FAST DEPTH CODING IN 3D-HEVC USING DEEP LEARNING

A DISSERTATION
SUBMITTED TO THE
DEPARTMENT OF ELECTRONIC AND INFORMATION ENGINEERING
OF THE HONG KONG POLYTECHNIC UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Zhen-xiang WANG
December 2017

CERTIFICATE OF ORIGINALITY

I hereby declare that this dissertation is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written nor material which has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____(Signed)

_____(Date)

Abstract

The 3D Extension of the High Efficiency Video Coding standard (3D-HEVC), which has been finalized by the Joint Collaborative Team on Video Coding (JCT-VC) in February 2015, is the new industry standard for 3D applications. The 3D-HEVC provides plenty of advanced coding tools specifically for addressing the coding of auto-stereoscopic videos which have the format of multiple texture views along with the depth maps which are responsible for synthesising intermediate views with sufficient quality for auto-stereoscopic display. The provided tools take advantage of the statistical redundancies amongst texture views and depth maps in the video sequences, as well as the unique characteristics of depth maps to significantly shrink the bit-rate while preserving the objective visual quality of the 3D videos. However, those tools with high capability in terms of compression come with the high complexity of computation which has made the encoding time of the 3D video sequences much longer than ever by traversing a lot more candidates, calculating time-consuming RD Cost for each of them, especially in the wedgelet searching process for depth maps. While this full-search style method can promise to find the best candidate in depth intra mode decision, the time cost is expensive.

In this dissertation we address the time cost by presenting a new intra mode decision method for depth maps, leveraging the deep convolutional neural networks to predict the wedgelet angles for the depth blocks. The predictions from the learned models are capable of reducing the number of wedgelet candidates by half as well as the angular modes in depth map coding. The size of the neural network has been carefully designed to balance the trade-off between the time cost of model prediction and the model prediction accuracy. Confusion matrix is used to monitor the training process. Top-K criteria is employed for the prediction. We have integrated the learned models into the reference software of 3D-HEVC for the experiments. The compiled executable binaries are able to harness the power of the simultaneous computation of CPU, as well as the parrallel computation of GPU to accelerate the predictions. The simulation results show that the proposed algorithm provides 64.6% time reduction in average while the BD performance has a tiny decrease comparing with the state-of-the-art 3D-HEVC standard.

Acknowledgments

First and foremost, I would like to give sincere thanks to my supervisor, Dr.Yui-Lam Chan, for his extremely generous support, most insightful advices and innumerable yet constructive feedback. I learned from him to first identify a problem, by reading a vast amount of articles to know what people have achieved and what bottlenecks they have encountered. I learned how to read papers, how to organize them to become the inner comprehension. He guided me to use the machine learning approach to solve the problem that has been found in the first stage. Without his guidance I will not have the idea to learn the deep learning technology and apply it to optimize the video coding. His encyclopedic knowledge and charming personalities made him my mentor in both research and life. I wish to thank Dr.Sik-Ho Tsang, for our in-depth discussions from which I can always find useful clues to proceed to next step. His great expertise in video coding significantly benefits me during my intensive period of learning. Also I would like to thank my friends Alex and Jacky, for our extensive discussions about artificial intelligence and their applications. Finally thank you my parents, for the great love and constant encouragement which give me confidence to face and handle all the challenges at every moment.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Video is the medium to record, copy, playback, broadcast and display the motion images in an electronic style [1]. Watching videos is becoming an important way for our entertainment as well as education. The high definition (HD) and ultra high definition (UHD) video are increasingly demanding nowadays. People prefer videos with higher definitions than those with lower resolutions because the former one provides much better viewing experience. However, challenges emerged for delivering videos with high definition. HD videos typically contain much more information in every picture frame than the standard definition videos. More data needs to be squeezed into the same capacity for transmission. For example, the uncompressed video with the dimension 720×480 at 30 frames per second requires 0.03 gigabytes per second, while the uncompressed video with the dimension 2880×2048 at 120 frames per second requires 2.12 gigabytes per second. Since bit rate is proportional to system bandwidth for transmission [2], and expanding the bandwidth in a large scale is too expensive, the significantly increased bit rate for transmitting the video data is becoming one of the major obstacles for HD video services.

To cope with the growing need for higher compression of moving pictures [3], Joint Collaborative Team on Video Coding (JCT-VC) [4] has developed the High Efficiency Video Coding standard which is the newest international video coding standard for substantially ameliorate the compression performance against the previous standards. Comparing with the H.264 Advanced Video Compression Standard [5], the H.265 High Efficiency Video Coding Standard provides fifty percent bit rate reduction while maintaining the objective video quality at the same level.

While Two-dimensional video is the most common video type, Three-dimensional (3D) video has been brought to market via lots of ways, including Blu-Ray disc, cable and satellite transmission, terrestrial broadcast, and streaming or downloading from the Internet [6]. 3D video provides the perception of depth information which augments the vividness of the video contents. Currently most 3D videos in the market are using stereo display technology. Two similar views, one for left eye, the other for right eye, are presented at the same time with the multiplexing techniques

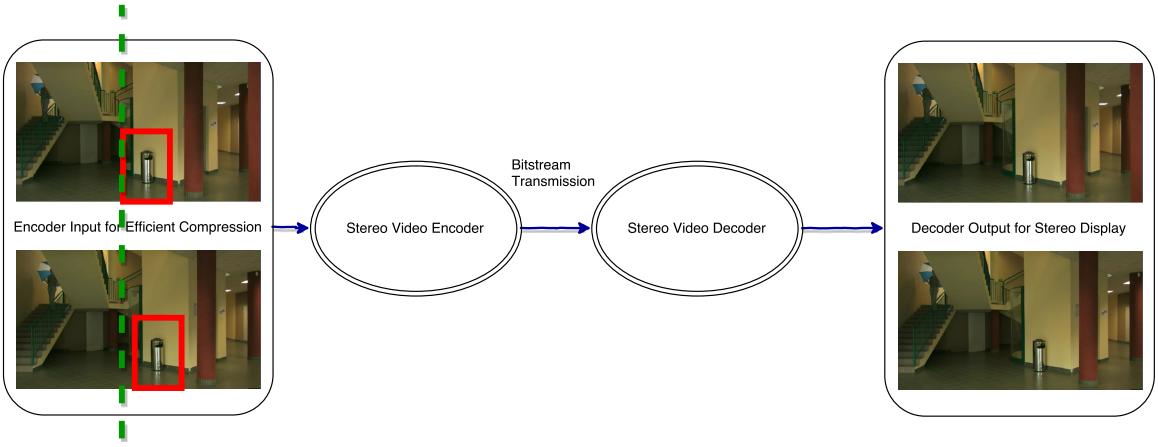

Figure 1.1: System Structure for transmitting videos targeting stereo display.

enabling the adjustments of video geometry information [7] to provide the 3D effect. Figure 1.1 illustrates the typical system structure for transmitting videos targeting stereo display. It can be observed that there exists a displacement between the two views. The green vertical left margins of the red rectangles in the two views at encoder side are different. Such a displacement is the visual disparity for 3D perception. Stereoscopic videos [8] have achieved great profitability for movie theatres in recent years. For example, IMAX 3D has became the most popular one that offering the immersing multimedia experiences around the world. Special 3D glasses are needed for watching the IMAX 3D movies. The current 3D film industry is very successful in terms of attracting customers, however, it is not the end of the story. Myopic people do not like to wear one more pair of glasses when watching 3D movies. Some people will experience discomfort after wearing the 3D glasses for a period of two hours. To get rid of the undesired 3D glasses, autostereoscopic multi-view technology [8] is coming to our rescue. The two major different characteristics between stereo display and autostereoscopic display are listed in Table 1.1 [9]. The impact of different view numbers for autostereoscopic display is shown in Table 1.2 [9]. Comparative ease can be brought to the 3D video audience since they do not need to wear 3D glasses for watching autostereoscopic videos. At each different view position, scenes with minor differences are available from multiple stereo pairs which are provided by autostereoscopic display [9]. As a result, when audience make a move for various

Table 1.1: Characteristics comparison of stereoscopic display and autostereoscopic display

Characteristic	Stereo Display	Autostereoscopic Display
Glass-Free	No	Yes
Multiple Stereo Pairs	No	Yes

Table 1.2: Impact of Available View Amount for Autostereoscopic Display

Characteristic	Small Number of Views	Large Number of Views
Seamless View Transition	No	Yes
High Quality of Scene Depth	No	Yes

view positions, scenes not viewable from the previous locations are revealed during the movement. The autostereoscopic multi-view display demands more than two views. With a sufficient amount of views present in autostereoscopic display, the disparities between every two adjacent views can be small enough to offer seamless transitions from scene to scene, such that when multiple views meet eyes sequentially, the scenes as a whole can be gorgeous. The visual quality of the autostereoscopic display is highly proportional to the number of available views. Due to limited available bandwidth, transmitting arbitrary number of views is not practical. Researchers have proposed a new format which only requires limited number of view and their associated depth maps for the capability of generating arbitrary amount of views theoretically. The typical system structure for using this new format to compress and supply 3D video resources is shown in Figure 1.2. An enormous amount of views in the medium positions which are able to guarantee the high quality of the 3D video can be synthesized from the decoded texture frames in combination with decoded depth maps.

To employ multi-view plus depth format for 3D video, efficient compressing methods are desired, which has led to the 3D Video Coding Extension of the High Efficiency Video Coding Standard (3D-HEVC) by the Joint Collaborative Team on 3D Video Coding Extension Development (JCT-3V) [10]. The 3D Extension of the HEVC standard gives extra coding efficiency for encoding a few texture views along with the corresponding depth maps by using new tools which exploit the

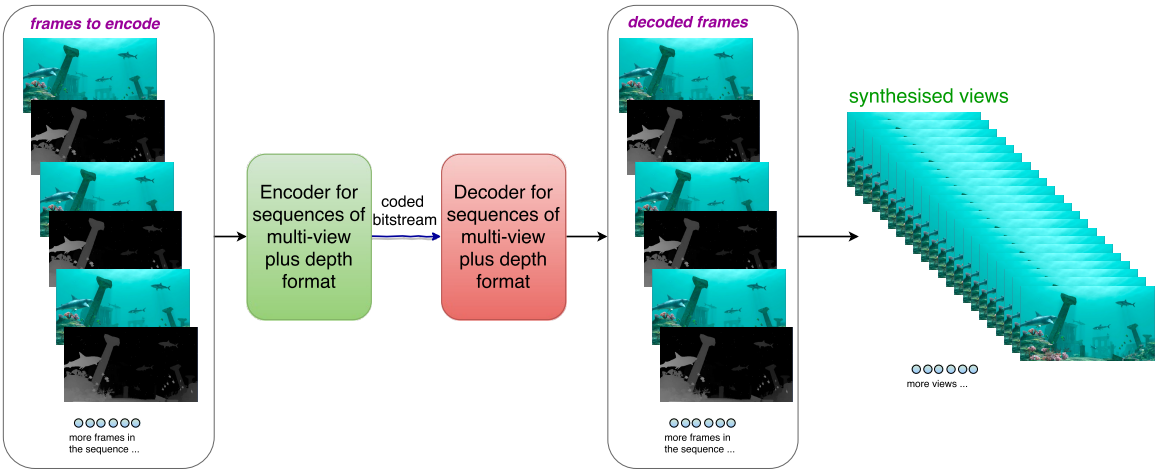

Figure 1.2: System Structure for transmitting videos of Multi-view Plus Depth format.

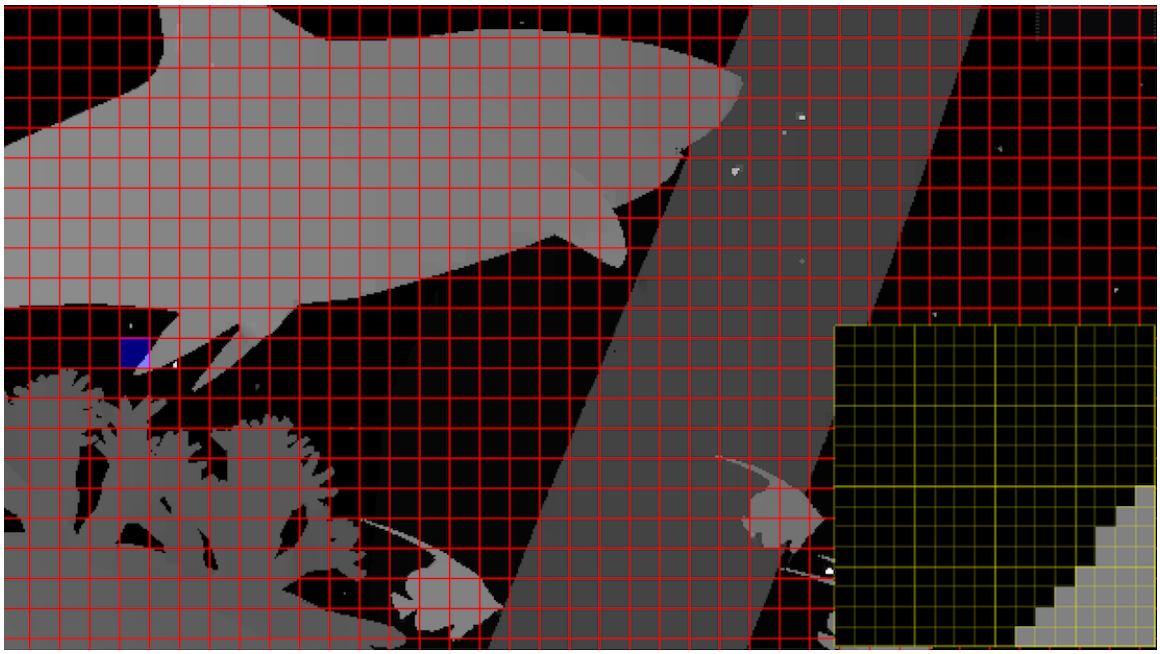

Figure 1.3: Example of wedgelet partition in a block of size 16 by 16 in depth map from Shark video sequence.

redundancies amongst texture and depth views, and pay attention to the unique characteristics of the depth maps, such as large homogeneous regions separated by sharp boundaries [11].

Depth information measures of the distance between the object in the far position and the object in the near position from a static viewpoint, which is expressed in the format of depth map. Instead of presenting depth maps directly to the viewer, views in the medium positions are generated by Depth-Image-Based Rendering (DIBR) technique. The qualities of the depth maps are vital to the DIBR process. Corona artifacts (a.k.a. ringing artifacts) [9] can be discovered in synthesized views if the edge sharpness in depth maps can not be well preserved. Therefore, retaining the edge sharpness in depth map is the key to avoid the artifacts in the synthesized views. In 3D-HEVC, new intra-picture prediction tools and residual coding methods have been applied to preserve the special properties of depth maps. Depth Modelling Mode (DMM) which is one of the new intra-picture prediction tools, is designed to provide much more granularity for encoding the depth maps than the normal angular intra prediction modes. DMM is more capable of approximating the depth maps to be encoded due to the fact that it provides a vast amount of non-rectangle partitions. Figure 1.3 presents an example of the wedgelet partition from the depth map in Shark video sequence. The small block highlighted by blue color amongst the blocks separated by the red grid is magnified at the right-bottom position in Figure 1.3. A straight line is used for the partition in wedgelet mode. Figure 1.4 shows a sample of the contour partition from the same depth map as Figure 1.3. The

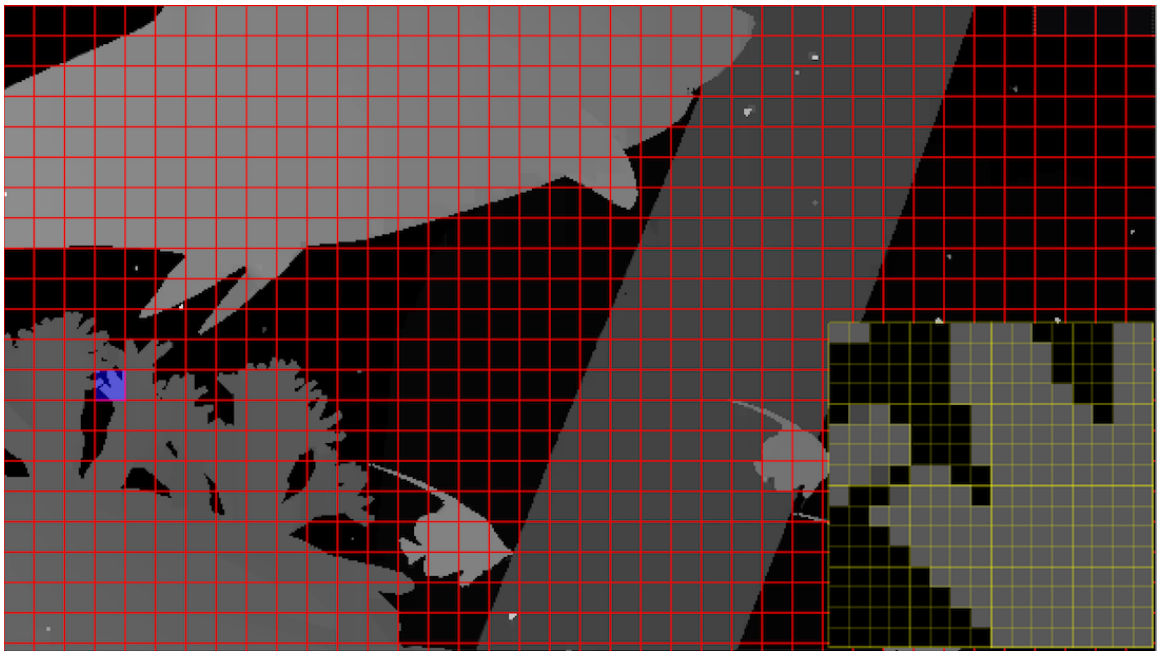

Figure 1.4: Example of contour partition in a block of size 16 by 16 in depth map from Shark video sequence.

partition pattern comprises contour lines instead of one single straight line. Wedgelet partition and contour partition for depth maps are enabled by DMM1 and DMM4 separately.

1.1 Motivation

The idea of this work originates from the discovery of the computational complexity of the wedgelet searching process in depth modelling modes. The immense complexity for searching the best wedgelet candidate lead to the strongly marked increase of encoding time. The time consumed for compressing a single depth map in 3D-HEVC encoder is roughly a sixfold increase relevant to the encoding time of a single texture frame wherein the all intra configuration in HTM-16.2 is used. Thus we designed a deep neural network architecture which is trained subsequently for predicting the most probable wedgelet candidates. The learned model achieves 92.2% to 97.3% top-16 accuracy for various block sizes. The inference engine is integrated into the reference software (HTM-16.2) of 3D-HEVC. The learned models reduce roughly half of the wedgelet searching candidates. It provides 64.6% time reduction in average while the BD performance has a negligible decrease comparing with the unmodified 3D-HEVC encoder.

Motivation for Wedgelet Candidates Reduction: The encoding time consumed in HTM-16.2 encoder for each view (including both texture and depth) by default can be observed from the

```
Layer   0   POC   143 TId: 0 ( I-SLICE, nQP 25 QP 25 )     932224  bits [Y  41.6950 dB    U  44.3646 dB    V  45.2432 dB] [ET    14 ] [L0 ] [L1 ]
Layer   1   POC   143 TId: 0 ( I-SLICE, nQP 34 QP 34 )      60160  bits [Y  45.0108 dB    U   0.0000 dB    V   0.0000 dB] [ET    89 ] [L0 ] [L1 ]
Layer   2   POC   143 TId: 0 ( I-SLICE, nQP 25 QP 25 )     932592  bits [Y  41.7195 dB    U  44.3713 dB    V  45.2450 dB] [ET    14 ] [L0 ] [L1 ]
Layer   3   POC   143 TId: 0 ( I-SLICE, nQP 34 QP 34 )      59616  bits [Y  44.0645 dB    U   0.0000 dB    V   0.0000 dB] [ET    69 ] [L0 ] [L1 ]
Layer   4   POC   143 TId: 0 ( I-SLICE, nQP 25 QP 25 )     932312  bits [Y  41.7154 dB    U  44.3855 dB    V  45.2770 dB] [ET    14 ] [L0 ] [L1 ]
Layer   5   POC   143 TId: 0 ( I-SLICE, nQP 34 QP 34 )      61920  bits [Y  44.1105 dB    U   0.0000 dB    V   0.0000 dB] [ET    69 ] [L0 ] [L1 ]
Layer   0   POC   144 TId: 0 ( I-SLICE, nQP 25 QP 25 )     933808  bits [Y  41.7343 dB    U  44.3498 dB    V  45.2481 dB] [ET    14 ] [L0 ] [L1 ]
Layer   1   POC   144 TId: 0 ( I-SLICE, nQP 34 QP 34 )      62288  bits [Y  44.6998 dB    U   0.0000 dB    V   0.0000 dB] [ET    90 ] [L0 ] [L1 ]
Layer   2   POC   144 TId: 0 ( I-SLICE, nQP 25 QP 25 )     932096  bits [Y  41.7303 dB    U  44.3655 dB    V  45.2215 dB] [ET    14 ] [L0 ] [L1 ]
Layer   3   POC   144 TId: 0 ( I-SLICE, nQP 34 QP 34 )      61496  bits [Y  43.9290 dB    U   0.0000 dB    V   0.0000 dB] [ET    69 ] [L0 ] [L1 ]
Layer   4   POC   144 TId: 0 ( I-SLICE, nQP 25 QP 25 )     931432  bits [Y  41.7151 dB    U  44.4146 dB    V  45.2913 dB] [ET    14 ] [L0 ] [L1 ]
Layer   5   POC   144 TId: 0 ( I-SLICE, nQP 34 QP 34 )      62440  bits [Y  43.7962 dB    U   0.0000 dB    V   0.0000 dB] [ET    69 ] [L0 ] [L1 ]
Layer   0   POC   145 TId: 0 ( I-SLICE, nQP 25 QP 25 )     927656  bits [Y  41.7556 dB    U  44.3820 dB    V  45.2680 dB] [ET    14 ] [L0 ] [L1 ]
Layer   1   POC   145 TId: 0 ( I-SLICE, nQP 34 QP 34 )      64216  bits [Y  44.6256 dB    U   0.0000 dB    V   0.0000 dB] [ET    90 ] [L0 ] [L1 ]
Layer   2   POC   145 TId: 0 ( I-SLICE, nQP 25 QP 25 )     928776  bits [Y  41.7512 dB    U  44.4091 dB    V  45.3110 dB] [ET    14 ] [L0 ] [L1 ]
Layer   3   POC   145 TId: 0 ( I-SLICE, nQP 34 QP 34 )      62608  bits [Y  44.0115 dB    U   0.0000 dB    V   0.0000 dB] [ET    68 ] [L0 ] [L1 ]
Layer   4   POC   145 TId: 0 ( I-SLICE, nQP 25 QP 25 )     929248  bits [Y  41.7290 dB    U  44.3781 dB    V  45.2933 dB] [ET    14 ] [L0 ] [L1 ]
Layer   5   POC   145 TId: 0 ( I-SLICE, nQP 34 QP 34 )      65928  bits [Y  44.0145 dB    U   0.0000 dB    V   0.0000 dB] [ET    70 ] [L0 ] [L1 ]
Layer   0   POC   146 TId: 0 ( I-SLICE, nQP 25 QP 25 )     928136  bits [Y  41.7692 dB    U  44.4027 dB    V  45.2841 dB] [ET    14 ] [L0 ] [L1 ]
Layer   1   POC   146 TId: 0 ( I-SLICE, nQP 34 QP 34 )      62480  bits [Y  44.4089 dB    U   0.0000 dB    V   0.0000 dB] [ET    89 ] [L0 ] [L1 ]
Layer   2   POC   146 TId: 0 ( I-SLICE, nQP 25 QP 25 )     923304  bits [Y  41.7896 dB    U  44.3323 dB    V  45.2464 dB] [ET    14 ] [L0 ] [L1 ]
Layer   3   POC   146 TId: 0 ( I-SLICE, nQP 34 QP 34 )      61344  bits [Y  43.5797 dB    U   0.0000 dB    V   0.0000 dB] [ET    69 ] [L0 ] [L1 ]
Layer   4   POC   146 TId: 0 ( I-SLICE, nQP 25 QP 25 )     927120  bits [Y  41.7560 dB    U  44.3389 dB    V  45.2676 dB] [ET    14 ] [L0 ] [L1 ]
Layer   5   POC   146 TId: 0 ( I-SLICE, nQP 34 QP 34 )      65248  bits [Y  43.6238 dB    U   0.0000 dB    V   0.0000 dB] [ET    69 ] [L0 ] [L1 ]
Layer   0   POC   147 TId: 0 ( I-SLICE, nQP 25 QP 25 )     918384  bits [Y  41.7766 dB    U  44.4255 dB    V  45.2442 dB] [ET    14 ] [L0 ] [L1 ]
Layer   1   POC   147 TId: 0 ( I-SLICE, nQP 34 QP 34 )      64480  bits [Y  44.3403 dB    U   0.0000 dB    V   0.0000 dB] [ET    90 ] [L0 ] [L1 ]
Layer   2   POC   147 TId: 0 ( I-SLICE, nQP 25 QP 25 )     926144  bits [Y  41.8009 dB    U  44.3506 dB    V  45.2566 dB] [ET    14 ] [L0 ] [L1 ]
Layer   3   POC   147 TId: 0 ( I-SLICE, nQP 34 QP 34 )      64000  bits [Y  43.5365 dB    U   0.0000 dB    V   0.0000 dB] [ET    69 ] [L0 ] [L1 ]
Layer   4   POC   147 TId: 0 ( I-SLICE, nQP 25 QP 25 )     923752  bits [Y  41.7702 dB    U  44.4198 dB    V  45.3113 dB] [ET    14 ] [L0 ] [L1 ]
Layer   5   POC   147 TId: 0 ( I-SLICE, nQP 34 QP 34 )      64664  bits [Y  43.6823 dB    U   0.0000 dB    V   0.0000 dB] [ET    70 ] [L0 ] [L1 ]
```

Figure 1.5: An example showing a piece of the command line outputs during the encoding process for Shark sequence.

command line outputs. Figure 1.5 shows a piece of command line outputs from the encoding process of Shark sequence. The numbers in red blocks stands for the encoding time of certain views, while the corresponding layer Id and Picture Order Count (POD) are in the green blocks. A repetitive pattern of the encoding time for each view can be observed every six numbers vertically. A simple calculation using six numbers within the top-most red block, $(90+69+69)/(90+69+69+14*3) \approx 0.84$, shows that approximately 84% of the total encoding time is busy with encoding the depth maps. Similarly, it is reported in [12] that the coding for depth map consumes near 86% of total 3D-HEVC encoding time. A trial of time profiling for 3D-HEVC encoder is performed using Instruments which is available on macOS. After encoding the Newspaper sequence for more than one hour, Figure 1.6 clearly shows 97.8% time is used to compress the CUs recursively. The first recursive xCompressCU

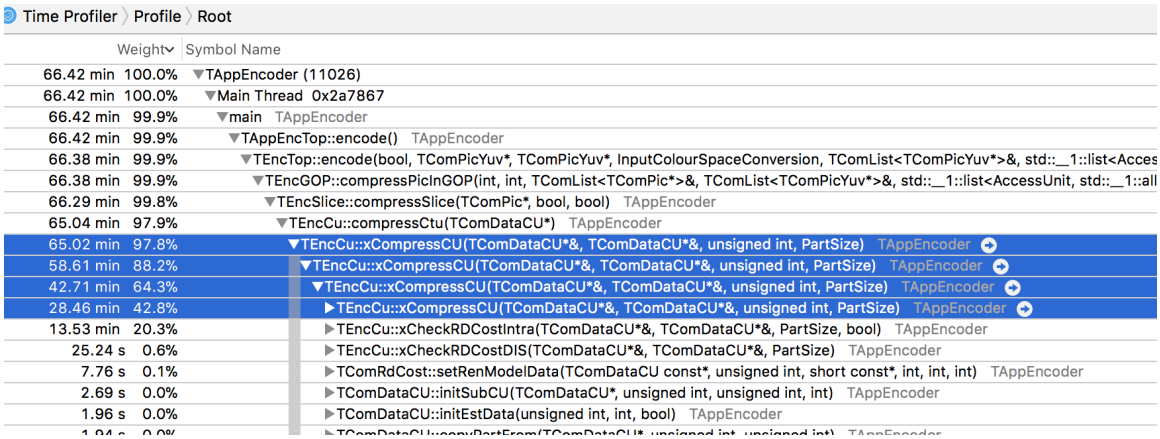

Figure 1.6: A screen capture of the time profiling information for Newspaper sequence.

Table 1.3: The summary of the time percentages occupied by DMM1 searching in the process for compressing CUs

size of CU	Recursive xCompressCU Function	Time percentage of DMM1 searching process
32×32	XC2	30.0%
16×16	XC3	25.6%
8×8	XC4	18.8%

Table 1.4: The summary of the time percentages occupied by VSO in DMM1 searching

size of CU	process	Time percentage of VSO in DMM1 searching
32 by 32	VSO in DMM1 searching from XC2	80.1%
16 by 16	VSO in DMM1 searching from XC3	83.7%
8 by 8	VSO in DMM1 searching from XC4	78.8%

function (denoted as XC1 thereafter) is for CUs of size 64×64, the second recursive xCompressCU (denoted as XC2 thereafter) is targeting CUs of size 32×32, the third one (denoted as XC3 thereafter) is dedicated to CUs of size 16×16, and the last one (denoted as XC4 thereafter) is bound to CUs of size 8×8. It is observed that the most time consuming part during the process of compressing the depth CUs is DMM1 searching. The DMM1 searching time percentages are summarized in Table 1.3 wherein the summary for XC1 is omitted since DMM1 is not applicable to CUs of size 64 by 64 in HTM-16.2. [t] The major reason leading to the time consuming property of DMM1 searching is the View Synthesis Optimization (VSO) Method for improving quality of synthesized views [13], wherein the Synthesized View Distortion Change (SVDC) is computed. The time percentages of the VSO processes in DMM1 searching are summarized in Table 1.4. In HTM-16.2, many wedgelet candidates are evaluated using the VSO which has a high computational complexity. Evaluating less wedgelet candidates will help to relieve the burden of heavy computation required by VSO, thereby certain time reduction can be achieved.

Motivation for Using Deep Learning: Deep learning such as Multi-Layer Perceptron (MLP) is a subfield of representation learning, which is in turn a major subset of machine learning [14]. Machine learning such as the support vector learning [15] is applied to many methods in the domain of Artificial Intelligence (AI). Deep learning based on back propagation training has been found hard to proceed in the late 1980s [16], however, starting from 2012, it kicks off the glorious comeback. The deep Convolutional Neural Network (CNN) has won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) from 2012 to 2015 with the CNN architecture of the winner going deeper and deeper year by year. The great achievements attract attentions from people all over the world and make deep learning the hottest topic in our daily lives. Inspired by the fact that supervised deep learning can learn multiple layers of abstract representations in the visual recognition tasks, it should

be applicable to recognize the angular modes of the intra-picture prediction in the 3D-HEVC. The final DMM1 candidate selected in depth map coding is essentially determined by the angle pattern of the depth blocks. If we can make use of deep learning to predict the most probable angles of the target pixel block, a vast amount of angular modes and DMM1 wedgelet candidates can be naturally skipped by which the time saving can be achieved without decreasing the coding performance.

Motivated by the discussions above, we adopt deep learning approach with deep convolutional neural network to accelerate the depth map coding in 3D-HEVC.

1.2 Contribution and Dissertation Outline

We accelerate the depth map coding by leveraging the power of deep learning. The contributions of the dissertation are:

- A deep convolutional neural network with 32 layers comprising ResNet units [17] has been designed and trained for recognizing the angular directions of the blocks from intra-picture prediction in 3D-HEVC encoder. The learned models have high top-k precisions which work well on the tasks of recognizing intra angular patterns in 3D-HEVC.
- A way of integrating the learned model into the HTM-16.2 encoder has been suggested. By making use of Bazel [18] to compile the encoder binary, the data level parallelism (instead of concurrency) functionality in CPU as well as the parallel architecture in GPU are fully utilized for efficient computations of matrix operations.
- An algorithm, illustrated in Figure 1.7 on page 9, for fast depth map coding based on the predictions from learned deep models has been proposed and implemented. The simulation results show that the proposed algorithm is capable of reducing 64.6% time in wedgelet searching during 3D-HEVC encoding process while the BD performance only has a trivial decrease.

The first two contributions lay the foundation for the third one, which is the main objective of this work: to accelerate the depth map encoding process in 3D-HEVC.

Chapter 2 supplies the background of video coding history, video coding standards, and deep learning using artificial neural network. Prior arts in video coding and deep learning are surveyed in this chapter.

Chapter 3 describes the methodology which has been implemented to collect the data to be used for deep learning. The pro-processing steps for the data are provided in details along with the reasons behind the scene. We also visualize lots of collected data to help with the understanding of their properties.

Chapter 4 presents the designed deep convolutional neural network which has been adopted in the deep learning process. Discussions on choosing proper hyper-parameters for the devised neural network are given. The stopping criteria are presented with the training results.

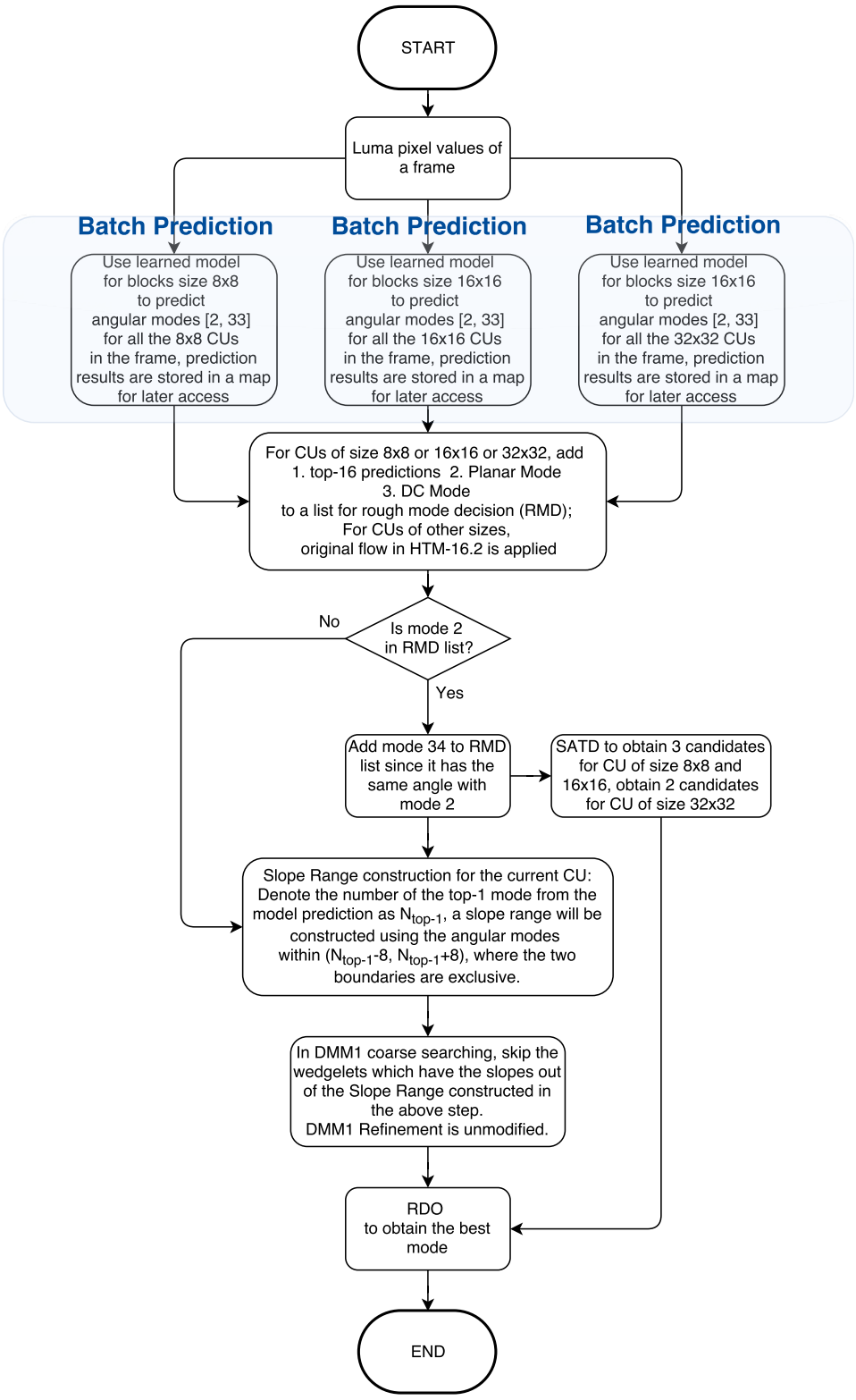

Figure 1.7: Flowchart for proposed fast depth coding algorithm.

Chapter 5 provides evaluation results for the learned models. Block resizing with different approaches are compared using the accuracy of prediction.

Chapter 6 shows the methods been used for integrating the learned model into the 3D-HEVC encoder. Advantages and drawbacks of the integration are discussed. Simulation results comparing with the original HTM-16.2 are given in this chapter.

Chapter 7 concludes the thesis and discusses the future work for fast depth coding using deep learning.

Chapter 2

Background

To start with, we bring up what is video coding, why it is needed, and its challenges. Next we discuss what is deep learning, the history of deep learning and how it works for vision tasks. Furthermore, we introduce how we plan to apply deep learning to optimize video coding tasks and why it should work. In the end, a survey of related works on the topic of video coding is given.

2.1 Video Coding

Video playback is the most straightforward way for human to perceive dynamic scenes that exist across a time series. More than half of the neurons in human brain are born to process the visual information which is supplied by human eyes. It becomes effortless for human to understand things presented by the video playback instead of a long paragraph of words. Videos are made up of consecutive sets of image frames, which in turn are made up of pixel matrices. Visual information of a cosmic scale is first stored by various methods then delivered during a period of video playback.

In 1950s, video tapes were employed to store the videos. Video tape is able to serve for about eight to twelve years before the video quality starts to degrade. In 1970s, laser disc appeared in the US market as an alternative of video tapes. Start from laser disc, the video storage started its new era in digital world. In 1990s, DVDs were released after laser disc. Data is stored in spiralling tracks on the disc. A laser beam can be utilized to read the data. In addition, hard drives, flash drives and SD cards were also starting to become popular in the late 90s. Nowadays, the cloud storage is very common in daily lives. It is capable of storing data on the servers which are accessible from any devices via internet connections.

Although so many formats are available for video storage, they share a common feature: the more storage you use, the more cost it will be. Let's take the cloud storage as an example. Google cloud is one of the most popular cloud services in our daily lives. It provides cloud storage with a price of $0.026 per GB/month [19] (this price is observed on 21 Nov 2017, it may change in the

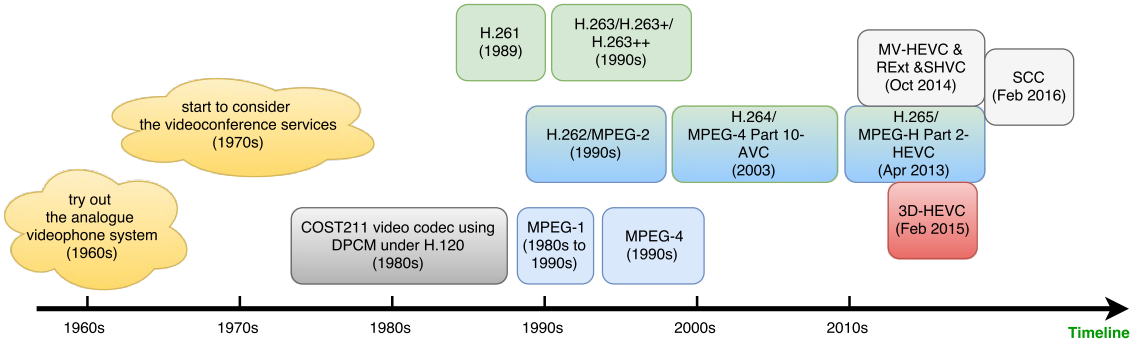

Figure 2.1: The brief history of the video coding standards

future). If a 4K video with a resolution of 4096*2160, at 120 frames per second, 8 bits for each of the RGB component, needs to be stored without any compression in Google cloud, we need to pay a monthly fee: $(4096 * 2160 * 120 * 60 * 90 * 3 * 0.026)/(1024 * 1024 * 1024) \approx 416.47$ \$. Without doubt, this figure is relatively not acceptable for just storing the video. High compression is needed to store the videos in a practical way.

From the other perspective, let us take the bandwidth into consideration. To deliver the uncompressed 4K video which has been mentioned in the previous paragraph, we need a bandwidth of: $(4096 * 2160 * 120 * 3)/(1024 * 1024 * 1024) \approx 2.97$ Gigabytes per second. The maximum bandwidth of Wireless 802.11ac, which is one of the common internet access technologies, is 1.3 Gigabytes per second [20]. Apparently, the wireless connection is not able to deliver such kind of 4K videos. High compression is desired to deliver the video through the internet.

Despite the fact that raw videos usually contain a large amount of data, a lot of redundancies exist. For every video sequence, two types of redundancies are ubiquitous: Spacial Redundancy and Temporal Redundancy. Video coding technologies are taking advantages of those redundancies to achieve the efficient compression for video data. Many of the useful video coding technologies have been adopted by the international video coding standards, such as MPEG-4, H.264, H.265, etc.

Figure 2.1 shows the brief history of the video coding standards. In 1980s, the COST211 video codec, built on top of Differential Pulse Code Modulation (DPCM), was standardized under H.120 standard by CCITT (now known as ITU-T). In late 1989, the H.261 was completed and its success marked a milestone for video coding at low bit rate with fairly good quality [21]. The Motion Picture Experts Group (MPEG) kicked off the exploration of video storage, such as CD-ROMs. Their objective was to achieve a competitive performance with cassette recorders in terms of compression of videos which have rich motions. The framework of H.261 had been used to start the codec design of MPEG-1. MPEG-2 was one generation after the MPEG-1. It featured higher capabilities when handling videos with high bit rates and high resolutions. In MPEG-2, the encoder is allowed to make its own decision on the the number of bi-directionally predicted pictures according to a suitable

coding delay. ITU-T found this technique applicable to telecommunication applications, as a result MPEG-2 has been adopted as H.262 for telecommunications. Right after the MPEG-2 standard, MPEG-3 was designed mainly for coding of high definition videos. However, MPEG-3 was discarded due to the versatility of MPEG-2, which can be used to encode videos of any resolutions. In the late 1998, MPEG-4 was introduced as a way of defining compression of both audio and visual digital data. Later on MPEG-4 was divided into several parts during its continuously evolving. Among its sub-parts, MPEG-4 part 10 (a.k.a. Advanced Video Coding) is mainly for the video compression. With the rising popularity of the high definition videos, the new standard termed High Efficiency Video Coding (HEVC) for compressing videos in a more efficient way comparing with previous standards, such as H.264/AVC, has emerged under the efforts from the Joint Collaborative Team on Video Coding (JCT-VC). In the meanwhile, five extensions of the HEVC standard, comprising Format Range Extension (RExt), Scalability Extension (SHVC), Multi-view Extension (MV-HEVC), 3D Extension (3D-HEVC), Screen Content Coding Extension (SCC), have been finalized from 2014 to 2016 to fulfill extra requirements in various video coding scenarios.

In this work, we focus on the depth map coding in 3D-HEVC. The 35 angular modes and depth modeling modes have been embraced in the depth map coding tools in 3D-HEVC. The DMM1 mode introduces an huge increase for the encoding time of 3D videos. Acceleration of the depth map coding is needed.

2.2 Deep Learning

Deep learning is an approach of representation learning (a.k.a. feature learning), which is essentially a method to learn from data. Numerous layers of computational units together with appropriate activating mechanism comprise the basic architecture for deep learning. Multitudinous data sets are needed for those computational architectures to learn data abstractions for tasks such as image classification, speech recognition, object detection, etc. Each layer learns a level of abstraction from the data sets using back-propagation algorithm [22]. Making use of those learned abstractions, the computational architectures are able to solve complex problems which are typically non-linear and normally hard to solve by using specific rules that are designed in advance.

Deep learning has been attracting wide attention from all over the world in recent years, not only because of the great achievements it has made in various application scenarios, but also due to the promise of an intelligent future it gives. Such a learning methodology makes people believe it is possible for the formation of wise machines that they have long dreamed to possess. The growing data accessibility provides rich examples for deep computational architectures to adjust their internal weights and bias until their predictions have low error rate. On the other hand, the computational devices are relatively affordable than in the previous years by the society, with the help of which, accelerations of learning processes has been achieved, hence a bunch of time consuming deep learning

architectures can be tried within acceptable periods.

In the ILSVRC-2012 competition [23], AlexNet [24] received the championship with the 15.3% top-5 error rate, compared to 26.2% achieved by the runner-up. Such a large margin of error rate claimed a breakthrough in object recognition history. It kicked off a blistering pace of trying out deep learning by both academia and industry, which in turn led to an increase of the convolutional neural networks' submissions to ILSVRC-2013, in which ZF Net [25] was the winner. It fine-turned the architecture of AlexNet based on the gorgeous visualizations of trained models. Both AlexNet and ZF Net are of the same structure which is built up by simply stacking computational layers while GoogLeNet [26] is composed of Inception modules. This new architecture was the most successful candidate in ILSVRC-2014. It has not only set the new height of object recognition but also started to optimize the computational resources of the network by design. It consists of 22 layers, which was deeper than all the previous networks in ILSVRC. However, it is still not deep enough. In ILSVRC-2015, Residual Neural Network (ResNet) [17] with 152 layers won the championships in all the five main tracks. ResNet introduced a brand new notion into the neural network architecture named identity mapping. The shortcut connection in the identity mapping prevents the degradation of training accuracy when the network goes deeper. Besides, the converging speed of ResNet is faster than the network built up with Inception modules when both are of the similar size.

Despite the fact that neural networks built up from Inception modules converge slower than those built up from ResNet modules, it is still worth it for a brief review of the valuable insights residing in the Inception networks. A typical incarnation of the first generation of Inception networks is named GoogLeNet [26]. It was intricately carved with a responsibility to win computer vision tasks in ILSVRC-2014, on which it performed better than all the other deep neural network architectures. There exist philosophical reflections which are intend to serve as guidelines for the construction of Inception networks. Two major downsides of a enlarged neural network have been discussed in [26]. One is the higher chances of overfitting while the other is the strikingly increased requirements of computational resources with the enlarged network size. For handling those drawbacks, based on the new ideas which were introduced in [27] about how to construct the reasonable architecture of neural networks, new experiments orienting sparse network structure have been tried out. One year later after GoogleNet hold the championship of ILSVRC-2014, a method named Batch Normalization [28] has been proposed by Google researchers to accelerate and ease the training of deep neural networks. The core idea behind Batch Normalization is to normalize the inputs to each layer for every batch of training data. More importantly, based on the observation that the normalization process essentially is matrix multiplications followed by adding biases, the Batch Normalization is implemented as additional layers which makes it part of the network architecture. This fairly novel method started a new chapter for the training of deep neural networks. With the adoption of Batch Normalization, higher learning rates no longer impede the convergence of the deep networks, oppositely faster training speed is brought to scene which can achieve a better accuracy of prediction with considerably

less time. Additionally, in some cases, it can even replace the Dropout [29] which is an effective method to prevent overfitting. The incorporation of Batch Normalization into the first generation of Inception network architecture led to the formation of Inception-v2, which improved the best accuracy on ImageNet classification with less training steps. In the same year, Inception-v3 [30] joined the party, the objective of which was to effectively leverage the power of additional computation by factorizing to smaller size convolutions and regularizing the classifier layer with the estimation of minor effect of label-dropout in the training process. The network architectures were scaled up in Inception-v3, which consequently imposed higher requirements of available computational resources. With the ResNet [17] stealing the show in ILSVRC-2015, the influence of the identity connections in residual units on the learning process has been investigated in [31]. The filter concatenation stage of in Inception-v3 is replaced using identify connections which led to the layout of a new model named Inception-ResNet-v1. A more advanced version which was named Inception-ResNet-v2 has a larger network size than the first version. Besides the mixed architectures of Inception-ResNet, a pure Inception incarnation named Inception-v4 was also presented with comparison to Inception-ResNet-v2. Both Inception-v4 and Inception-ResNet-v2 have significant gain of performance mainly benefiting from the enlarged size of network.

2.3 Related Work

In this section, the prior arts working on optimizations for video coding are reviewed.

Before the occurrence of Depth Modeling Mode (DMM) and View Synthesis Optimization (VSO) in [32], a lot of literature on depth map coding which have been published are mainly focusing on improving the effectiveness of depth map coding. Based on the observation that the depth map is characterized by vast smooth regions separated by sharp edges, an algorithm to effectively encode homogeneous regions has been proposed in [33]. It improves coding performance for depth maps by copying pixel values for homogeneous blocks from values of neighboring reference pixels. In [34], Depth Lookup Table (DLT) has been proposed for encoding the depth maps in 3D-HEVC standard. It offers the benefits of 1.3% bit-rate reduction. To further improve the coding performance for depth map, more dedicated tools for depth map coding are needed. Depth Modeling Modes (DMM) and View Synthesis Optimization (VSO) are proposed in [32]. VSO provides 17% bit rate reduction in average while DMM provides 6% savings on bit rate. Although the introduction of DMM and VSO have brought the effectiveness of depth map coding into a new level, the computational complexity has increased a lot due to the complex nature of VSO. Consequently the time cost of depth map encoding becomes fairly expensive.

The computational complexity of depth map coding raised the question of whether it is possible to reduce the computational complexity for saving encoding time. In [35], a fast wedgelet searching scheme achieves significant reduction for computational complexity with minor BD-rate increase.

It takes advantage of the result from Sum of Absolute Transform Difference (SATD) to reduce the wedgelet searching candidates. Rough RD cost from Rough Mode Decision is used as mode selection threshold in [36] to speed up the bi-partition modes decision. A two-step fast searching approach for wedgelet partition appears in [37]. It features a coarse search in conjunction with a further refinement step. Another fast approach for wedgelet searching [38] is to make use of the Most Probable Mode (MPM) to reduce wedgelet searching candidates. Since intra angular modes will lead to ringing artifacts when utilized for depth map coding, the idea of skipping intra angular prediction by making use of edge detector is shown in [39]. Bayesian classifier is used in [40] to alleviate the computational complexity of intra mode decision in 3D-HEVC. The optimal mode of the parent prediction unit (PU) in the hierarchical quad-tree coding structure has been utilized to select the mode for child prediction unit (PU) in [41], and early decision for segment-wise DC coding is used together to achieve faster depth intra coding. Edge classification in Hadamard transform domain is used in [42] to skip the DMM decision process conditionally. The minimum RD cost of the candidates in the full-RD searching list is taken as a threshold to bypass DMM decision based on the comparisons with the header rates in [43]. Most probable region for DMM1 mode decision is identified with the help of sharp edges in [44], and DMM3 is skipped when depth prediction unit (PU) does not match with co-located texture counterpart. Variance is utilized in [45] to estimate the most promising sub-region for DMM1. Corner point is used for fast quad-tree decision of depth intra coding in [46]. Variance distribution is studied in [12], based on which the method termed Squared Euclidean distance of variances (SEDV) is proposed to substitute the long-standing View Synthesis Optimization (VSO) process. Besides, a new scheme termed probability-based early depth intra mode decision (PBED) is employed to skip modes and the RD cost in Rough Mode Decision (RMD) is used to terminate segment-wise depth coding (SDC) [34] as early as possible. The correlation between depth maps and texture views are explored in [47] to alleviate the complexity of the compression for depth map. In [48], comparing RD cost with pre-calculated threshold for fast intra mode decision together with early decision for the CU depth are used to accelerate the encoding process. Making use of RD cost results of the angular modes, only the most promising DMMs are evaluated in [49] and, moreover, an innovative method using golden ratio to further improve the depth map coding is proposed. The characteristics of depth map are studied in [50], as a result only four conventional intra modes are used for depth map intra coding and only six directions are used in DMM1 searching. Block edge along with the border gradient are used together in [51] to accelerate the depth map coding. Information of neighbouring blocks and threshold which is derived from lots of experiments are used in [52] for improving depth map coding.

Despite the aforementioned works which are using heuristic approaches, machine learning approaches are also applied to optimize the video coding process. In [53], the decision for the depth of the coding unit in High Efficiency Video Coding (HEVC) is modeled as a classification problem which is solved by machine learning approach. A shallow convolutional neural network (CNN) is

used in [54] to determine coding unit depth in High Efficiency Video Coding (HEVC) while in [55], a deeper convolutional neural network together with long- and short-term memory (LSTM) network are employed to address the same issue. In additional to the works which are targeting the coding unit depth decision using machine learning approaches, it is found in [56] that deep learning is used for the intra mode selection in Screen Content Coding (SCC) Extension of High Efficiency Video Coding (HEVC).

The researching focus in this thesis is the same as [56]. However, there exist three important differences. Firstly, the network in this thesis comprising 32 layers is much deeper than the 4-layer network in [56]. Secondly, unlike the server-client setup in [56], we have managed to integrate the learned models into the codebase of HTM16.2. Executable binary can be obtained which is totally self-contained in the sense that they are not relying on the remote server to do the prediction, just the binary itself is capable of doing prediction for mode selection in depth maps. Thirdly, in this work the deep learning is for depth map coding in Three Dimension Extension of High Efficiency Video Coding (3D-HEVC) while in [56] the learning is for Screen Content Coding (SCC) Extension of High Efficiency Video Coding (HEVC).

Chapter 3

Prepare the Data for Deep Learning

The success of a deep learning procedure heavily relies on the size of available data. It only works when a considerably large set of data can be provided. Moreover, since we are using supervised learning which is the most popular form of deep learning for the time being, our datasets must be well labeled. A large dataset can contain enormous and different classes, with each class has its own label. The labels are used to adjust the inner parameters of a pre-constructed deep model according to the pre-defined loss function during the training process. We need to prepare a large set of labeled data before starting the model training. In this chapter, we start with the data collection, in which the data source and method for collecting data are shown in detail. After that the necessary pre-processing for the collected data are described. Furthermore, plenty of visualizations for the collected raw data are shown with discussion explaining the reason for the data pre-processing.

3.1 Data Collection

To collect the data for training a deep model to predict the most probable intra angular directions for depth blocks, we need to identify two questions: Firstly, where does the data come from? Secondly, how to collect the data from the source? In this section, the two questions are answered one by one.

3.1.1 Source of Data

The data are collected from four video sequences as shown in Table 3.1. Balloons sequence and Kendo sequence are of the resolution 1024 by 768 while Poznan Street sequence and Undo Dancer sequence are of the resolution 1920 by 1088. Both of the former two sequences have 300 frames, all of which are used to collect the data. The latter two sequences both have 250 frames, and all the

Table 3.1: Source of data for deep learning

#	Name of the Sequence	Resolution	Usage	Number of Frames
1	Balloons	1024×768	train,test,validate	300
2	Kendo	1024×768	train,test,validate	300
3	Poznan Street	1920×1088	train,test,validate	250
4	Undo Dancer	1920×1088	train,test,validate	250

frames are involved in the data collection. The collected data for each sequence will be separated into three sets for training, testing and validating. The training data sets are used for the deep model to learn the best representations by the back-propagation algorithm [57], during which the inner parameters typically weights and bias are adjusted along the gradient as instructed by the back-propagation. After the learned model will be obtained, the validating datasets are used to fine turn the hyper-parameters. With the reasonably adjusted hyper-parameters, the training process will be performed again. After certain loops of the train-validate circle, the testing datasets will be used to evaluate the final learned model which by then will not be further turned any more. After learned model will be applied to the testing datasets, the performance results of evaluation can indicate the generalization of the learning model.

3.1.2 Algorithm for Collecting Data

In this subsection, the algorithm used for collecting data are presented. We collect data by encoding four video sequences shown in Table 3.1 from the coding unit (CU) level. Most of the hybrid video coding features from HEVC remain unchanged in 3D-HEVC, including the sizes allowed for each type of block. There are totally four types of block, namely coding tree unit (CTU), coding unit (CU), prediction unit (PU) and transform unit (TU). Table 3.2 shows the allowed sizes for four types of block. A CTU itself can be used as a single CU while in some scenarios it can be split into

Table 3.2: Allowed sizes of each type of block

Block Type	Allowed Sizes				
CTU			16×16	32×32	64×64
CU		8×8	16×16	32×32	64×64
PU	4×4	8×8	16×16	32×32	64×64
TU	4×4	8×8	16×16	32×32	

multiple CUs [58]. CU sits on top the PU partition structure. CU can be partitioned to form TUs recursively in residual coding. The maximum coding unit (CU) size in 3D-HEVC is 64. The quad-tree splitting syntax allows CUs of largest size to be further split into smaller sizes. The encoder chooses the minimum allowed CU size based on the syntax in the sequence parameter sets (SPS).

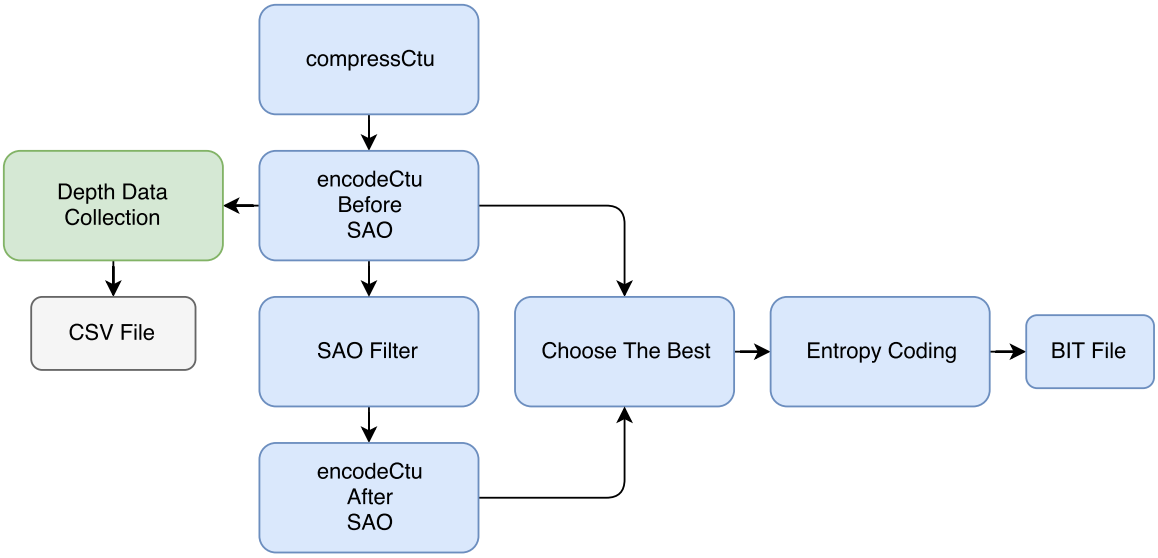

Figure 3.1: Data collecting diagram.

For luma CU samples, the minimum allowed size is larger than or equal to 8×8. The encoder first needs to make the basic decision of whether to code a block with inter-picture or intra-picture prediction at CU level. After that the best mode of the intra-picture prediction is obtained at PU level. The maximum block size allowed for DMM1 is 32×32 while the minimum block size allowed is 4×4. Hence we need to collect data for each block size from 4×4, 8×8, 16×16 to 32×32. To collect the data, we first need to identify to which part in the reference software we should insert the data collecting module. Since supervised learning is chosen, labels for data samples are also required to be collected. The label is the best mode that has been chosen by the HTM16.2 encoder. The diagram in Figure 3.1 illustrates the relationships among the core modules for data collection. The rectangular blocks with light blue background are the modules from HTM16.2 while others are the newly added modules for data collection. In the module of *compressCtu*, the encoder recursively invokes anther module named *xCompressCtu* which is not on the diagram to try different kinds of CU, PU, TU and to decide the best mode for them. Once the *compressCtu* module will be finished, all the needed data can be obtained in the *encodeCtu Before SAO* module. During the data collecting process, only Luma samples in depth blocks are used since we are trying to reduce the computational complexity of DMM1. Moreover, as shown in Figure 3.2 on page 21, the Luma samples are flattened from rectangle CU blocks into a single row which will be subsequently written into associated CSV file.

The detailed implementation for the module of *depth data collection* is shown in Algorithm 1 on page 35. The inputs to the data collecting workflow are exactly the inputs to the module of

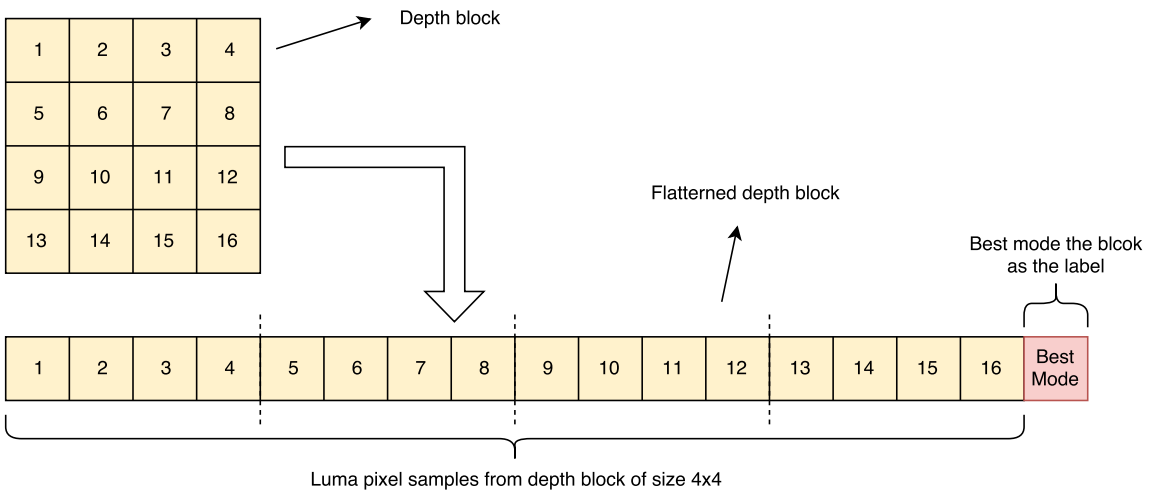

Figure 3.2: Flattern Luma samples into one line and append best mode at the end.

encodeCtu, namely the CU data structure, the absolute partition index and the corresponding quad-tree depth. Firstly, the partition mode of the CU is obtained which can further indicate the partition number to either be one or four. After that, based on the partition number, the best modes for blocks are stored into an four dimensional array of integer values within the range from 0 (inclusive) to 36 (inclusive). If the partition number is equal to one, which means the current CU has not been split and it has its own best mode. The luma pixel samples of this block are collected into a single row in the CSV file. In the end of the same row, the best mode of the CU is signaled. If the partition number is equal to four, which means the current CU has been split to form four smaller sub-blocks with each sub-block has its own best mode. The luma samples for each sub-block are collected into four different rows in the CSV file with the last value in each row to be the best mode for each sub-block. HTM16.2 [59] is used in this work, which is the newest version of the reference software of 3D-HEVC. Considering we are focusing on the intra prediction, All-Intra configuration is used.

3.2 Data Visualization

Visualizing data is a good way for human to better understand and memorize information. Complex patterns hidden in the data are easier to be discovered when they are aesthetically presented via the graphical format. The collected data are visualized with the hope of discovering more information from them. Discussions based on the observations of the visualized blocks are given, which convince us about the necessity of data pre-processing in Section 3.3 before training the deep models.

3.2.1 Visualized Data

We have four sets of data which are from blocks of size 4×4, 8×8, 16×16 to 32×32. The intra prediction modes in 3D-HEVC include DC mode, Planar mode, 33 angular modes, DMM1 and DMM4. In this thesis, several indices are assigned to each mode: 0 for DC mode, 1 for Planar mode, [2,34] for 33 angular modes, 35 for DMM1 and 36 for DMM4.

After all the four sets of data have been visualized, it is found that four corresponding sets of visualizations give the same hints as discussed in Subsection 3.2.2. Besides, to avoid too much visualizations' occupation of the thesis, only the visualizations of blocks of size 8×8 are shown. There are totally 37 figures presented, from Figure 3.3, Figure 3.4, ... to Figure 3.39.

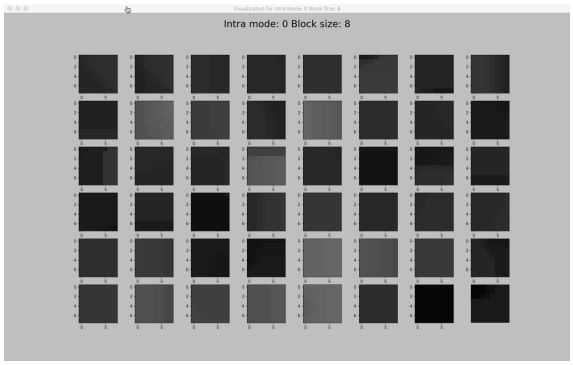

Figure 3.3: Visualizations for blocks tagged with intra DC.

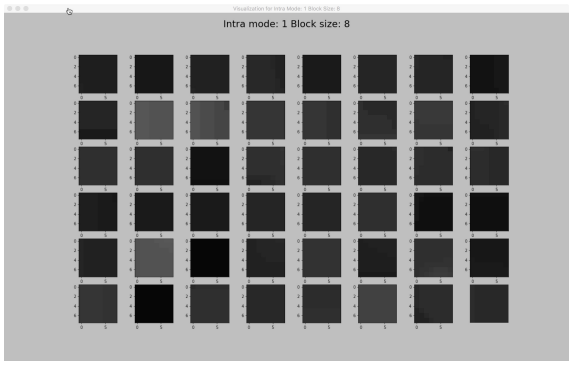

Figure 3.4: Visualizations for blocks tagged with intra PLANAR.

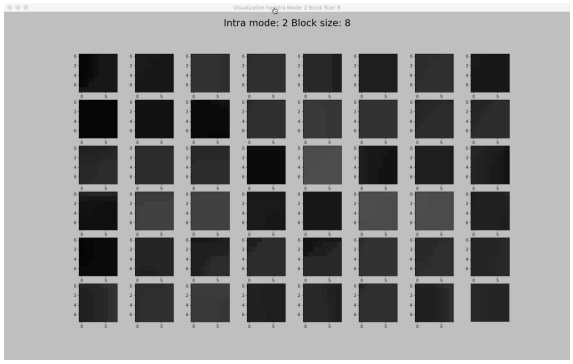

Figure 3.5: Visualizations for blocks tagged with intra mode 2.

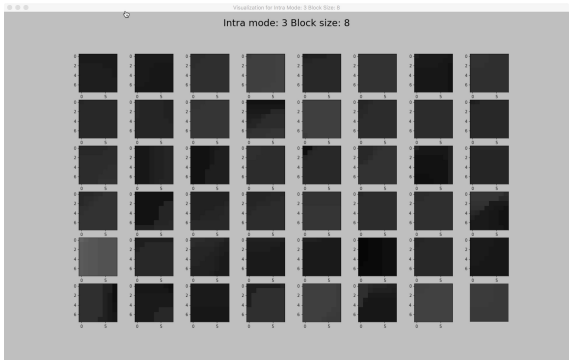

Figure 3.6: Visualizations for blocks tagged with intra mode 3.

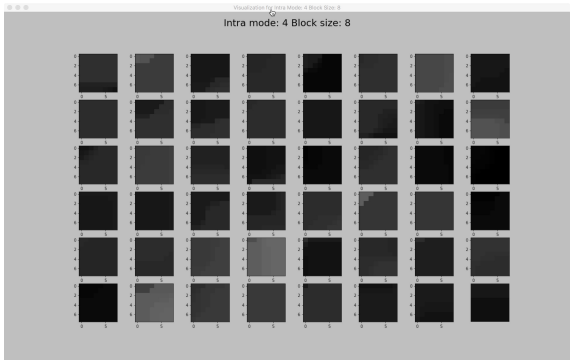

Figure 3.7: Visualizations for blocks tagged with intra mode 4.

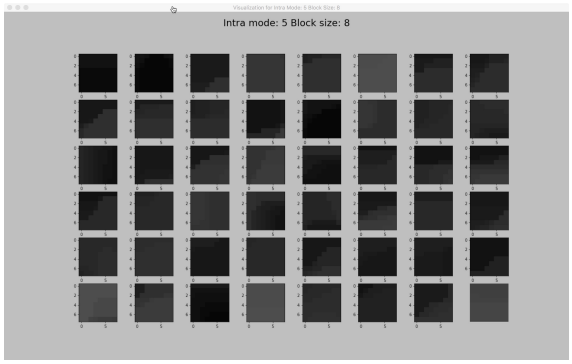

Figure 3.8: Visualizations for blocks tagged with intra mode 5.

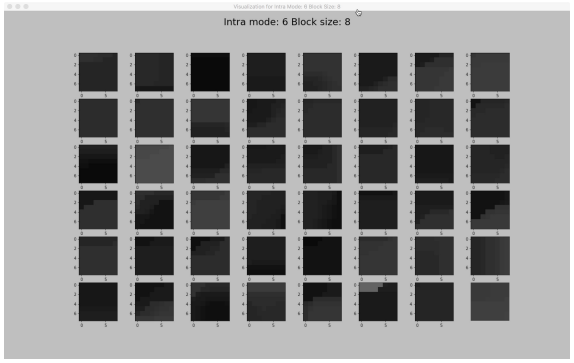

Figure 3.9: Visualizations for blocks tagged with intra mode 6.

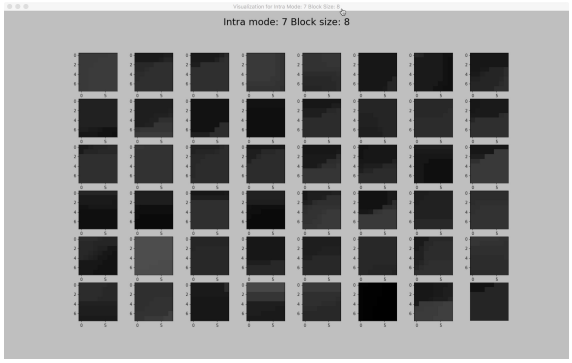

Figure 3.10: Visualizations for blocks tagged with intra mode 7.

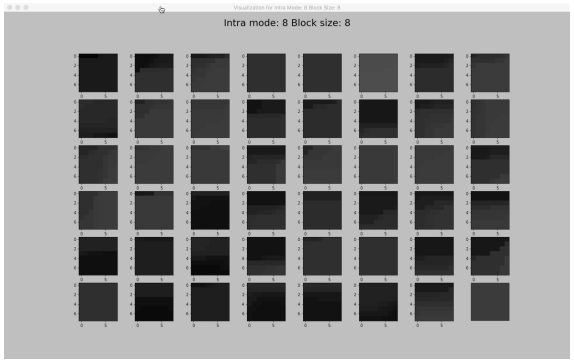

Figure 3.11: Visualizations for blocks tagged with intra mode 8.

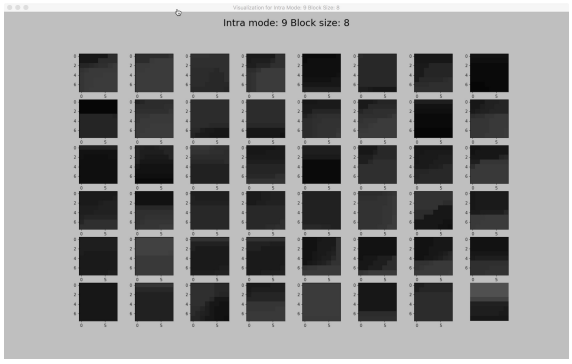

Figure 3.12: Visualizations for blocks tagged with intra mode 9.

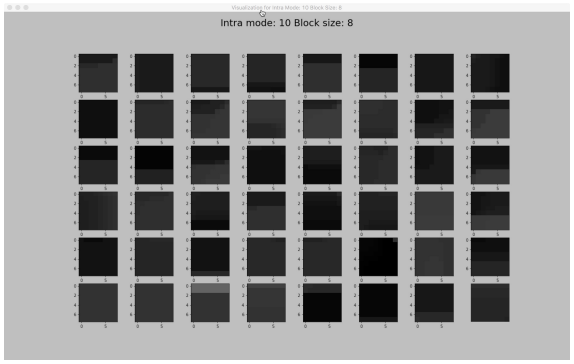

Figure 3.13: Visualizations for blocks tagged with intra mode 10.

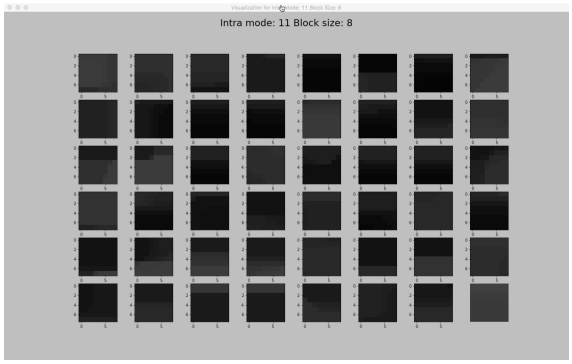

Figure 3.14: Visualizations for blocks tagged with intra mode 11.

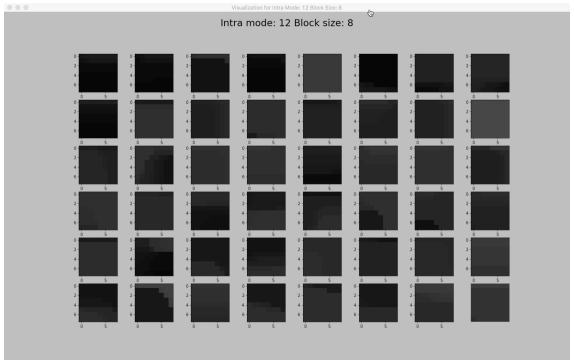

Figure 3.15: Visualizations for blocks tagged with intra mode 12.

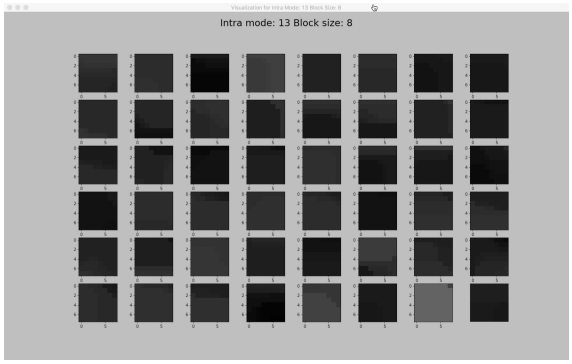

Figure 3.16: Visualizations for blocks tagged with intra mode 13.

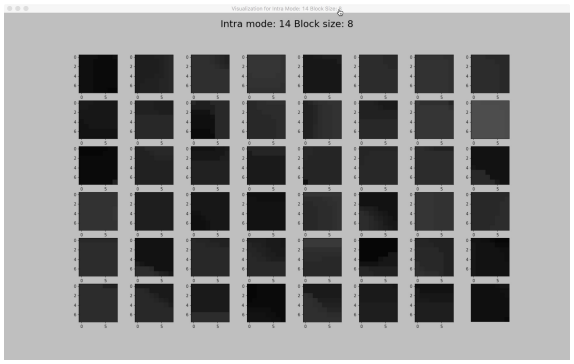

Figure 3.17: Visualizations for blocks tagged with intra mode 14.

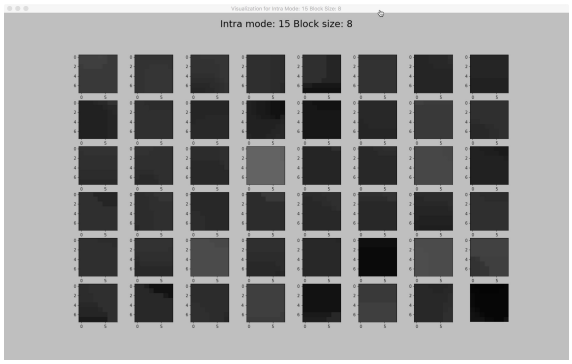

Figure 3.18: Visualizations for blocks tagged with intra mode 15.

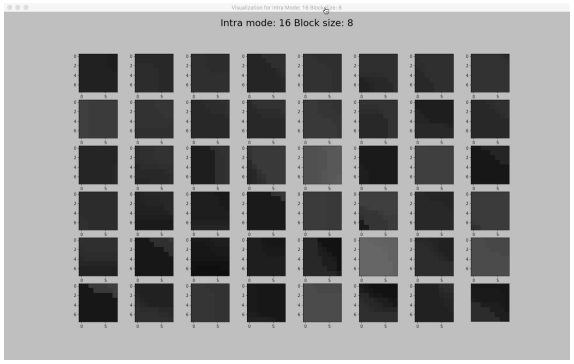

Figure 3.19: Visualizations for blocks tagged with intra mode 16.

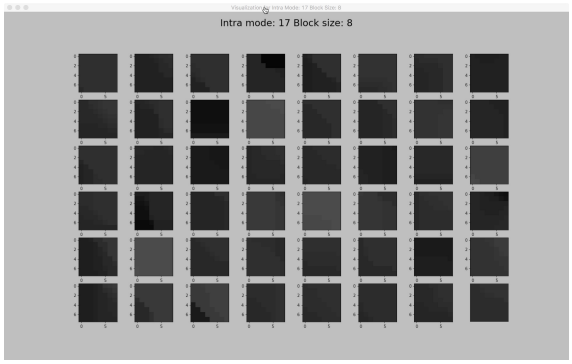

Figure 3.20: Visualizations for blocks tagged with intra mode 17.

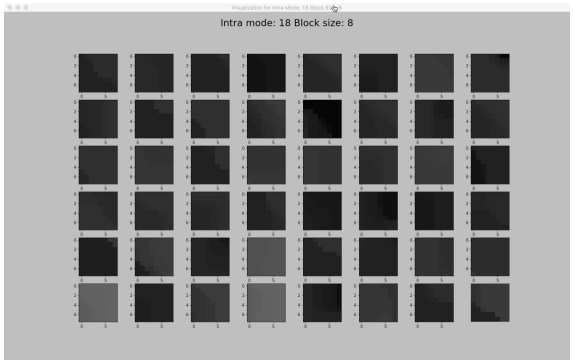

Figure 3.21: Visualizations for blocks tagged with intra mode 18.

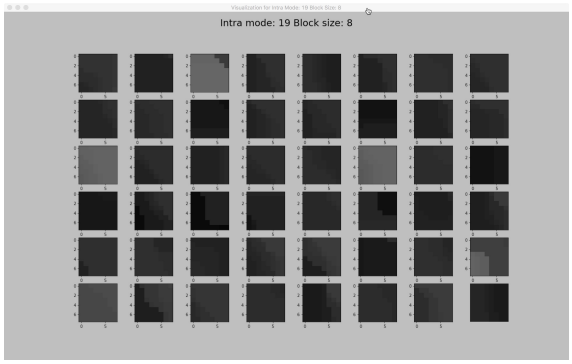

Figure 3.22: Visualizations for blocks tagged with intra mode 19.

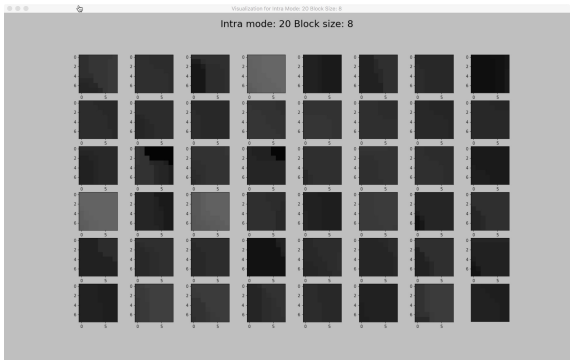

Figure 3.23: Visualizations for blocks tagged with intra mode 20.

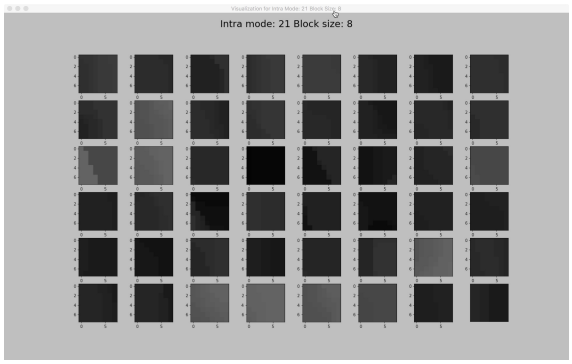

Figure 3.24: Visualizations for blocks tagged with intra mode 21.

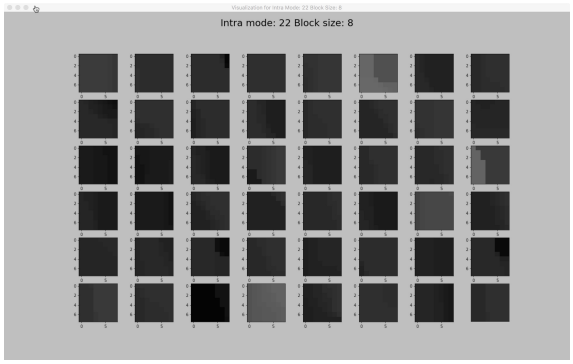

Figure 3.25: Visualizations for blocks tagged with intra mode 22.

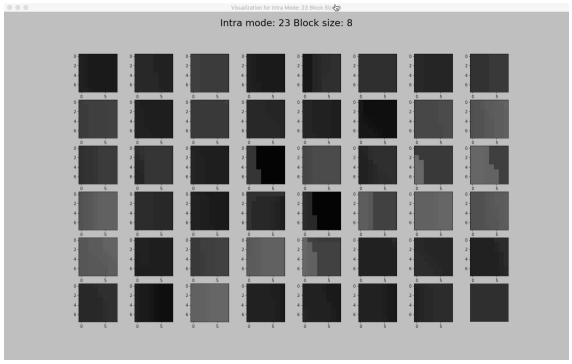

Figure 3.26: Visualizations for blocks tagged with intra mode 23.

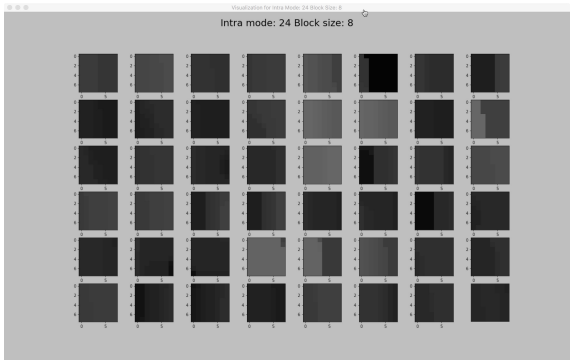

Figure 3.27: Visualizations for blocks tagged with intra mode 24.

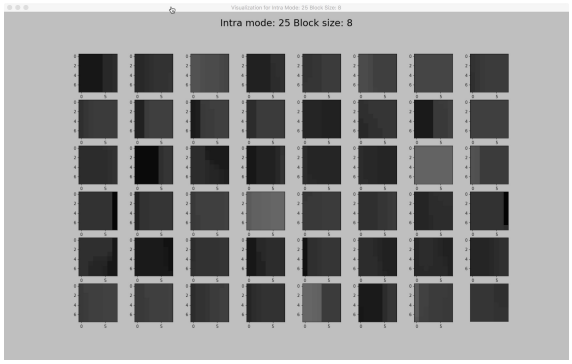

Figure 3.28: Visualizations for blocks tagged with intra mode 25.

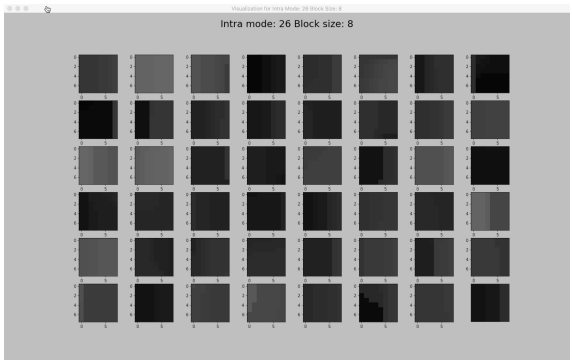

Figure 3.29: Visualizations for blocks tagged with intra mode 26.

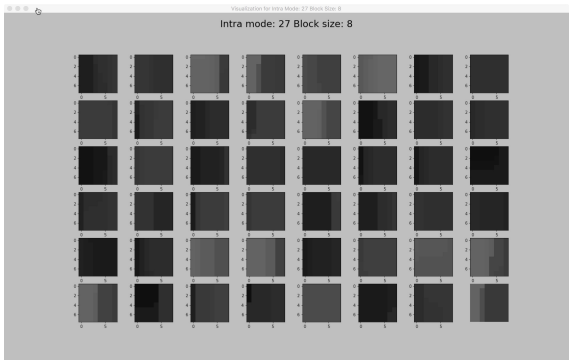

Figure 3.30: Visualizations for blocks tagged with intra mode 27.

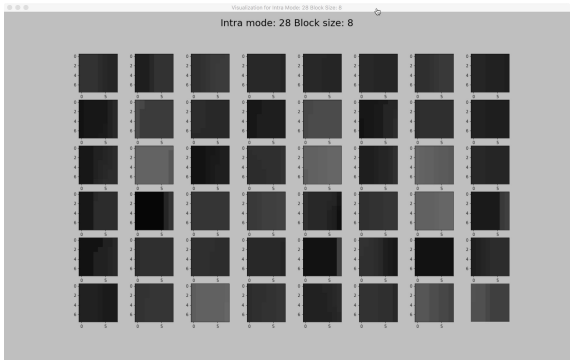

Figure 3.31: Visualizations for blocks tagged with intra mode 28.

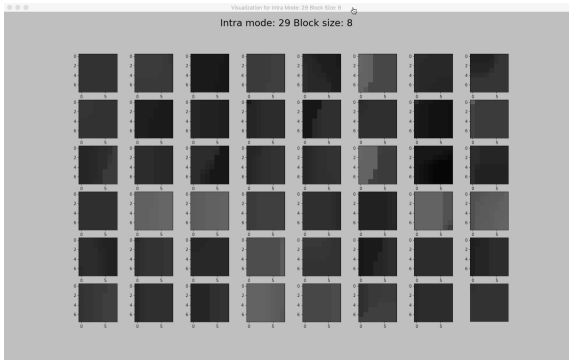

Figure 3.32: Visualizations for blocks tagged with intra mode 29.

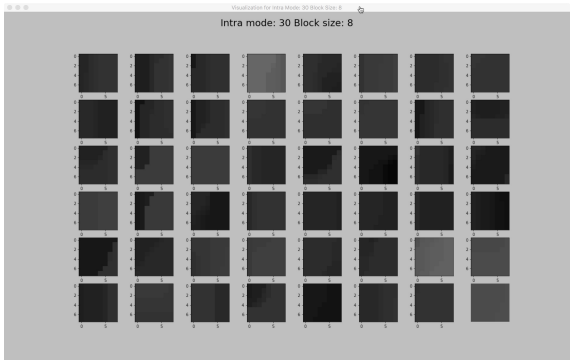

Figure 3.33: Visualizations for blocks tagged with intra mode 30.

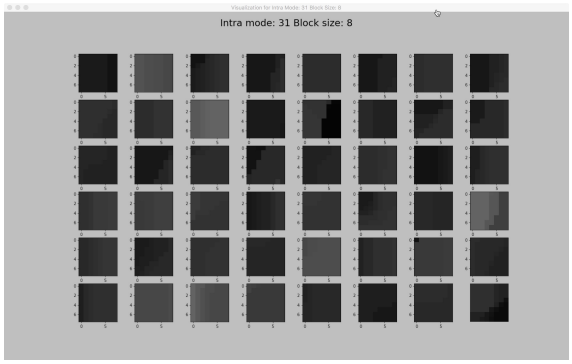

Figure 3.34: Visualizations for blocks tagged with intra mode 31.

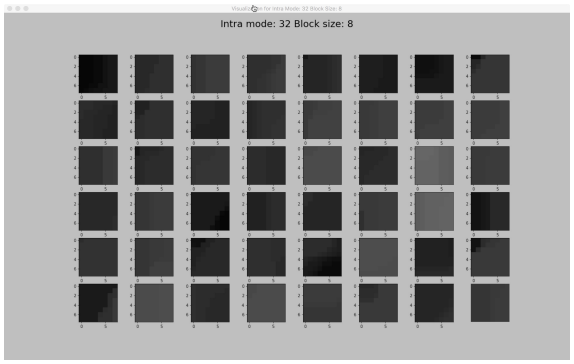

Figure 3.35: Visualizations for blocks tagged with intra mode 32.

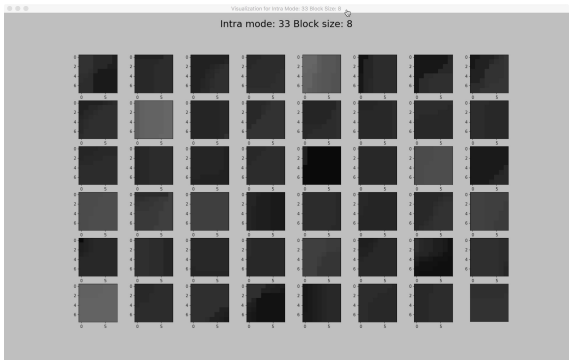

Figure 3.36: Visualizations for blocks tagged with intra mode 33.

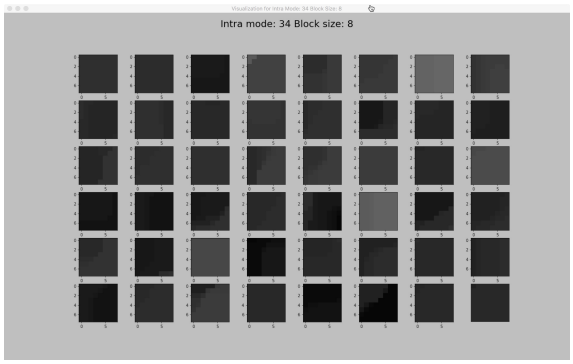

Figure 3.37: Visualizations for blocks tagged with intra mode 34.

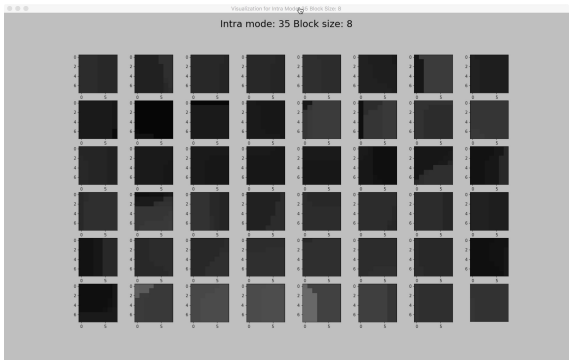

Figure 3.38: Visualizations for blocks tagged with DMM1

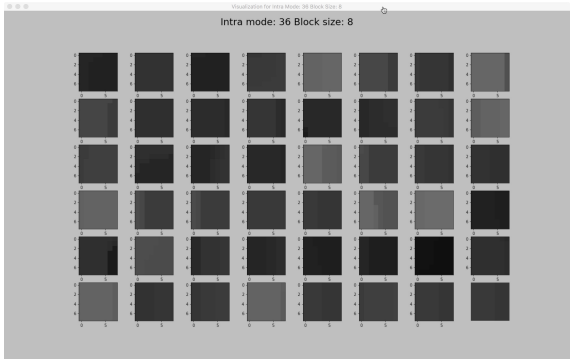

Figure 3.39: Visualizations for blocks tagged with DMM4

3.2.2 Discussion

From the visualizations of blocks of size 8×8, it is found that within blocks of each single angular mode, part of them have sharp edges that can be easily perceived by human sights while the others are very smooth such that their edges can not be clearly seem unless they are enlarged by multiple times. When blocks with such a mixed style are fed into the convolutional neural networks, the prediction accuracy of the constructed computational model is always not ideal enough to be used inside the reference software of 3D-HEVC. However, the same model learns well on other benchmark datasets such as MNIST [60] and CIFAR [61]. There exist two explanations to this phenomenon. One is that the mix of the extreme smoothness and clear sharpness inside each single intra prediction mode yields impure datasets such that the neural networks are not able to figure out the intrinsic abstractions layer by layer. The other is that our network size is not deep enough to have the capability of learning representations for such a mixed style, and meanwhile, the size of the dataset cannot satisfy larger networks due to the limited features available in the training dataset. The limited size of the collected data means that there is no chance of trying to train the deep learning model with a much more rich datasets currently. Moreover, larger neural networks require more computational power which can be very expensive. Combining the two considerations above, eliminating extreme smoothness is the way to go, by which the blocks with vague edges are removed from the datasets.

Special attentions need to be given to mode DC, PLANAR, DMM1 and DMM4. For DC mode and PLANAR mode, since most of their blocks still have weak edges with various patterns, intuitively it seems not practical to require neural network to learn to distinguish them from angular modes. For DMM1 which is specially designed for depth maps, it is noticed that lots of their blocks contain sharp edges with arbitrary angles. Hence the learned model may predict DMM1 into any of the 33 angular modes according to the angle of the partition line in DMM1 blocks. And DMM4 which is another dedicated mode for depth maps, most of their blocks feature contour partitions instead of straight lines while still some contours cannot show their clear characteristics that can discriminate themselves from angular modes with some curvilinear distortions.

In fact, it has been tried to train the deep neural networks which work well on benchmark datasets by using 37 classes, including modes [DC, PLANAR, 2, ..., 34, DMM1, DMM4]. During the training process, the validations are performed in a fixed frequency to monitor the performance of the learned model. Confusion matrix [62] is obtained after every validation process. Figure 3.40, Figure 3.41, Figure 3.42, and Figure 3.43 on page 30 show the confusion matrices after 12, 24, 36 and 48 epochs of model training separately. The color thickness of block with coordinates (i, j) expresses the frequency with which a block with best mode i is predicted as j. For each horizontal line in the matrix, the probabilities of all blocks sum up to 100%. From epoch 12 to epoch 48, most of the thicknesses in matrices have gradually been aggregated to the main diagonal, but there are nevertheless four exceptional classes that predictions for them have never been right in all the confusion matrices. Those four classes are exactly mode DC, PLANAR, DMM1 and DMM4 which

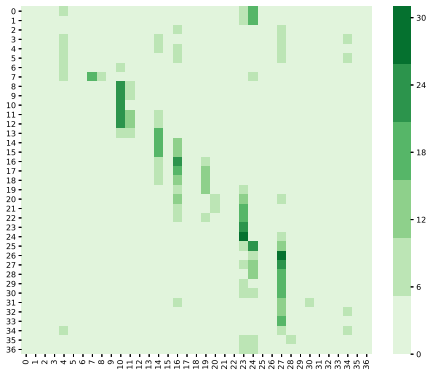

Figure 3.40: Confusion matrix obtained after 12 epochs of model training

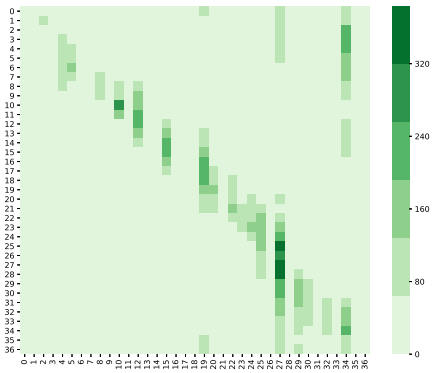

Figure 3.41: Confusion matrix obtained after 24 epochs of model training

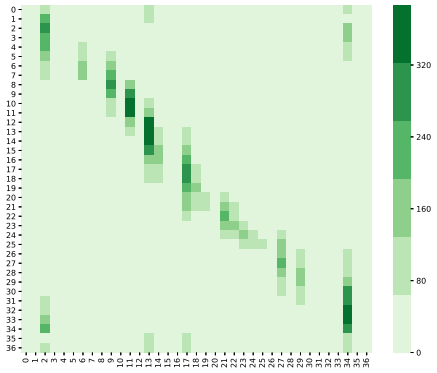

Figure 3.42: Confusion matrix obtained after 36 epochs of model training

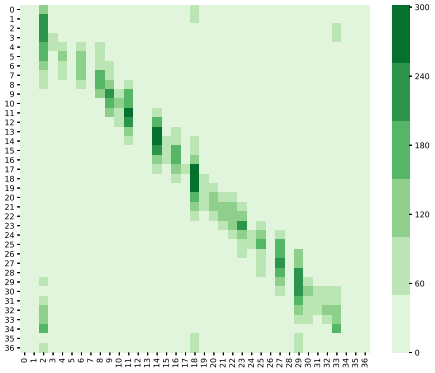

Figure 3.43: Confusion matrix obtained after 48 epochs of model training

are tagged with mode index 0, 1, 35 and 36 separately. It turns out the neural networks will misclassify them into angular modes instead of giving predictions that are identical to their label of ground truth. To cure this illness, it has been decided to remove those four modes from target classes.

Different from the object recognition problems, when convolutional neural networks are employed to predict the intra patterns, some popular data pre-processing steps such as random crop, vertical flip are not applicable anymore. For example, angular mode 18 in Figure 3.21 on page 25 and angular mode 34 in Figure 3.37 on page 28 would be identical to each other in terms of the edge angle if either is vertically flipped.

Angular mode 2 and angular mode 34 are on the same diagonal such that they cannot be treat as two separate classes in our case. For this reason, collected blocks of angular mode 34 is removed from the datasets. Instead of directly predicting angular mode 34, both angular mode 2 and angular mode 34 will be deemed as equal when the prediction result is angular mode 2. Further comparisons between the two modes will be preformed using conventional encoder decision.

3.3 Data Pre-processing

According to the discussions in Subsection 3.2.2 on page 29 which are based on the observations of the visualized data in Subsection 3.2.1 on page 22, several pre-processing actions need to be taken to clean up the collected data for deep learning. The pre-processing steps are very crucial to the success of the learning process for the convolutional neural networks. Without pre-processing the collected data would be inappropriate for achieving required model performance for the prediction activities. In this section, the data pre-processing details in this work are explained in details.

Table 3.3: Information of the datasets after merging

#	Name of the file	Size	Samples	Usage
1	size04.csv	206 MegaBytes	3675428	train,test,validate
2	size08.csv	513 MegaBytes	2372324	train,test,validate
3	size16.csv	1.25 GigaBytes	1439773	train,test,validate
4	size32.csv	2.02 GigaBytes	567554	train,test,validate

Right after encoding the four video sequences for data collection, four CSV files would be obtained for each of them. Every CSV file contains two unique identifications, one is the *size of blocks* collected, the other is the *name of the source video sequence*. For example, one of CSV files shall include depth Luma data of blocks of size 16×16 from *Newspaper* video sequence. In the first step of data-preprocessing, all the CSV files having the same identification of *size of blocks* shall be merged into

a single dataset which will be further divided into training dataset, testing dataset and validating dataset in the subsequent processing steps. The specifications of the four CSV files after merging are shown in Table 3.3 on page 31. There is no data for blocks of size 64×64 since the largest block size for DMM modes is 32×32. As the size of block increases, the total volume of the collected data is growing while the total number of samples of the collected data is decreasing. This phenomenon is reasonable in the sense that the decreasing speed of the number of samples is slower than the increasing speed of the volume of every single record. More specifically, from block size 4×4 to 8×8, the number of samples is decreased by 1.55 times; however, there exists a fourfold increase of the volume size in each sample.

The statistics of the datasets after the first step of merging are shown in Table 3.5 on page 33 and Table 3.6 on page 34. The statistics are unsorted or sorted in terms of the percentages of each mode. The statistics of each mode can be quickly found in Table 3.5 on page 33 by looking at the first column. From Table 3.6 on page 34, it can be observed that the size differences of collected samples for each mode vary in a large range. For example, the blocks of mode 0 are 96.9 times as many the blocks of mode 16 while it is only 1.27 times as many the blocks of mode 35 for blocks of size 32×32. This is introducing the topic of imbalanced learning [63] in which the data sizes of each class are very different from each other.

According to the discussions presented in Subsection 3.2.2, mode 0, 1, 34, 35 and 36 will be removed from the datasets in the second step. The specifications of the data after the second step are shown in Table 3.4. More than half of the collected data are removed.

Table 3.4: Information of the datasets after removing mode 0, 1, 34, 35 and 36

#	Name of the file	Size	Samples	Usage	Percent of removed data (%)
1	msize04.csv	75.7 MegaBytes	3675428	train,test,validate	64
2	msize08.csv	130.8 MegaBytes	2372324	train,test,validate	74
3	msize16.csv	377.3 MegaBytes	1439773	train,test,validate	70
4	msize32.csv	708.7 MegaBytes	567554	train,test,validate	65

In the third step, we start to remove the smooth blocks. The reasons have been discussed in Subsection 3.2.2. An algorithm based on the coarse edge strength analysis in [54] has been developed to define the sharpness of depth blocks.

Table 3.5: Unsorted statistics of datasets obtained after merging

Block Size	4×4		8×8		16×16		32×32	
Mode Idx	Samples	Percent (%)	Samples	Percent (%)	Samples	Percent (%)	Samples	Percent (%)
0	717,274	19.52	642,520	27.08	459,291	31.90	158,824	27.98
1	482,776	13.14	249,061	10.50	164,050	11.39	57,528	10.14
2	97,629	2.66	25,101	1.06	12,868	0.89	4,561	0.80
3	17,991	0.49	12,489	0.53	7,946	0.55	2,863	0.50
4	14,375	0.39	11,688	0.49	8,642	0.60	2,175	0.38
5	15,849	0.43	13,428	0.57	8,829	0.61	2,164	0.38
6	17,144	0.47	15,318	0.65	9,768	0.68	2,958	0.52
7	18,187	0.49	17,238	0.73	15,988	1.11	6,625	1.17
8	19,146	0.52	15,785	0.67	20,357	1.41	11,642	2.05
9	23,462	0.64	12,362	0.52	18,207	1.26	18,195	3.21
10	42,752	1.16	9,740	0.41	7,978	0.55	18,972	3.34
11	23,727	0.65	12,836	0.54	17,696	1.23	21,142	3.73
12	21,992	0.60	17,837	0.75	23,143	1.61	13,262	2.34
13	24,613	0.67	20,254	0.85	19,260	1.34	6,740	1.19
14	22,620	0.62	17,784	0.75	13,851	0.96	2,995	0.53
15	21,169	0.58	18,268	0.77	12,834	0.89	2,073	0.37
16	20,289	0.55	15,418	0.65	10,214	0.71	1,639	0.29
17	21,869	0.60	17,501	0.74	10,010	0.70	1,977	0.35
18	52,552	1.43	19,889	0.84	9,862	0.68	1,998	0.35
19	23,871	0.65	16,171	0.68	9,797	0.68	2,170	0.38
20	22,992	0.63	15,656	0.66	10,227	0.71	1,925	0.34
21	25,416	0.69	16,706	0.70	11,978	0.83	2,504	0.44
22	27,593	0.75	16,446	0.69	12,251	0.85	2,925	0.52
23	33,250	0.90	16,783	0.71	12,744	0.89	3,707	0.65
24	40,677	1.11	17,262	0.73	12,257	0.85	4,457	0.79
25	36,018	0.98	13,841	0.58	7,812	0.54	5,123	0.90
26	404,933	11.02	70,417	2.97	30,896	2.15	17,897	3.15
27	48,843	1.33	17,062	0.72	12,205	0.85	7,414	1.31
28	34,217	0.93	24,051	1.01	16,725	1.16	6,169	1.09
29	37,756	1.03	23,486	0.99	15,647	1.09	5,436	0.96
30	30,910	0.84	20,972	0.88	12,782	0.89	3,945	0.69
31	35,102	0.95	20,499	0.86	13,331	0.93	3,475	0.61
32	25,756	0.70	18,811	0.79	12,254	0.85	3,289	0.58
33	33,270	0.91	19,088	0.80	11,943	0.83	3,526	0.62
34	73,107	1.99	31,114	1.31	15,848	1.10	5,382	0.95
35	789,662	21.48	710,089	29.93	299,368	20.79	126,427	22.28
36	276,639	7.53	139,353	5.87	70,914	4.93	23,450	4.13

Table 3.6: Sorted statistics of datasets obtained after merging

Block Size		4×4			8×8			16×16			32×32	
Row idx	Mode idx	Samples	Percent (%)	Mode idx	Samples	Percent (%)	Mode idx	Samples	Percent (%)	Mode idx	Samples	Percent (%)
0	4	14,375	0.39	10	9,740	0.41	25	7,812	0.54	16	1,639	0.29
1	5	15,849	0.43	4	11,688	0.49	3	7,946	0.55	20	1,925	0.34
2	6	17,144	0.47	9	12,362	0.52	10	7,978	0.55	17	1,977	0.35
3	3	17,991	0.49	3	12,489	0.53	4	8,642	0.60	18	1,998	0.35
4	7	18,187	0.49	11	12,836	0.54	5	8,829	0.61	15	2,073	0.37
5	8	19,146	0.52	5	13,428	0.57	6	9,768	0.68	5	2,164	0.38
6	16	20,289	0.55	25	13,841	0.58	19	9,797	0.68	19	2,170	0.38
7	15	21,169	0.58	6	15,318	0.65	18	9,862	0.68	4	2,175	0.38
8	17	21,869	0.60	16	15,418	0.65	17	10,010	0.70	21	2,504	0.44
9	12	21,992	0.60	20	15,656	0.66	16	10,214	0.71	3	2,863	0.50
10	14	22,620	0.62	8	15,785	0.67	20	10,227	0.71	22	2,925	0.52
11	20	22,992	0.63	19	16,171	0.68	33	11,943	0.83	6	2,958	0.52
12	9	23,462	0.64	22	16,446	0.69	21	11,978	0.83	14	2,995	0.53
13	11	23,727	0.65	21	16,706	0.70	27	12,205	0.85	32	3,289	0.58
14	19	23,871	0.65	23	16,783	0.71	22	12,251	0.85	31	3,475	0.61
15	13	24,613	0.67	27	17,062	0.72	32	12,254	0.85	33	3,526	0.62
16	21	25,416	0.69	7	17,238	0.73	24	12,257	0.85	23	3,707	0.65
17	32	25,756	0.70	24	17,262	0.73	23	12,744	0.89	30	3,945	0.69
18	22	27,593	0.75	17	17,501	0.74	30	12,782	0.89	24	4,457	0.79
19	30	30,910	0.84	14	17,784	0.75	15	12,834	0.89	2	4,561	0.80
20	23	33,250	0.90	12	17,837	0.75	2	12,868	0.89	25	5,123	0.90
21	33	33,270	0.91	15	18,268	0.77	31	13,331	0.93	34	5,382	0.95
22	28	34,217	0.93	32	18,811	0.79	14	13,851	0.96	29	5,436	0.96
23	31	35,102	0.95	33	19,088	0.80	29	15,647	1.09	28	6,169	1.09
24	25	36,018	0.98	18	19,889	0.84	34	15,848	1.10	7	6,625	1.17
25	29	37,756	1.03	13	20,254	0.85	7	15,988	1.11	13	6,740	1.19
26	24	40,677	1.11	31	20,499	0.86	28	16,725	1.16	27	7,414	1.31
27	10	42,752	1.16	30	20,972	0.88	11	17,696	1.23	8	11,642	2.05
28	27	48,843	1.33	29	23,486	0.99	9	18,207	1.26	12	13,262	2.34
29	18	52,552	1.43	28	24,051	1.01	13	19,260	1.34	26	17,897	3.15
30	34	73,107	1.99	2	25,101	1.06	8	20,357	1.41	9	18,195	3.21
31	2	97,629	2.66	34	31,114	1.31	12	23,143	1.61	10	18,972	3.34
32	36	276,639	7.53	26	70,417	2.97	26	30,896	2.15	11	21,142	3.73
33	26	404,933	11.02	36	139,353	5.87	36	70,914	4.93	36	23,450	4.13
34	1	482,776	13.14	1	249,061	10.50	1	164,050	11.39	1	57,528	10.14
35	0	717,274	19.52	0	642,520	27.08	35	299,368	20.79	35	126,427	22.28
36	35	789,662	21.48	35	710,089	29.93	0	459,291	31.90	0	158,824	27.98

Algorithm 1: Collect data

Input: CU data structure pcCU, absolute partition index of CU uiAbsPartIdx, quad-tree depth uiDepth

Output: Flattened luma pixel values of each block together with the index of its best intra mode in each row of the output csv file

```
1  begin
2      for each CU in depth maps do
3          uiCuSize ←getCUSize(pcCU, uiDepth)
4          pOrgPel ←getYPelCU(pcCU)
5          if DISFlag ≡ 0 then
6              partitionMode ←getPartitionSize(pcCU, uiAbsPartIdx)
7              if partitionMode ≡ sizeOfNByN then
8                  iPartNum ← 4
9              else
10                 iPartNum ← 1
11             for j ← 0 to iPartNum do
12                 iDir[j] ← getIntraDir(pcCU, uiAbsPartIdx)
13             if iPartNum ≡ 1 then
14                 Create a new csv file, append the value of uiDepth at the end of the name of
                    the new csv file
15                 for y ← 0 to uiCuSize do
16                     for x ← 0 to uiCuSize do
17                         Write pOrgPel[x] into row_m in csv file
18                     pOrgPel ← pOrgPel + iStride
19                 Write iDir[0] into the end of row_m in the csv file
20             else
21                 Create a new csv file, append the value of (uiDepth + 1) at the end of the name
                    of the new csv file
22                 sizeOfSubBlk ←getSizeOfSubBlk(pcCU, uiDepth)
23                 for j ← 0 to iPartNum do
24                     if j ≡ 0 then
25                         yStartPos ← 0 & xStartPos ← 0 & yEndPos ← sizeOfSubBlk & xEndPos
                            ← sizeOfSubBlk
26                     else if j ≡ 1 then
27                         yStartPos ← 0 & xStartPos ← sizeOfSubBlk & yEndPos ← sizeOfSubBlk
                            & xEndPos ← sizeOfSubBlk × 2
28                     else if j ≡ 2 then
29                         yStartPos ← sizeOfSubBlk & xStartPos ← 0 & yEndPos
                            ← sizeOfSubBlk × 2 & xEndPos ← sizeOfSubBlk
30                     else if j ≡ 3 then
31                         yStartPos ← sizeOfSubBlk & xStartPos ← sizeOfSubBlk & yEndPos
                            ← sizeOfSubBlk × 2 & xEndPos ← sizeOfSubBlk × 2
32                 for y ← yStartPos to yEndPos do
33                     for x ← xStartPos to xEndPos do
34                         Write pOrgPel[x] into row_m in the csv file
35                     pOrgPel ← pOrgPel + iStride
36                 Write iDir[j] into the end of row_m in the csv file
```

Chapter 4

Train the Deep Model for Prediction

4.1 The Architecture of the Deep Convolutional Neural Network

4.2 The Hyper-parameters of the Deep Convolutional Neural Network

4.3 Stopping criteria and Training Results

Chapter 5

Evaluate the Learned Deep Model

5.1 Evaluate the Deep Model trained using blocks of size 08x08

5.2 Evaluate the Deep Model trained using blocks of size 16x16

5.2.1 Evaluate the Deep Model on blocks of size 16x16

5.2.2 Evaluate the Deep Model on blocks of size 32x32

5.3 Discussion

Chapter 6

Employ the Learned Deep Model

With the rising popularity of the high definition videos, the new standard termed High Efficiency Video Coding (HEVC) for compressing videos in a more efficient way comparing with previous standards, such as H.264/AVC, has emerged under the efforts from the Joint Collaborative Team on Video Coding (JCT-VC). In the meanwhile, five extensions of the HEVC standard, comprising Format Range Extension (RExt), Scalability Extension (SHVC), Multi-view Extension (MV-HEVC), 3D Extension (3D-HEVC), Screen Content Coding Extension (SCC), have been finalized from 2014 to 2016 to support fulfill extra requirements in various scenarios.

6.1 The Analysis and Optimisation for the Time of Prediction

6.2 The Integration of the Learned Model

6.3 Results of Experiments

6.4 Discussion

Chapter 7

Conclusion

With the rising popularity of the high definition videos, the new standard termed High Efficiency Video Coding (HEVC) for compressing videos in a more efficient way comparing with previous standards, such as H.264/AVC, has emerged under the efforts from the Joint Collaborative Team on Video Coding (JCT-VC). In the meanwhile, five extensions of the HEVC standard, comprising Format Range Extension (RExt), Scalability Extension (SHVC), Multi-view Extension (MV-HEVC), 3D Extension (3D-HEVC), Screen Content Coding Extension (SCC), have been finalized from 2014 to 2016 to support fulfill extra requirements in various scenarios.

Bibliography

[1] Web Page, 2017. [Online]. Available: http://hidefnj.com/video.html.

[2] C. E. Shannon, *The mathematical theory of communication*. Urbana: Urbana : University of Illinois Press, 1949.

[3] "Itu-t recommendation database," 2017. [Online]. Available: http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=12905&lang=en.

[4] "Jct-vc - joint collaborative team on video coding," 2017. [Online]. Available: http://www.itu.int/en/ITU-T/studygroups/2013-2016/16/Pages/video/jctvc.aspx.

[5] I. E. Richardson, *The H.264 Advanced Video Compression Standard*, 2nd ed.., ser. H.264 Advanced Video Compression Standard 2e. Hoboken: Hoboken : Wiley, 2010.

[6] A. Vetro, T. Wiegand, and G. J. Sullivan, "Overview of the stereo and multiview video coding extensions of the h.264/mpeg-4 avc standard," *Proceedings of the IEEE*, vol. 99, no. 4, pp. 626–642, 2011, ISSN: 0018-9219. DOI: 10.1109/JPROC.2010.2098830.

[7] J. Konrad and M. Halle, "3-d displays and signal processing," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 97–111, 2007, ISSN: 1053-5888. DOI: 10.1109/msp.2007.905706.

[8] I. Sexton and P. Surman, "Stereoscopic and autostereoscopic display systems," *Signal Processing Magazine, IEEE*, vol. 16, no. 3, pp. 85–99, 1999, ISSN: 1053-5888. DOI: 10.1109/79.768575.

[9] Mu, amp, X, K. Ller, P. Merkle, and T. Wiegand, "3-d video representation using depth maps," *Proceedings of the IEEE*, vol. 99, no. 4, pp. 643–656, 2011, ISSN: 0018-9219. DOI: 10.1109/JPROC.2010.2091090.

[10] "Jct-3v - joint collaborative team on 3d video coding extension development," 2017. [Online]. Available: http://www.itu.int/en/ITU-T/studygroups/2013-2016/16/Pages/video/jct3v.aspx.

[11] G. Tech, K. Ying Chen, J.-R. Muller, A. Ohm, A. Vetro, and A. Ye-Kui Wang, "Overview of the multiview and 3d extensions of high efficiency video coding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 26, no. 1, pp. 35–49, 2016, ISSN: 1051-8215. DOI: 10.1109/TCSVT.2015.2477935.

[12] H.-B. Zhang, C.-H. Fu, Y.-L. Chan, S.-H. Tsang, and W.-C. Siu, "Probability-based depth intra mode skipping strategy and novel vso metric for dmm decision in 3d-hevc," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2016, ISSN: 1051-8215. DOI: 10.1109/TCSVT.2016.2612693.

[13] H. Dou, Y.-L. Chan, K.-B. Jia, and W.-C. Siu, "Segment-based view synthesis optimization scheme in 3d-hevc," *Journal of Visual Communication and Image Representation*, vol. 42, pp. 104–111, 2017, ISSN: 1047-3203. DOI: 10.1016/j.jvcir.2016.11.012.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, ser. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016, xxii, 775 pages, ISBN: 9780262035613 0262035618.

[15] B. Schlkopf, C. J. C. Burges, and A. J. Smola, *Advances in kernel methods : support vector learning*. Cambridge, Mass. ; London: Cambridge, Mass. ; London : MIT Press, 1999.

[16] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks : the official journal of the International Neural Network Society*, vol. 61, p. 85, 2015.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[18] Web Page, 2017. [Online]. Available: https://www.bazel.build/.

[19] Web Page, 2017. [Online]. Available: https://cloud.google.com/storage/.

[20] C. Y. Chou, "Advances in grid and pervasive computing: First international conference, gpc 2006," in. 2006, ISBN: 3540338098. [Online]. Available: https://en.wikipedia.org/wiki/Bandwidth_(computing)#Network_bandwidth_capacity.

[21] M. Ghanbari, *Video coding: an introduction to standard codecs*. London: Institution of Electrical Engineers, 1999.

[22] Y. A. LeCun, L. Bottou, G. B. Orr, and K. R. Mller, "Efficient backprop," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7700, pp. 9–48, 2012, ISSN: 03029743. DOI: 10.1007/978-3-642-35289-8-3.

[23] Web Page, 2017. [Online]. Available: http://www.image-net.org/challenges/LSVRC/.

[24] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017, ISSN: 0001-0782. DOI: 10.1145/3065386.

[25] Generic, 2014. DOI: 10.1007/978-3-319-10590-1_53.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

[27] S. Arora, A. Bhaskara, R. Ge, and T. Ma, "Provable bounds for learning some deep representations," *CoRR*, vol. abs/1310.6343, 2013. [Online]. Available: `http://arxiv.org/abs/1310.6343`.

[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014, ISSN: 1532-4435.

[30] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015.

[31] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," 2016.

[32] K. Muller, P. Merkle, G. Tech, and T. Wiegand, *3d video coding with depth modeling modes and view synthesis optimization*, Generic, 2012.

[33] S. H. Tsang, Y. L. Chan, and W. C. Siu, "Efficient intra prediction algorithm for smooth regions in depth coding," *Electronics Letters*, vol. 48, no. 18, pp. 1–2, 2012, ISSN: 00135194. DOI: `10.1049/el.2012.1768`.

[34] F. Jager, *Simplified depth map intra coding with an optional depth lookup table*, Generic, 2012. DOI: `10.1109/IC3D.2012.6615142`.

[35] Z. Mengmeng, Z. Chuan, X. Jizheng, and B. Huihui, *A fast depth-map wedgelet partitioning scheme for intra prediction in 3d video coding*, Generic, 2013. DOI: `10.1109/ISCAS.2013.6572473`.

[36] Z. Gu, J. Zheng, N. Ling, and P. Zhang, *Fast bi-partition mode selection for 3d hevc depth intra coding*, Generic, 2014. DOI: `10.1109/ICME.2014.6890324`.

[37] P. Merkle, K. Muller, and T. Wiegand, *Coding of depth signals for 3d video using wedgelet block segmentation with residual adaptation*, Generic, 2013. DOI: `10.1109/ICME.2013.6607429`.

[38] Z. Gu, J. Zheng, N. Ling, and P. Zhang, *Fast depth modeling mode selection for 3d hevc depth intra coding*, Generic, 2013. DOI: `10.1109/ICMEW.2013.6618267`.

[39] G. Sanchez, M. Saldanha, G. Balota, B. Zatt, M. Porto, and L. Agostini, *Complexity reduction for 3d-hevc depth maps intra-frame prediction using simplified edge detector algorithm*, Generic, 2014. DOI: `10.1109/ICIP.2014.7025649`.

[40] H. R. Tohidypour, M. T. Pourazad, and P. Nasiopoulos, *A low complexity mode decision approach for hevc-based 3d video coding using a bayesian method*, Generic, 2014. DOI: `10.1109/ICASSP.2014.6853726`.

[41] H. Zhang, S. H. Tsang, Y. L. Chan, C. H. Fu, and W. C. Siu, *Early determination of intra mode and segment-wise dc coding for depth map based on hierarchical coding structure in 3d-hevc*, Generic, 2015. DOI: 10.1109/APSIPA.2015.7415297.

[42] C. S. Park, "Edge-based intramode selection for depth-map coding in 3d-hevc," *IEEE Trans. Image Process.*, vol. 24, no. 1, pp. 155–162, 2015, ISSN: 1057-7149. DOI: 10.1109/TIP.2014.2375653.

[43] C. S. Park, "Efficient intra-mode decision algorithm skipping unnecessary depth-modelling modes in 3d-hevc," *Electronics Letters*, vol. 51, no. 10, pp. 756–758, 2015, ISSN: 00135194. DOI: 10.1049/el.2014.3874.

[44] H.-B. Zhang, C.-H. Fu, Y.-L. Chan, S.-H. Tsang, W.-C. Siu, and W.-M. Su, "Efficient wedgelet pattern decision for depth modeling modes in three-dimensional high-efficiency video coding," *Journal of Electronic Imaging*, vol. 25, no. 3, p. 033 023, 2016, ISSN: 1017-9909. DOI: 10.1117/1.JEI.25.3.033023.

[45] C. H. Fu, H. B. Zhang, W. M. Su, S. H. Tsang, and Y. L. Chan, "Fast wedgelet pattern decision for dmm in 3d-hevc," in *2015 IEEE International Conference on Digital Signal Processing (DSP)*, pp. 477–481, ISBN: 1546-1874. DOI: 10.1109/ICDSP.2015.7251918.

[46] H. B. Zhang, Y. L. Chan, C. H. Fu, S. H. Tsang, and W. C. Siu, "Quadtree decision for depth intra coding in 3d-hevc by good feature," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1481–1485. DOI: 10.1109/ICASSP.2016.7471923.

[47] Q. Zhang, M. Chen, X. Huang, N. Li, and Y. Gan, "Low-complexity depth map compression in hevc-based 3d video coding," *EURASIP Journal on Image and Video Processing*, vol. 2015, no. 1, pp. 1–14, 2015. DOI: 10.1186/s13640-015-0058-5.

[48] K. K. Peng, J. C. Chiang, and W. N. Lie, "Low complexity depth intra coding combining fast intra mode and fast cu size decision in 3d-hevc," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 1126–1130. DOI: 10.1109/ICIP.2016.7532533.

[49] P. Liu, G. He, S. Xue, and Y. Li, "A fast mode selection for depth modelling modes of intra depth coding in 3d-hevc," *VCIP 2016 - 30th Anniversary of Visual Communication and Image Processing*, pp. 0–3, 2017, ISSN: 9781509053162. DOI: 10.1109/VCIP.2016.7805548.

[50] Q. Zhang, N. Zhang, T. Wei, K. Huang, X. Qian, and Y. Gan, "Fast depth map mode decision based on depthtexture correlation and edge classification for 3d-hevc," *Journal of Visual Communication and Image Representation*, vol. 45, pp. 170–180, 2017, ISSN: 1047-3203. DOI: 10.1016/j.jvcir.2017.03.004.

[51] G. Sanchez, M. Saldanha, G. Balota, B. Zatt, M. Porto, and L. Agostini, "Dmmfast: A complexity reduction scheme for three-dimensional high-efficiency video coding intraframe depth map coding," *Journal of Electronic Imaging*, vol. 24, no. 2, p. 023 011, 2015, ISSN: 1017-9909. DOI: 10.1117/1.JEI.24.2.023011.

[52] G. Sanchez, L. Jordani, C. Marcon, and L. Agostini, "Dfps: A fast pattern selector for depth modeling mode 1 in three-dimensional high-efficiency video coding standard," *J. Electron. Imaging*, vol. 25, no. 6, 2016, ISSN: 1017-9909. DOI: 10.1117/1.JEI.25.6.063011.

[53] 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," *Image Processing, IEEE Transactions on*, vol. 24, no. 7, pp. 2225–2238, 2015, ISSN: 1057-7149. DOI: 10.1109/TIP.2015.2417498.

[54] Z. Liu, X. Yu, Y. Gao, S. Chen, X. Ji, and D. Wang, "Cu partition mode decision for hevc hardwired intra encoder using convolution neural network," *Image Processing, IEEE Transactions on*, vol. 25, no. 11, pp. 5088–5103, 2016, ISSN: 1057-7149. DOI: 10.1109/TIP.2016.2601264.

[55] M. Xu, T. Li, Z. Wang, X. Deng, and Z. Guan, "Reducing complexity of HEVC: A deep learning approach," *CoRR*, vol. abs/1710.01218, 2017. arXiv: 1710.01218. [Online]. Available: http://arxiv.org/abs/1710.01218.

[56] T. Laude and J. Ostermann, "Deep learning-based intra prediction mode decision for hevc," ISBN: 9781509059669. DOI: 10.1109/PCS.2016.7906399.

[57] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, Ed. Morgan-Kaufmann, 1990, pp. 396–404. [Online]. Available: http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf.

[58] G. J. Sullivan, J. Ohm, T. Woo-Jin Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1649–1668, 2012, ISSN: 1051-8215. DOI: 10.1109/TCSVT.2012.2221191.

[59] Web Page, 2017. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware/tags/HTM-16.2/.

[60] Y. LeCun, C. Cortes, and C. J. Burges, *The mnist database of handwritten digits*, Web Page. [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[61] A. Krizhevsky, V. Nair, and G. Hinton., *The cifar-10 and cifar-100 datasets*, Web Page. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html.

[62] *Confusion matrix*, Web Page, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Confusion_matrix.

[63] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009, ISSN: 1041-4347. DOI: 10.1109/TKDE.2008.239.