0.0173
Bmx Bumps	1.2346	0.0250	0.9588	0.0278	0.9161	0.0256
Bmx Tress	1.3709	0.0220	1.0459	0.0218	1.0423	0.0248
Boat	1.0428	0.0165	0.7232	0.0176	0.7689	0.0172
Boxing Fisheye	1.7231	0.0362	1.3988	0.0352	1.2687	0.0352
Breakdance	1.8790	0.0393	1.5697	0.0392	1.3975	0.0409
Break Flare	1.3262	0.0287	1.0520	0.0292	1.0244	0.0297

Table 6.4: Test Rsult for QP 37

	Network I		Network II		Network III	
QP	Time	WRD	Time	WRD	Time	WRD
22	1.7085	0.0363	1.1825	0.0393	1.0630	0.0404
27	1.5737	0.0382	1.2431	0.0376	1.1045	0.0362
32	1.6321	0.0341	1.2086	0.0339	1.1621	0.0324
37	1.4259	0.0265	1.1108	0.0272	1.0663	0.0279

Table 6.5: Over All Test Result

can find there are quite large number of mislabeled blocks for depth 2 and 3. However, the penalty of mislabel depth 2 to depth 3 or vise versa is relatively small compare to mislabel depth 3 to depth 0, therefore the performance of all three neural networks are still acceptable when dealing with such videos which has a lot of details.

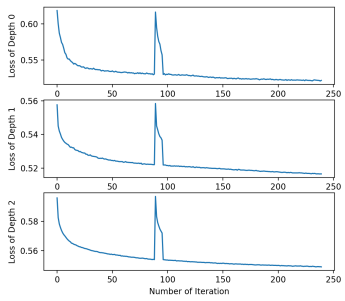
Since the total parameters for each neural network are almost the same, therefore the quantity of parameters will not bring large influence on the difference in processing time for different networks. But from the neural network's structure level, we can then explain the difference in processing time for all three networks. First we use the top-down method to predict the splitting quad-tree for an input frame, therefore the network for depth 0 will be applied to all blocks and network for depth 1 will be applied to almost all blocks, then how fast can a network determined whether to split in depth 0 and 1 became critical in processing time. From our test results, we

noticed that the Network I is the slowest one among all three networks, this is determined by its network structure, from figure 4.1 in section IV, we denoted there are seven layers in depth 0 and 5 layers in depth 1 which is the deepest network over all three networks and this is also the main reason for the longest processing time, but this also gives the greatest power of learning the ground truth to the network I compare to others.

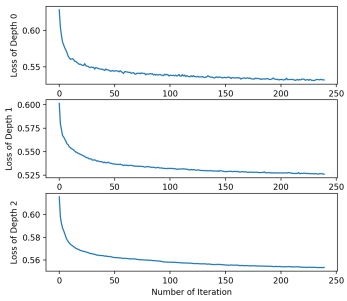
Different than network I, there are three sub-nets in network II for each depth, but it is shallower than the network I and all its convolution kernels are non-overlapping, therefore even though it needs 9 layers to do prediction and the deepest path in network II has 6 layers for each depthit is still faster than network I. The structure of network II provides the ability of learning the difference between the key feature of current depth and key features from higher depths which also enhance the accuracy of the predicted output.

Similar to network II, the proposed method network III is also using sub-net structure for each depth, but instead of focusing more on the features from higher depths, we decided to put evenly focus on both current depth and next level depth. This decision is made based on the nature of the original HEVC algorithm, as we have discussed in section IV the decision of splitting the current block will be determined by compare current depth and higher depth recursively, therefore we only have two sub-nets for each depth, and each sub-net extracts the key features for their designed depth. Eventually this structure make the proposed method the fastest one among all three networks.

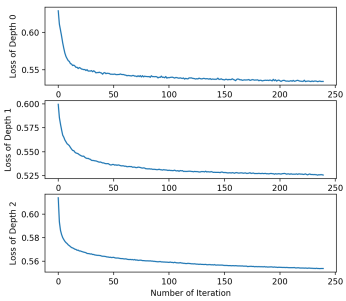
Figure 6.1 and 6.2 are the training loss and accuracy among three networks for different QPs. All three networks converged after 240 iterations. By looking at figure 6.1 we noticed that for QP 22 and 27 network I has a huge jump during training, which is quite strange, because it may be trapped in a local optimal, and after several iterations it jumped out of this local optimal and continues to the global optimal. Figure 6.3 represents the predicted output for a given frame from three networks on different QPs, from the observation we can conclude that the network I tend to divide the frame into smaller blocks over all QPs, network II tend to divide the frame into larger blocks, and network III is in the middle of the network I and the network II.



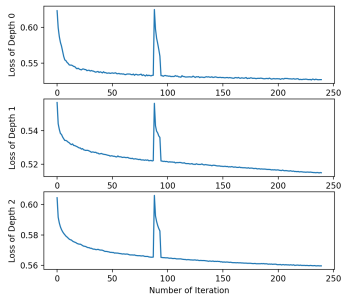
(a) Loss for Network I in QP 22



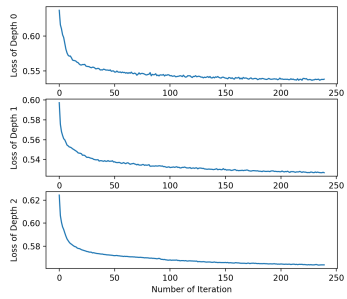
(b) Loss for Network II in QP 22



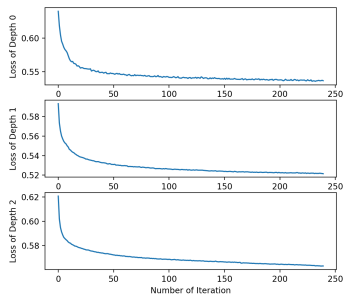
(c) Loss for Network III in QP 22



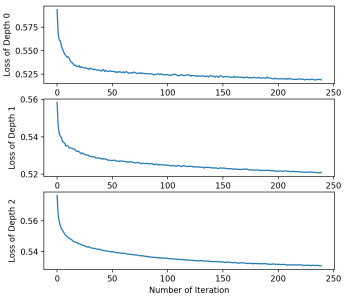
(d) Loss for Network I in QP 27



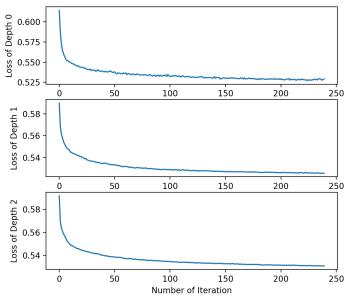
(e) Loss for Network II in QP 27



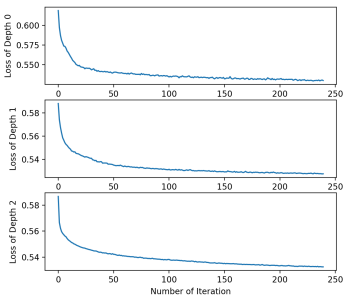
(f) Loss for Network III in QP 27



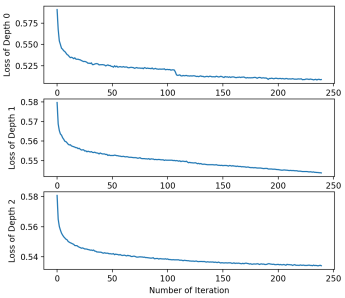
(g) Loss for Network I in QP 32



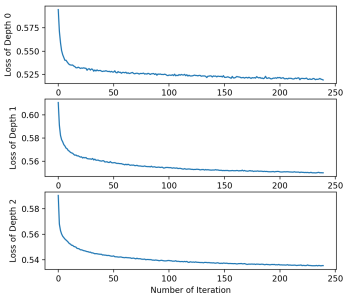
(h) Loss for Network II in QP 32



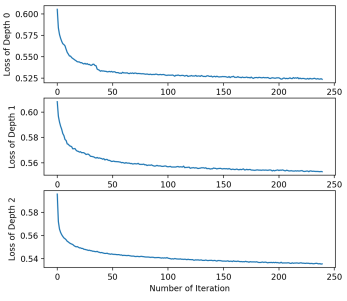
(i) Loss for Network III in QP 32



(j) Loss for Network I in QP 37

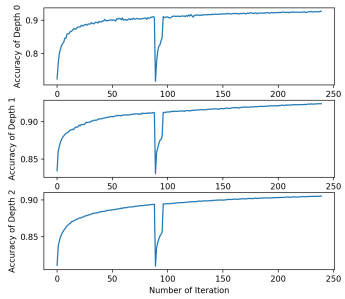


(k) Loss for Network II in QP 37

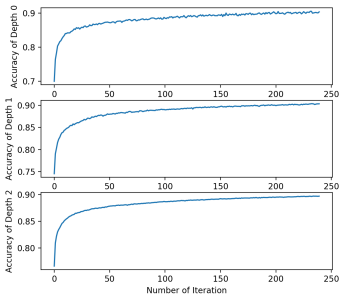


(l) Loss for Network III in QP 37

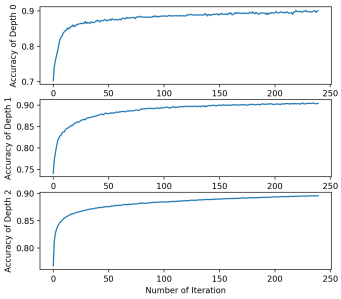
Figure 6.1: Loss for 3 Networks in different QPs



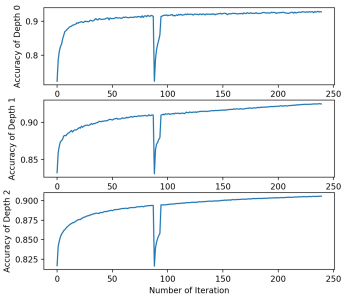
(a) Accuracy for Network I in QP 22



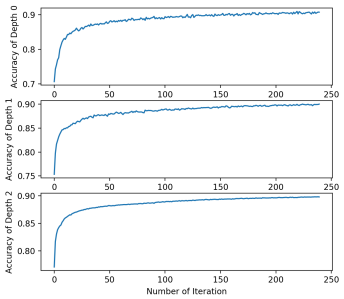
(b) Accuracy for Network II in QP 22



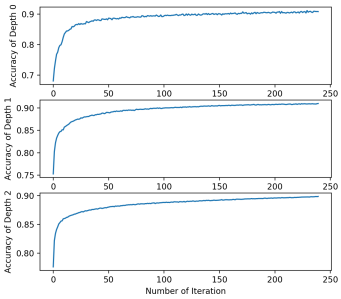
(c) Accuracy for Network III in QP 22



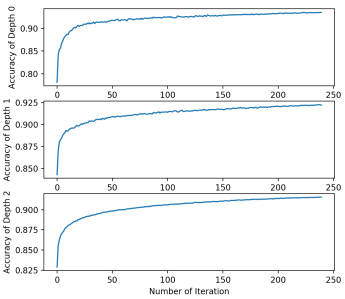
(d) Accuracy for Network I in QP 27



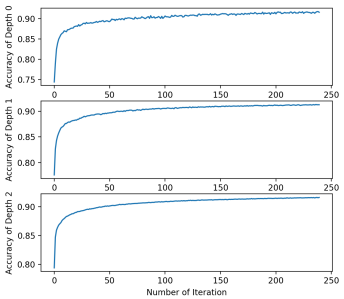
(e) Accuracy for Network II in QP 27



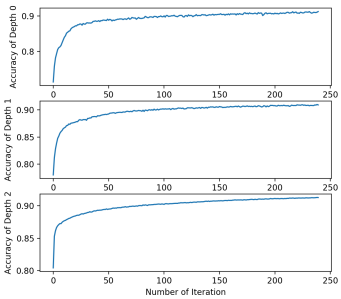
(f) Accuracy for Network III in QP 27



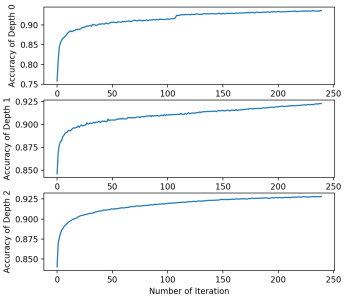
(g) Accuracy for Network I in QP 32



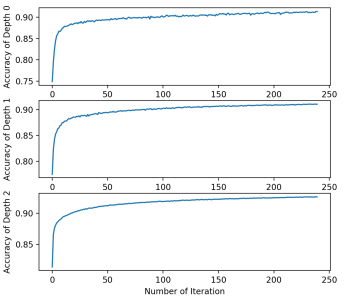
(h) Accuracy for Network II in QP 32



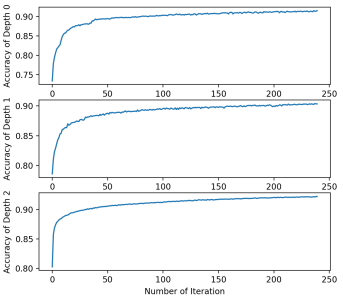
(i) Accuracy for Network III in QP 32



(j) Accuracy for Network I in QP 37

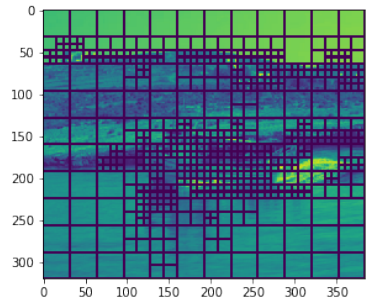


(k) Accuracy for Network II in QP 37

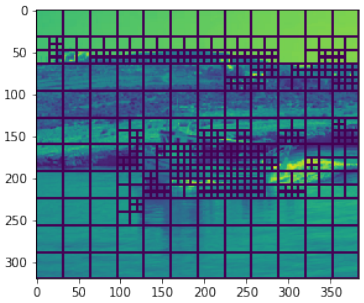


(l) Accuracy for Network III in QP 37

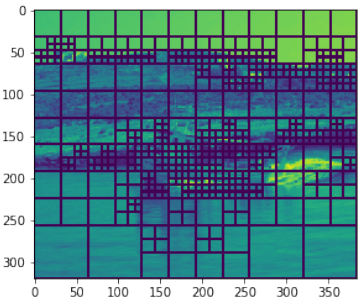
Figure 6.2: Accuracy for 3 Networks in different QPs



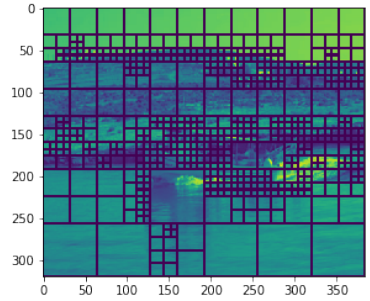
(a) Prediction for Network I in QP 22



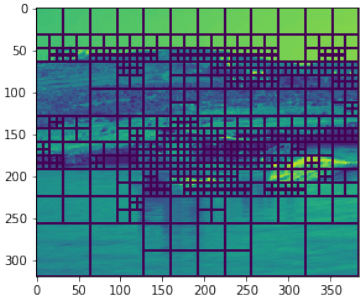
(b) Prediction for Network II in QP 22



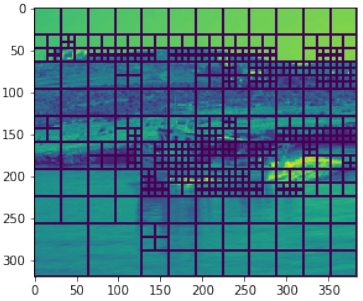
(c) Prediction for Network III in QP 22



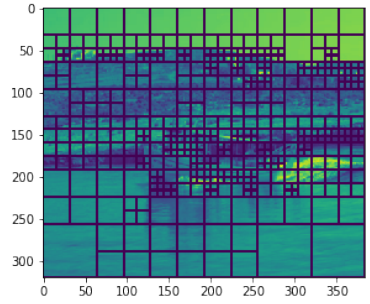
(d) Prediction for Network I in QP 27



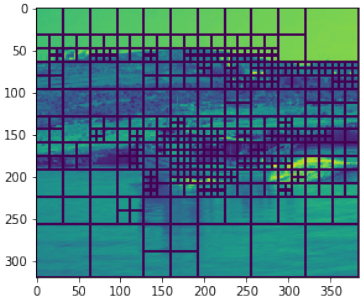
(e) Prediction for Network II in QP 27



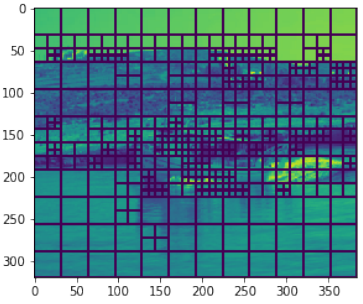
(f) Prediction for Network III in QP 27



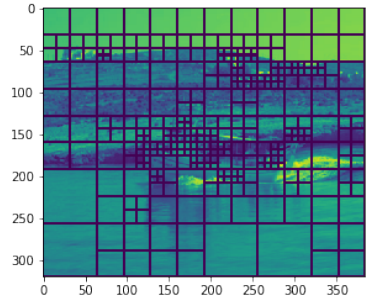
(g) Prediction for Network I in QP 32



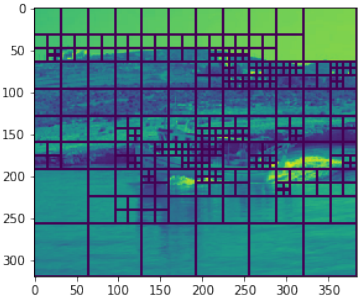
(h) Prediction for Network II in QP 32



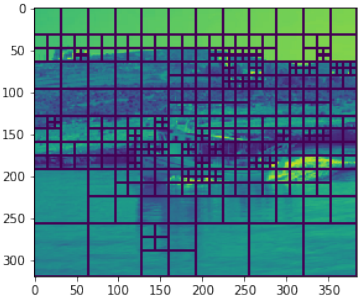
(i) Prediction for Network III in QP 32



(j) Prediction for Network I in QP 37



(k) Prediction for Network II in QP 37



(l) Prediction for Network III in QP 37

Figure 6.3: Prediction for 3 Networks in different QPs

7. CONCLUSION

In this paper, we have improved the CU depth decision method in HEVC by applying neural network based deep learning algorithm. We have built the database and designed the structure of our neural network. Our algorithm is the fastest algorithm compares to another two algorithms with a competitive quality of reconstructed frames. Therefore the encoding time can be future reduced by using our method.

REFERENCES

[1] K. Kim and W. W. Ro, "Fast cu depth decision for hevc using neural networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 8, 2015.

[2] T. Li, M. Xu, and X. Deng, "A deep convolutional neural network approach for complexity reduction on intra-mode hevc," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1255–1260, July 2017.

[3] W.-J. H. Gary j.Sullivan, Jens-Rainer Ohm 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, 2012.

[4] T. C.E.Rhee, K.Lee and H.J.Lee, "A survey of fast mode decision algorithms for inter-prediction and their applications to high efficiency video coding," *IEEE Transactions on Consumer Electronics.*, vol. 58, no. 4, 2012.

[5] S. L.Zhao, L.Zhang and D.Zhao, "Fast mode decision algorithm for intra prediction in hevc,," *Proc. Visual Communications and Image Processing.(VCIP'2011)*, 2011.

[6] H. Kim and R. Park, "Fast cu partitioning algorithm for hevc using an online-learning-based bayesian decision rule," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, pp. 130–138, Jan 2016.

[7] Y. Zhang, S. Kwong, X. Wang, H. Yuan, Z. Pan, and L. Xu, "Machine learning-based coding unit depth decisions for flexible complexity allocation in high efficiency video coding," *IEEE Transactions on Image Processing*, vol. 24, pp. 2225–2238, July 2015.

[8] A. Heindel, T. Haubner, and A. Kaup, "Fast cu split decisions for hevc inter coding using support vector machines," in *2016 Picture Coding Symposium (PCS)*, pp. 1–5, Dec 2016.

[9] D. Ruiz-Coll, V. Adzic, G. Fernández-Escribano, H. Kalva, J. L. Martínez, and P. Cuenca, "Fast partitioning algorithm for hevc intra frame coding using machine learning," in *2014*

IEEE International Conference on Image Processing (ICIP), pp. 4112–4116, Oct 2014.