Image And Video Compression: Human And Computer Vision Perspectives

by

Hossam Amer

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2020

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

As we start a new decade, image and video compression should further improve to satisfy each of the human and computer visions. Human and computer visions have different perspectives on the perceived images and videos, which are compressed due to bandwidth and storage requirements. From a human vision (HV) perspective, one key aspect of human satisfaction is the perceived quality of these compressed images and videos. From a computer vision (CV) perspective, especially in image classification, one crucial aspect for machine satisfaction is the ability to accurately recognize patterns or objects in these compressed images and videos. This thesis is motivated to address a variety of image/video compression problems to serve each of human and computer vision perspectives. For HV, our goal is focused on video compression to improve the trade-off between compression rate, compression distortion, and time complexity, while our goal for CV is to show that compression if used in the right manner, helps improve deep neural network (DNN) machines in terms of classification accuracy while reducing the size in bits of the input image.

Towards the HV perspective, we first introduced a global rate-distortion optimization (RDO) model rather than the existing RDO in the state-of-the-art video codec, High Efficiency Video Coding (HEVC), that is traditionally performed within each frame with fixed quantization parameters (QPs), without fully considering the coding dependencies between the current frame and future frames within a temporal propagation chain. To further improve the coding efficiency of HEVC, it is desirable to perform a global RDO among consecutive frames while maintaining a similar coding complexity. To address this problem, temporal dependencies are first measured via a model for the energy of prediction residuals that enables the formulation of the global RDO in low-delay (LD) HEVC. Second, we introduce the notion of propagation length, which is defined as the impact length of the current frame on future frames. This length is estimated via offline experiments and used to propose two novel methods to predict the impact of the coding distortion of the current frame on future frames from previous frames of similar coding properties. Third, we apply these two methods to adaptively determine the Lagrangian multiplier and its corresponding QP for each frame in the LD configuration of HEVC. Experimental results show that, in comparison to the default LD HEVC, the first method can achieve, on average, BD-rate savings of 5.0% and 4.9% in low-delay-P (LDP) and low-delay-B (LDB) configurations, respectively, and the second can achieve, on average, BD-rate savings of 4.9% and 4.9% in the LDP and LDB configurations, respectively, all with only 1% increase in the encoding time. This work has piqued serious interest from industry, such as Google.

Along with the HV perspective, despite the rate distortion performance improvement that HEVC offers, it is computationally expensive due to the adoption of a large variety of

coding unit (CU) sizes in its RDO. Thus, we investigated the application of fully connected neural networks (NNs) to this time-sensitive application to improve its time complexity, while controlling the resulting bitrate loss. Specifically, four NNs are introduced with one NN for each depth of the coding tree unit. These NNs either split the current CU or terminate the CU search algorithm. Because the training of NNs is time-consuming and requires large training data, we further propose a novel training strategy in which offline training and online adaptation work together to overcome this limitation. Our features are extracted from original frames based on the Laplacian Transparent Composite Model (LPTCM). Experiments carried out on all-intra configuration for HEVC reveal that our method is among the best NN methods, with an average time saving of 32% and an average controlled bitrate loss of 1.6%, compared to the original HEVC. In our CU partition algorithm, a fully connected NN machine 'saw' extracted LPTCM features to help reduce the computational intensity of compression at a controlled trade-off between compression rate and compression distortion.

Turning to CV perspective where DNNs typically 'see' the input as a JPEG image, we revisited the impact of JPEG compression on deep learning (DL) in image classification. Given an underlying DNN pre-trained with pristine ImageNet images, we demonstrated that if for any original image, one can select, among its many JPEG compressed versions including its original version, a suitable version as an input to the underlying DNN, then the classification accuracy of the underlying DNN can be improved significantly while the size in bits of the selected input is, on average, reduced dramatically in comparison with the original image. This is in contrast to the conventional understanding that JPEG compression generally degrades the classification accuracy of DL. Specifically, for each original image, consider its 10 JPEG compressed versions with their quality factor (QF) values from $\{100, 90, 80, 70, 60, 50, 40, 30, 20, 10\}$. Under the assumption that the ground truth label of the original image is known at the time of selecting an input, but unknown to the underlying DNN, we presented a selector called Highest Rank Selector (HRS). It is shown that HRS is optimal in the sense of achieving the highest top-k accuracy on any set of images for any k among all possible selectors. When the underlying DNN is Inception V3 or ResNet-50 V2, HRS improves, on average, the top-1 classification accuracy and top-5 classification accuracy on the whole ImageNet validation dataset by 5.6% and 1.9%, respectively, while reducing the input size in bits dramatically—the compression ratio (CR) between the size of the original images and the size of the selected input images by HRS is 8 for the whole ImageNet validation dataset. When the ground truth label of the original image is unknown at the time of selection, we further propose selectors that either maintain the top-1 accuracy, the top-5 accuracy, or the top-1 and top-5 accuracy of the underlying DNN, while achieving CRs of 8.8, 3.3, and 3.1, respectively.

Acknowledgements

First and foremost, I am heartily thankful to my thesis supervisor, Prof. En-hui Yang, whose guidance and support paved the road to complete my PhD degree. Throughout my PhD training, Professor Yang taught me the fundamentals of high-quality research, presentation, and writing. Professor Yang not only inspired my research quality with his constant feedback, but also shaped my logical and critical thinking process. Professor Yang always took the extra step to show his care to my current and future success.

I wish to thank Professor Alex Wong and Professor Fakhri Karray, and Professor Zhou Wang, for being my thesis committee members and their valuable feedback on my work. Also, I wish to thank Professor Jie Liang from the Simon Fraser University for his commitment to serve as my external examining committee member.

Not forgetting also the support of my lab-mates in Multimedia Lab who are the invaluable resource of my improvement and happiness. Special thanks also to the graduate team whose support was vital to complete my PhD degree and to my friends for their great memories, continuous support, and care.

Finally, an honorable mention goes to my family for their understanding and support throughout the PhD journey. They were behind my back during the hard times before the good times. In particular, I would like to express my sincere gratitude to my father, Prof. Mostafa Amer, whose words were constant fuel to overcome any challenges. Sign of love and gratitude especially go to my mother, Mrs. Soad Lotfy, who is a flowing river of love and kindness. Since I was born, I achieved what I achieved because of my mother. A lot of times, my mum listened and supported me to always be the best version of myself. I also would like to thank my brother and companion, Dr. Ihab Amer, whose advice and encouragement helped me a lot in my life. When my brother completed his thesis, he wrote that he saw himself in me, and I am happy that I reached this stage with his support and motivation. Special thanks also to my sister, Mrs. Samah Amer, whose non-stop understanding, love, and support right from my childhood are pillars in my life. Seeing how my sister successfully manage her life is no doubt an immense source of motivation. May my brother and sister see their kind legacy reflect on their kids: Salma, Seif, Mostafa, Serene, Selim, Ahmed. Sign of special gratitude to my sister in-law, Mrs. Nora Ahmed, for her true and valuable care, friendship, and support that helped ease a lot of things in my life. Many thanks to my brother in-law, Dr. Samer AlGhazaly, for his support.

Any acknowledgement ends up with incomplete list inevitably, but I still wish to thank those professors, who are the instructors of classes I have attended, mentors, who I learnt a lot from, people that I met during this period, who supported me to earn the PhD degree, and experiences, which shaped who I am today.

***My family**: I always feel the pleasure to be a part of such a kind family. Your care and support are keys to where I am today and everyday.*

***My mother, Mrs. Soad Lotfy**: Your love, understanding and support were always inspiring my success. Your happiness in general, for me, and about me is my life-time goal.*

***My father, Prof. Mostafa Amer**: My entire life is all about you. Seeing you, or even remembering you is enough to get through any challenges.*

My mission, My life is just to make you happy.

You are in my heart and I will never let you down.

This is for your soul, Dady

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Every second 725 images are posted on Instagram, more than 2,000 to Facebook and about 50 hours of video are uploaded to YouTube. On top of that, recent statistics from Cisco forecast an increase in video traffic by four-fold, and an increase by more than seven-fold in machine-to-machine communication in the period between 2017 and 2022. This shows that this ongoing and increasing flow of images or videos, which are typically compressed to meet storage or bandwidth requirements, could be 'seen' by either humans or machines.

To see the big picture, this thesis addresses a group of image/video compression problems in today's practical multimedia system. As shown in Figure 1.1, this system consists of an information source or a camera that captures digital images and videos, an encoder to compress these images to meet the expensive channel bandwidth requirements or limited space of the storage media, and decoder to reconstruct these images and videos, which are seen by either humans or machines. From a human vision perspective, one key aspect for human satisfaction is the perceived fidelity of these images and videos. On the other hand, from a computer vision perspective, one crucial aspect for machine satisfaction in the context of image classification is their ability to accurately recognize patterns or objects in these images and videos. Thus, the first major part of this thesis takes the human vision perspective into account and tackles video compression to find an improved trade-off between compression rate, compression distortion, and computational intensity. As for the computer vision perspective, we can see from Figure 1.1 that compression has an impact on the task of image classification in accurately recognizing patterns or objects in the input images, which we investigate in the second major part of this thesis.

Towards human vision, the foundation of video compression is to reduce the size of video data to the least possible accompanied with a negligible difference in the video quality that

Figure 1.1: A Typical Digital Image/Video Compression Scenario

can not be noticed with a human eye [101]. From this view, to reach the best coding efficiency, the state-of-the-art video coding standard, High Efficiency Video Coding (HEVC), encompass several advanced techniques that have many coding parameters, such as quantization parameters (QPs), block structures called CU structures, and coding modes, that require optimization in the midst of the encoding process [119]. A complex procedure called the rate-distortion optimization (RDO) is pivotal to select these parameters in order to minimize the coding distortion subject to a given coding rate [116, 96, 135, 131, 119]. Despite the need of RDO and other complex encoding operations, the encoding speed is rather an important factor that should be considered due to its impact on the video viewing experience. In the first major part thesis, we propose better RDO algorithms that improve video compression in terms of the trade-off between compression rate, compression distortion, and encoding time. These algorithms are applied on the state-of-the-art video coding standard, HEVC (see Chapter 3 and 4 for details).

Turning to computer vision, deep neural network (DNN) machines proved a great success in several tasks such as image classification and recognition due to their computer vision system [124, 35, 84, 91]. The visual system of these machines were inspired from human brains, whereby humans solves the image classification task via hierarchical processing along the ventral pathway of the visual cortex [38, 39]. Similarly, DNNs learn the parameters of non-linear activation functions using a mini-batch gradient descent learning algorithm, where these functions progressively transform raw pixels of the input image to produce the output predicted label. These non-linear functions provide multi-layer representations to the input image, where each representation is a feature that amplifies certain aspects of the raw pixels required for classification and suppress irrelevant information. Traditionally, images in DNNs' large datasets are stored in JPEG format [127]. The design of the JPEG codec influences the raw pixels of the input image to DNNs and their subsequent representations. Thus, we dedicate the second major portion of this thesis to study the impact of JPEG compression on DNN computer vision in the context of image classification in terms of classification accuracy, which turns out to be positive in contrast to the conventional understanding (as demonstrated in Chapter 5).

Future work of this thesis can be well pictured in the multimedia system in Figure

1.1. The first one is for human vision where improved versions of the proposed RDO algorithm can be introduced to further improve the coding efficiency of video compression. The second one is for computer vision where we believe that compressionists should think about designing better compression algorithms favourable to computer vision rather than the existing ones that are solely focusing on the bitrate vs perceived quality criterion. Both pieces of work will be discussed in more detail in the last chapter.

1.1 Thesis motivation

This thesis is primarily motivated by a desire to answer the following three questions inspired by the multimedia system shown in Figure 1.1. Figure 1.2 shows a helicopter view for these research questions, which we describe in details as follows:

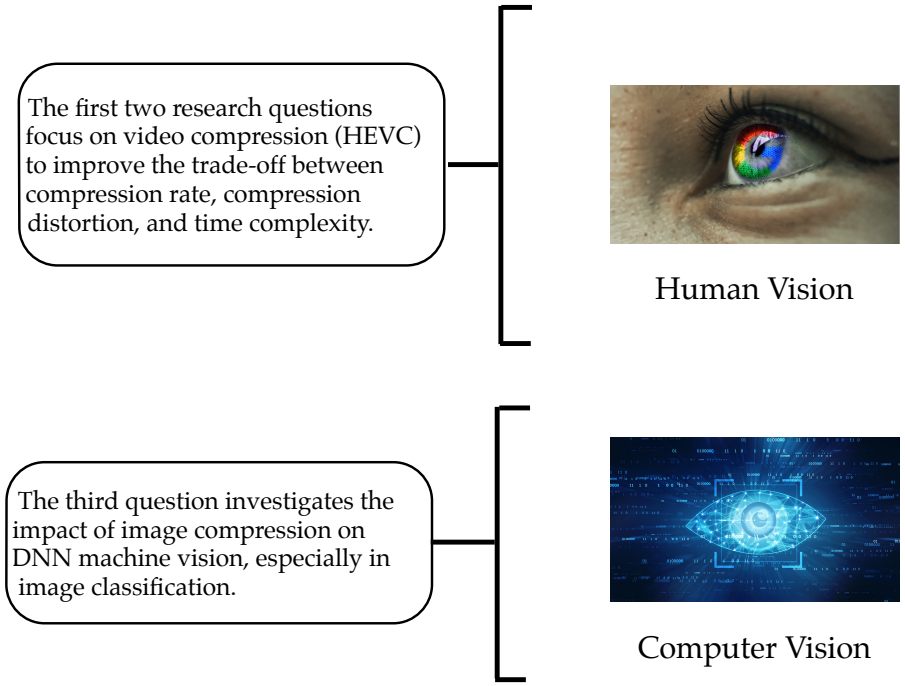


Figure 1.2: Overview on Thesis Research Questions.

1. For human vision perspective, how can RDO further be improved for video compression to achieve a better trade-off between compression rate and compression distortion with insignificant increase in encoding time?

RDO is a pivotal component in video compression to select the best coding parameters in the midst of the encoding process [131, 119]. Due to the inherent complexity of HEVC, RDO is typically performed on each frame by identifying the best coding mode given predefined QP values. In low-delay HEVC, it typically encodes P/B frames that are strongly connected to the coding efficiency of all their past reference frames, which leads to inter-frame dependency. This dependency results in a coding propagation effect, in which the coding of the current frame can impact the coding of its future frames in the temporal chain. Without fully considering this propagation effect, the potential impact on the trade-off between compression rate and compression distortion, or the coding efficiency, is bounded.

Therefore, it is desirable to revisit the global RDO problem among consecutive frames while controlling the time complexity. To address this problem, we are motivated to measure the temporal dependencies via a model for the energy of prediction residuals that enables the formulation of the global RDO in low-delay HEVC. Then, we are motivated to introduce the notion of propagation length, which is defined as the impact length of the current frame on future frames. This length aligns with one's intuition, in view that the impact of the coding distortion for the current frame on future frames has a damping effect. If this length can be estimated, methods that predict the coding dependencies via past frames can be created. Accordingly, a Lagrangian multiplier and QP for each frame can adaptively be calculated, which in turn can improve the coding efficiency.

2. Along the human vision perspective, how can you adopt a neural network machine online to minimize the encoding time in the HEVC CU partition process, while controlling the trade-off between the compression rate and compression distortion?

   There is a quite rich literature for optimizing the time complexity of HEVC through predicting the CU partition structure while controlling the trade-off between compression rate and compression distortion. Some of these methods are threshold-based and have fixed designs [143, 111]. Other methods extracted features from the newly emerged Laplacian Transparent Composite Model (LPTCM) [100]. These features were employed to train a Baye's model to decrease the complexity of HEVC [62]. Similar features were also used to train an SVM for the same purpose [110]. All of these methods were developed due to the inherent complexity of HEVC owing to its large variety of CU sizes that are selected via RDO.

   With the increasing success of neural network machines in several tasks, we are motivated to adopt neural networks to improve the complexity of the HEVC CU partition process in intra coding as a classification problem. Applying NNs, however,

to time-sensitive applications such as the HEVC CU partition problem is challenging. First, the learning process requires a large amount of data and long training time. Second, even if an NN is properly populated, the time required for it to make a decision may not be negligible, especially when it is deep. Techniques in the literature typically utilize deep networks trained offline without updation to circumvent these two problems [44, 88, 132]. Therefore, we are motivated to examine the effect of creating a neural network based learning strategy to work online and minimize the encoding time in the HEVC CU partition process, while controlling the trade-off between compression rate and compression distortion.

3. For the computer vision perspective, which version of compressed raw data is good to deep learning (DL) and its related applications?

   The literature investigated this question to some extent in the context of image classification on the basis of constant quality factors (QFs) which are the same for all images in a whole set of JPEG images [66, 40, 145, 47, 50, 90]. It has been shown that compression has a negative impact on the top-1 and top-5 classification accuracy of deep neural network computer vision. Based on this understanding, some other methods such as stability training took this negative impact into account to improve the robustness of NNs against compression [145]. Inspite of the robustness improvement, there is still a significant degradation (as high as 10%) in classification accuracy when these newly trained DNN models are applied to low quality JPEG compressed images. Based on these findings, it is generally believed that compression, especially JPEG compression, would hurt the classification accuracy of deep learning (DL) in image classification.

   We are motivated to investigate this question in the context of JPEG compression from a different perspective. Instead of using a constant QF in JPEG compression for all images, we would allow each image to be compressed first with a possibly different QF and then fed into a DNN. Suppose that each image is compressed with QF values from 10 to 100 with a step size of 10. For each image, there are now 11 different versions: 1 original version plus 10 compressed versions. For each original image, we are free to select one version out of its 11 versions to be fed into the DNN. This discussion culminates into this interesting question: Is there any selector that can select, for each original image, a suitable version to be fed into the DNN so that both the top-1 classification accuracy and top-5 classification accuracy of the DNN can be improved significantly while the size (in bits) of the input image to the DNN can be reduced dramatically in general?

1.2 Thesis contributions

The scope of this thesis is to contribute to the three questions raised in the last section as follows:

1. In answering the first question in Section 1.1, we proposed an adaptive frame-level QP selection algorithm for the hierarchical coding structure in LD HEVC by characterizing the coding propagation effect through a notion called temporal propagation length, which is defined as the impact length of the current frame on its future frames. This length is estimated via offline experiments that led to solving the global RDO problem and creating two novel methods that predict the coding dependencies via past frames without the need of look-up-tables to store pre-estimated values of the reconstruction distortion of future frames. Using these methods, our overall coding efficiency savings go up to 5.0% for LD HEVC, respectively at insignificant increase in encoding time of 1%. In addition, the proposed methods provide considerable improvements to the coding efficiency of slow motion sequences and video conferencing sequences that can on average go up to 12.9%, 9.5%, respectively. Our algorithms and results piqued serious interest from Google[1] due to their practicality and interoperability.

2. in answering the second question in Section 1.1, we presented a new CU partition prediction method equipped with hierarchical fully connected NN models and features from LPTCM to minimize the encoding time in the HEVC CU partition process, while controlling the trade-off between the compression rate and compression distortion. Also, we highlighted challenges to equip NN online learning with HEVC and proposed adaptive training strategies to these challenges. Experimental results demonstrated that our technique is among the best NN methods with controlled BD-rate loss and of comparable performance to others. Our technique achieves 32% TS average with 1.6% BD-rate average, and attains time savings that can go as high as 60% at low bitrates. In this CU partition algorithm, a fully connected NN machine 'saw' manually extracted LPTCM features to make a classification decision to help reduce the encoding time of compression at a controlled trade-off between compression rate and compression distortion.

3. in answering the third question in Section 1.1, we formulated a new framework to investigate the impact of JPEG compression on DL in image classification. Fix

[1] A collaboration proposal was accepted based on this work.

an underlying DNN pre-trained with pristine images. For any original image, the framework allows one to select, among many JPEG compressed versions of the original image including possibly the original image itself, a suitable version as an input to the underlying DNN. It is demonstrated that within this framework, a selector can be designed so that the classification accuracy of the underlying DNN can be improved significantly while the size in bits of the selected input is, on average, reduced dramatically in comparison with the original image. Therefore, compression, if used in the right manner, helps DL in image classification.

1.3 Thesis organization

The remainder of this thesis is organized as follows: Chapter 2 provides an overview to the core concepts required for this thesis contributions. Chapter 3 introduces the adaptive QP algorithm for the LD HEVC. Chapter 4 proposes the fast CU split algorithm based on NN and LPTCM for HEVC coding. Chapter 5 introduces the framework with JPEG compression and deep learning and show that compression can help deep learning. Last but not least, Chapter 6 concludes this thesis and proposes related future work.

Chapter 2

Background

This chapter covers the core concepts and background related to the research conducted in this thesis. Section 2.1 and Section 2.2 review the details of JPEG and HEVC compression schemes, respectively. Section 2.3 introduces classification and present popular neural network architectures within the context of classification. Last but not least, Section 2.4 summarizes this chapter.

2.1 Overview on the JPEG Standard

JPEG is a well-known lossy and DCT-based image compression standard for 2D digital images that appeared in late 1980s [127, 98]. It contains the basic structure of other block-based video codecs [131, 119]. As shown in Figure 2.1, RGB channels are first converted into YCbCr channels to exploit the psychovisual redundancy. Then, each channel of the YCbCr channels of the input image is partitioned into 8x8 non-overlapping blocks and then chroma sub-sampling is applied. Luminance is indicated by Y, while chrominance is denoted by Cb and Cr. Typically, 4:2:0 chroma sub-sampling is the one applied in JPEG compression, where each 8x8 luma block corresponds to two 2x2 chroma blocks. Next, the current 8x8 block is transformed using the 2D DCT transform to produce 64 DCT coefficients divided into one DC coefficient, $d_{(0,0)}$, and 63 AC coefficients, $d_{(i,j)}$, where $0 \leq i \leq 7$, $0 \leq j \leq 7$ and $i \neq j \neq 0$. The DCT transformation is defined as follows:

$$d_{u,v} = \frac{1}{4}\alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y} cos[\frac{(2x+1)u\pi}{16}] cos[\frac{(2y+1)v\pi}{16}] \tag{2.1}$$

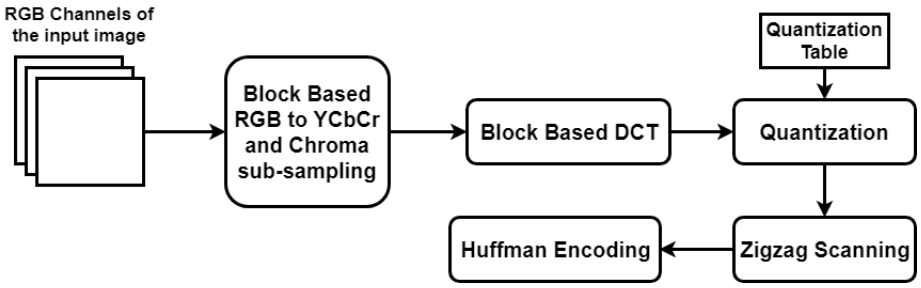


Figure 2.1: Overview on JPEG image encoder.

where $u \in [0, 7]$ and $v \in [0, 7]$ are the horizontal and vertical spatial coordinate in frequency domain, $g_{x,y}$ is the spatial domain value at coordinate (x, y). $\alpha(\cdot)$ is a normalizing scale factor to make the transformation orthogonal, where $\alpha(i)$ equals to $\frac{1}{\sqrt{2}}$ if $i = 0$ and 1 otherwise. DCT is typically applied to achieve energy compaction and decorrelation, as depicted in Figure 2.2. Clearly, most of the energy of the transformed block resides in the top-left corner of the transformed coefficients matrix. Hence, preserving a selected number of these coefficients, while removing the remainder, will have a minor effect on the reconstructed block when converted back to the spatial domain. An example that demonstrates the unimportance of the high-frequency coefficients due to limitations of the human visual system is shown in Figure 2.3.

The more coefficients that are ignored, the more distortion is introduced in the reconstructed block. Figure 2.3 illustrates that keeping only the 6 top-left coefficients (out of 64) of the transformed matrix per 8x8 block would result in a reconstructed frame with an extremely high similarity with the original frame (all coefficients are preserved) which shows the importance of the next step: quantization.

Quantization is a non-linear operation responsible for removing unimportant high-frequency coefficients (equivalent to low-pass filtering the blocks) in each 64 coefficients per block. The produced 64 coefficients per block are quantized and rounded via user-specified quantization tables. In this thesis, we utilize the default quantization tables for

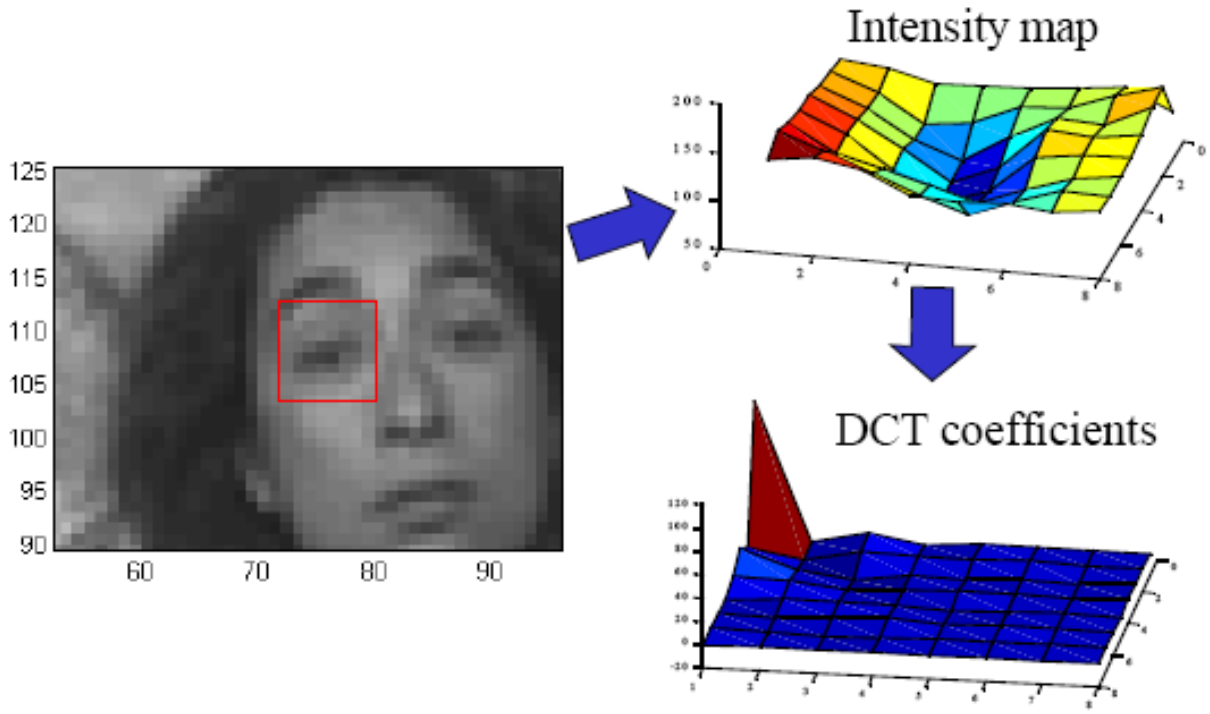


Figure 2.2: Example of DCT decorrelation and compaction features (reprinted from [102]).

(a) Keep all coefficients per block

(b) Keep 6 top-left coefficients per block

(c) Keep 3 top-left coefficient per block

(d) Keep 1 top-left coefficient per block

Figure 2.3: Effect of removing less-important coefficients per block on the input JPEG image.

luminance (Q_Y) and chrominance (Q_C) for JPEG compression to produce quantized DCT coefficients ($d'_{(i,j)}$). The entries of these quantization tables are customized based on the desired QF. Default quantization tables and the quantization process can be expressed in the following mathematical equations [127, 99, 32]:

$$q(Y)_{i,j} = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix} \tag{2.2}$$

$$q(C)_{i,j} = \begin{pmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{pmatrix} \tag{2.3}$$

$$q'_{i,j} = round(\frac{50 + S \times q_{i,j}}{100}) \tag{2.4}$$

$$d'_{i,j} = round(\frac{d_{i,j}}{q'_{i,j}}) \tag{2.5}$$

Here, $q(\cdot)_{i,j}$, where $0 \leq i \leq 7$ and $0 \leq j \leq 7$, represents the entries of the default quantization tables for luminance and chrominance, respectively. $q'_{i,j}$ are the actual quantization table entries after have been adjusted with the input QF and a parameter called S equals to $5000/QF$ if $QF < 50$ and $200 - 2 \times QF$ otherwise. These entries are used to produce the 64 quantized DCT coefficients, $d'_{i,j}$.

After quantization, these coefficients are scanned in zigzag manner depending on their frequency order. Last but not least, the differential coded DC and run-length coded AC coefficients will be passed to Huffman encoding. Huffman encoding encodes the image

based on its input statistics and produce a new JPEG image. To decompress the image and extract its YCbCr channels, JPEG decoder simply performs the reverse of the aforementioned steps.

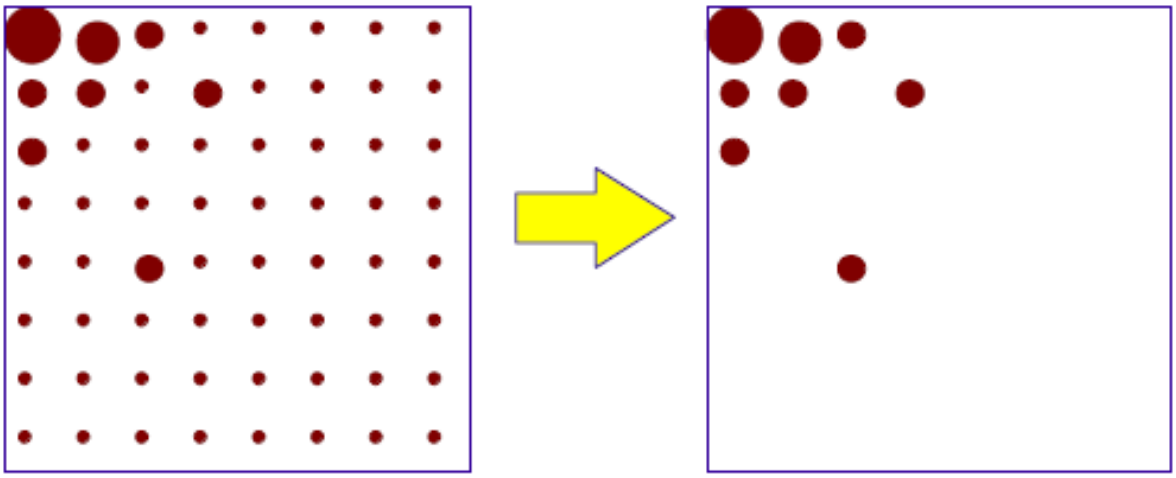


Figure 2.4: Effect of quantization on the transformed coefficients matrix (reprinted from [102]).

The most common coding process in JPEG is the Baseline method, which supports two modes of operation: sequential and progressive. The key difference between sequential and progressive modes is that JPEG encodes all the DCT coefficients of the current block in one scan, whereas progressive JPEG encodes similar-positioned batch of DCT coefficients of all blocks in one scan.

We wrote an optimized implementation of Baseline JPEG decoder and encoder that supports sequential and progressive modes of operation, which we make available publicly[1].

2.2 Overview on the HEVC Standard

Due to the cumulative experience of about four decades of research and three decades of international standardization for digital video coding technology from JPEG in 1980 until 2012, a new standard now known as High Efficiency Video Coding (HEVC) has emerged

[1] https://github.com/HossamAmer12/jpeg_customised_huffman

[119, 117, 27]. Collective efforts from two highly reputable international video coding standardization organizations, namely the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) defined the HEVC standard. Compared to its predecessor, H.264/AVC standard [101, 103], the High Efficiency Video Coding Standard (HEVC) standard improves video coding rate distortion (RD) performance significantly, especially for high resolution sequences.

Many experiments were carried out in the literature to prove the superiority of HEVC in coding efficiency over other codecs such as its predecessor, H.264/AVC and Google VP9 [93]. The VP9 video codec includes similar coding tools as in AVC and HEVC, yet it offers some alternative tools such as adaptive mixing strategies for artificial reference frames, processor adaptive real time encoding, or a low complexity loop filter [9]. Relative to H.264/AVC high profile and Google VP9 codec, HEVC can save approximately 43.3% and 39.3%, respectively, of bit rate at the same video reconstruction quality [51, 14].

Compared to JPEG, HEVC is a predictive video codec that inherited the block-based structure of JPEG, but added advanced coding tools such as variable block size, intra/inter prediction and in-loop deblocking filtering. In the next few sections, we will provide an overview on the structure of the HEVC encoder shown by Figure 2.5.

2.2.1 Block Structure in HEVC

HEVC is designed along the successful principle of block-based hybrid video coding. Following this principle, as depicted in Figure 2.5, a picture is first partitioned into non-overlapping blocks and then each block is predicted by using either intra-picture (I-picture) or inter-picture prediction (P-picture or B-picture). In most of the previous standards, the basic coding unit was the macroblock, which contains a 16x16 pixel area. Conversely, the analogous structure in HEVC is the coding true unit (CTU) [67]. The size of the CTU is typically set to 64x64 in order to boost the coding efficiency for high resolution videos. Each CTU can be split into multiple coding units (CUs) of different sizes. Luma CUs cover a picture area of 2Nx2N, where N can be any of 32 (depth=0), 16 (depth=1), 8 (depth=2), 4 (depth=3). Allowed CU sizes are determined based on the CU type: intra or inter. In the following section, we give an overview of these types.

2.2.2 Inter/Intra-Picture Prediction & Mode decision in HEVC

Inter and intra prediction methods are switched based on configuration and features of current frame. For example, intra (I) frames only allow intra prediction while inter (P or

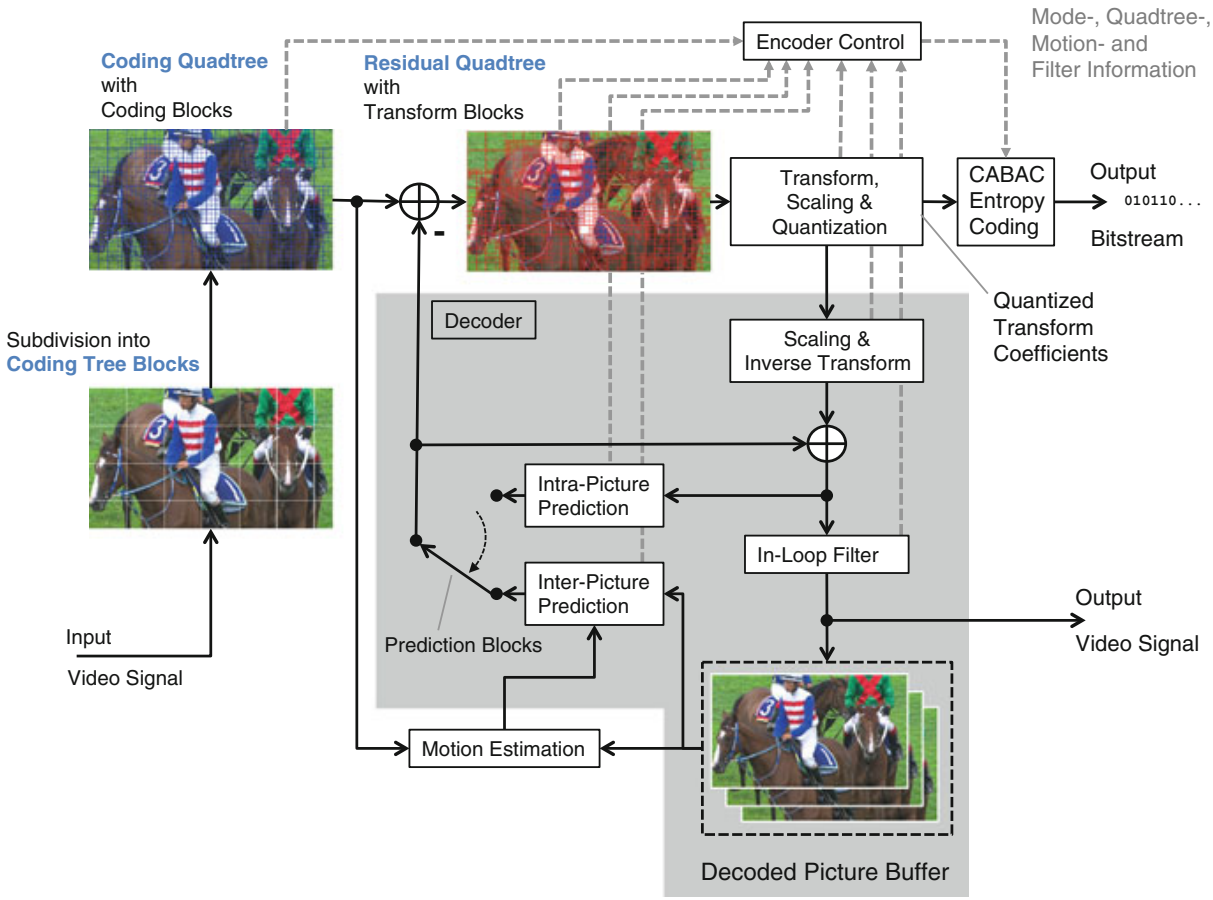


Figure 2.5: Generalized block diagram of the HEVC video encoder with motion compensation (reprinted from [119]).

B) frames allow both inter and intra prediction methods. Intra-picture prediction exploits the spatial redundancy within the same frame while inter-picture prediction (motion estimation) exploits the temporal redundancy by using the displaced blocks of already decoded pictures as a reference. For P frames, inter-prediction is done only from previously decoded frames, whereas B frames are predicted from previous and future frames in the temporal order. Typically, the prediction compensates for the motion of real-world objects between pictures of the given video sequence. The difference between the original block and its prediction, or the so-called residual signal, is transmitted using transform coding.

Based on whether the current CU is encoded as skip, inter or intra, HEVC supports different symmetric and asymmetric partitioning into prediction units (PUs), as shown

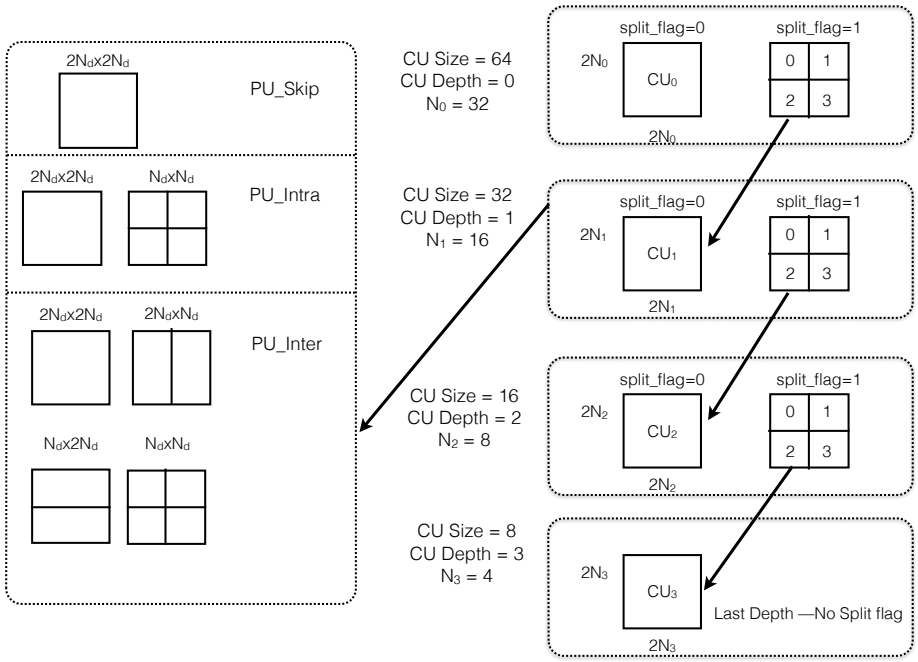


Figure 2.6: Quad-tree structure and supported modes of the CU in HEVC.

in Figure 2.6. Skip mode often happens when the residual energy between the current and predicted block is equal to zero. Given that 2N is the side length of the CU, any 2Nx2N can be encoded as skip, intra, or inter modes. Skip implies a size of 2Nx2N to the current CU. In the inter-prediction case, a CU can be further recursively split into four symmetrically sized CUs until the size of the CU becomes 8x8 pixels which is the smallest default size of the CU. In addition, if the side length of the CU (2N) is greater than 16, it is typically allowed to further split into any of these partitions: 2N/4×2N from either left or right, 2N/4×2N from either up or down [117]. In intra prediction, HEVC only supports splitting the current CU into either the 2Nx2N mode or four NxN sub-CUs. Typically, homogeneous areas are coded in larger blocks, whereas highly textured regions are coded in smaller blocks.

Quad-tree CTU partitioning structure and mode decision in HEVC is a depth-first process which starts at depth 0 with the CU size equal to 64x64 pixels until it reaches a maximum depth set to three by default with CU size equal to 8x8 pixels. At every split, a comprehensive rate-distortion optimization (RDO) is done for every CU. The CU partitioning structure is decided after comparing and selecting the best RD cost among the CU itself and its four sub-CUs. This RD cost also depends on the prediction mode of

the current CU and its four sub-CUs. Take intra coding as an example. In addition to the variable CU structure, each CU is further predicted from its neighboring reconstructed pixel samples with up to 35 different Intra Prediction Modes (IPMs). Figure 2.7 illustrates the recursive process to obtain the best CU structure for each CTU in HEVC's intra coding, where a check RD cost function is continuously called to get the best CU structure in terms of IPM and CU splitting structure. We first start with a CU size set to 64x64 and check the RD costs if we encode the CU as is. As long as we did not reach the maximum depth or the minimum CU size, we recursively split the parent CU into four sub CUs; otherwise, we compute the RD cost of each sub CU and backtrack the recursion to get the best CU structure in terms of RD costs. The RD cost of the quadtree partitioning, J_{RDO}, is given by:

$$J_{RDO} = SSE + \lambda * R_{Total} \tag{2.6}$$

where SSE is the sum of squared errors between the original image $I(x, y)$ and reconstructed image $I'(x, y)$, R_{Total} is the total number of bits to encode the current CU, and λ is the Lagrangian multiplier that decides the trade-off between the rate and distortion based on a selected quantization parameter (QP). SSE is defined as follows:

$$SSE = \sum_{x,y} |I(x, y) - I'(x, y)|^2 \tag{2.7}$$

In HEVC's intra coding, the RDO process also includes the calculation of the rate-distortion (RD) cost for each of 35 IPMs and the determination of the best IPM for each CU. These 35 IPMs include 33 angular modes, 1 DC mode, and 1 planar mode. To reduce the IPM candidates of HEVC and in turn slightly reduce its complexity, HEVC adopts a Rough Mode decision (RMD) based on an approximated RD-cost, J_{RMD} [123], and it is given by:

$$J_{RMD} = SATD + \lambda * R_{Est} \tag{2.8}$$

where $SATD$ is the absolute Summation of Hadamard Transformed Coefficients (SATD) of residual data, and R_{Est} is an estimated rate cost. Table 2.1 shows the steps for the RMD algorithm, which consists of three main stages. The first stage encompasses applying the 35 IPMs to the current CU, calculating the $SATD$ for the 35 intra modes available in HEVC, and selecting N modes with minimum $SATD$ for further RDO. At the second stage, these selected modes are appended to a candidate list of MPMs, which already includes three

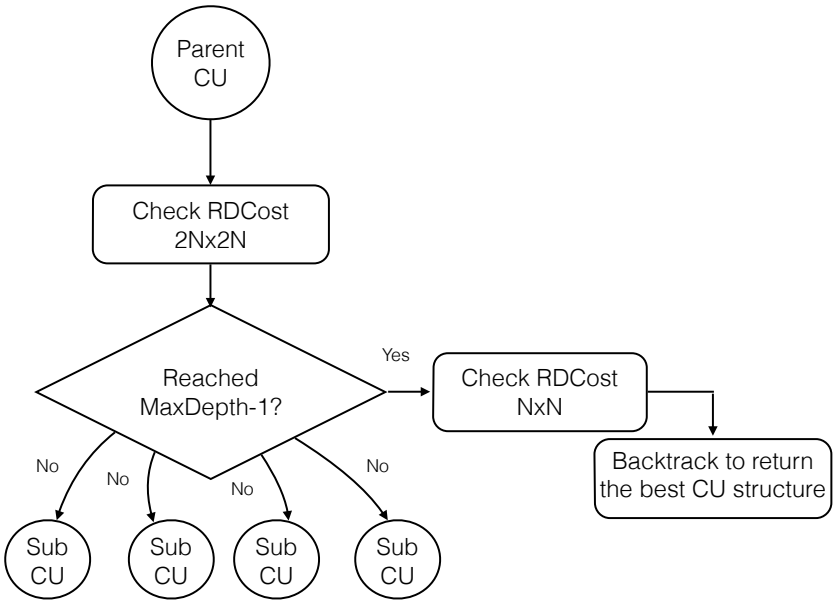


Figure 2.7: HEVC Recursive Process for Quad-Tree Intra CTU Partitioning.

MPMs based on the best MPM of the left and top neighbouring CUs. Algorithm 1 shows the method to obtain the indexes of these three MPMs. Table 2.2 indicates the indexes of the IPM modes, their corresponding IPM mode type, and target content. The third stage encompasses taking this candidate list to apply complete RDO and selecting the best IPM. According to [123], values of N are set to 1, 2, 2, and 4 for 64x64, 32x32, 16x16, 8x8, and 4x4 CUs, respectively.

Figure 2.8 illustrates the intra/inter mode decision as well as the quad-tree CU structure in HEVC under the uni-directional inter prediction for frames 139 and 140 of Kimono video sequence. It can be seen from this figure that areas with strong temporal correlation (green) utilize inter prediction, and areas with weak temporal correlation and strong spatial correlation (blue) utilize intra prediction. The majority of frame 140 in Kimono video sequence was encoded with intra prediction because it is a scene change (SC) and there is no correlation between this frame and its past frames in the encoding order. To illustrate the meaning of residuals in HEVC, Figure 2.9 shows Frames 139, 140, and 141 in the Kimono video sequence and their corresponding residual frames generated from HEVC. Due to the sudden change in temporal correlation, frame 140 has a relatively higher residual energy compared to frame 139 and frame 141.

Owing to the variable prediction block size in HEVC, it can achieve noticeable coding

Table 2.1: Rough Mode Decision in HEVC Intra coding.

	Steps
Stage One	Apply 35 IPMs and generate CU data
	Calculate estimated cost of for 35 IPMs
	Select N IPMs with minimum costs and append them to the candidate set
Stage Two	Add MPM in the candidate set
Stage Three	Calculate RD cost for IPMs in the candidate set
	Return the best IPM

Algorithm 1 Pseudo code to return the index of the three MPMs in HEVC Intra coding. A: the best IPM in the left CU neighbour, B: the best IPM in the right CU neighbour.

```
1:  procedure getMPMs
2:      Input: A, B
3:      Output: MPM[0], MPM[1], MPM[2]
4:      if A == B
5:          MPM[0] = A
6:          MPM[1] = 2 + ((A − 2 − 1 + 32)%32)
7:          MPM[2] = 2 + ((A − 2 − 1 + 32)%32)
8:      else
9:          MPM[0] = A
10:         MPM[1] = B
11:         if MPM[0] ≠ 1 & MPM[1] ≠ 1
12:             MPM[2] = 1
13:         else
14:             if MPM[0] ≠ 0 & MPM[1] ≠ 0
15:                 MPM[2] = 0
16:             else
17:                 MPM[2] = 26
18:     return MPM[0], MPM[1], MPM[2]
```

Table 2.2: Indexes of intra prediction modes.

Mode Index	Mode Type	Target Content
0	DC	Homogeneous
1	Planar	Gradually Changing
2-34	Angular	Directional

efficiency improvements. For example, [119] conducted a coding efficiency comparison between HEVC with its different block sizes (configuration A) and HEVC with a restricted set of only {16x16, 8x8} prediction block sizes (configuration B) relative to a configuration with only 16x16 prediction block sizes. The coding efficiency improvement of configuration A can go up to 30.3% vs 4.4% for configuration B. These coding efficiency improvements show the importance of more flexible prediction block partitioning in HEVC. However, these benefits come at the expense of significant increase in time complexity. Indeed, it is reported that HEVC encoding is 3.2 times more complex than its predecessor, H.264/AVC encoding [126].

After prediction, each frame can be further divided into transform units (TUs) to start the transform coding process.

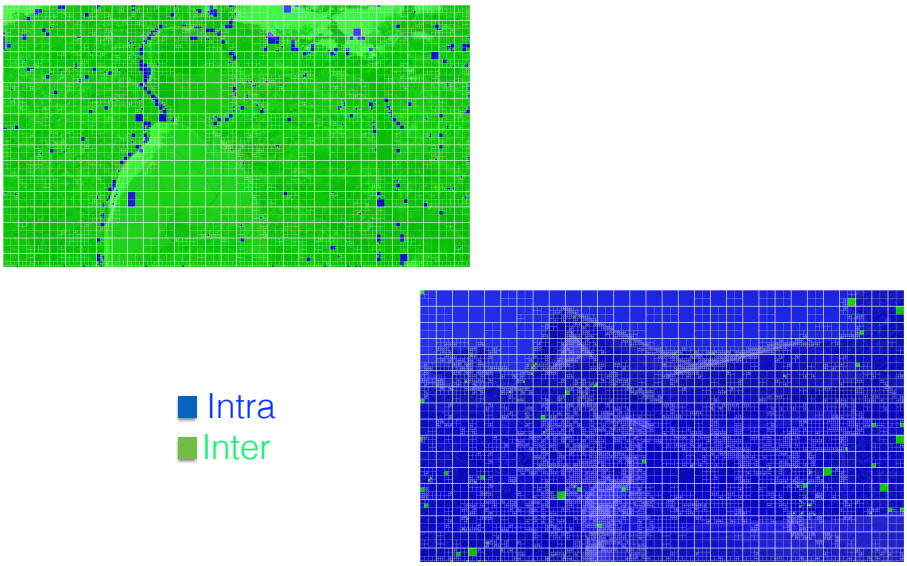
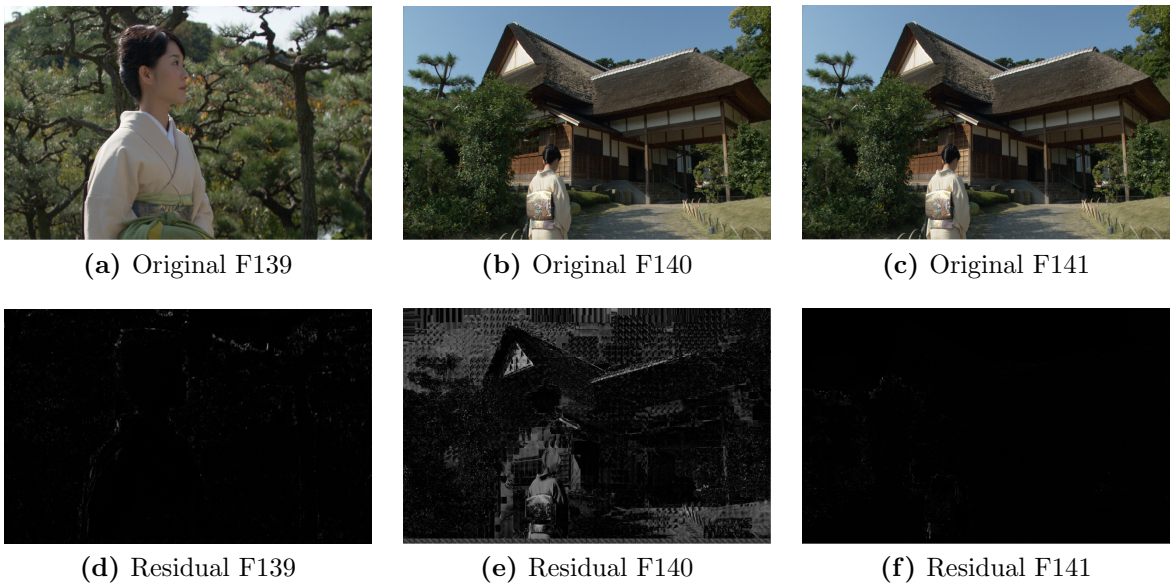


Figure 2.8: HEVC Encoder Mode Decision using uni-directional Inter-prediction for Frame 139 and 140 in the Kimono Video Sequence.



(a) Original F139 **(b)** Original F140 **(c)** Original F141

(d) Residual F139 **(e)** Residual F140 **(f)** Residual F141

Figure 2.9: Original and corresponding residual frames for picture order counts 139, 140 and 141 of Kimono Video Sequence, where in residual frames, black and white indicate low and high levels of residual energy, respectively.

2.2.3 HEVC Transform and Quantization

Transform coding consists of a decorrelating linear transformation, scalar quantization of the transform coefficients and entropy coding of the resulting transform coefficient levels. In HEVC, discrete cosine transform (DCT) is employed for all possible sizes of TUs, namely 4×4, 8×8, 16×16, and 32×32. It also uses discrete sine transform (DST) which is only applied on 4×4 TUs. Similar to the JPEG standard, the DCT in HEVC shares similar characteristics with the DCT in JPEG explained in Subsection 2.1. Yet, one of the major differences is the variable transform block size. After residuals are obtained via prediction, a residual quad tree is built and the TU size is also decided based on rate-distortion optimization process. With respect to fixed 4x4 TU size on high resolution sequences, HEVC with maximum TU size of 8x8 can achieve coding efficiency improvements up to 8.5%. When the maximum TU size is set to 16x16, the coding efficiency improvements increase to 14.7% and further increase to 17.5% when the maximum TU size is set to 32x32 [119].

In HEVC, quantization is done by dividing each element in the transformed coefficients matrix by the corresponding element in pre-determined quantization matrix, which results in having a highly sparse matrix of non-zero coefficients ready for entropy coding. This quantization strategy is similar to JPEG, but HEVC uses different quantization tables for different size and types of the transform block [119]. These quantization tables are customized based on a parameter called quantization parameter (QP) that controls the trade-off between rate and distortion.

2.2.4 HEVC Entropy Coding

After exploiting the temporal and spatial redundancies, the remaining data are highly correlated (e.g long runs of zero quantized transform coefficients), and can be further compressed using some statistical lossless methods, or, entropy coding. Entropy is a mathematical measure of the amount of information contained in a series of numbers of systems [94]. In addition, entropy is used as a measure of image (or a frame) compressibility. Entropy coding is a lossless compression scheme that uses the statistical properties to compress data such that the number of bits used to represent the data is logarithmically proportional to the probability of the data. For example, if the data being compressed is a string of characters, the frequently used characters are each represented by a few bits, while infrequently used characters are each represented by many bits. HEVC entropy coding scheme is called context adaptive binary arithmetic coding (CABAC) [92]. In entropy coding, the quantized coefficients are coded in groups of 16 coefficients for each TU no

matter what size of the TU is. A group of TU coefficients is called coefficients group (CG). Scan order of each CG may vary unlike H.264/AVC which uses only zigzag scan for every 4x4 coefficients. The encoded coefficients information, such as levels and coefficient bits, uses a scan index from 0 to 15 in each CG.

2.2.5 In-Loop Filters in HEVC

Due to lossy compression, the existence of visible artifacts, or the so-called checker board effect at block boundaries in reconstructed video sequences are unavoidable in lossy compression engines such as HEVC. Independent block-based inter and intra prediction coding of the blocks is essentially the main reason for these discontinuities. To get rid of these artifacts and obtain better reconstruction, a deblocking filter and sample adaptive (SAO) filter are employed in the HEVC encoder. SAO is essentially used to reduce the mean sample distortion of a region by first classifying the region samples into multiple categories. For each category, an offset is derived and added to each sample in that category.

2.2.6 HEVC Configurations

HEVC defines three configurations: All Intra (AI), Low Delay (LD) and Random Access (RA) configurations. HEVC utilizes only intra frame coding under the AI configurations. As for the LD configuration, HEVC enforces the first frame of any given video sequence to be an I frame while the rest are P or B frames. RA, however, has a relatively frequent intra-frame refresh (every 32 frames) with B frames in between. Frames in each configuration are encoded via pre-defined QP settings based on the configuration type to reduce the inherent complexity of HEVC.

2.3 Classification and Neural Networks

Classification is a supervised machine learning task whereby the computer program is asked to identify for each input its corresponding correct category out of k categories. More formally, suppose that we are given n observations. Each observation consist of a pair: a vector $x_i \subset \mathbb{R}^d$, $i = 1, 2, \cdots, n$, and the associated label y_i. Here $\mathbf{x}_i = (x_{i1}, x_{i2}, ...x_{id}) \in \mathcal{X} \subset \mathbb{R}^d$, and the associated label Y_i belongs to some finite set $\mathcal{Y}$ [49]. A machine learning learning algorithm such as neural networks should learn the label y_i of each input from its

corresponding features x_i using a certain predefined loss function, where this loss should decrease by experience over time.

Inspired by the biological brain, neural networks (NNs) were born in 1988 for computer vision and were named as the "Optimal Brain Damage" regularization methods [75]. NNs, especially deep neural networks (DNNs), have become the de facto approach in the classification task under the image and video domain due to their steady classification accuracy improvement from 74.2% to 95.2% on large-scale datasets [73, 44, 124, 88, 35, 132, 68, 91]. For example, DNNs have been used to speed up the HEVC encoding mode and partition decisions at the minimum coding efficiency loss [44, 124, 88, 35, 132, 16, 91]. Also, DNNs have been used to enhance the quality of the reconstructed video at the decoder end [35, 137]. Further, neural networks have recently been utilized to replace some or all basic components of the image/video compression for better coding efficiency [124, 132, 104]. Not to mention, DNNs have been widely used in image classification and recognition tasks. All in all, it is worth noting that DNN machines have positively contributed in several fronts of image/video compression and understanding.

DNNs or specifically convolutional neural networks (CNNs) are known to be universal approximators because of their ability in extracting the required features for classification from the raw pixels of images [49]. To extract these features, CNNs encompass one or more convolutional layers [49] with several fully connected layers at their tail, as indicated in Figure 2.10. These layers include parameters of non-linear activation functions, which are learnt using a backpropagation learning algorithm. These functions progressively transform raw pixels of the input image to produce the output predicted label. These non-linear functions provide multi-layer representations to the input image, where each representation is a feature that amplifies certain aspects of the raw pixels required for classification and suppress irrelevant information. Typically, the first few representations project primitive features such as existence of edges, their orientations, and their arrangements. Subsequent layers combine representations from previous layers to classify familiar objects [73]. As such, CNNs are machines that "see" images as group of pixels and extrapolate relationships among these pixels to finally reach a decision, which is hopefully be the ground truth label decision in the supervised learning context.

2.3.1 AlexNet Architecture

Figure 2.11 illustrates the AlexNet architecture, one of the early CNN architectures in the domain of image classification [70]. As seen by the figure, an RGB image is typically the input stimuli to the architecture with pixel intensties range from 0 to 255. A series of

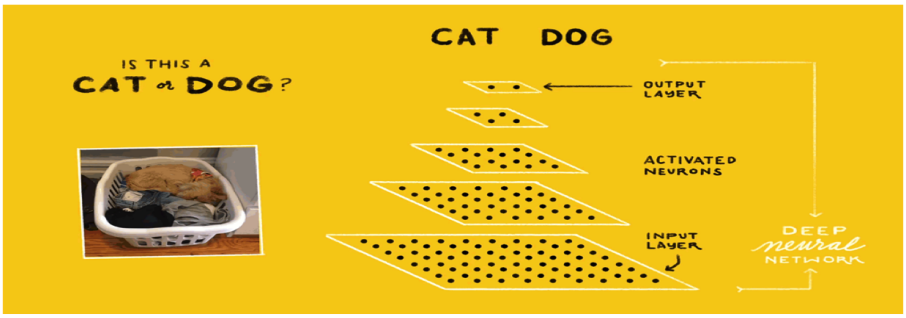


Figure 2.10: Convolutional Neural Network (CNN) For Image Classification Schematic Diagram.

convolution and max pooling layers with variable number of kernels of variable dimensions are applied to the input image to extract the required features for classification. These convolutional layers include non-linear activation functions, while max pooling layers spatially subsample the output from convolutional layers i.e, feature maps. Last but not least, the output from the last max pooling layer is connected to fully connected layer. The name "fully connected" originated from having each neuron from the last max pooling layer connected to the output neurons (the prediction). The fully connected layer typically utilizes matrix multiplication, matrix flattening to yield a 1D probability vector indicating the probability of each class. As shown in Figure 2.11, the number of classes for AlexNet is 1000.

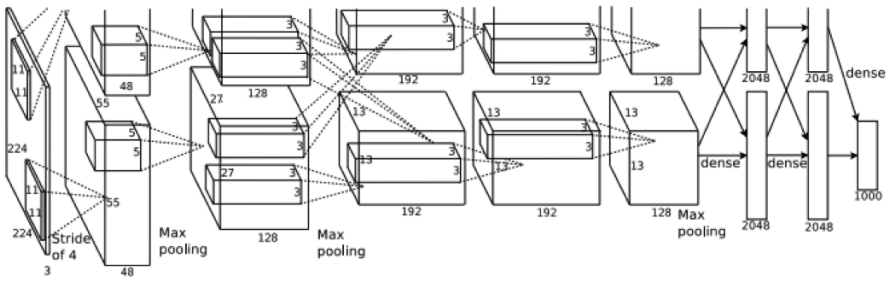


Figure 2.11: AlexNet Convolutional Neural Network Architecture.

2.3.2 Inception and Residual Architectures

Based primarily on AlexNet and many others, inception and residual types of architectures were introduced and led to major advances in the accuracy of deep neural network in the task of image classification [122, 120, 56, 57, 121]. Both types of architectures were developed through continuous efforts that started in 2014. For inception, its first version, Inception V1, appeared with lower computational complexity than its predecessors because it was constructed using multiple parallel inception modules that can capture several features from input images. These modules contain convolutions, pooling, and non-linear activation functions such as ReLU. Inception V2 and V3 were later created and introduced new strategies of constructing the model and training such as convolutional factorization, auxiliary classifier, batch normalization, and label smoothing [120]. Turning to residual type of architectures, the accuracy degradation phenomenon of deeper networks compared to shallower networks was observed due to vanishing gradients. As a result, residual networks called ResNet V1 and ResNet V2 added identity mappings to transfer the output of the former layer directly to the latter layer(s). The main difference between ResNet V1 and V2 lies in the underlying structure of the residual unit, which improved the accuracy of ResNet V1.

In this thesis, we consider Inception V3 and ResNet-50 V2 architectures pre-trained for the machine vision experiments [5, 7]. Despite the widespread of DNN architectures for image classification, Inception V3 and ResNet-50 V2 architectures are common representatives of inception and residual type of learning. On one hand, Inception V3 contains 42 layers and 24M parameters. Inception V3 starts off by decoding the input JPEG image and resizing it to $299 \times 299 \times 3$ using bilinear interpolation. A series of convolutions, 10 inception blocks, a pooling operation, softmax activation function are applied to the resized image in order to produce a sorted probability vector for the predicted labels. This probability vector is sorted based on the confidence of the predicted labels. On the other hand, ResNet-50 V2 is 50 layers deep and contains 25M parameters. ResNet-50 V2 resizes the input image to $299 \times 299 \times 3$ for which it applies convolution, pooling, and then four ResNet blocks. Each block contains the necessary residual units to finally produce a sorted probability vector. This vector contains an ordered list of predicted labels.

2.3.3 ImageNet Dataset

ImageNet dataset is our input stimuli for all of our machine vision experiments and one of the keys to improving DNNs in the task of image classification [36, 70, 65, 28, 105, 114, 122, 120, 56, 57]. Images of this dataset were collected from several search engines

and labeled by a majority vote from human labers using Amazon's Mechanical Turk crowsourcing tool. Using this dataset, an annual competition called the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) was held. Although this challenge for image classification was stopped in 2017, this dataset is considered as a standard benchmark for image classification [65, 28, 114, 122, 120, 56, 57]. This dataset consists of 1.3 million training images, 50,000 validation images, and 150,000 testing images. All of these images are split into 1000 distinct classes. The vast majority of these images have high QF with a relatively-high average resolution of 469x387 pixels.

2.4 Chapter Summary

In this chapter, we covered the core concepts of compression and neural network machines to easily layout our thesis contributions in the human vision as well as machine vision perspectives. As discussed in this chapter, to reduce the complexity of RDO in HEVC, frames are encoded via pre-defined QP settings, which does not fully consider the inter-frame dependency. These settings bounds the potential improvements in terms of the trade-off between compression rate and compression distortion for humans. In the next chapter, we propose an adaptive frame-level QP selection algorithm for the hierarchical coding structure in LD HEVC by characterizing the coding propagation effect through a notion called temporal propagation length.

Chapter 3

Adaptive Quantization Parameter Selection for Low-Delay HEVC via Temporal Propagation Length Estimation

This chapter proposes an adaptive frame-level QP selection algorithm for the hierarchical coding structure in LD HEVC by characterizing the coding propagation effect and integrating it into rate-distortion optimization (RDO) through a notion called temporal propagation length. Section 3.1 reviews the related work, Section 3.2 formulates the QP selection problem for the LD HCS in HEVC, and Section 3.3 investigates the inter-frame dependencies in LD HEVC and derives its accumulated propagation effects. Our proposed adaptive QP selection algorithms along with their initialization settings are presented in Section 3.5. The experimental settings, results and analysis are illustrated in Section 3.7. Finally, Section 3.8 concludes the chapter.

3.1 Literature Review

According to Cisco, video traffic is forecasted to increase by four-fold from 2017 to 2022 [1]. To cope with this increasing demand, the state-of-the-art video coding standard, High Efficiency Video Coding (HEVC), introduces an average of 50% improvements to the coding efficiency of its predecessors [126, 119, 117]. In HEVC, an important key to

these improvements is the adoption of many advanced coding techniques such as quad-tree block partitioning [67], more intra prediction modes [71], merge mode [58], advanced motion vector prediction (AMVP) [119], etc. These advanced techniques have many coding parameters, such as coding modes and quantization parameters (QPs), that require optimization in the midst of the encoding process. Thus, an encoding procedure known as RDO is pivotal to select these parameters in order to minimize the coding distortion subject to a given coding rate [116, 96, 135].

To alleviate HEVC's coding complexity, RDO is typically performed on each basic unit by identifying the best coding mode given predefined QP values. This scheme is employed in the random access (RA) and low-delay (LD) hierarchical coding structures (HCS) of the HEVC test model, HM [4], in which a basic unit can be a frame, a slice, or a coding unit (CU). Adopting this RDO scheme, however, is detrimental to the coding performance. For instance, low-delay (LD) HEVC typically encodes P/B frames that are strongly connected to the coding efficiency of all their past reference frames, which leads to inter-frame dependency. This dependency results in a coding propagation effect, in which the coding of the current frame can impact the coding of its future frames in the temporal chain. Running RDO with a predefined QP for the current frame ignores the coding propagation effect to some extent, leading to a coding performance plateau. To further improve the coding performance, one way is to integrate the coding propagation effect into RDO to select a QP for each frame adaptively, which is the purpose of this chapter.

Reviewing the literature, a number of independent RDO techniques did not consider the coding propagation effect aiming at relatively simpler implementations of video encoders. Approaches in [79, 78, 142] are instances of the independent RDO. Specifically, RDO schemes in [79, 78] presented a retrained relationship that can compute a QP value for any given λ and can be utilized for any basic unit. The method in [78] was dubbed as the QP refinement (QPR) method and provided one retrained relationship based on all coding configurations of HEVC. Other methods as in [142] leveraged some perceptual features, such as the spatial energy ratio and the temporal motion activity, to adaptively select λ. All these methods did not consider the temporal dependency among basic units, which in turn bounds the potential impact on the coding efficiency.

On the contrary, there are some works in the literature that proposed dependent RDO schemes to incorporate the coding propagation effect among blocks or frames into the encoding process of HEVC's predecessors aiming at a better coding efficiency [86, 53, 85, 63, 138]. For example, a trellis-based dynamic programming technique was implemented for bit allocation in H.263+ [33], in which the authors derived the best combination of QP values for each frame by minimizing the RD cost [86]. This dynamic programming

method aggravates the computational complexity due to the required multi-pass encoding process. Also, a graph based approach was developed in [135, 53] to jointly optimize the motion estimation, quantization, and entropy coding in HEVC's predecessor, H.264/AVC encoder [131]; hence, the optimal coding performance can be achieved by considering the coding dependencies among different macroblocks (MBs) within a frame. Further, an RDO scheme was designed in [138] under IPPP/IBBB coding structure in H.264. This RDO scheme was based on a source distortion propagation model to account for the temporal propagation effect of the current MB on its future MBs in the temporal propagation chain. To pre-estimate the reconstructed distortion of future MBs and in turn estimate the coding dependency among MBs in [138], a well-trained look-up table was constructed with different MB content and quantization step sizes. Method in [138] successfully improved the coding efficiency of H.264. However, relative to HEVC, the referencing structure in H.264 is simpler because in H.264 frames only affect one following frame in encoding order and affect other frames indirectly through this following frame. On the other hand, multiple reference frames in the hierarchical structure of LD HEVC can affect the to-be-coded frame leading to dispersed temporal relationship. Hence, the temporal propagation problem in LD HEVC has more aspects that make the problem settings different.

Along the same line, other dependent RDO methods incorporated the coding propagation effect to improve the coding efficiency of RA HEVC [46, 54]. For example, a temporally dependent RDO scheme was proposed in [46] based on the temporal relationship among layers within RA HEVC. In [46], they modeled the uni-prediction and bi-prediction of RA HEVC and enhanced its coding efficiency. Furthermore, authors in [54] considered the inter-dependency of RA HEVC to obtain adaptive QP values. The main contribution of [54] is a linear distortion model that represents the distortion of a predicted frame as a function of its reference frame, in which they successfully improved the coding efficiency of RA HEVC. Unlike RA HEVC, all frames in LD HEVC are used as references, i.e., multiple frames may be directly affected by the to-be-coded frame, creating a dispersed temporal chain. On the other hand, RA HEVC always has a periodic I-frame segregating this temporal chain into independent groups. In addition, some frames in RA HEVC are never used as references. Based on these characteristics of RA HEVC, algorithms in [54] and [46] operate on every distinct group of frames independently. As alluded in the future work of [54], therefore, the straightforward application of adaptive QP algorithms for RA HEVC to LD HEVC is somewhat limited due to the inherent differences in terms of coding structure between LD HEVC and RA HEVC.

To serve the LD HEVC configuration, some RDO methods in the literature were proposed to take into consideration the temporal dependency of LD HEVC [81, 45, 144]. In [81], a temporally dependent RDO scheme adapted the Lagrangian multiplier on LD

HEVC. This scheme relies on the high-rate distortion approximation and in turn overcomes the issue of distortion propagation. As a result, the dependent RDO formulation in [81] was simplified by only modelling the inter-frame rate relationship. Also, the Lagrangian multiplier was further regularized to ensure that the high-rate approximation assumption still holds. In [45], a temporally dependent RDO method extended the source distortion temporal propagation model in [138] to account for the temporal dependency among CUs in the hierarchical structure of LD HEVC. Similar to [138], the RDO algorithm in [45] required the re-construction of a look-up table to store the reconstructed distortion of future CUs under the LD HEVC configuration. The RDO scheme in [45] showed success in improving the coding efficiency of LD HEVC, but the interoperability of this method may have some limitation because it requires the storage and construction of a well-trained look-up-table that accounts for different types of CU-based content and video encoders [138, 45]. In [144], the characteristics of a video sequence, i.e., video content textures, motion, and inter-layer dependencies were integrated into an adaptive quantization parameter cascading (QPC) scheme. Nonetheless, the inter-layer dependency is not video sequence adaptive. Furthermore, the so-called CU tree was adopted in the X.265 open source codec, in which it enables the use of look-aheads low-res motion vector fields to determine the amount of reuse of each block to tune adaptive quantization factors [11]. Due to the need of look-aheads, this implementation may not be very well-suited for low-delay communication.

Enable the use of lookahead's low-res motion vector fields to determine the amount of reuse of each block to tune adaptive quantization factors. CU blocks which are heavily reused as motion reference for later frames are given a lower QP (more bits) while CU blocks which are quickly changed and are not referenced are given less bits. This tends to improve detail in the backgrounds of video with less detail in areas of high motion. Default enabled

Looking at all the aforementioned dependent methods in both HEVC and its predecessors, we can conclude that each method successfully made progress towards solving the temporal propagation problem in LD HEVC. However, these methods did not fully consider all aspects. Therefore, we believe that the frame-level temporal propagation problem in LD HEVC still has room for improvement.

To take some steps towards this improvement, we propose in this chapter an adaptive frame-level QP selection algorithm for the hierarchical coding structure in LD HEVC by characterizing the coding propagation effect through a notion called temporal propagation length. First, after reviewing the linear distortion model of [54] between a coding frame and its reference frame that is still valid in LD HEVC, we use it to help characterize the inter-frame coding dependencies in LD HEVC by taking the accumulated coding propagation

effects into consideration. Second, we introduce the notion of propagation length, which is defined as the impact length of the current frame on its future frames. Estimation of this length is done via offline experiments. With the given propagation length, we then propose two novel methods for predicting the impact of the current frame's coding distortion on future frames based on previous frames of similar coding characteristics. The first method makes a prediction in a group-of-frames manner based on a group of frames, while the second one makes a prediction individually based on each frame on its own. Third, each of these two methods is applied to adaptively determine Lagrangian multiplier and its corresponding QP for each frame in the LD configuration of HEVC. Experimental results prove the effectiveness of our RDO schemes in terms of BD-rate results: (1) the RDO scheme based on the first prediction method can outperform the HM-16.0 by -5.0% and -4.9% in low-delay-P (LDP) and low-delay-B (LDB) configurations, respectively, and (2) the second prediction method can outperform the HM-16.0 by 4.9% and -4.9% in the LDP and LDB configurations, respectively. These improvements are achieved with an insignificant 1% increase in encoding time as well as insignificant and consistent increase in terms of quality fluctuation of the coded video in the majority of content types compared to HM-16.0. It is worth noting that our experiments also show the effectiveness of our methods on low-motion sequences with BD-rate savings that can go up to 12.9%, which piqued serious interest from industry, such as Google.

3.2 Adaptive QP Selection Problem For Low-Delay HEVC

HEVC encompasses two LD coding configurations, namely LDP and LDB. The first frame in both LDP and LDB is an I-frame. For LDP, the rest are P frames, whereas they are all B-frames for LDB. In these configurations, every group of frames form a predefined coding structure called a Group-of-Picture (GOP) [108, 6]. Each GOP contains a fixed number of frames (four by default). These frames follow a pre-defined content-independent pattern of relatively small and large QP values that correspond to the level of involvement in prediction according to the referencing structure. In this section, we review the LD HEVC coding structure and present the formulation of the adaptive QP selection problem.

3.2.1 Low-Delay Coding Structure in HEVC

For real-time video applications, HEVC adopts LDP and LDB configurations because they employ forward predictions in order to limit the coding latency. Figure 3.1 illustrates the

LD coding structure in HEVC, in which the numbers denote the picture order count (POC) of each frame according to display order. Arrows in this figure represent the prediction relationship among these frames and their temporal levels (from $L0$ to $L3$). In figure 3.1, we refer to frames in one GOP by the relative POC (rPOC) from 1 to 4 as shown in Table 3.1. Frames sharing the same rPOC, for example POC=1 and POC=5, have the same QP pattern and referencing structure. Furthermore, QP offsets define the values added to the QP of the I-frame to calculate QP of the current rPOC. For example, frames at rPOC=4 always have a relatively lower QP offset than the rest of the frames because they are key frames. A key frame is the most frequently used reference frame; assigning a low QP offset to it results in high reconstruction quality, which in turn provides better referencing for future frames. Looking at Table 3.1, we can see that multiple reference frames are employed in LD HEVC. Any given current frame has four references represented by Ref_i that defines the difference between the POC of the reference frame and that of the current frame. Specifically, the referencing structure for each rPOC always contains the immediately previous frame and three previous frames that are utilized in the inter-prediction process. The coding distortion of each reference frame can indirectly or directly affect the coding distortion of future frames leading to a coding propagation effect.

According to RDO, each frame chooses an optimal reference frame from its set of references for each prediction unit (PU). Such a prediction scheme disperses the temporal dependency relationship. However, the work in [81, 45] showed that the immediately preceding frame is the most frequently used reference in the inter-prediction process of LD HEVC; thus, this frame should have the highest impact on the coding distortion. For this reason, we consider only the nearest reference frame in the following analysis as shown in Fig. 3.1. However, all our experimental results are carried out on the default LD HEVC reference settings.

Table 3.1: QP Pattern and Referencing Structure Under LD HEVC.

rPOC	Ref_1	Ref_2	Ref_3	Ref_4	QP Offset
1	-1	-5	-9	-13	+3
2	-1	-2	-6	-10	+2
3	-1	-3	-7	-11	+3
4	-1	-4	-8	-12	+1

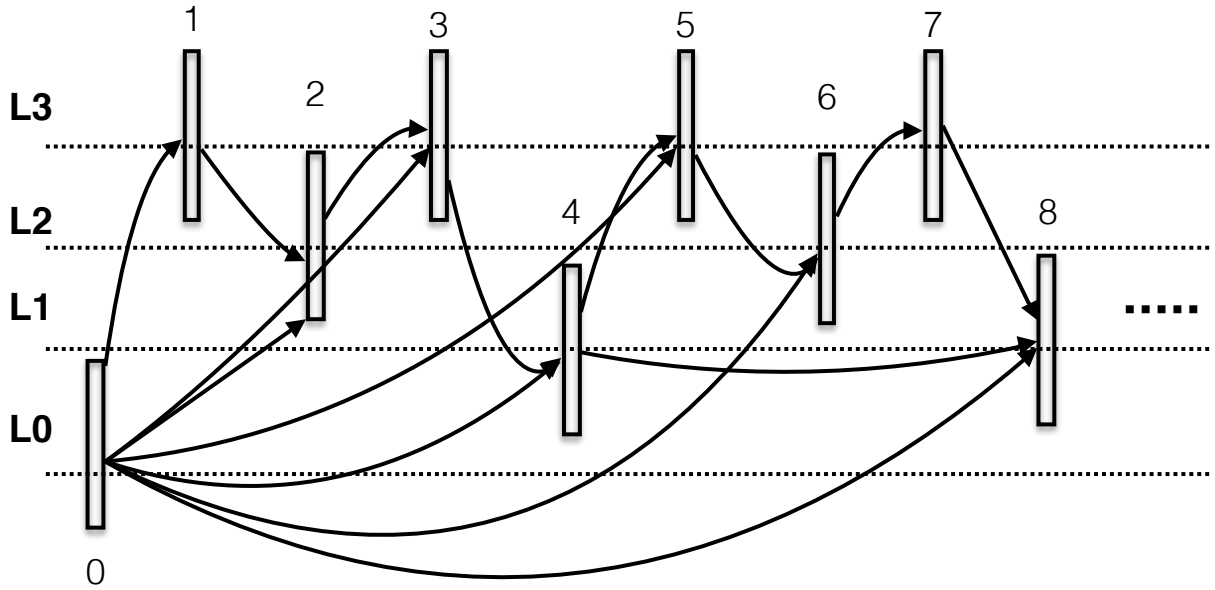


Figure 3.1: HEVC LD Coding Structure.

3.2.2 Problem Formulation of the QP Selection in LD HEVC

Taking the coding propagation effect into account, the QP selection problem in LD HEVC can be described as an RDO problem of minimizing the total coding distortion D across multiple frames subject to a target bit budget R^T for all frames as follows [119]:

$$\begin{aligned} \boldsymbol{Q^*} &= (Q_0^*, Q_1^*, Q_2^*, \cdots, Q_N^*) \\ &= \arg\min_{\boldsymbol{Q}} \sum_{i=0}^{N} D_i \quad s.t \quad \sum_{i=0}^{N} R_i \leq R^T \end{aligned} \tag{3.1}$$

where $N+1$ is the total number of frames, and D_i and R_i are the coding distortion and the number of coding bits of the ith frame, respectively. Both D_i and R_i depend on $\boldsymbol{Q} = (Q_0, Q_1, \cdots, Q_i)$ with Q_i representing the QP for the ith frame while the set of optimal QP values is $\boldsymbol{Q^*} = (Q_0^*, Q_1^*, \cdots, Q_N^*)$. To solve the constrained optimization problem above, the Lagrangian multiplier method [64] is employed to convert Equation (3.1) into an unconstrained form as follows:

$$
\begin{aligned}
\boldsymbol{Q}^{*} &= (Q_0^{*}, Q_1^{*} \cdots Q_N^{*}) \\
&= \arg\min_{\boldsymbol{Q}} J, \\
\text{where } \; J &= \sum_{i=0}^{N} D_i + \lambda \sum_{i=0}^{N} R_i
\end{aligned} \tag{3.2}
$$

In Equation (3.2), we define J as the total rate-distortion (RD) cost function with λ being the global Lagrangian multiplier that represents the trade-off between the coding distortion and the number of coding bits. A straightforward solution for the present problem is to exhaustively search for all possible QP combinations for all frames and select the best combination producing the minimum RD cost. Such a solution is clearly impractical due to its expensive computational requirements. These requirements become even more challenging in LD HEVC, because encoding any frame i in LD HEVC impacts all frames in the future leading to a coding propagation effect. To circumvent this challenge, we analyze the coding propagation effect of LD HEVC in the following sections to solve this optimization problem.

3.3 Accumulated Coding Propagation Effect in Low-Delay HEVC

This section continues building the adaptive QP problem formulation. In Subsection 3.3.1, we review the linear distortion model derived from [54] that is still applicable in LD HEVC. In Subsection 3.3.2, we utilize this model to develop the problem formulation of LD HEVC, which leads to considering the accumulated propagation effects.

3.3.1 Review for the Linear Distortion Model

In the inter-prediction process, motion compensation is executed to produce the residual signal between the original and reconstructed frames. Afterward, this residual signal is transformed, quantized and entropy coded to further compress it. The reconstructed (i.e., distorted) version of a frame is the one used as a reference for other frames; thus, the coding distortion of the current frame may impact the RD performance of the future frames in the encoding order.

Experiments were carried out in [54] to investigate the relationship between the coding distortion and the number of coding bits of the predicted frame $i+1$ and coding distortion of its directly preceding reference frame i. In these experiments, the first three frames of some standard HEVC sequences were encoded using I-P-P structure. The last two P frames in this structure only used their directly preceding frame as reference. For each of the last two P frames, the QP of frame $i+1$ was fixed to 32 while the QP of frame i was varied from 20 to 31. Based on this experiment, it was observed in [54] that the coding distortion of frame $i+1$ varies linearly with the coding distortion of frame i. In addition, the number of coding bits of frame $i+1$ had approximately a constant relation with the coding distortion of frame i, which could also be confirmed according to [138, 45]. From the findings of these experiments, the following equations were defined in [54] for coding distortion and number of coding bits of frame $i+1$:

$$\begin{aligned} D_{i+1} &\approx \mu_{i+1,i} D_i + D_{i+1}^{(0)} \\ R_{i+1} &\approx R_{i+1}^{(0)} \end{aligned} \tag{3.3}$$

Here D_{i+1} and R_{i+1} are the coding distortion and the number of coding bits, respectively, of the predicted frame $i+1$, whereas D_i and R_i are the coding distortion and the number of coding bits, respectively, of the reference frame i. $\mu_{i+1,i}$ is the linear coefficient or in other words, the slope of the linear distortion between frame $i+1$ and frame i. $D_{i+1}^{(0)}$ denotes the coding distortion of the predicted frame $i+1$ when all its preceding reference frames are coded losslessly or without distortion, while $R_{i+1}^{(0)}$ is the equivalent number of coding bits under the same condition.

Clearly, accurately estimating the distortion dependency, $\mu_{i+1,i}$, between the predicted and reference frame is an essential key to solve the adaptive QP optimization problem in (3.2).

Estimating the distortion dependency coefficient, μ, has recently been done in [54] based on the high rate theory. Suppose that p_{i+1} defines the original pixel value of the predicted frame $i+1$, p_i defines the original pixel value of the reference frame i, and $\hat{p}_i$ defines the reconstructed pixel value of the reference frame i. Define D_{i+1}^{MCP} as the variance of the motion prediction residual. D_{i+1}^{MCP} was given in [54] as follows:

$$\begin{aligned} D_{i+1}^{MCP} &= E[(p_{i+1} - \hat{p}_i)^2] \\ &\approx E[(p_{i+1} - p_i)^2 + (p_i - \hat{p}_i)^2] \\ &= \sigma_{ori}^2 + D_i \end{aligned} \tag{3.4}$$

where σ^2_{ori} is the variance of the residual signal produced by motion estimation based on original reference frames and D_i is the coding distortion of the nearest reference frame i.

The high rate theory provides an approximate relationship among the coding distortion of the predicted frame D_{i+1}, the variance σ^2, and the coding rate R (in bits per pixel), which is defined as follows [54, 48, 34]:

$$D_{i+1} = 2^{-2R_{i+1}} * D^{MCP}_{i+1} \tag{3.5}$$

From Equation (3.4) and Equation (3.5),

$$D_{i+1} = 2^{-2R_{i+1}} * (D_i + \sigma^2_{ori}) \tag{3.6}$$

We know that D_i does not impact R_{i+1} so we can make an observation that there is a linear relationship between D_{i+1} and D_i for a given R_{i+1} and the slope is $2^{-2R_{i+1}}$. In a practical video encoding scenario, it is difficult to find an estimate to the coding rate R_{i+1} given the complex CABAC entropy coding scheme in HEVC. Therefore, the distortion dependency metric, μ, in [54] was given by:

$$\mu_{i+1,i} = \frac{D_{i+1}}{D_i + \sigma^2_{ori}} \tag{3.7}$$

Here σ^2_{ori} is the variance of the residual signal, which is produced by executing a 2Nx2N, where N=16, mode motion estimation based on original reference frames. D_i is the coding distortion of the nearest reference frame i in terms of mean squared error (MSE), and D_{i+1} is the coding distortion of the predicted frame $i+1$ in terms of MSE. The definition in (3.7) is consistent with one's intuition, because if σ^2_{ori} is large, then the coding distortion of the reference frame and predicted frame is less correlated, and as a result, the distortion dependency, μ, is less.

3.3.2 Accumulated Propagation Effects for LD HEVC

The first frame in LD HEVC is coded as an I-frame, and thus its coding distortion depends only on itself, i.e.

$$D_0 = D^{(0)}_0 \tag{3.8}$$

Based on the linear model we reviewed in (3.3), the distortion of the rest of the frames can also be written as follows:

$$\begin{aligned}
D_1 &= \mu_{1,0} D_0 + D_1^{(0)} \\
&= \mu_{1,0} D_0^{(0)} + D_1^{(0)} \\
D_2 &= \mu_{2,1} D_1 + D_2^{(0)} \\
&= \mu_{2,1}\mu_{1,0} D_0^{(0)} + \mu_{2,1} D_1^{(0)} + D_2^{(0)} \\
D_3 &= \mu_{3,2} D_2 + D_3^{(0)} \\
&= \mu_{3,2}\mu_{2,1}\mu_{1,0} D_0^{(0)} + \mu_{3,2}\mu_{2,1} D_1^{(0)} + \mu_{3,2} D_2^{(0)} + D_3^{(0)} \\
D_4 &= \mu_{4,3} D_3 + D_4^{(0)} \\
&= \mu_{4,3}\mu_{3,2}\mu_{2,1}\mu_{1,0} D_0^{(0)} + \mu_{4,3}\mu_{3,2}\mu_{2,1} D_1^{(0)} \\
&+ \mu_{4,3}\mu_{3,2} D_2^{(0)} + \mu_{4,3} D_3^{(0)} + D_4^{(0)} \\
&\vdots
\end{aligned} \tag{3.9}$$

From (3.9), we can write the distortion of frame i as follows:

$$D_i = D_i^{(0)} + \sum_{k=0}^{i-1} \theta_{i,k}(\mu) D_k^{(0)} \tag{3.10}$$

Here $\theta_{i,k}$, $k = 0, 1, \cdots, i-1$, is a function of μ indicating the total impacts of the distortion of frame k on frame i. Looking at (3.10), we can see that D_i is described as a linear combination of $D_0^{(0)}, D_1^{(0)}, \cdots, D_i^{(0)}$. By substituting (3.3) and (3.10) into our optimization problem in (3.2):

$$
\begin{aligned}
J &= \sum_{i=0}^{N} D_i + \lambda \sum_{i=0}^{N} R_i \\
&= \sum_{i=0}^{N} \left(D_i^{(0)} + \sum_{k=0}^{i-1} \theta_{i,k}(\mu) D_k^{(0)} \right) + \lambda \sum_{i=0}^{N} R_i^{(0)} \\
&= \sum_{i=0}^{N} \varepsilon_i D_i^{(0)} + \lambda \sum_{i=0}^{N} R_i^{(0)}
\end{aligned} \tag{3.11}
$$

In (3.11), ε_i is the impact of the coding distortion of frame i propagated to all future frames due to the nature of the LD configuration, which is recursively expressed as follows:

$$
\varepsilon_i = \begin{cases} 1 & \text{if } i = N \\ 1 + \varepsilon_{i+1} \mu_{i+1,i} & \text{if } i \neq N. \end{cases} \tag{3.12}
$$

Looking at (3.11), $D_i^{(0)}$ and $R_i^{(0)}$ are the coding distortion and the number of coding bits of frame i when all references frames are losslessly coded or frame i is motion compensated from original reference frames. In this particular case, if all impact factors ε_i are known, the total RD cost function in (3.11) can be the aggregate sum of the RD cost functions of each coding frame. Hence, the RDO function in (3.11) can be solved by individually optimizing the RDO function of each frame as follows:

$$
\begin{aligned}
&min\{J_i\}, \ \text{ where } \ J_i = \varepsilon_i D_i^{(0)} + \lambda R_i^{(0)}, \\
&for \ \ i = 0, 1, \cdots, N.
\end{aligned} \tag{3.13}
$$

3.4 Estimating the propagation parameters for Lagrangian Multiplier Determination

It is clear that accurately estimating ε_i for each frame is another essential key to solving the present QP optimization problem. As seen in Equation (3.12), getting exact values for ε_i is intractable because all the information about the future frames must be available for each frame i, which is certainly undesirable in LD scenarios. Instead, we introduce the

notion of the propagation length, estimate it, and propose two novel methods to predict ε_i. This contribution will be discussed in this section and will be assessed in the experimental results section.

3.4.1 Estimation of the Impact Propagation Length

If we expand the definition of ε_i in (3.12), we can see the following:

$$\begin{aligned}\varepsilon_i &= 1 + \mu_{i+1,i} + \mu_{i+2,i+1}\mu_{i+1,i} + \mu_{i+3,i+2}\mu_{i+2,i+1}\mu_{i+1,i} \\ &\quad + \mu_{i+4,i+3}\mu_{i+3,i+2}\mu_{i+2,i+1}\mu_{i+1,i} \\ &\quad + \cdots + \mu_{N,N-1}\mu_{N-1,N-2}\mu_{N-2,N-3}\cdots\mu_{i+1,i} \\ &= 1 + \theta_{i+1,i}(\mu) + \theta_{i+2,i}(\mu) + \theta_{i+3,i}(\mu) \\ &\quad + \theta_{i+4,i}(\mu) + \cdots + \theta_{N,i}(\mu) \end{aligned} \tag{3.14}$$

From (3.14), it is difficult to calculate ε_i because all $\mu_{i+1,i}\cdots\mu_{N,N-1}$ are unavailable for the current frame i. To overcome this difficulty, we carried out two experiments to examine the possibility of truncating the expansion of ε_i in (3.14) at a certain length p, where $0 \leq p \leq N$, and to estimate p, respectively. Here, p is the propagation length or in other words, the impact length of the current frame on future frames in the temporal propagation chain.

First, we carried out an experiment to investigate the possibility of truncating the expansion of ε_i in (3.14). In this experiment, three sequences were used: namely BasketballPass, BlowingBubbles, and BQSquare. These sequences were encoded under the I-P-P coding structure with P frames using only their most adjacent frame as reference. In addition, we fixed the QP pattern of all frames except one reference frame, F_{ref}, for which we change its QP from 20 to 31. The predicted frame, F_{pred}, had a fixed QP equal to 32. We started by setting the F_{ref} to the nearest reference frame and then progressively decreased its POC until we see a negligible impact on the coding distortion of frame F_{pred}. Negligible impact means that the coding distortion of F_{ref}, MSE_{ref}, does not really change the coding distortion of F_{pred}, MSE_{pred}. Figure 3.2 plots the relationship between MSE_{ref} and MSE_{pred} for F_{ref} from 15 to 11 when $F_{pred} = 16$. At $F_{ref} = 11$, a negligible impact for the sequences under test between MSE_{ref} and MSE_{pred} starts to occur compared to $F_{ref} = 15, 14, 13, 12$ until it becomes apparent at a lower POC. To further support our observation, we did the same experiment for $F_{pred} = 17, 18, 19, 20, 21, 22, 23$. Table 3.2 shows the average Pearson correlation coefficient between MSE_{ref} and MSE_{pred} for all

F_{pred} from 16 to 23 at different p across BasketballPass, BQSquare, and BlowingBubbles where the correlation coefficient decreases as p increases. The same observation can be seen in other videos sequences, which enlightens the possibility of introducing the notion of propagation length and limiting the expansion of ε_i in Equation (3.14).

Table 3.2: Average Pearson correlation coefficient for each p from 1 to 8 across BasketballPass, BQSquare, BlowingBubbles for all F_{pred} from 16 to 23.

	1	2	3	4	5	6	7	8
AVERAGE	0.9	0.7	0.5	0.5	0.2	0.2	0.2	0.1

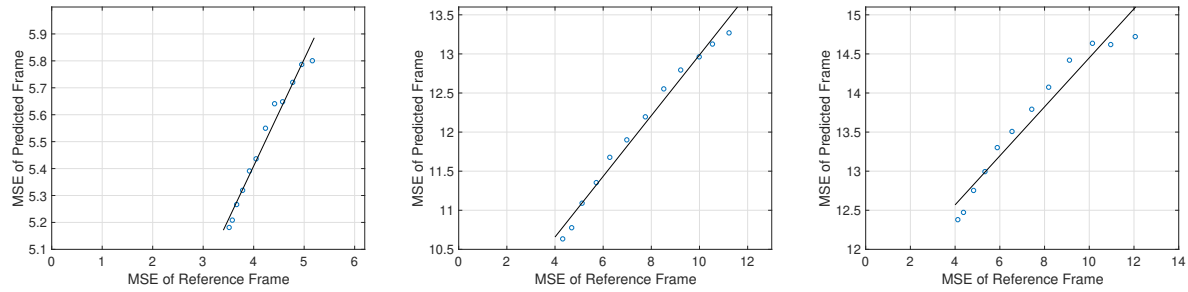
Second, we examined the values of each $\theta_{j,i}(\mu)$, where $i+1 \leq j \leq N$, to estimate p in order to enable ε_i computation. We ran an experiment on the first 100 frames of two sequences, BasketballPass and Racehorses at QP = 22, 27, 32, 37 and default λ_{HM}. To give enough room to observe $\theta_{j,i}(\mu)$, we choose i to be three GOPs before the end of the sequence. Therefore, at $i = 88$, we calculate all $\theta_{i+1,i}(\mu) \cdots \theta_{N,i}(\mu)$ and plot them as shown in Figure 3.3; this figure includes a plot and its zoomed-in version. As seen in the figure, $\theta_{i+1,i}(\mu)$, $\theta_{i+2,i}(\mu)$, $\theta_{i+3,i}(\mu)$, $\theta_{i+4,i}(\mu)$ are significant while the rest are insignificant or zero, which was also a typical behaviour in other i and other sequences. Hence, $\theta_{i+1,i}(\mu)$, $\theta_{i+2,i}(\mu)$, $\theta_{i+3,i}(\mu)$, $\theta_{i+4,i}(\mu)$ can be enough to approximate ε_i. This finding aligns with one's intuition, in view that the impact of the coding distortion for frame i on future frames has a damping effect. In other words, the significance of ε_i decreases with time until it finally dies out at a certain propagation length, p.

Based on the outcomes of the previous two experiments, we can introduce the notion of the propagation length and can truncate the series of ε_i at p. In addition, these experiments enabled the possibility of setting p to 4 in the rest of our experiments. Hence, we can now write ε_i as follows:

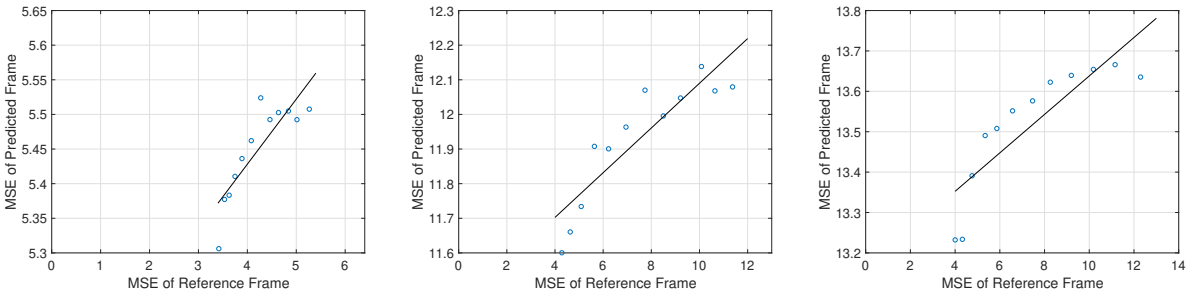
$$\begin{aligned}\varepsilon_i \approx 1 &+ \mu_{i+1,i} + \mu_{i+2,i+1}\mu_{i+1,i} + \mu_{i+3,i+2}\mu_{i+2,i+1}\mu_{i+1,i} \\ &+ \mu_{i+4,i+3}\mu_{i+3,i+2}\mu_{i+2,i+1}\mu_{i+1,i}\end{aligned} \tag{3.15}$$

3.4.2 Two Methods for ε_i Prediction

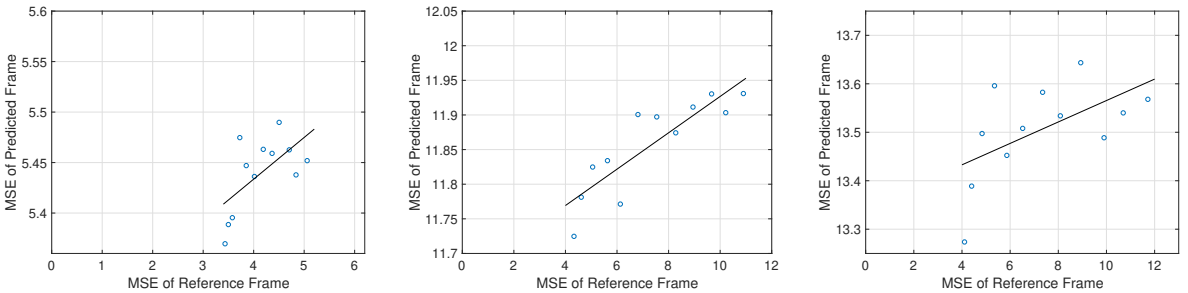
Equation (3.15) truncates ε_i at the fourth order term, but still leaves out the challenge of estimating ε_i from future frames. There are two ways to circumvent this challenge: (1)



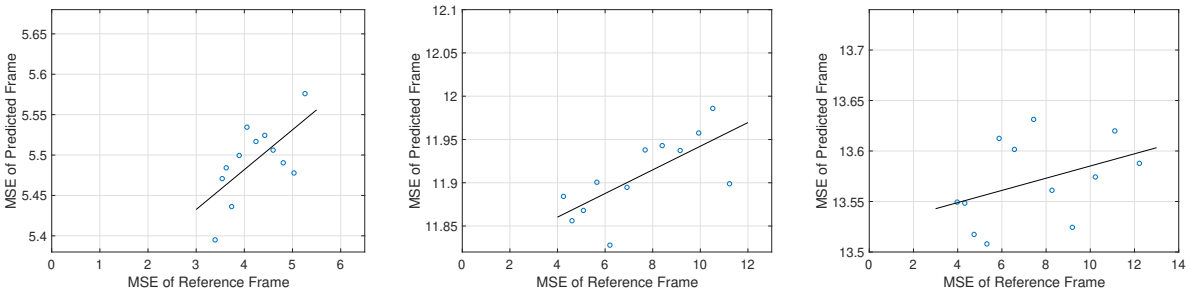
(a) $F_{pred} = 16,\ F_{ref} = 15$ **(b)** $F_{pred} = 16,\ F_{ref} = 15$ **(c)** $F_{pred} = 16,\ F_{ref} = 15$



(d) $F_{pred} = 16,\ F_{ref} = 14$ **(e)** $F_{pred} = 16,\ F_{ref} = 14$ **(f)** $F_{pred} = 16,\ F_{ref} = 14$

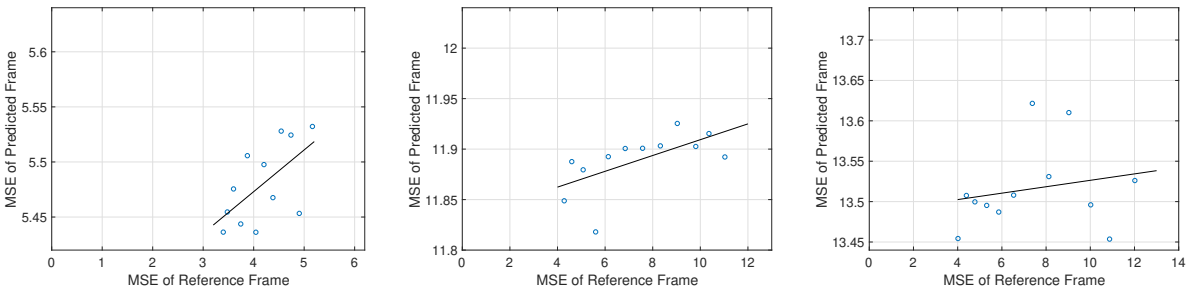


(g) $F_{pred} = 16,\ F_{ref} = 13$ **(h)** $F_{pred} = 16,\ F_{ref} = 13$ **(i)** $F_{pred} = 16,\ F_{ref} = 13$



(j) $F_{pred} = 16,\ F_{ref} = 12$ **(k)** $F_{pred} = 16,\ F_{ref} = 12$ **(l)** $F_{pred} = 16,\ F_{ref} = 12$

Figure 3.2: Relationships between MSE_{ref} and MSE_{pred} at different F_{ref}: Left: BasketballPass, Middle: BlowingBubbles, Right: BQSquare.



(m) $F_{pred} = 16$, $F_{ref} = 11$ **(n)** $F_{pred} = 16$, $F_{ref} = 11$ **(o)** $F_{pred} = 16$, $F_{ref} = 11$

Figure 3.2: Relationships between MSE_{ref} and MSE_{pred} at different F_{ref}: Left: BasketballPass, Middle: BlowingBubbles, Right: BQSquare.

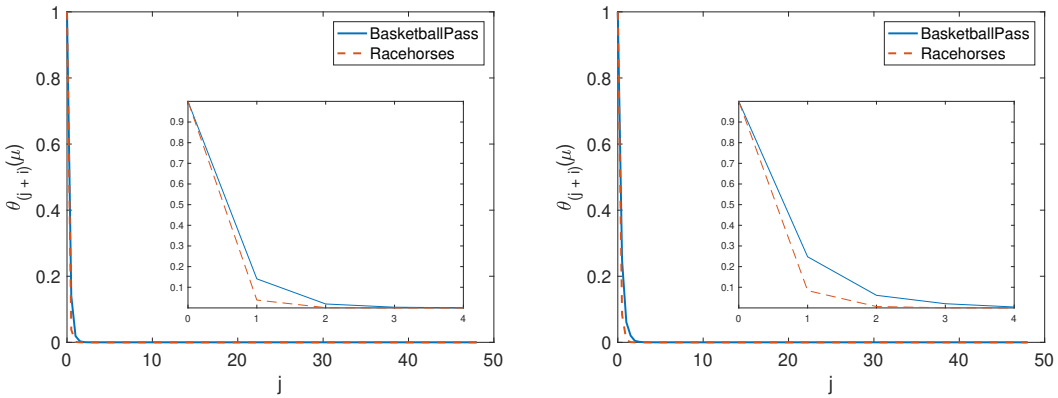
predict the current ε_i via previous potentially similar impact propagation factors, dubbed as the ε-prediction method, and (2) predict the current ε_i by estimating the coding dependency elements $\theta_{i+1,i}(\mu)$, $\theta_{i+2,i}(\mu)$, $\theta_{i+3,i}(\mu)$, $\theta_{i+4,i}(\mu)$ from past frames with potentially similar characteristics, dubbed as the μ-prediction method. In LD HEVC, frames with the same $rPOC$ share the same QP pattern and referencing structure. Following this fact, we assume that the impact of frame i on the future is approximately stationary among frames with the same $rPOC$, and hence we write the following:

$$\varepsilon_{i+4} \approx \varepsilon_i \tag{3.16}$$

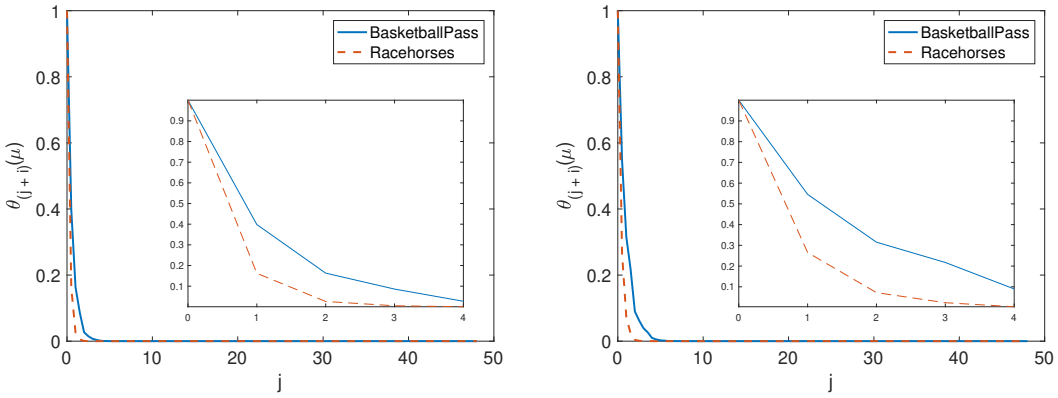
From this assumption, we outline the first method for approximating ε_i in a sliding window in Algorithm 2. This algorithm starts off by calculating ε_k from k=1 to k=4 using (3.15). Afterward, ε_j, where $9 \leq j \leq 12$, is substituted with ε_{j-8}. Then, the algorithm continues its group-based prediction to ε_i by incrementing k and j by four frames.

Algorithm 2 $\varepsilon-$Prediction Method

1: **procedure** epsilonPredMethod
2: $start \leftarrow 1$; $end \leftarrow 4$;
3: Compute ε_k from (3.15), where $start \leq k \leq end$
4: Based on (3.16), $\forall_{j \in [start+8, end+8]}\ \varepsilon_j \leftarrow \varepsilon_{j-8}$;
5: $start \leftarrow start + 4$;
6: $end \leftarrow end + 4$; go to #3



(a) BasketballPass and RaceHorses; QP = 22 **(b)** BasketballPass and RaceHorses; QP = 27



(c) BasketballPass and RaceHorses; QP = 32 **(d)** BasketballPass and RaceHorses; QP = 37

Figure 3.3: Results of $\theta_{i+1,i} \cdots \theta_{N,i}$ from Equation (3.14) that enable setting p to 4. A zoomed-in version is stacked on each plot.

Another approach for predicting ε_i is to predict μ from frames with similar $rPOC$, and thus we replace Equation (3.15) with the following:

$$\begin{aligned}\varepsilon_i \approx 1 &+ \mu_{i-3,i-4} + \mu_{i-2,i-3}\mu_{i-3,i-4} \\ &+ \mu_{i-1,i-2}\mu_{i-2,i-3}\mu_{i-3,i-4} \\ &+ \mu_{i-4,i-5}\mu_{i-1,i-2}\mu_{i-2,i-3}\mu_{i-3,i-4}\end{aligned} \tag{3.17}$$

From the definition in (3.17), we propose another method to approximate ε_i in a sliding

window as outlined in Algorithm 3. The first step in this algorithm is to calculate $\mu_{k+1,k}$, where $3 \leq k \leq 7$, from (3.7). This calculation enables computing ε_8 from (3.17). This algorithm continues predicting ε_i in a frame-based manner by moving its sliding window by one frame.

Algorithm 3 $\mu-$Prediction Method

1: **procedure** muPredMethod
2: $start \leftarrow 3$; $end \leftarrow 7$;
3: Compute $\mu_{k+1,k}$ from (3.7) for $start \leq k \leq end$
4: Compute ε_i from (3.17), where $i \leftarrow end + 1$
5: $start \leftarrow start + 1$;
6: $end \leftarrow end + 1$; go to #3

Using these two algorithms, we adaptively determine the Lagrangian multiplier and QP of each frame in LD HEVC as will be shown in the following parts of this paper.

3.4.3 Adaptive Lagrangian Multiplier Determination

Given the optimization problem in (3.13), the optimal RD cost for each frame can be achieved when

$$\begin{aligned} \frac{\partial J_i}{\partial R_i^{(0)}} &= \frac{\partial\left(\varepsilon_i D_i^{(0)} + \lambda R_i^{(0)}\right)}{\partial R_i^{(0)}} \\ &= \varepsilon_i \frac{\partial D_i^{(0)}}{\partial R_i^{(0)}} + \lambda = 0 \end{aligned} \tag{3.18}$$

Using $\frac{\partial D_i^{(0)}}{\partial R_i^{(0)}} = -\lambda_i^*$, then

$$\lambda_i^* = \frac{\lambda}{\varepsilon_i} \tag{3.19}$$

Here the Lagrangian multiplier for each frame i is adapted based on its impact propagation factor ε_i. From (3.19), it can be observed that there is an inverse correlation between the adapted λ_i^* and its propagation factor ε_i. This observation is reasonable because when ε_i

is large i.e, this frame is a key frame and has a high impact on subsequent frames, λ_i^* will be small leading to smaller coding distortions in subsequent frames. Thus, less propagated distortions will flow through the temporal chain and a better coding performance can be attained. Conversely, small values of ε_i result in large values of λ_i^*, which always happens in frames not frequently used as references i.e $rPOC = 1, 2, 3$.

3.5 Adaptive QP Selection

3.5.1 QP Determination via QP-λ Relationships

In the HEVC test model, HM [4], there is a fixed relationship between QP and λ. For each frame i, λ is defined as:

$$\lambda_{HM_i} = qp_{factor,HM_i} * 2^{\frac{(QP_{HM_i}-12)}{3}} \tag{3.20}$$

Here λ_{HM_i} is the lambda used in the HEVC's reference software given a qp_{factor,HM_i} related to the coding configuration and QP_{HM_i} is calculated by knowing the QP offset of frame i depending on its $rPOC$ as specified in Table 3.1.

In LD HEVC, qp_{factor,HM_i} is fixed for each $rPOC$ and is defined as follows:

$$qp_{factor,HM_i} = \begin{cases} 0.4845, & for \text{ POC=0,} \\ 0.4624, & for \text{ rPOC=1,} \\ 0.4624, & for \text{ rPOC=2,} \\ 0.4624, & for \text{ rPOC=3,} \\ 0.5780, & for \text{ rPOC=4,} \end{cases} \tag{3.21}$$

To obtain λ_i^*, the global Lagrangian multiplier λ has to be estimated. The RDO schemes in HM and the temporally dependent RDO produce different rates and distortions. And yet the temporally dependent RDO scheme takes the propagation factor ε_i into consideration. For fair comparision between the default HM and the proposed RDO scheme, we follow the way proposed in [138, 45] to estimate λ based on the high rate theory [34] from (3.5) as follows:

$$\begin{aligned}\lambda_{HM_i} &= -\frac{\partial D_{HM_i}}{\partial R_{HM_i}} \\ &= \ln(2) * 2^{-2R_{HM_i}} * D_{HM_i}^{MCP} \\ &= \ln(2) * D_{HM_i}\end{aligned} \tag{3.22}$$

In correspondence with (3.22) while taking the propagation factor ε_i into consideration, we write the following from (3.18):

$$\begin{aligned}\lambda &= -\varepsilon_i \frac{\partial D_i^{(0)}}{\partial R_i^{(0)}} \\ &= \varepsilon_i * \ln(2) * 2^{-2R_i^{(0)}} * D_i^{MCP^{(0)}} \\ &= \ln(2) * \varepsilon_i D_i^{(0)}\end{aligned} \tag{3.23}$$

Based on (3.22) and (3.23), we write:

$$\lambda = \lambda_{HM_i} * \frac{\varepsilon_i D_i^{(0)}}{D_{HM_i}} \tag{3.24}$$

Since D_{HM_i} and $D_i^{(0)}$ are unavailable in our practical implementation, Equation (3.24) may be approximated as follows:

$$\lambda \approx \lambda_{HM_i} * \frac{\sum_{i=0}^{n} \varepsilon_i * D_i}{\sum_{i=0}^{n} D_i} \tag{3.25}$$

where n is the number of available frames for which ε_i can be computed according to either the ε or μ methods.

Now substituting (3.25) in (3.19), we get the following:

$$\lambda_i^* \approx \frac{\lambda_{HM_i}}{\varepsilon_i} * \frac{\sum_{i=0}^{n} \varepsilon_i * D_i}{\sum_{i=0}^{n} D_i} \tag{3.26}$$

We update λ_i^* every frame in our implementation. By updating λ_{HM_i} into λ_i^* and reversing (3.20), we update QP_{HM_i} accordingly as follows:

$$QP_i = 3log_2(\frac{\lambda_i^*}{qp_{factor,HM_i}}) + 12 \tag{3.27}$$

Due to the accumulated propagation impact factor ε_i, λ_i^* is adaptive to each frame i, and so is QP_i.

3.5.2 Initialization

Because our $\varepsilon-$prediction and $\mu-$prediction methods predict each ε_i from previous frames, we decrease the QP value of the I-frame to reduce the respective distortion and in turn improves prediction. The QP values of the first 9 frames in the case of $\varepsilon-$prediction and the first 8 frames in the case of $\mu-$prediction are initialized to be the same as the those default values of the respective frames in HM except that the QP value of the I-frame is lowered by a notch and set to be 22, 22, 27, 32, which are corresponding, respectively, to the HM's default values QP = 22, 27, 32, 37.

3.5.3 Overall Adaptive QP Selection Algorithm

To put together the analysis conducted in the previous sections, we list the steps of our overall algorithm to adaptively determine QP values for all frames as follows:

Step 1 Specify the start and end of the sliding window according to either $\varepsilon-$Prediction method or $\mu-$Prediction method.

Step 2 For the first set of frames in the sliding window, select their initialized QP values according to subsection 3.5.2, and encode them.

Step 3 For the rest of the frames, follow these steps:

Step 3.1 Compute $\mu_{i+1,i}$ between each predicted frame $i+1$ and its nearest reference frame i in the sliding window according to Equation (3.7). D_{i+1} and D_i are the coding distortions in MSE of a predicted frame $i+1$ and its nearest reference frame i.

Step 3.2 Get ε_i according to either $\varepsilon-$Prediction method or $\mu-$Prediction method.

Step 3.3 Calculate λ_i^* and its corresponding QP_i using Equation (3.27).

Step 3.4 Encode the current frame with the updated QP_i.

Step 3.5 Move the sliding window as instructed by $\varepsilon-$Prediction or $\mu-$Prediction methods.

3.6 HEVC Encoder Testing Methodology: Objective Video Assessment Tool (MCTest)

To have fast access of experimental results and to facilitate the comprehensive analysis, we implemented a testing framework called MulticomTest or abbreviated as MCTest using bash script for the HM encoder [1]. Figure 3.4 explains the sequence diagram for the MCTest, or the testing process. First, we start inputting n video sequences, their configuration files, arbitrary number of encoder configuration files, and arbitrary number of frames required for analysis to the main bash script which executes the HM encoder. The main script also instructs the encoder to generate its outputs in a specific naming convention and collects HM-generated files from the log files and places them in formatted text. Afterwards, the main bash script decodes all generated bitstreams using HM decoder creating YUVs for comparison. Then, it forces the decoder to generate the reconstructed YUVs into a specific filename format. Finally, the results bash scripts generate integrated bitrate and PSNR log files of selected frames at arbitrary number of QP values. They also gather formatted text HM generated files and create a log file containing the bitrate and PSNR per frame for n sequences (at different configurations under test) with j different QP values for every configuration. Figure 3.5 explains how the bash scripts are integrated into the HM encoder and decoder.

3.7 Experimental Results

In this section, we assess the effectiveness of our pair of techniques, namely ε and μ prediction techniques under both the LDP and LDB configurations of the common test conditions (CTC) of HEVC [26]. Four classes of 20 sequences covering a wide range of

[1] https://github.com/HossamAmer12/hevc

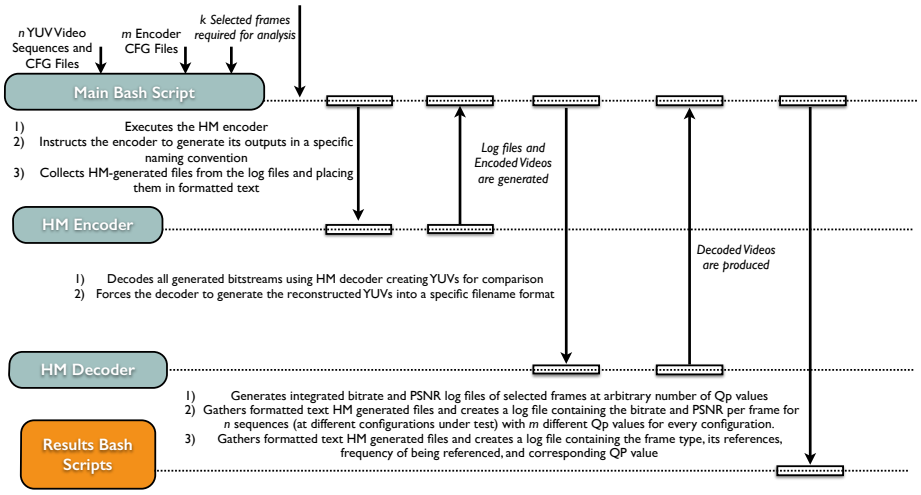


Figure 3.4: Sequence Diagram for the Objective Video Quality Assessment Framework (MCTest).

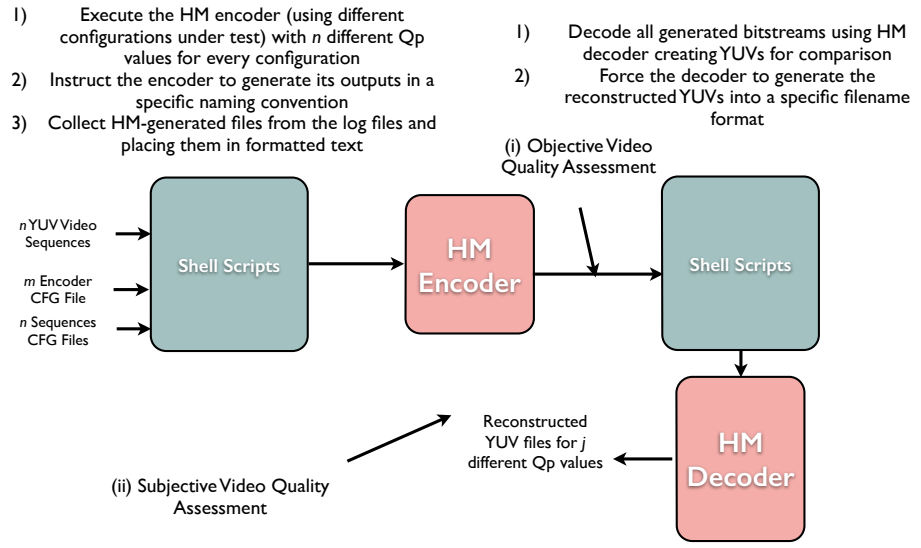
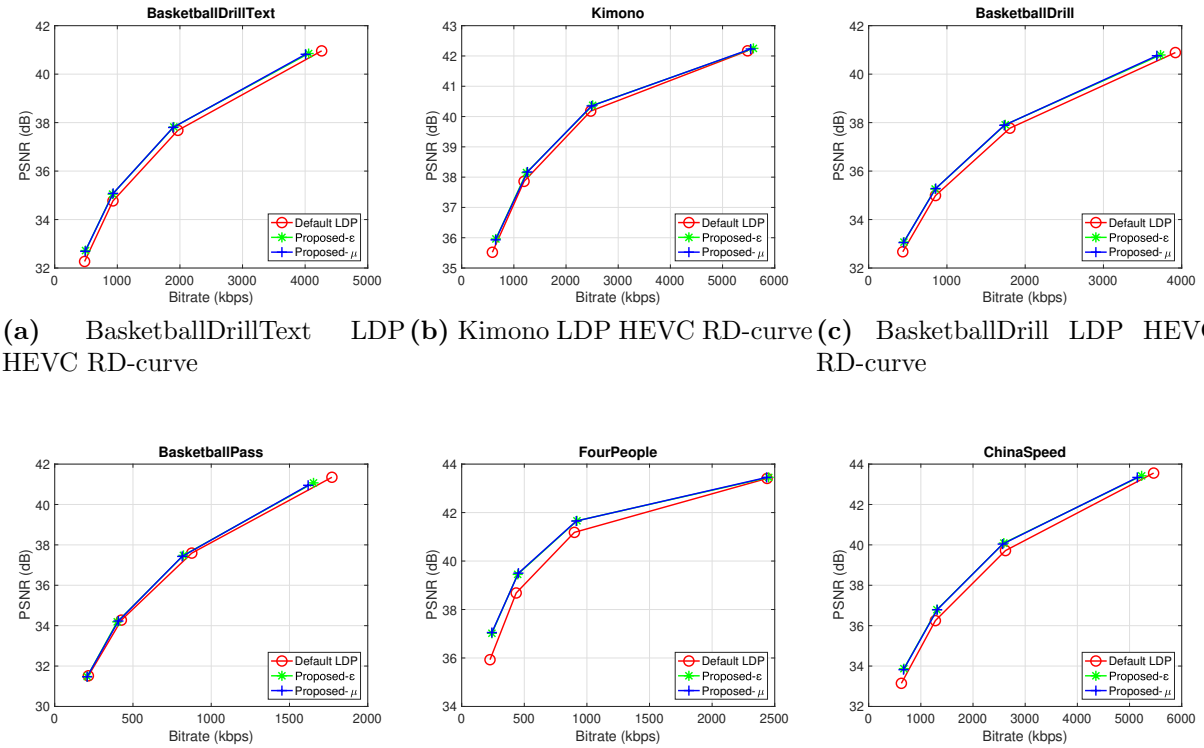


Figure 3.5: Objective Video Quality Assessment Framework (MCTest).

content and resolutions were tested under the HM 16.0 software version of HEVC. Four different initial QPs equal to 22, 27, 32, and 37 are used to observe the R-D performance of these sequences under different bitrates. Three other related methods are used for coding efficiency comparisons with respect to the default LD configurations. These methods are the QP cascading algorithm (QPC) in [144], the QP refinement (QPR) method in [78], and the CTU-based method that accounts for temporal dependencies specified in [45]. All experiments were carried out on an 8GB memory Quad-Core Intel Xeon (2×2.26GHz).



(a) BasketballDrillText LDP HEVC RD-curve **(b)** Kimono LDP HEVC RD-curve **(c)** BasketballDrill LDP HEVC RD-curve

(d) BasketballPass LDP HEVC RD-curve **(e)** FourPeople LDP HEVC RD-curve **(f)** ChinaSpeed LDP HEVC RD-curve

Figure 3.6: RD curves for the proposed algorithms and the default LDP.

3.7.1 Coding Efficiency Comparison

The R-D performance for the proposed techniques are measured via the BD-rate (BDBR) for the luminance component [82] where negative numbers of BDBR indicate performance

Table 3.3: Coding Efficiency Comparisons Between LD HEVC And the Proposed Methods With Respect To The Default HM In Terms of Luma BDBR(%)

Class	Sequence	[144]		[78]		[45]		Prop ε		Prop μ	
		LDP	LDB	LDP	LDB	LDP	LDB	LDP	LDB	LDP	LDB
B	BasketballDrive	/	/	-1.5	-1.2	-4.1	-3.7	-1.6	-1.7	-1.2	-1.2
	BQTerrace	/	/	-1.3	-0.2	-1.6	0.0	2.6	2.3	3.6	2.7
	Cactus	/	/	-2.3	-1.7	-3.8	-3.2	-6.8	-7.1	-6.5	-6.8
	Kimono	/	/	-1.7	-1.2	-1.8	-0.9	-3.5	-3.5	-3.1	-2.8
	ParkScene	/	/	-1.1	-0.9	-4.0	-4.0	-2.9	-2.9	-6.5	-2.7
	Average	/	/	-1.6	-1.1	-3.0	-2.4	-2.5	-2.6	-2.0	-2.2
C	BasketballDrill	-2.8	-2.9	-2.1	-1.6	-9.1	-8.4	-6.0	-5.9	-5.8	-5.6
	BQMall	-0.6	-0.7	-1.1	-0.8	-2.8	-2.8	-2.4	-2.5	-2.4	-2.3
	PartyScene	-2.4	-2.5	-1.5	-1.0	-2.9	-2.6	-4.5	-4.6	-4.4	-4.5
	RaceHorses	1.9	2.0	-1.5	-1.3	-0.3	0.0	0.1	0.1	0.5	0.4
	Average	-1.0	-1.0	-1.5	-1.2	-3.8	-3.5	-3.2	-3.2	-3.0	-3.0
D	BasketballPass	-0.2	-0.1	-1.9	-1.7	-4.6	-4.5	-3.7	-3.9	-3.5	-3.4
	BlowingBubbles	-1.9	-1.8	-1.2	-1.0	-2.9	-2.6	-3.5	-3.5	-3.8	-3.9
	BQSquare	-2.5	-2.0	-1.4	-1.1	-0.3	0.6	-2.7	-2.2	-2.6	-2.5
	RaceHorses	1.0	0.9	-1.3	-1.2	-1.3	-1.3	-0.7	-0.7	-0.5	-0.5
	Average	-0.9	-0.8	-1.4	-1.1	-2.3	-2.0	-2.7	-2.6	-2.6	-2.6
E	FourPeople	/	/	-3.1	-2.7	-8.6	-7.7	-13.4	-13.0	-13.6	-12.9
	Johnny	/	/	-2.5	-1.7	-3.2	-2.4	-10.4	-9.5	-10.4	-9.5
	KristenAndSara	/	/	-3.4	-2.3	-4.4	-4.1	-14.5	-14.1	-14.6	-14.2
	Average	/	/	-3.0	-2.3	-5.4	-4.7	-12.8	-12.2	-12.9	-12.2
F	BasketballDrillText	-3.0	-3.0	-2.0	-1.7	-7.2	-6.7	-6.2	-6.1	-6.0	-5.7
	ChinaSpeed	-2.4	-2.3	-2.1	-2.1	-2.9	-2.7	-7.4	-7.6	-7.2	-7.1
	SlideEditing	-0.3	-2.3	-0.5	-0.5	-0.2	-0.6	-2.0	-1.9	-0.7	-1.7
	SlideShow	-1.6	-1.6	-1.2	-0.9	-6.6	-6.4	-9.5	-9.2	-13.3	-12.9
	Average	-1.8	-2.0	-1.5	-1.3	-4.2	-4.1	-6.3	-6.2	-6.8	-6.8
Overall Average		**-1.2**	**-1.3**	**-1.7**	**-1.3**	**-3.6**	**-3.2**	**-5.0**	**-4.9**	**-4.9**	**-4.9**

Table 3.4: Coding Efficiency Improvements For Our Proposed Methods With Respect To LD HEVC In Terms of Luma BDBR(%) For Video Conferencing Sequences.

Sequence	**Prop ε**		**Prop μ**	
	LDP	LDB	LDP	LDB
vidyo1	-10.5	-9.9	-10.7	-10.1
vidyo3	-6.8	-7.2	-7.2	-7.2
vidyo4	-10.2	-9.5	-9.5	-10.1
AVERAGE	**-9.2**	**-8.9**	**-9.5**	**-9.1**

Table 3.5: Coding Efficiency Improvements For Our Proposed Methods With Respect To LD HEVC In Terms of Luma BDBR(%) For Merged Sequences.

Sequence	**Prop μ**	
	LDP	LDB
BasketballDrillPartyScene400	-6.2	-6.2
BasketballPassBQSquare400	-3.3	-2.9
KristenAndSaraJohnny400	-8.5	-8.1
AVERAGE	**-6.0**	**-5.73**

gains over the default LD HEVC benchmark. Table 3.3 shows the overall luminance BDBR results for the QPC [144], QPR [78], CTU-based [45], and the proposed methods (Prop), respectively. From Table 3.3, it can be seen that the QPC method achieves -1.2% and -1.3% BDBR improvements for LDP and LDB, respectively, the QPR method achieves -1.7% and -1.3% BDBR improvements for LDP and LDB, respectively, and the CTU-based method achieves -3.6% and -3.2% BDBR improvements for LDP and LDB, respectively. Table 3.3 also shows that the $\varepsilon-$prediction achieves -5.0% and -4.9% BDBR improvements for LDP and LDB, respectively, and the $\mu-$prediction achieves -4.9% and -4.9% BDBR improvements for LDP and LDB, respectively. From these results, it can be seen that our proposed methods provide higher coding efficiency gains than the QPC, QPR, and CTU-based methods. Figure 3.6 shows some RD curve comparisons for our methods for some sequences, where the proposed methods on average outperform the default LDP HEVC. Furthermore, we can observe from Table 3.3 that there are more than 10.0% BDBR savings for sequences in Class E, and more than 3.0% BDBR saving for Kimono, which will be discussed in the following subsection.

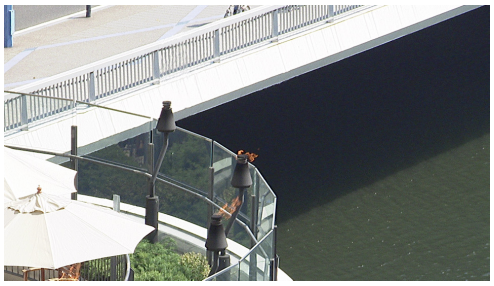
3.7.2 Analysis for the Coding Efficiency Results

It is expected for our proposed methods to work well in terms of coding efficiency on sequences with strong temporal dependency because they exploit the temporal dependency among the input frames, especially that the proposed RDO model predicts its parameters from past frames. This is indeed the case as shown in Class E sequences, for example. Conversely, our proposed methods do not achieve the same gains with Racehorses, BQTerrace, and SlideEditing because they contain complex scenes and large motion leading to poor leverage of the frame-level temporal dependency.

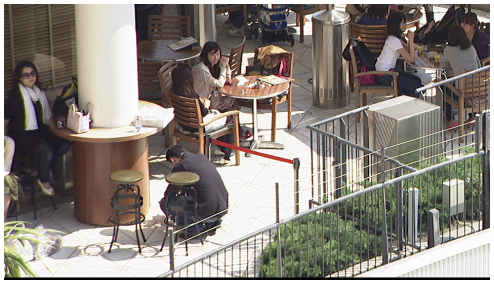
To get some insight about our methods with sequences with complex scenes, we carried out an experiment where we spatially divided BQTerrace video sequence into four different quads. The four quads of the first frame are shown in Figure 3.7. As shown in this figure, quad 1 and quad 3 contain less complex structures than quad 2 and quad 4, which contain more subtle movements of water leading to difficulty of leveraging the temporal correlation. The coding efficiency of our proposed methods computed on the first 100 frames of BQTerrace with respect to the default HM confirmed our understanding: (1) the μ-prediction method achieved a coding efficiency of -2.2%, -0.4%, -1.0%, and -1.2%, for quad 1, 2, 3, and 4, respectively under the LDP configuration, (2) the ε-prediction method achieved a coding efficiency of -3.2%, -1.5%, -2.6%, and -0.4%, for quad 1, 2, 3, and 4, respectively under the LDP configuration. From these results, we can conclude that our proposed methods can work on regions of frames where temporal dependencies can

be leveraged. In addition, the $\varepsilon-$prediction method can work better in sequences with complex scenes than the $\mu-$prediction method.

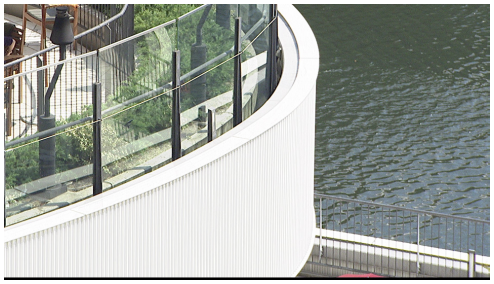
(a) BQTerrace Quad One



(b) BQTerrace Quad Two



(c) BQTerrace Quad Three



(d) BQTerrace Quad Four

Figure 3.7: First frame of the BQTerrace video sequence divided into quads where quad one and three contain less complex structures than quad two and four.

To confirm that our methods perform particularly well on sequences with strong temporal dependency, we carried an experiment on extra three stationary and low-motion sequences, namely vidyo1, vidyo3, and vidyo4. This experiment followed the adaptive QP algorithm explained in subsection 3.5.3 for ε and μ methods. Table 3.4 indicates that the proposed ε has 9.2%, 8.9% BDBR savings for LDP and LDB, respectively, and the μ has 9.5%, 9.1% BDBR savings for LDP and LDB, respectively. From these results, it can be observed that the proposed μ-method works better than $\varepsilon-$method in slow motion sequences. On top of that, we can deduce that our proposed methods can improve the coding efficiency of video conferencing sequences by a large margin.

As seen by Algorithms 2 and 3, our proposed methods are predictive in nature because they use past distortion information to predict the future distortion information to calculate

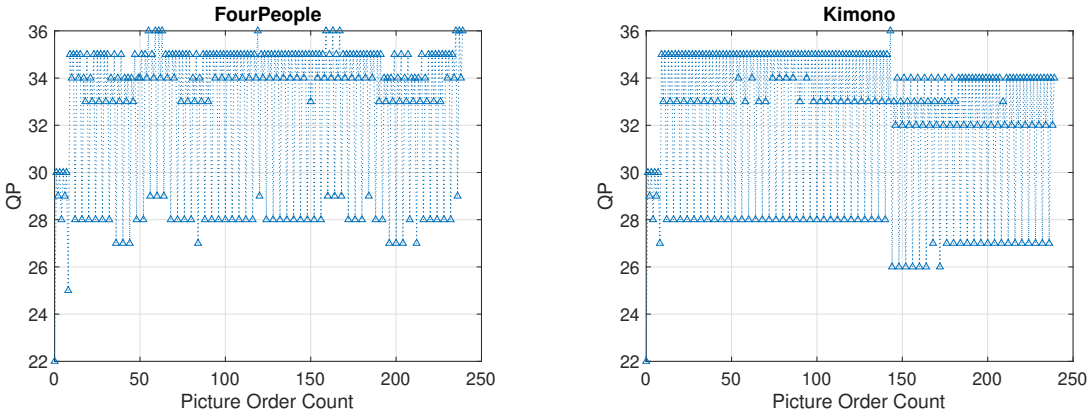
the temporal distortion dependency relationship, μ, and eventually the impact factor of each frame, ε_i. This approach is beneficial in sequences that have scene changes or sudden changes of content as reflected by Kimono where our adaptive QP methods can achieve -3.5% coding efficiency improvement in the LDP configuration.

To gain more insight about this phenomenon into our proposed algorithms, we carried out two experiments. First, we plot the QP per frame from the $\mu-$prediction LDP method for FourPeople and Kimono at initial QP = 27 as shown in Figure 3.8. From this figure, it can be seen that FourPeople has a relatively fixed QP pattern because this sequence has a very strong temporal dependency. Comparing FourPeople with Kimono, we can see that Kimono's QP pattern suddenly changes at POC=144 and is relatively decreased with respect to the first part of Kimono because a scene change occurs at POC=140 and Kimono's second part has a stronger temporal correlation than its first part. Second, to get an even deeper look at this phenomenon for our methods, we carried out another experiment where we created three sequences by merging the first 200 frames of two sequences in Class C, D, E. Similar to Kimono, each of these sequences has only one scene change. For Class C, we merged BasketballDrill and PartyScene, merged BasketballPass and BQSquare for Class D, and merged KristenAndSara and Jhonny for Class E. Afterward, we run our $\mu-$prediction method under the LDP and LDB configurations on these three sequences and observe the results. The experiment settings are the same as the ones for the standard HEVC sequences. Table 3.5 demonstrates that the $\mu-$prediction method provides some average coding efficiency improvements in the extra merged sequences under test. Based on these two experiments, our proposed methods can react to sequences with sudden changes in content and can on average improve the RD performance of scene change sequences.

3.7.3 Quality Fluctuation

Quality fluctuation is one of the important aspects to evaluate for the encoded video because if the amount of quality fluctuations is relatively high, then the decoded video will not be as appealing to human perception.

To assess the quality fluctuations of our proposed methods, we carried out an experiment to measure the standard deviation (std) of PSNR for the default HM, the $\varepsilon-$method, and the $\mu-$method under the LDP configuration, as indicated in Table 3.6. This table shows the average std of PSNR for each sequence across QPs equals to $\{22, 27, 32, 37\}$. Each row in Table 3.6 corresponds to particular sequence, its average std of PSNR under default LDP HM, its average std of PSNR under the $\varepsilon-$method with a number between parenthesis indicating the difference between the std of $\varepsilon-$method and default HM, its



(a) Fourpeople at QP = 27 **(b)** Kimono at QP = 27

Figure 3.8: QP Per Frame for Fourpeople, Kimono at QP = 27 using $\mu-$prediction for LDP.

average std of PSNR for the $\mu-$method with a number between parenthesis indicating the difference between the std of $\varepsilon-$method, respectively. At the end of each class of sequences, we compute the overall average of HM and proposed methods as well as the average differences between the proposed methods and the default HM.

Table 3.6 indicates that the overall average std for the proposed $\varepsilon-$method is 0.14 dB away from default HM, and the overall average std for the proposed $\mu-$method is 0.17 dB away from the default HM. Except for sequences that contain significantly large motion, this average difference is approximately consistent across the vast majority of sequences and also decreases to 0.06 dB in some sequences. For example, the BDBR savings in class E under LDP HEVC are 12.8% and 12.9% in $\varepsilon-$method and $\mu-$method, respectively. These BDBR savings are at the expense of an average difference in terms of PSNR std of only 0.12 dB, 0.13 dB for $\varepsilon-$method and $\mu-$method, respectively. Furthermore, Table 3.6 shows that sequences with significantly large motion may lead to high PSNR std produced from our methods because they predict their parameters from past frames. For instance, the average PSNR std of SlideEditing is 0.36 dB and 0.43 dB away from the default HM for $\varepsilon-$method and $\mu-$method, respectively. These differences are also reasonable because SlideEditing in particular contains a lot of large motion. Additionally, although the QP pattern for our proposed methods shown in Figure 3.8 for Kimono and FourPeople appear to cause more quality fluctuations compared to default HM, the average std of Kimono and FourPeople in the μ-method is only 0.02 dB and 0.14 dB away from the default HM.

Further, Table 3.6 demonstrates that the differences in terms of PSNR std align with the way how the ε and μ methods predict the coding dependency parameters. The ε method is a group-based prediction method and thus it is expected to have slightly less PSNR std than that of the μ-method. All the aforementioned observations were also found with our methods under the LDB HEVC configuration. Based on this experiment, we can conclude that the difference in terms of PSNR std between our proposed methods and the default HM is on average range from only 0.14 dB to 0.17 dB and can decrease to $\approx$0.06 dB for some sequences. Relative to the default HM encoding, we believe that these differences may not have a significant impact on human perception.

3.7.4 Computational Complexity Analysis

To evaluate the computational complexity of our techniques, we measure the execution time of the default LD HEVC and compare it against the execution time of our methods. Encoding time (ET) ratio is the ratio of geometric means of encoding time. Table 3.7 shows the ET ratio per sequence, class, and overall ET ratio of the $\mu-$prediction method with default initialization under the LDP configuration (similar results under other settings or configurations). It can be seen that the $\mu-$prediction method increases the encoding time only by 1% on average given the present coding efficiency gains. This complexity increase is due to the additional motion estimation process required to estimate the coding dependency for each frame (see the subsection 3.3.1). In some cases, the complexity may decrease with respect to the default HM because if one frame in our proposed methods have a high reconstruction quality, this will lead to smaller residuals, which in turn decreases the time for transform, quantization, and entropy coding processes.

3.8 Chapter Summary

In this chapter, we proposed adaptive frame-level QP selection methods for the LD HEVC by integrating the coding propagation effect into RDO through a notion called the temporal propagation length. Based on a linear model to model the inter-frame distortion dependency and analysis for the temporal propagation length, we adaptively determined the QP value and the Lagrangian Multiplier of each frame. Experimental results demonstrated that our methods can achieve BDBR savings for the LD configurations of 5.0% and 4.9% for the $\varepsilon-$prediction method, and 4.9% and 4.9% for the $\mu-$prediction method. All these improvements at the expense of an insignificant average increase of 1% time overhead. In addition, the results also show that our proposed method can provide considerable coding

efficiency improvements for video conferencing sequences in specific. These improvements can go up to -12.9% and -12.2%, respectively. Overall, this chapter focused on the human vision perspective and introduced improvements in terms of the trade-off between compression rate and compression distortion at insignificant increase in encoding time. Yet, this chapter did not tackle the huge computational intensity of HEVC due to its numerous advanced coding tools. Along the same line of the human vision perspective, in the next chapter, we examine the effect of creating a fully connected neural network machine based learning strategy to work online and minimize the computational intensity in the HEVC CU partition process, while controlling the trade-off between the compression rate and compression distortion.

Table 3.6: Quality Fluctuation Comparison Between The Default HM And The Proposed Methods Under LDP Configuration. Numbers Between Parenthesis Indicate The Differences Between The Proposed Methods And The Default HM In Terms of std.

Class	Sequence	HM std (dB)	ε std (dB)	μ std (dB)
B	BasketballDrive	0.83	0.86 (0.03)	0.87 (0.04)
	BQTerrace	0.88	0.96 (0.08)	0.98 (0.1)
	Cactus	0.48	0.53 (0.05)	0.54 (0.06)
	Kimono	0.71	0.7 (-0.01)	0.72 (0.02)
	ParkScene	0.48	0.63 (0.15)	0.66 (0.18)
AVERAGE		**0.68**	**0.74 (0.06)**	**0.75 (0.07)**
C	BasketballDrill	0.52	0.6 (0.08)	0.66 (0.14)
	BQMall	0.89	0.94 (0.05)	0.95 (0.06)
	PartyScene	0.77	0.97 (0.2)	0.99 (0.22)
	RaceHorses	1.44	1.47 (0.03)	1.5 (0.06)
AVERAGE		**0.9**	**0.99 (0.09)**	**1.0 (0.1)**
D	BasketballPass	1.2	1.3 (0.1)	1.36 (0.16)
	BlowingBubbles	0.8	1 (0.2)	1 (0.2)
	BQSquare	0.7	0.8 (0.1)	0.85 (0.15)
	RaceHorses	1	1 (0)	1 (0)
AVERAGE		**0.93**	**1.0 (0.07)**	**1.0 (0.07)**
E	FourPeople	0.3	0.43 (0.13)	0.44 (0.14)
	Johnny	0.2	0.3 (0.1)	0.31 (0.11)
	KristenAndSara	0.3	0.41 (0.11)	0.42 (0.12)
AVERAGE		**0.26**	**0.38 (0.12)**	**0.39 (0.13)**
F	BasketballDrillText	0.6	0.75 (0.15)	0.77 (0.17)
	ChinaSpeed	0.9	1 (0.1)	1 (0.1)
	SlideEditing	0.4	0.9 (0.5)	1 (0.6)
	SlideShow	3.24	3.6 (0.36)	3.67 (0.43)
AVERAGE		**1.28**	**1.56 (0.28)**	**1.6 (0.32)**
Overall AVERAGE		**0.83**	**0.97 (0.14)**	**1.0 (0.17)**

Table 3.7: Encoding Time Ratio of $\mu-$prediction Under LDP Configuration

Class	Sequence	Encoding Time Ratio
B	BasketballDrive	102.3%
	BQTerrace	101.6%
	Cactus	101%
	Kimono	101.9%
	ParkScene	101.4%
Encoding Time Ratio		**101.6%**
C	BasketballDrill	100.1%
	BQMall	101.7%
	PartyScene	100.4%
	RaceHorses	101.4%
Encoding Time Ratio		**100.9%**
D	BasketballPass	99.9%
	BlowingBubbles	99.4%
	BQSquare	99.8%
	RaceHorses	100.9%
Encoding Time Ratio		**100%**
E	FourPeople	102.1%
	Johnny	102.7%
	KristenAndSara	102.3%
Encoding Time Ratio		**102.4%**
F	BasketballDrillText	99.9%
	ChinaSpeed	102.5%
	SlideEditing	102.8%
	SlideShow	102.7%
Encoding Time Ratio		**101.9%**
Overall Encoding Time Ratio		**101%**

Chapter 4

Neural Network for HEVC CU Split Decision equipped with Laplacian Transparent Composite Model

This chapter examines the effect of creating a fully connected neural network based learning strategy to work online and minimize the computational intensity in the HEVC CU partition process, while controlling the trade-off between the compression rate and compression distortion. Section 4.1 reviews some related method to shrink the computational intensity of HEVC. Section 4.2 outlines our fully connected based method to reduce the the computational intensity of HEVC and shows some analysis, Section 4.3 demonstrate our method with some experimental results, and Section 4.4 concludes this chapter.

4.1 Literature Review

In the literature, many methods that do not rely on neural networks (NNs) have been proposed to shrink the computational intensity gap between HEVC and its predecessor(s). Some of these methods are threshold-based and have fixed designs [143, 111]. For example, method in [143] consisted of two levels: micro-level and macro-level. At the micro-level, the number of modes of the rough mode decision (RMD) of the HEVC has been reduced via a progressive rough mode decision (pRMD). In pRMD, a chosen subset of the 35 modes were first tested and their corresponding sum of absolute differences (SATDs) are pre-estimated. For the mode of the minimum SATD, another RMD would be applied to its neighbouring

modes, and the mode with lowest SATD is added to the result set. This algorithm halts when there are specific number of modes added to the result set, which already has the three MPMs. At the macro-level, an early termination for the CU splitting process exists if the predicted RD cost of split CUs was larger than that of current CU.

Other methods, as in [62], extracted a feature dubbed as the Outlier Block Flag (OBF) extracted from the newly emerged model, Laplacian Transparent Composite Model (LPTCM). This feature was employed to train a Bayes model in order to decrease the complexity of HEVC encoding. In [110], the OBF feature was extended and further combined with two support vector machines (SVM) to make fast CU decisions. Each of the two SVMs were designed to either precisely terminate or continue the partitioning process without checking the current depth. Based on the precision statistics of each of these SVMs, a final decision to either terminate or continue CTU partitioning will be taken. Furthermore, SVMs were also used in [87], in conjunction with other features, for fast CU decisions.

In this chapter, we take a different approach by applying neural networks (NNs) to improve the complexity of the HEVC CU partition process in intra coding as a classification problem. Our motivation comes from the recent noticeable activities of NNs in image and video applications. For instance, an entire framework based on NNs was proposed in [124] to build an image compression engine for the JPEG standard. In [35], a convolutional NN (CNN) was used as a post-processing step to reduce video artifacts resulting from HEVC's encoding.

However, applying NNs to time-sensitive applications such as the HEVC CU partition process is challenging. First, the learning process requires a large amount of data and long training time. Second, even if an NN is properly populated, the time required for it to make a decision may not be negligible, especially when it is deep. In the literature, offline training without updation is often used to circumvent the first problem. For example, Duanmu et. al proposed a method that utilizes CU raw pixel statistics and input them to an NN to optimize the screen content coding (SCC) version of HEVC [44]. In their method, the first 30 frames of the SCC sequences were extracted to offline train the NN to provide enough samples for the NN to converge. The technique in [72] followed a similar approach to Duanmu's method. Likewise, CNNs were employed to ameliorate the computational intensity of the intra encoding for HEVC in [88, 132]. In the CNN method, the training was done offline using specific set sequences, which decreases this method's content adaptivity.

Instead, in this chapter, we examine the effect of creating a learning strategy to work online and minimize the computational intensity in the HEVC CU partition process, while

controlling the resulting bitrate loss. Also, our goal is to propose a method to be among the best of its NN's counterparts. In our work, four NNs are hierarchically inserted with one NN for each depth of the CTU. Each NN instructs the HEVC encoder to either split the current CU and skip the RDO process at the current depth or terminate the CU search algorithm. To alleviate the the above mentioned problems associated with NNs, we adopt the following strategies:

- Our NNs are trained using a novel feature called CTU-based LPTCM derived from LPTCM. This model proved its success in video content analysis [62, 110, 19].
- We propose a novel training strategy to resolve the big data challenge, allow for adaptivity, and maximize our gains. Specifically, we periodically segment the input video to online train the NNs and adapt them afterwards. These NNs are only trained offline at low resolutions.
- In both online and offline training, we define a new RD-weighted cross entropy loss and use it to maximize the accuracy.
- To control the bitrate loss and increase the time savings (TS), we selectively listen to our NNs based on accuracy constrains.

4.2 Fully Connected Network for HEVC's CU Partition Problem

In this section, we explain the main components of our algorithm to classify between two classes: skip and termination. The skip class, or class 1, means to skip the current CU size mode check and jump directly to the next depth in the CTU, whereas the termination class, or class 0, means to stop the RDO search and move on to the next CU. CU labelling is done by running HEVC's reference software (HM 16.0) and getting its optimal CTU structure. All our NNs are designed via one of the strongest NN design platforms, Google's Tensorflow [15].

4.2.1 Feature Extraction: CTU-based LPTCM

A lot of effort has been done in the literature to estimate the statistical distribution of images in the DCT frequency domain. Most of those distributions suffer from the so-called *heavy tail* phenomenon which happens when the tail portion decay faster than the

actual distribution of the frequency coefficients of the image. Transparent Composite Model (TCM) solves this problem by modeling the tail frequency coefficients separately from the central portion [134]. A uniform distribution is used to model the tail portion while the main portion is modeled using a parametric distribution. To balance between simplicity and accuracy, the Laplacian distribution is used to fit the main portion of the image frequency coefficients. The Laplacian Transparent Composite Model (LPTCM) is given by:

$$p(y|y_c, b, \lambda) \triangleq \begin{cases} \frac{b}{1-e^{\frac{-y_c}{\lambda}}} \frac{1}{2\lambda} e^{\frac{-|y|}{\lambda}} & \text{if } |y| < y_c \\ \frac{1-b}{2(a-y_c)} & \text{if } y_c < |y| \leq a \\ \max\left\{ \frac{b}{1-e^{\frac{-y_c}{\lambda}}} \frac{1}{2\lambda} e^{\frac{-|y|}{\lambda}}, \frac{1-b}{2(a-y_c)} \right\} & \text{if } |y| = y_c \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

where $0 \leq b, \leq 1$, $0 < y_c < a$, and a represents the largest magnitude an input sample y can take. As seen by equation 4.1, the LPTCM separates the tail of the input DCT coefficients from its main body. It models the tail of the input DCT coefficients with a uniform distribution while a Laplacian distribution is used to model the DCT coefficients in the main body. According to [136], LPTCM offers very high modelling accuracy with simplicity similar to those of pure Laplacian models.

Based on LPTCM, we create a feature extraction (FE) method that consists of four main steps as shown in Figure 4.1. First, we divide each input CTU into 4x4 non-overlapping blocks and transform them by a 4x4 DCT to get a coefficient map for the frame. Second, for each 64x64 CTU, we collect 15 AC coefficient vectors corresponding to all 4x4 coefficient blocks in this 64x64 CTU. Third, we model these vectors via LPTCM and produce 15 LPTCM models. Each model segregates the DCT coefficients in the main portion, inliers, from the DCT coefficients in the tail portion, outliers, via y_c. As stated in [134], outliers carry important information about the image. Therefore, our fourth step is setting all inliers to zero and quantizing the outliers to one based on the y_c of the given LPTCM model. Then, for each w*w CU, we use its corresponding inliers and outliers as features, while ignoring the DC coefficients from DCT [134]. This procedure is repetitively done for all CTUs in a given input frame. For instance, Figure 4.2 illustrates the produced features for the zeroth picture order count (POC) of Racehorses video sequence at QP = 22. In this figure, highly textured regions, such as the front horse's head, are generally characterized by more outliers and typically encoded with smaller blocks, while the opposite is true for homogeneous regions as the CU in the top left corner.

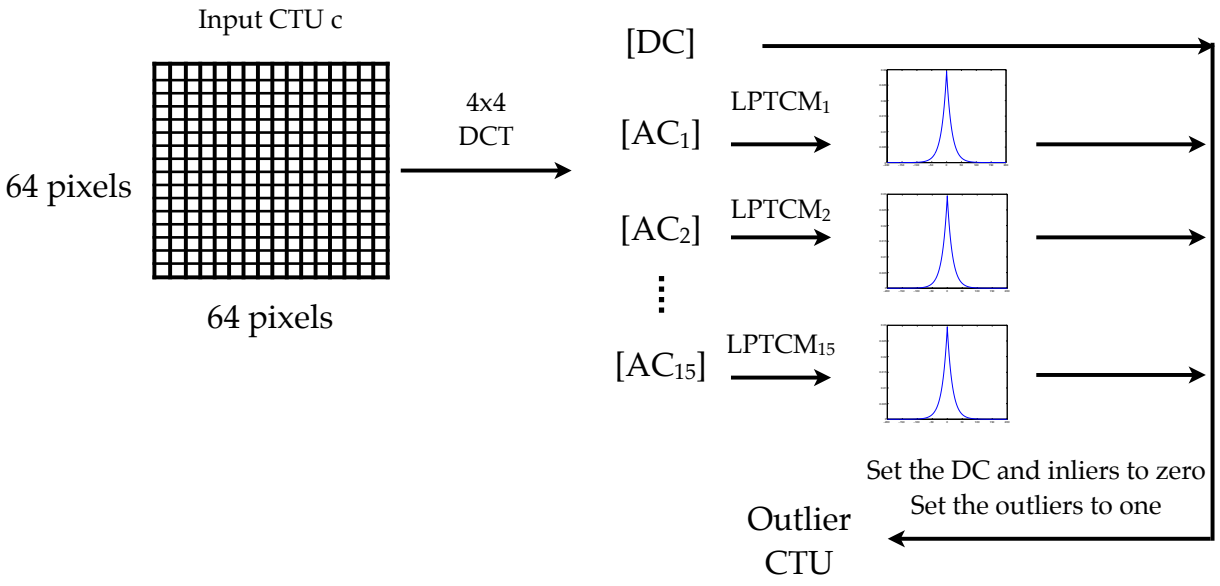


Figure 4.1: Overview on the CTU-based LPTCM Feature Extraction Method.

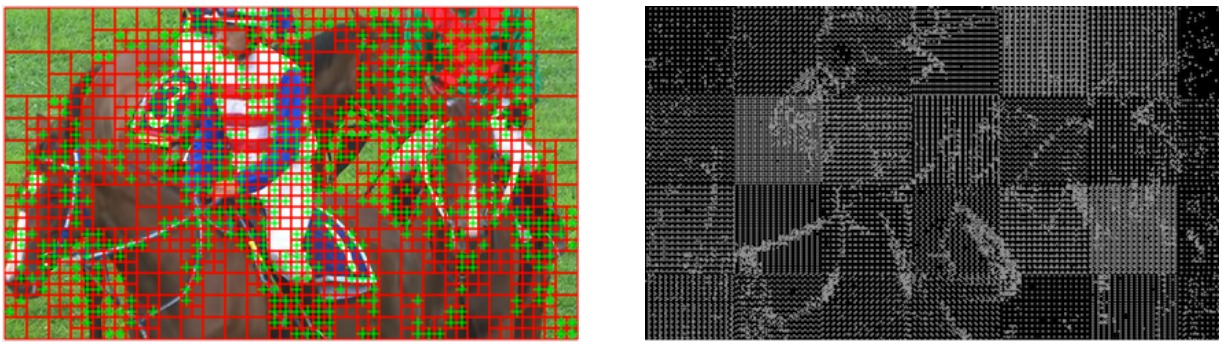


Figure 4.2: *Left:* Original POC = 0 Racehorses at QP = 22 *Right:* CTU-based LPTCM POC = 0 Racehorses at QP = 22. Black = Inliers and DC, White = Outliers

4.2.2 Neural Network Structure

As shown in Figure 4.3, our NNs are fully connected and consist of an input layer, two hidden layers, and an output layer. Here, only two hidden layers were employed to align with HEVC's time complexity demands. To avoid loss of information, the number of neurons in the input layer is adaptive to the CU input size and equal to $w * w$, where w is the width of the CU. Next, the first hidden layer consists of 32 neurons, while the second hidden layer encompasses two neurons for skip and termination classes. At the end, there is a softmax layer to compute a 2-dimensional vector p corresponding to the probabilities of the two classes for each CU c, where $1 \leq c \leq n$, and n denotes the total number of input CUs.

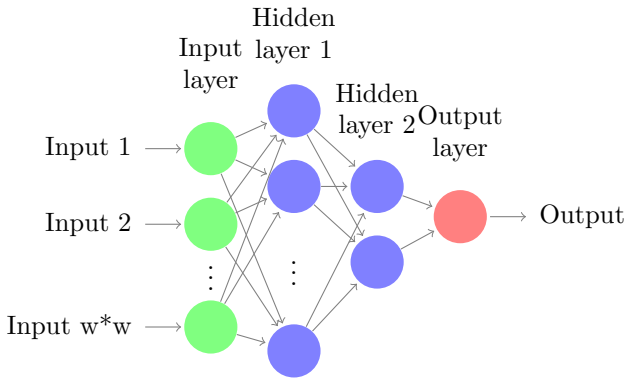


Figure 4.3: Proposed Fully Connected Neural Network Structure.

For each CU, we define an equivalent size feature vector x corresponding to its CTU-based LPTCM output. Given that W_1, W_2, b_1, and b_2 are the weights and biases of hidden layer one and two, respectively, we compute the output probabilities as follows:

$$o_1 = ReLU(W_1^T x + b_1), \qquad o_2 = W_2^T o_1 + b_2 \tag{4.2}$$

$$p = softmax(o_2) \tag{4.3}$$

Here o_1 and o_2 are the outputs from the first and second hidden layers, respectively. $ReLU$ is the rectifier linear unit activation function defined as $ReLU(x) = max(0, x)$, and $softmax$ is the softmax activation. Softmax computes for every CU c a normalized probability vector as $p_{ck} = \frac{e^{o_{2_k}}}{\sum_j e^{o_{2_j}}}$, where $1 \leq c \leq n$ and k is 0 or 1.

Figure 4.4 shows the encoded structure of the second 64x64 CTU in the first row of POC = 0 in Racehorses at QP = 22. As shown by the figure, despite the existence of homogeneous regions that are visually similar such as the green grass, the RDO's optimal structure for some of these regions require smaller blocks than others. Normally, focusing the neural network training on CUs with major differences in terms or R-D cost is important to the overall trade-off between the overall coding efficiency loss and encoding time [88]. Therefore, we train our NNs with a weighted cross entropy (CE) loss to accurately predict CUs with significant difference in R-D cost between the skip and termination classes. We define this loss for each CU c as follows:

$$L_c = -w_c * \sum_{i=1}^{K} t_{ci} \ln p_{ci} \tag{4.4}$$

where L_c is each CU's loss, w_c is defined as $\frac{C_{2N}-C_N}{C_{2N}+C_N}$, C_{2N} is the RD cost for the CU encoded as 2Nx2N, C_N is for the NxN mode, and K is the total number of classes (two in our case). Here, t_{ci} is either 1 or 0. If $t_{ci} = 1$, then $c \in$ class one, and $t_{ci} = 0$, otherwise. From (4.4), our NNs are penalized when they misclassify CUs with significant difference between 2Nx2N and NxN modes, which controls the resulting bitrate loss as will be shown by our experiments.

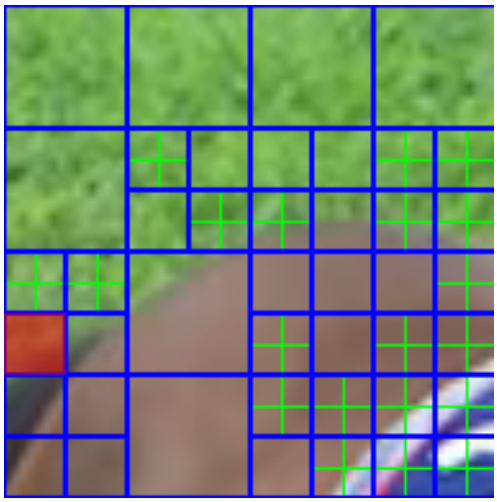


Figure 4.4: Encoded Structure of the second 64x64 CTU in the first row of POC = 0 in Racehorses at QP=22

4.2.3 Offline Training Stage For Low Resolutions

In general, enough data samples are necessary for training any NN to have better estimation of the weight parameters. As a result, at low resolutions when there are not enough samples, we employ an offline training method in order to obtain initial NN models. This training begins with partitioning the input video into periodic segments $0 \cdots i \cdots N$ of 60 frames (P = 60). Next, we obtain an initial NN model, $NN(i = 0)$, for each of the four depths by considering the first 30 frames as training (T = 30), and the next 30 frames as validation (V = 30). For all QPs, initial NNs are created for all CUs in 416x240 videos and for only 16x16 and 8x8 CUs in 832x480 sequences. For these initial NNs, a learning rate equals to 0.001 was used.

4.2.4 Online Training/Adaptation Stage

Enabling adaptivity in our algorithm is done through online training and adaptation of neural networks created within segments as shown in Figure 4.5. For any input video sequence, we set P to 60, but T and V are set to 2 and 1, respectively. To encode the current segment i, we restore the parameters for $NN(i-1)$ that is previously trained by segment $i-1$ to leverage the temporal similarities among video segments. At the start, if $i = 1$ and we have offline-trained models, we consider them as a starting point. Adaptation of $NN(i-1)$ to suit the current segment i is done via the first two frames in segment i to produce a new model $NN(i)$ used for testing. A relatively slow learning rate equals to 0.0001 was used to avoid drifting the initial offline weights.

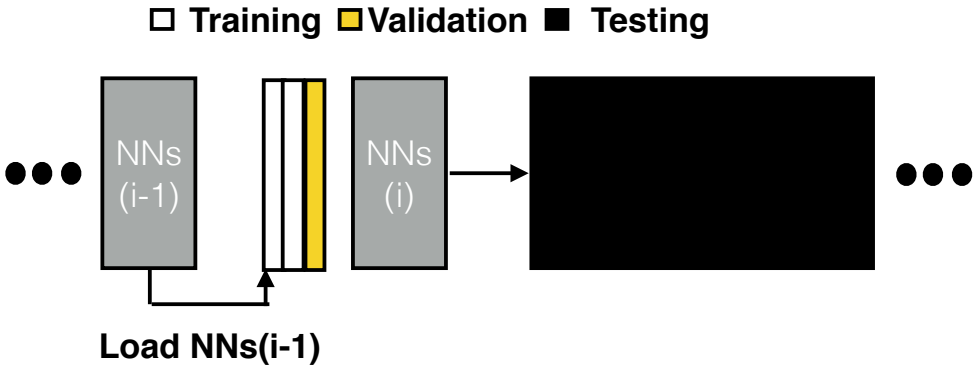


Figure 4.5: Training Strategy for the CU Partition Problem.

To show the importance of adaptation, we carried out an experiment on five standard

HEVC sequences: Traffic, Kimono, RacehorsesC, RacehorsesD, and FourPeople. In this experiment, we executed HEVC default intra coding and recorded the percentage of encoded CUs at each depth from 0 to 4 and each QP from 22 to 37 with a step size of 5. Figures 4.6, 4.7, 4.8, 4.9, and 4.10 show these percentages for Traffic, Kimono, RacehorsesC, RacehorsesD, and FourPeople, respectively. These depth percentages show the different behaviour of HEVC's intra coding at each depth and each QP for each input sequence. Therefore, online training and adaptation for our neural networks is important.

(a) Traffic @ QP = 22

(b) Traffic @ QP = 27

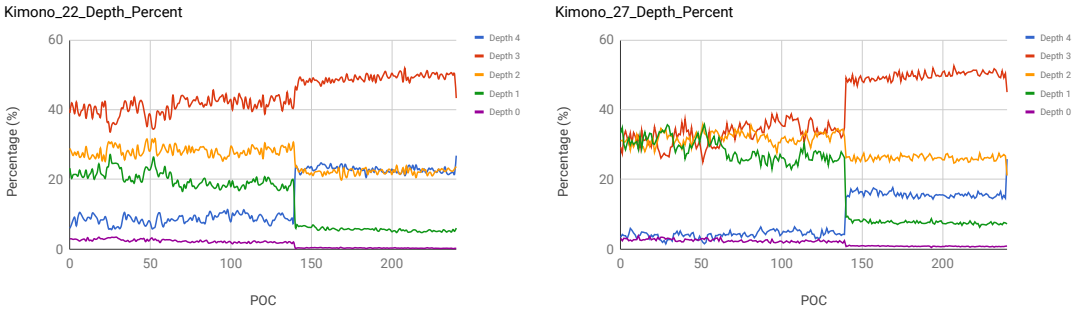
(c) Traffic @ QP = 32

(d) Traffic @ QP = 37

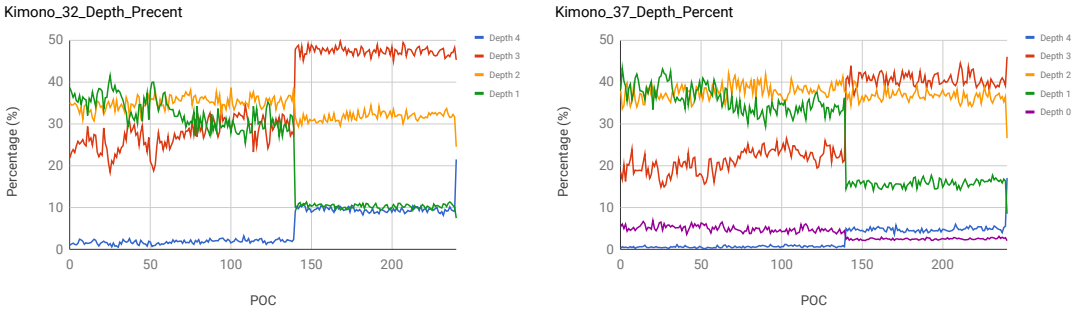
Figure 4.6: Percentages of each depth for Traffic video sequence using the default HEVC Intra Coding.

4.2.5 Online Testing Stage

Accuracy statistics, represented via recall (R) and precision (P), are collected every segment in the validation stage to minimize the resulting bitrate loss and maximize R and P

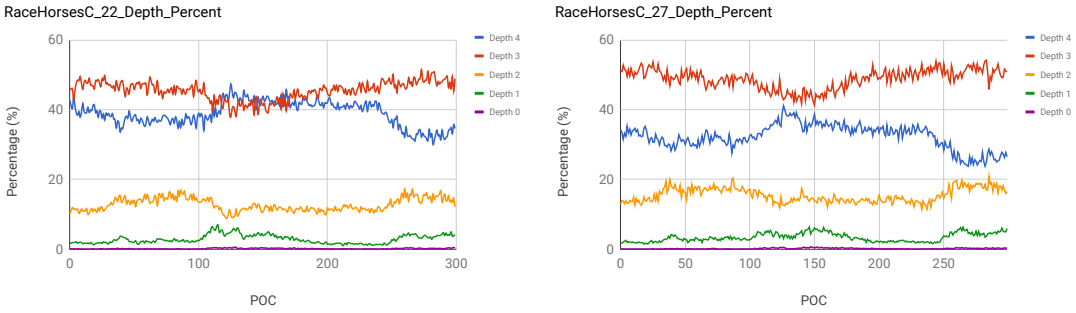


(a) Kimono @ QP = 22 **(b)** Kimono @ QP = 27



(c) Kimono @ QP = 32 **(d)** Kimono @ QP = 37

Figure 4.7: Percentages of each depth for Kimono video sequence using the default HEVC Intra Coding.



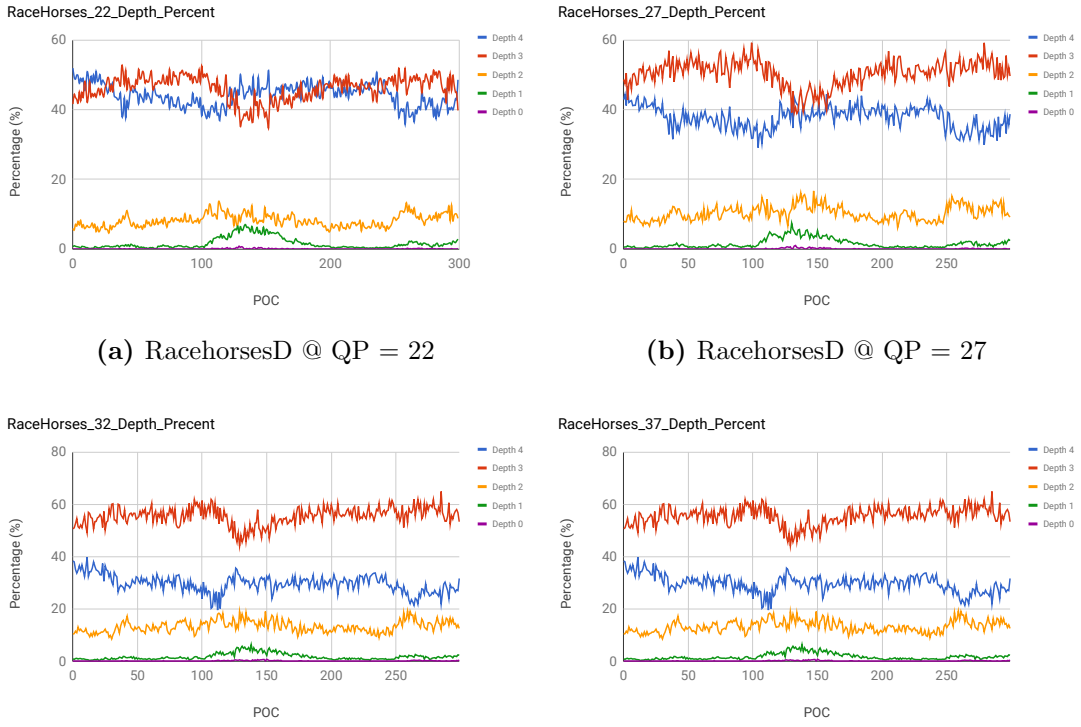
(a) RacehorsesC @ QP = 22

(b) RacehorsesC @ QP = 27

(c) RacehorsesC @ QP = 32

(d) RacehorsesC @ QP = 37

Figure 4.8: Percentages of each depth for RacehorsesC video sequence using the default HEVC Intra Coding.



(a) RacehorsesD @ QP = 22

(b) RacehorsesD @ QP = 27

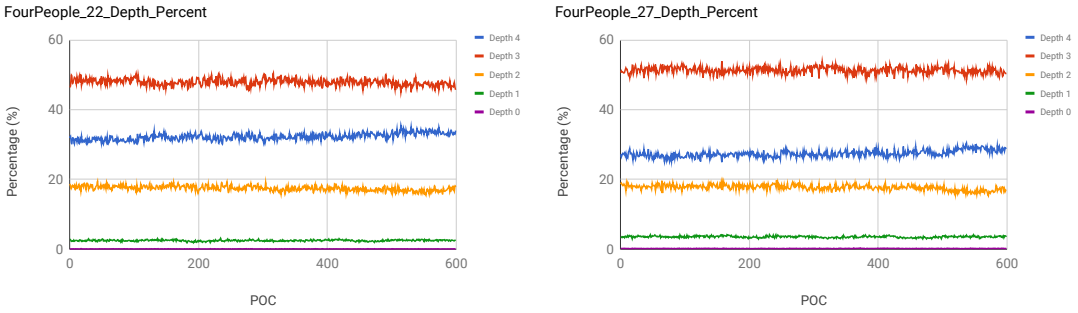
(c) RacehorsesD @ QP = 32

(d) RacehorsesD @ QP = 37

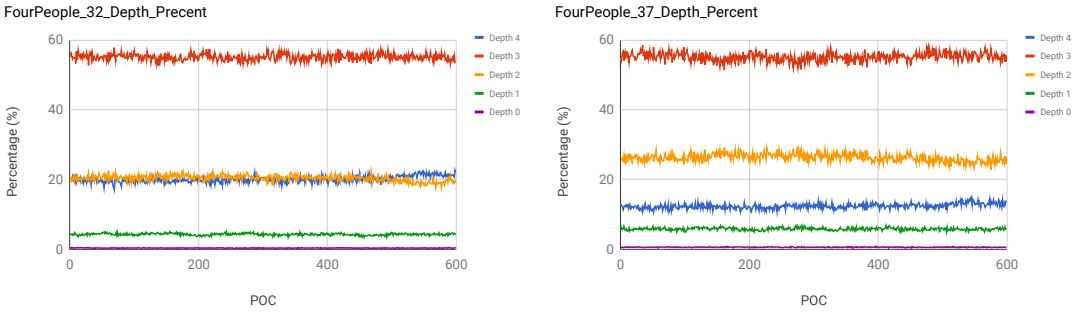
Figure 4.9: Percentages of each depth for RacehorsesD video sequence using the default HEVC Intra Coding.

of our skip and termination classes. R is the percentage of true decisions made for class 0 or 1 detected during the experiment and P is the percentage of detected decisions for class 0 or 1 that are actually correct. Now, assume that the probability vector $p_c = (a_0, a_1)$ is obtained by the NN for a CU c at depth d. Here, a_0 is the probability of the termination class, whereas a_1 is for the skip class. Also, suppose that pre_0 and pre_1 are the overall precision of the NN at depth d for termination and skip classes, respectively. The default execution mode for any CU is done via the default HM, but any CU executes a termination mode only if $a_0 > t_0$ & $pre_0 > g_0$ is satisfied. On the other hand, any CU does the skip mode if $a_1 > t_1$ & $pre_1 > g_1$. The thresholds t_0, t_1 are used to boost the precision so that the majority of the predicted decisions are correct. Also, we listen to our NNs if their precision exceed g_0, g_1.

After extensive simulations, we derived Algorithm 4 and Look-up Table 4.1 to select



(a) FourPeople @ QP = 22 **(b)** FourPeople @ QP = 27



(c) FourPeople @ QP = 32 **(d)** FourPeople @ QP = 37

Figure 4.10: Percentages of each depth for FourPeople video sequence using the default HEVC Intra Coding.

t_0, pre_0, t_1, pre_1, and g_0, g_1, respectively. In Algorithm 4, for class 0 and 1, we compute a recall list (r_l) and precision list (p_l) for *thresholds* from 0.5 to 0.8 with an offset equals to 0.1 and from 0.8 to 0.99 with an offset equals to 0.01. This offset setting is because R and P do not vary much for thresholds between 0.5 and 0.8.

Afterward, we look for the best index, o_{id}, to *thresholds* and p_l that has a good tradeoff between the bitrate loss represented by P and TS represented by R. Hence, we output t_0, pre_0 and t_1, pre_1 that either produce the maximum precision with index mp in the lists, or the maximum weighted sum of R and P. Max does this maximization with a second parameter as the lower bound for precision. For g_0 and g_1, they are set to 80% by default. However, to balance between the tradeoff between R and P, we change g_0 and g_1 at QP=22 or 37 when the CU width, w, meet Table 4.1's conditions in the corresponding HEVC classes, otherwise g_0 and g_1 are set as default (N/C).

Algorithm 4 Refinement Algorithm: Get t_0, pre_0 or t_1, pre_1

procedure searchThresholds
  Input: r_l, p_l
  Output: t_0, pre_0 or t_1, pre_1
  $c = 0.4$, $o_{id} = mp$
  if Class C, D, E **then** x = 65 **else** x = 100
  if $p_l(mp) \geq 80$ & $r_l(mp) < x$ OR
    $70 \leq p_l(mp) < 80$ & $r_l(mp) < 70$ **then**
    if $p_l(mp) \geq 80$ **then**
      if $416 \leq W < 832$ **then**
        $o_{id} = Max(c * r_l + (1 - c) * p_l, 80)$
      else $o_{id} = Max(r_l, 80)$
    else
      $o_{id} = Max(c * r_l + (1 - c) * p_l, 70)$
  if $r_l(o_{id}) < 45$ & $80 \leq p_l(o_{id}) < 83$ **then**
    if $p_l(o_{id}) \geq 80$ **then**
      $o_{id2} = Max(c * r_l + (1 - c) * p_l, 77)$
    else $o_{id2} = Max(c * r_l + (1 - c) * p_l, 70)$
  if $r_l(o_{id2}) > r_l(o_{id}) + 10$ **then** $o_{id} = o_{id2}$
  return $thresholds(o_{id}), p_l(o_{id})$

From our NNs, we found that incrementally obtaining the labels corresponding to each CU is inefficient. If labels are incrementally obtained, it takes about 500 milliseconds

to compute all the labels for the first frame in Racehorses (416x240 pixels). This high complexity is due to the large cost of the matrix multiplication operations. Figure 4.11 shows the timing diagram of the internal operations of the NN for one CU in the first frame of Racehorses at QP=22, where the majority of the time is spend on matrix multiplications. At QP = 22, it is important to note that this frame's encoding time using the original HEVC is ≈1.2 seconds. Adding the label-fetch overhead can diminish the possibility of time savings. Given that our features are available prior encoding, we overcome this challenge in the testing stage by leveraging these features and sending them in batches to NNs prior encoding to prepare the labels for all possible CUs and use them to encode the frame. This batching strategy enabled executing the matrix multiplications for all CUs all at once in parallel and in turn minimized the label-fetch time to ≈2-9 milliseconds on the same video frame, which enables the possibility of time savings as shown in section 4.3.

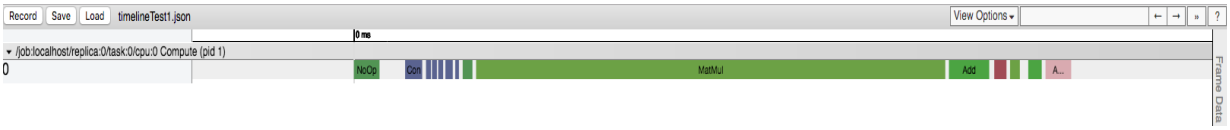


Figure 4.11: Tensorflow Timing Diagram for one sample of Racehorses video sequence at QP=22

Table 4.1: Look-up Table for g_0, g_1 when QP=22 or 37. N/C: No Change

	w=8	w=16	w=32
Class A	g_1=70, **if** QP=22 **then** g_0=70		
Class B, C, E	g_0, g_1=70		N/C
Class D	g_1=70	N/C	

4.3 Experimental Results

In this section, we demonstrate that our method is among the best NN methods with less overall BD-rate loss (BR), and it has a potential over other methods in the literature. All experiments are carried out on top of the intra_main configuration for HM 16.0 test model with QP=22, 27, 32, 37. All experiments are executed on a 16GB memory AMD six-core machine, where standard HEVC sequences were used for testing.

Table 4.2 shows the BR and TS for our technique. As seen in this table, we can conclude that our method can provide high time savings, 38% on average, while keeping the BD-rate

loss under control, 1.6% on average. Also, our method performs obviously better in high motion sequences than in low motion sequences as in Class E. To compare the CU partition structure of our fast CU algorithm vs HEVC, Figure 4.12 shows the CU split structure of frame 20 of the Traffic video sequence at QP = 22, 37.

Table 4.2 shows the detailed results for BR and TS for our technique. As seen in this table, we can conclude that our method can provide high time savings, 32% on average, while keeping the BD-rate loss under control, 1.6% on average. Also, our method performs obviously better in high motion sequences than in low motion sequences as in Class E. To get some insight about this phenomenon, we extracted the CU structure of the default HM at different QPs and compared it to the CU structure produced by the HM encoder equipped with our NN algorithm for the FourPeople sequence. For the majority of the smooth regions, Figure 4.13 indicate that our NNs tend to skip the current depth of the tree and go to smaller CU sizes. This behaviour will be effective for sequences that have large motion as Traffic, but may not be as effective for sequences characterized by low motion as FourPeople.

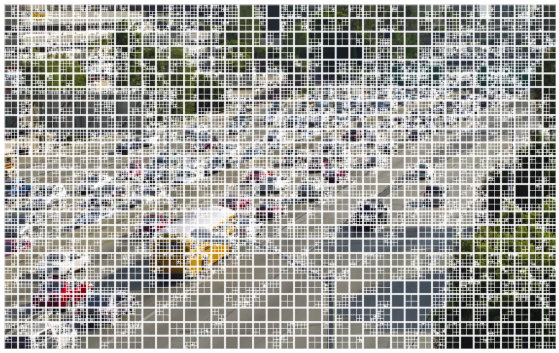
We compare our method against others from the literature in Table 4.3. It is clear that due to our adaptive training strategy and LPTCM features, our method provides TS while controlling the BR compared to the other NN methods [132, 88]. Similarly, compared to the first SVM approach [87], our method clearly controls the BR while providing time-savings. Because [110] utilizes multiple SVMs as classifiers, the method performs better; for this reason, multiple classifiers could be an interesting extension for our method in the future. Looking at [62] from TS perspective, our method has 39%-40% average TS for classes A, C, while it is 37% for [62]. In addition, our method's TS start from 35% and can go up to 60% for high resolution sequences, which beside our overall results show the effectiveness of the present method. Due to our online training/adaptation, we also believe that our methods could be also extended to inter-frame coding mode decision where inter-frame dependencies among frames and scene changes exist.

4.4 Chapter Summary

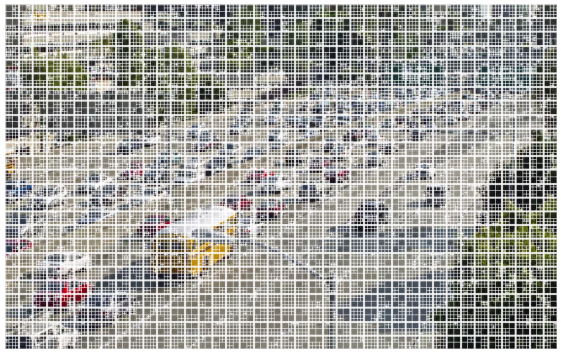
This chapter presented a new CU partition prediction method equipped with hierarchical fully connected NN models and features from LPTCM. We proposed some adaptive training strategies to counteract the challenges of equipping NN online learning with HEVC. Experimental results demonstrate that our technique is among the best NN methods with controlled BD-rate loss and of comparable performance to others. Our technique achieves 32% TS average with an 1.6% BR average. In this chapter, a fully connected neural network

(a) HM CU Split at QP = 22

(b) NN CU Split at QP = 22



(c) HM CU Split at QP = 37



(d) NN CU Split at QP = 37

Figure 4.12: Traffic CU partition comparisons between HM-16.0 and fast algorithm. POC = 20 (Best viewed in electronic format).

(a) HM CU Split at QP = 22

(b) NN CU Split at QP = 22

(c) HM CU Split at QP = 27

(d) NN CU Split at QP = 27

(e) HM CU Split at QP = 32

(f) NN CU Split at QP = 32

(g) HM CU Split at QP = 37

(h) NN CU Split at QP = 37

Figure 4.13: FourPeople CU partition comparisons between HM-16.0 and fast algorithm. POC = 20 (Best viewed in electronic format).

'saw' the manually extracted LPTCM features from an input image to make a classification decision to help reduce the computational intensity of compression at a controlled trade-off between compression rate and compression distortion. In the next chapter, we turn to computer vision where we utilize much deeper CNNs that 'see' images to extract features and recognize objects in these images, and formulate a new framework to investigate the impact of JPEG compression on deep learning (DL) in image classification for computer vision perspective.

Table 4.2: BD Rate Loss and Time Savings (TS) Percentage for the Standard HEVC Sequences.

Class	Sequence	BR loss (%)	TS (%)
A	Traffic	1.13	33
	PeopleOnStreet	2.4	44
AVERAGE		**1.76**	**39**
B	BasketballDrive	1.59	40.7
	BQTerrace	1.97	31
	Cactus	0.9	28.5
	Kimono	1.7	54
	ParkScene	0.2	21.4
AVERAGE		**1.2**	**36**
C	BasketballDrill	2.1	40.7
	BQMall	2.4	40.12
	PartyScene	0.96	37
	RaceHorses	1.8	40.27
AVERAGE		**1.8**	**40**
D	BasketballPass	2.1	22.12
	BlowingBubbles	0.78	37
	BQSquare	1.8	37
	RaceHorses	1.6	36
AVERAGE		**1.57**	**34**
E	FourPeople	1.1	13
	Johnny	2.9	6
	KristenAndSara	3.7	7.3
AVERAGE		**2.5**	**9**
Overall AVERAGE		**1.6**	**32**

Table 4.3: Comparison between the Proposed Method and Methods from the Literature.

Source	BR loss(%)	TS(%)
CNN-[88]	2.6	58
CNN-[132]	2.2	62
SVM-[87]	2.2	46.5
SVM-[110]	0.78	48.03
Baye's-[62]	0.46	39.29
Proposed	1.6	32

Chapter 5

Compression Helps Deep Learning In Image Classification

In Chapters 3 and 4, we focused our efforts on the human vision perspective. For example, in Chapter 4, we have proposed a fully connected neural network that 'sees' manually extracted LPTCM features to make a classification decision to help reduce the computational intensity of compression at a controlled trade-off between compression rate and compression distortion. In this chapter, we turn to machine vision where convolution neural networks directly 'see' the input image coming from JPEG compression and formulate a new framework to investigate the impact of JPEG compression on deep learning (DL) in image classification. Section 5.2 provides a case study that motivates this new framework, while Section 5.1 reviews the related work. In contrast to the conventional understanding that JPEG compression generally hurts the classification accuracy of DL, we show in Section 5.3 that our framework, for any original image, allows one to select, among many JPEG compressed versions of the original image including possibly the original image itself, a suitable version as an input to the underlying DNN, which helps improve the classification accuracy of the underlying DNN while the size in bits of the selected input is, on average, reduced dramatically in comparison with the input original image. Therefore, compression, if used in a right manner, helps DL in image classification. Section 5.4 also shows that within this framework other selectors can be designed to maintain the classification accuracy and reduce the size in bits of the selected input relative to the original image. Finally, Section 5.5 concludes the chapter.

5.1 Literature Review

Deep learning (DL) is becoming increasingly ubiquitous in the task of image classification due to its ability to extract desired features from raw data [70, 109, 65, 28, 112, 106, 122, 56, 120, 31]. DL is created through cascading non-linear layers that progressively produce multi-layers of representations with increasing levels of abstraction, starting from the raw input data and ending with the predicted output label [23, 141, 112, 73, 122, 59, 49, 91]. These multi-layers of representations are features not designed by human engineers with considerable domain expertise, but learned from the raw data through a backpropagation learning algorithm.

In image classification, the raw data fed into a DL machine is the pixel values of an image to be classified. Note that the meaning of raw data in the context of DL here is with respect to subsequently extracted features, but not in the context of compression. In the whole pipeline of data acquisition, data encoding (i.e., compression), data transmission, and data processing/utilization, the raw data fed into a DL machine is not "raw"; instead, it is generally compressed in a lossy manner. Since lossy compression is about the trade-off between compression ratio (CR) and compression quality, many versions of compressed raw data in the context of DL can be produced with each version having a different compression ratio and compression quality. This in turn brings forth the following interesting question to DL:

Question 1 which version of compressed raw data is good to DL and its related applications?

In practice, images are often compressed by JPEG encoders [10, 127, 98]. For most practical applications with JPEG, both the CR and compression quality of a JPEG image are controlled by a parameter called the quality factor (QF); the higher the QF, the lower the CR and the better the compression quality. With the maximum value of QF at 100, the majority of JPEG images in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 dataset [36, 105] have high QF values ranging from 91 to 100, implying that they all have high compression quality.

In the literature, Question 1 was investigated to some extent on the basis of constant QFs which are the same for all images in a whole set of JPEG images [66, 40, 145, 47, 50, 90], as shown in Figure 5.1. Specifically, four deep neural network (DNN) models were tested in [40] on a subset of the validation set of the ILSVRC 2012 dataset [36]. To evaluate the impact of compression on the classification performance of these four DNN models, all images in the subset were further compressed by JPEG with the same constant QF. These

compressed images with the constant QF were then fed into each of these four DNN models. Both the top-1 classification accuracy and top-5 classification accuracy were recorded. As the value of the constant QF decreases, curves of the top-1 classification accuracy vs QF and the top-5 classification accuracy vs QF were plotted in [40] in the QF range from 20 to 2. It was shown in [40] that both the top-1 classification accuracy and the top-5 classification accuracy of each of the four DNN models decay as the value of the constant QF decreases. This phenomenon of negative impact of compression on the classification performance of DNN models was also reported in [90].

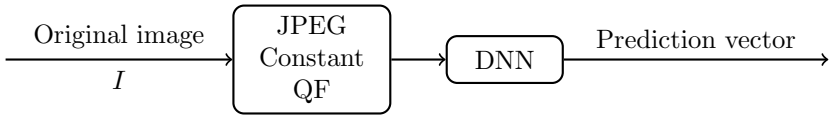


Figure 5.1: A DNN with a JPEG compressed version of an image as an input, where QF is a constant.

To alleviate the negative impact of JPEG compression on the classification performance of DNN models to some extent, several methods were proposed in the literature, including data augmentation, stability training, and due-channel training with preprocessing [145, 139, 90, 42, 24, 25]. For example, stability training was proposed in [145], where during the training stage of a DNN model, both the original image and its distorted version are fed into the model, and training is performed to minimize a modified cost function which takes stability into consideration. Although these methods improve the classification robustness of DNN models against JPEG compression and other types of distortion, there is still a significant degradation (as high as 10%) in classification accuracy when these newly trained DNN models are applied to low quality JPEG compressed images. Based on these findings, it is generally believed that compression, especially JPEG compression, would hurt the classification accuracy of deep learning in image classification.

In this chapter, we investigate Question 1 in the context of JPEG compression from a different perspective. Instead of using a constant QF in JPEG compression for all images in the ILSVRC 2012 dataset, we would allow each image to be compressed first with a possibly different QF and then fed into a DNN. Specifically, let QF take values from $\{100, 90, 80, 70, 60, 50, 40, 30, 20, 10\}$[1]. We associate each original image in the ILSVRC 2012 dataset with its 10 compressed versions, each compressed version corresponding to a different QF from $\{100, 90, 80, 70, 60, 50, 40, 30, 20, 10\}$. For each image, there are now 11

[1]This set of QF values is simply used as an example. The idea of this chapter, however, can be applied to any set of QF values. In addition, QF = 10 is regarded in this example as the lowest compression quality acceptable to human.

different versions: 1 original version plus 10 compressed versions. Fix a DNN. As shown in Figure 5.2, for each original image I, we now have freedom to select one version I_j out of its 11 versions I_i, $i = 0, 1, \cdots, 10$, to be fed into the DNN. Is there any selector that can select, for each original image I, a suitable version I_j to be fed into the DNN so that both the top-1 classification accuracy and top-5 classification accuracy of the DNN can be improved significantly while the size (in bits) of the input image to the DNN can be reduced dramatically in general?

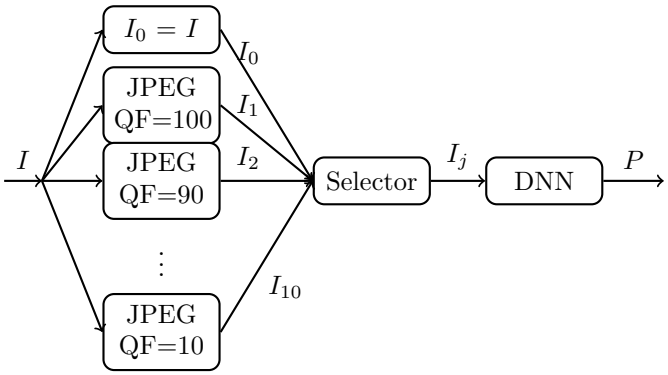


Figure 5.2: Selection of a compressed version of an image as an input to a given DNN, where P is the prediction vector of the DNN in response to the chosen I_j.

One of our purposes in this chapter is to settle the above question. We show that the answer to the above question turns out to be positive. Therefore, in contrast to the conventional understanding, compression, if used in a right manner, actually improves the classification accuracy of a DNN significantly while also reducing dramatically the number of bits needed to be fed into the DNN. Specifically, fix a DNN pre-trained with pristine ImageNet images. That is, the DNN was trained with the original images in the training set of the ILSVRC 2012 dataset. Suppose that the ground truth label of each original image I is known to the selector in Figure 5.2, but unknown to the DNN. Under this assumption, we propose a selector called Highest Rank Selector (HRS). For each original image I, HRS works as follows. Examine the prediction vector P_i of the DNN in response to each version I_i, determine the rank of the ground truth label in the sorted P_i, where labels in P_i are sorted according to their probabilities in descending order with rank 1 being the highest ranking, and then select the compressed version I_j as the desired input to the DNN if the rank of the ground truth label in the sorted P_j is the highest among all sorted P_i, where in the case of tie, HRS selects the compressed version with the lowest QF. It can be shown that among all selectors one could possibly design, HRS achieves the highest

top-1 and top-5 classification accuracy and hence is optimal. When applied to Inception V3 and ResNet-50 V2 architectures pre-trained with pristine ImageNet images [5, 7], HRS improves, on average, the top-1 classification accuracy by 5.6% and the top-5 classification accuracy by 1.9% on the whole ImageNet validation set. In addition, compared with the original image, the compressed version selected by HRS also achieves, on average, the CR of 8.

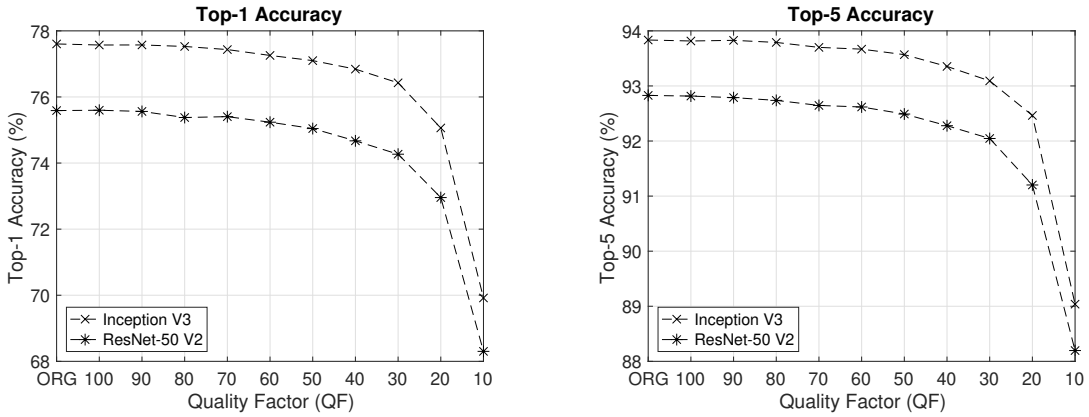
When the ground truth label of each input image is unknown to the selector in Figure 5.2 either, HRS is not applicable. Thus, we propose a selector which can maintain the same the top-1 classification accuracy and top-5 classification accuracy as those of the given DNN with the original image as its input. When applied to Inception V3 and ResNet-50 V2, the compressed version selected by the proposed selector achieves, on average, the CR of 3.1 in comparison with the original image.

5.2 Motivation: Case Study

This section motivates our approach to Question 1 as illustrated in Figure 5.2. To begin with, let us first reproduce results which lead people to the conventional understanding that JPEG compression generally degrades classification performance of DNNs.

The conventional understanding is based on the approach shown in Figure 5.1, where a constant QF is used to compress all images in a whole set of images. This approach can be dubbed as "one QF vs all images". For Inception V3 and ResNet-50 V2 pre-trained with the original images in the training set of the ILSVRC 2012 dataset, Figure 5.3 shows their respective curves of the top-1 classification accuracy and top-5 classification accuracy on the whole ImageNet validation dataset vs the constant QF as the value of the constant QF in Figure 5.1 decreases from 100 to 10 with a step size of 10. From Figure 5.3, it is clear that classification performance deteriorates as the value of the constant QF decreases, hereby reconfirming the conventional understanding.

Note that the concept of classification accuracy is a group notion with respect to a whole set of images. If, however, we focus on a particular image and examine the impact of JPEG compression with different QFs on the predicted vector of the underlying DNN—such a perspective is dubbed as "one image vs all QFs"—the rank and probability of the ground truth (GT) label of the image in the predicted vector do not necessarily go down as the value of QF decreases. This is indeed confirmed by Figure 5.4. With Inception V3 pre-trained with ImageNet pristine images as the underlying DNN, Figure 5.4 shows the ranks and probabilities of the GT labels of image #651 and #37 in the ImageNet validation

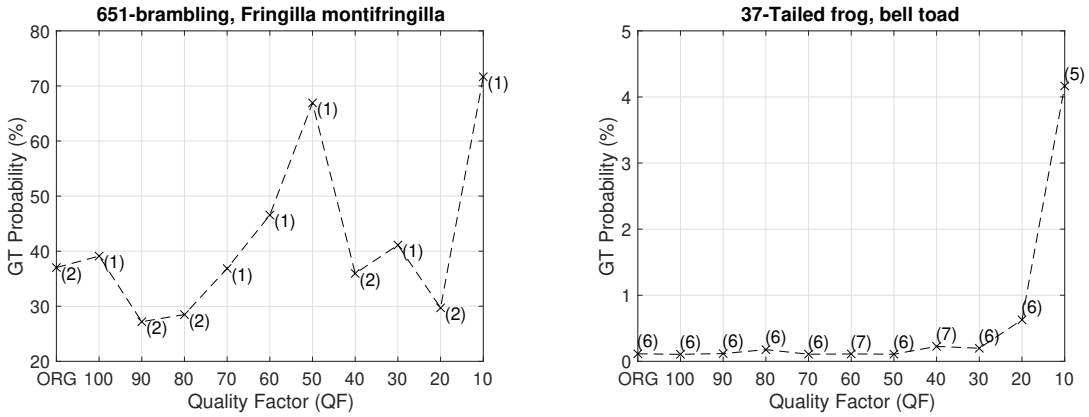


(a) Top-1 Accuracy Trend **(b)** Top-5 Accuracy Trend

Figure 5.3: Top-1 accuracy and Top-5 accuracy degradation phenomenon for Inception V3 and ResNet-50 V2 in the case of the "one QF vs all images" approach.

set as the value of QF decreases. From this Figure, it is clear that for a given image, a JPEG compressed version with a lower QF could yield a higher rank of the GT label and a larger probability of the GT label in comparison with the original image. For example, for image #651 shown in Figure 5.5, when the original image is fed into the underlying DNN, the GT label ranks second with probability 37% in the corresponding predicted vector. On the other hand, when its JPEG compressed version with QF = 10 shown in Figure 5.5 is fed into the underlying DNN, the GT label ranks first with probability 72% in the corresponding predicted vector; both the rank and probability of the GT label are improved. The same phenomenon is observed for image #37 and in the case of the ResNet-50 V2 architecture as well.

To shed light on why for a particular image, both the rank and probability of its GT label resulting from a JPEG compressed version with a low QF could be higher than those resulting from the original image, Figure 5.6 shows a pair of corresponding feature maps extracted from the original image #651 and its JPEG compressed version with QF = 10, respectively, by Layer 1 of Inception V3. In Figure 5.6, the feature map extracted from the JPEG compressed image with QF = 10 is a lot of cleaner and has much better contrast between the foreground and background than the one extracted from the original image. This is likely due to the unequal quantization performed by JPEG on different discrete cosine transform (DCT) coefficients, which is non-linear and reduces more energy in the background than the foreground. This, combined with the subsequent rectified linear unit



(a) Image # 651, GT label: Brambling

(b) Image # 37, GT label: Tailed Frog

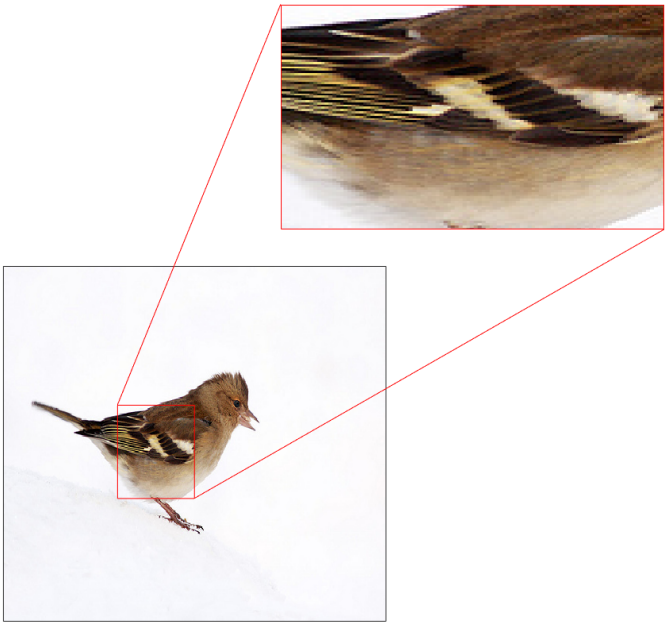
Figure 5.4: The perspective of one image vs all QFs—the ranks and probabilities of the GT label of an image across different QFs: (a) image # 651; (b) image # 37.

(ReLU) function in Inception V3, essentially wipes out the background information.

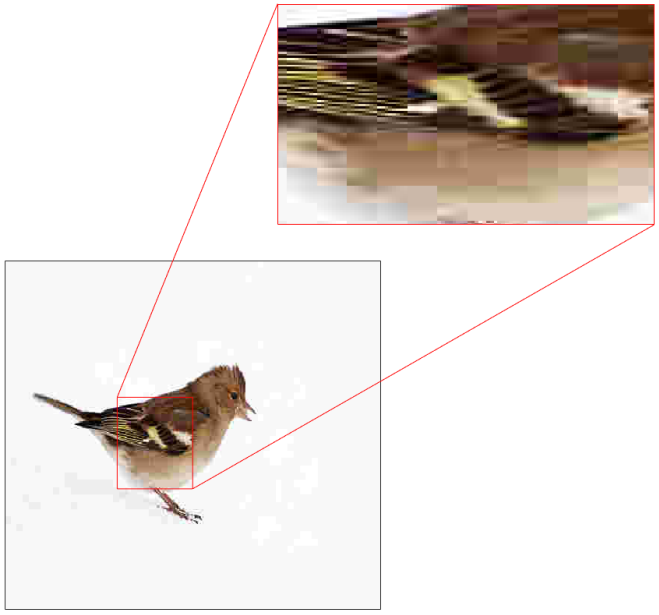
The above case study suggests that if for any image, one can select, among its many compressed versions including its original version, a suitable version as an input to the underlying DNN, then the classification accuracy of the underlying DNN could be improved. In addition, if a highly compressed version is selected most of time, then the size in bits of the input is also reduced dramatically in comparison with the original image. The question, of course, is how to select such a compressed version, which is addressed in the next section when the GT label of the image is known to the selector.

5.3 Highest Rank Selector

With reference to Figure 5.2, in this section, we assume that the GT label of each original image is known to the selector, but unknown to the underlying DNN. We present Highest Rank Selector (HRS) and demonstrate its optimality in the sense of achieving the highest classification accuracy for a given underlying DNN among all possible selectors. We also provide empirical analysis on the performance of HRS in terms of classification accuracy improvement and compression ratio for Inception V3 and ResNet-50 V2 pre-trained with the original images in the training set of the ILSVRC 2012 dataset.

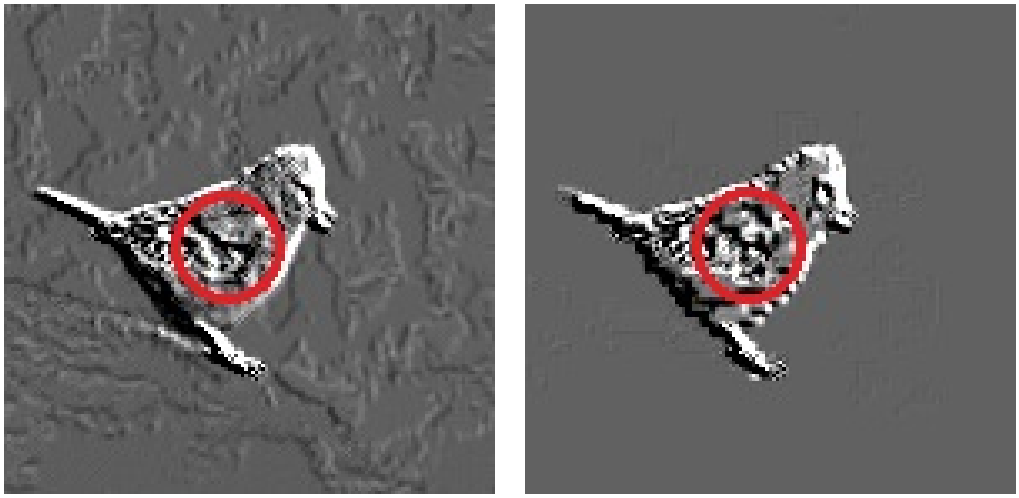


(a) Original Image



(b) QF=10 Image

Figure 5.5: Image #651 from ImageNet validation set with its GT GT label "Brambling": (a) the original image for which the GT label ranks second with probability 37%; (b) the JPEG compressed image with QF = 10 for which the GT label ranks first with probability 72%. Best viewed in electronic format.



(a) Original Feature Map **(b)** QF=10 Feature Map

Figure 5.6: Feature Maps extracted from the original image #651 and its JPEG compressed version with QF=10 by Layer 1 of Inception V3. Best viewed in electronic format.

5.3.1 HRS and its Optimality

Fix an underlying DNN. As illustrated in Figure 5.2, each original image I is now associated with 11 JPEG compressed images (including the original image itself) I_i, $i = 0, 1, \cdots, 10$. Let P_i denote the prediction vector of the underlying DNN in response to the input I_i. HRS now works as follows:

Step 1 For each $0 \leq i \leq 10$, determine the rank r_i of the GT label of I in the sorted P_i, where labels in P_i are sorted according to their probabilities in descending order with rank 1 being the highest ranking.

Step 2 Select I_j as an input to the underlying DNN if and only if

$$r_j = \min\{r_i : 0 \leq i \leq 10\} \tag{5.1}$$

where whenever there are multiple is achieving the above minimum, j is selected to be the largest among those is.

Example 1: Let the underlying DNN be Inception V3 pre-trained with the original images in the training set of the ILSVRC 2012 dataset. For image # 651, in view of

Figure 5.4, HRS selects I_{10}, the JPEG compressed image with QF=10 as an input to the underlying DNN. For image # 37, the same is true as well since for image # 37, r_{10} is the smallest as shown again in Figure 5.4.

For any selector S, let $P_S(I)$ denote the prediction vector at the output of the system shown in Figure 5.2 with S as the selector in response to I. Let $r_S(I)$ be the rank of the GT label of I in the sorted $P_S(I)$. For any set of images $\mathcal{A}$, let $A_S(k)$ denote the top-k classification accuracy of the system shown in Figure 5.2 with S as the selector on the image set $\mathcal{A}$. For convenience, $A_S(k)$ will be referred to as the top-k accuracy of the selector S on the image set $\mathcal{A}$ as well in the rest of the chapter. The following theorem implies that HRS achieves the highest top-k accuracy on any image set $\mathcal{A}$ among all possible selectors, and hence is optimal.

Theorem 1. *For any image set $\mathcal{A}$, any selector S, and any k, the following holds*

$$A_{HRS}(k) \geq A_S(k) \tag{5.2}$$

Proof. For any image $I \in \mathcal{A}$, it follows from (5.1) that

$$r_{HRS}(I) \leq r_S(I)$$

which further implies

$$\{I \in \mathcal{A} : r_{HRS}(I) \leq k\} \supseteq \{I \in \mathcal{A} : r_S(I) \leq k\}$$

Therefore

$$\begin{aligned} A_{HRS}(k) &= \frac{|\{I \in \mathcal{A} : r_{HRS}(I) \leq k\}|}{|\mathcal{A}|} \\ &\geq \frac{|\{I \in \mathcal{A} : r_S(I) \leq k\}|}{|\mathcal{A}|} \\ &= A_S(k) \end{aligned}$$

where for any set C, $|C|$ denotes the cardinality of C. This completes the proof of Theorem 1. □

Before we conclude this subsection, let us mention an application scenario where the GT label of the image I is indeed known to the selector in Figure 5.2, but unknown to the underlying DNN. Consider, for example, a party gathering with a lot of participants and high security requirements. Before the party gathering, each invited participant is

requested to provide his/her high quality photo along with his/her identification (ID) information to the party organizer. Upon receiving the photo and ID information of an invited participant, the organizer later on issues to the invited participant a formal invitation letter with the photo (possibly further compressed by JPEG) embedded in a chip. At the time of party gathering, each invited participant will go through security by presenting the invitation letter to an underlying DNN which in turn reads the photo inside the chip of the letter to determine the ID of the invited participant. In this case, the organizer can act as a selector with the knowledge of the ID of each invited participant (i.e., the GT label of the original high quality photo corresponding to the invited participant), which is unknown to the underlying DNN; the photo put inside the chip is the JPEG compressed version selected by the selector. Both the classification accuracy of the selector and the size in bits of the selected JPEG compressed image inside the chip of each letter are important.

5.3.2 Empirical Results and Analysis

Table 5.1 tabulates the top-1 and top-5 accuracy results of HRS on the whole ImageNet validation dataset for Inception V3 and ResNet-50 V2 pre-trained with the original images in the training set of the ILSVRC 2012 dataset, respectively. As shown in this table, the average accuracy improvement for Inception V3 and ResNet-50 V2 is 5.6% in terms of top-1 accuracy and 1.9% in terms of top-5 accuracy.

Table 5.1: Top-1 Accuracy and Top-5 Accuracy of HRS on the whole ImageNet validation dataset.

Underlying DNN	Default Top-1	HRS Top-1	Default Top-5	HRS Top-5
Inception V3	77.6%	83.37%	93.8%	95.79%
ResNet-50 V2	75.58%	80.95%	92.8%	94.64%
Average Diff	-	**5.6%**	-	**1.9%**

Figures 5.7 and 5.8 show the histograms of the QF values selected by HRS for Inception V3 and ResNet-50 V2, respectively. It is observed from Figures 5.7 and 5.8 that in both cases, a JPEG compressed version with a lower QF is selected by HRS more often than its counterpart with a higher QF, and the lowest QF value QF = 10 is selected by HRS more than 76% times. This in turn translates into a dramatic reduction in the size (in bits) of the selected input to the underlying DNN in comparison with the original image, as shown in Table 5.2, where the default size is the total size in Gigabytes (GB) of all original images

in the ImageNet validation dataset, and the HRS size is the total size of all selected images by HRS. The compression ratio achieved by HRS is, on average, 8.

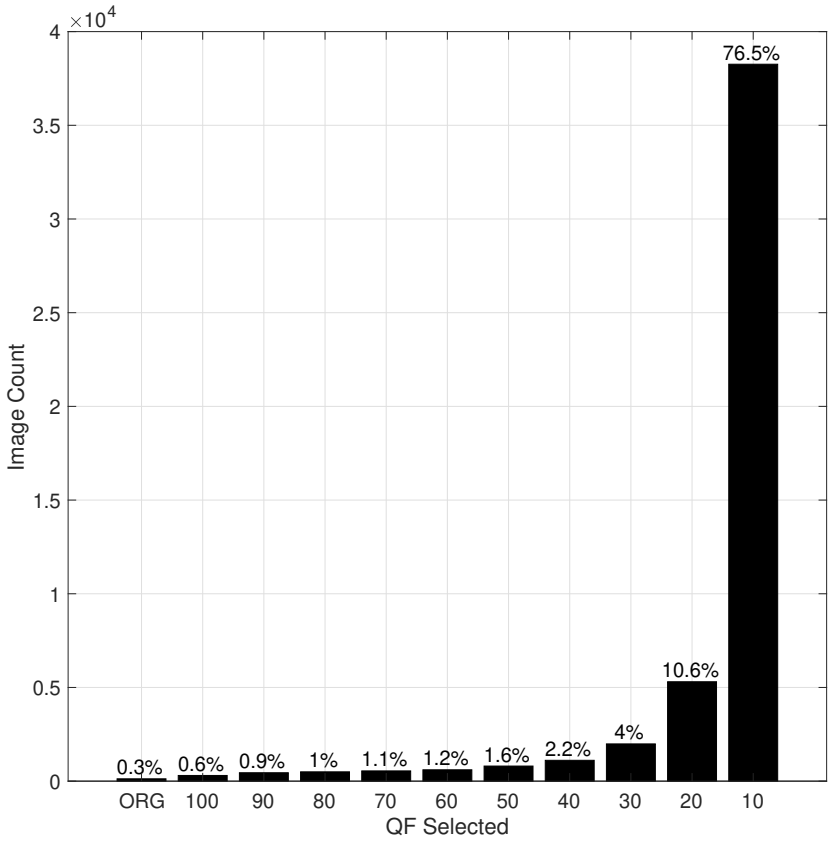


Figure 5.7: The histogram of QF values selected by HRS for Inception V3.

Now take $\mathcal{A}$ to be the whole ImageNet validation dataset. Let D be the default selector which always selects the original image I as an input to the underlying DNN. Note that for any $I \in \{I \in \mathcal{A} : r_{HRS}(I) \leq k\} - \{I \in \mathcal{A} : r_D(I) \leq k\}$, HRS improves the rank of the GT label of I from being below top-k to top-k. To have a better understanding on HRS, let us examine, through examples, feature maps extracted by some layers of the underlying DNN from the original image I and the compressed image selected by HRS, respectively, for $I \in \{I \in \mathcal{A} : r_{HRS}(I) \leq k\} - \{I \in \mathcal{A} : r_D(I) \leq k\}$ [2]. To be specific, take the underlying

[2] https://github.com/HossamAmer12/visualize_inception_featureMaps

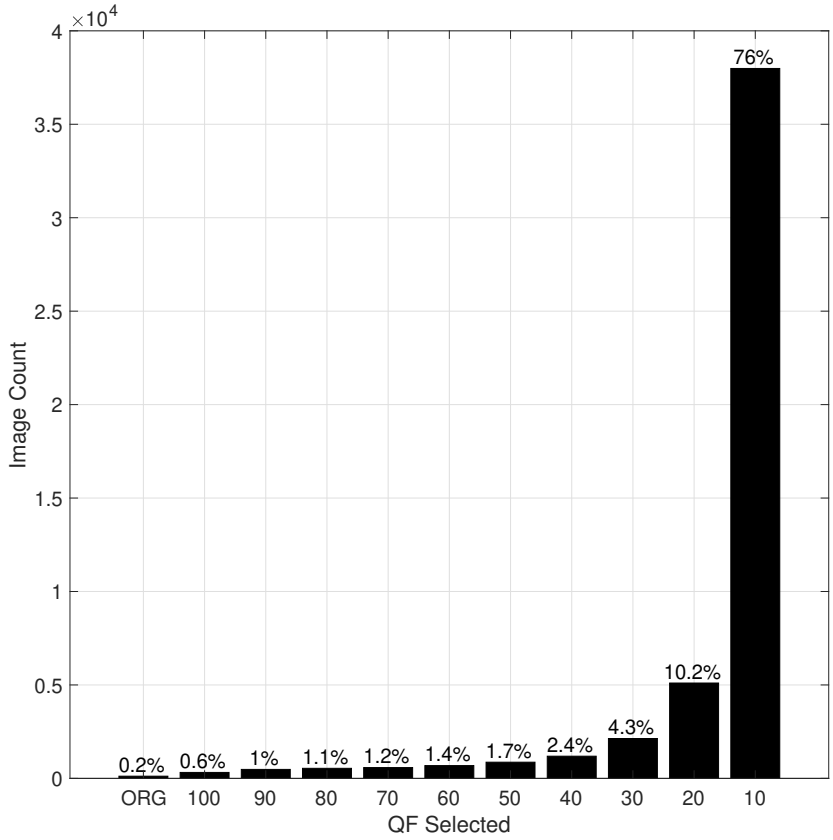


Figure 5.8: The histogram of QF values selected by HRS for ResNet-50 V2.

Table 5.2: Compression performance of HRS for the whole ImageNet validation dataset.

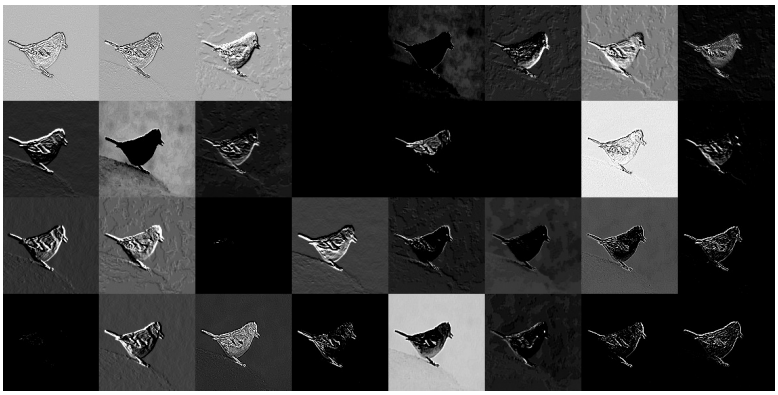
Underlying DNN	Default size (GB)	HRS size (GB)	CR
Inception V3	6.7	0.83	8.1x
ResNet-50 V2	6.7	0.84	8x
Average	**6.7**	**0.84**	**8x**

DNN to be Inception V3 pre-trained with the original images in the training set of the ILSVRC 2012 dataset.

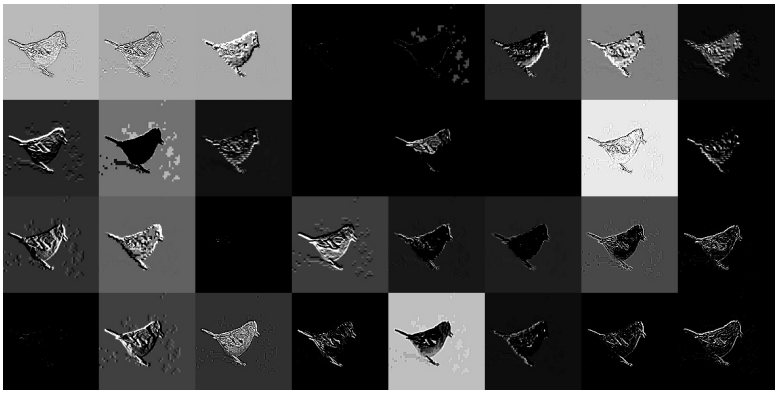
Example 2: Let I be image # 651. In view of Figure 5.4, HRS selects I_{10}, and $r_{HRS}(I) = 1$ while $r_D(I) = 2$. Therefore, $I \in \{I \in \mathcal{A} : r_{HRS}(I) \leq 1\} - \{I \in \mathcal{A} : r_D(I) \leq 1\}$. Figure 5.9 shows feature maps extracted by Layer 2 of Inception V3 from the original image #651 and the compressed image I_{10} selected by HRS, respectively. From this Figure, it can be observed that feature maps extracted from the compressed image are generally a lot of cleaner and have much better contrast between the foreground information, i.e the bird, and the background information than their counterparts from the original image. For any particular pair of feature maps extracted from the original image and the compressed image I_{10}, respectively, the bird is either visible or invisible in both of them. Whenever the bird is visible, the background of the feature map extracted from the compressed image is simple or more less wiped out, whereas the background of the feature map extracted from the original image is complicated and still contains significant energy most of times. These differences will be propagated to subsequent layers of Inception V3 (see Figure 5.10 for example). It is these difference along with the clearer texture on the body of the bird in the compressed image (see Figure 5.5) that makes the underlying DNN to distinguish a Brambling bird from a Junco snowbird that was original ranked first in the probability vector produced by the original image.

Example 3: Let I be image # 37. In view of Figure 5.4, HRS selects I_{10}, and $r_{HRS}(I) = 5$ while $r_D(I) = 6$. Therefore, $I \in \{I \in \mathcal{A} : r_{HRS}(I) \leq 5\} - \{I \in \mathcal{A} : r_D(I) \leq 5\}$. Figure 5.11 shows the original image I and its JPEG compressed version with QF = 10. Their respective feature maps extracted by Layer 2 of Inception V3 are illustrated in Figure 5.12. From these figures, the same understanding as explained in Example 2 can be confirmed. In feature maps extracted from the JPEG compressed version with QF = 10, the contrast is improved and interference information such as tiny spots on the frog's body is removed. Also, key features for the frog, such as the outlier of its body, are retained. The tiny spots in the frog's body make image #37 confused with a spotted salamander, which was ranked fifth with the original input image. Again, all of these differences will be propagated to subsequent layers of Inception V3 (see Figure 5.13 for example).

To further study impact of JPEG quality factor variations, we applied HRS on a different set of quality factors for each input original image than what we use in this chapter. This set of quality factors start from 0 until 100 with a step size of 5. In this case, we input the original input image in addition to its 21 quality factor versions to the HRS selector to output the final selection to feed it to the underlying subsequent DNN. Experimental results show that using HRS selector: (1) the top-1 accuracy of Inception V3 and ResNet-50 V2 became 85.302%, and 83.126%, respectively, (2) the top-5 accuracy of Inception

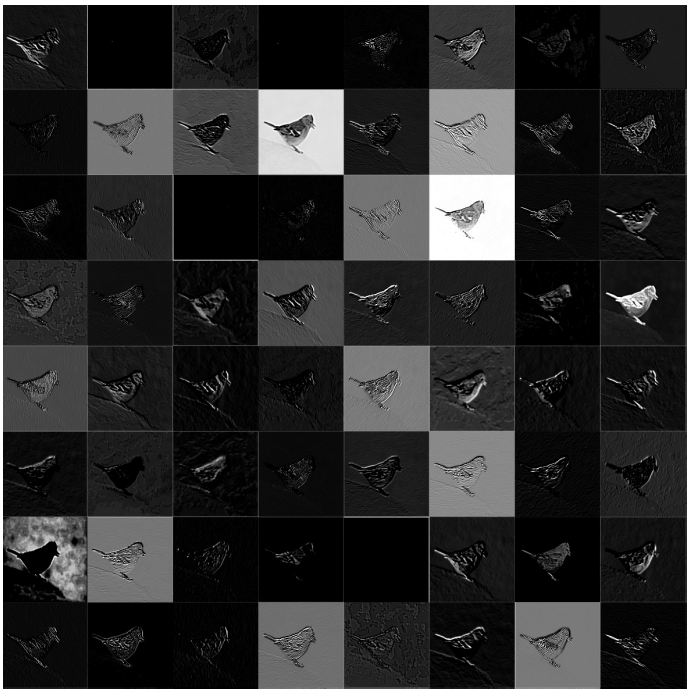


(a) Feature maps from the original image

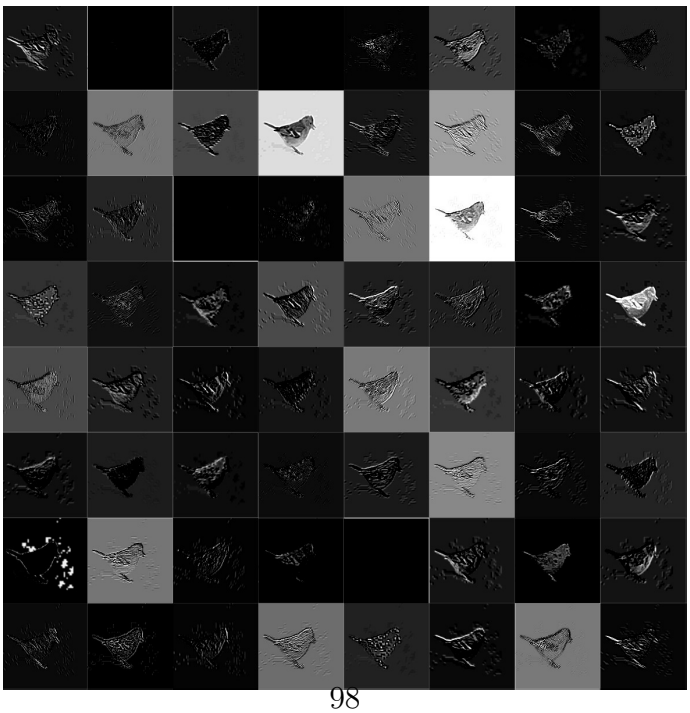


(b) Feature maps from the compressed image with QF = 10

Figure 5.9: Feature maps extracted by Layer 2 of Inception V3 from the original Image#651 with GT label "Brambling" from the ImageNet validation dataset and its JPEG compressed version with QF = 10. Best viewed in electronic format



(a) Original Image Visualization



(b) Compressed Image Visualization

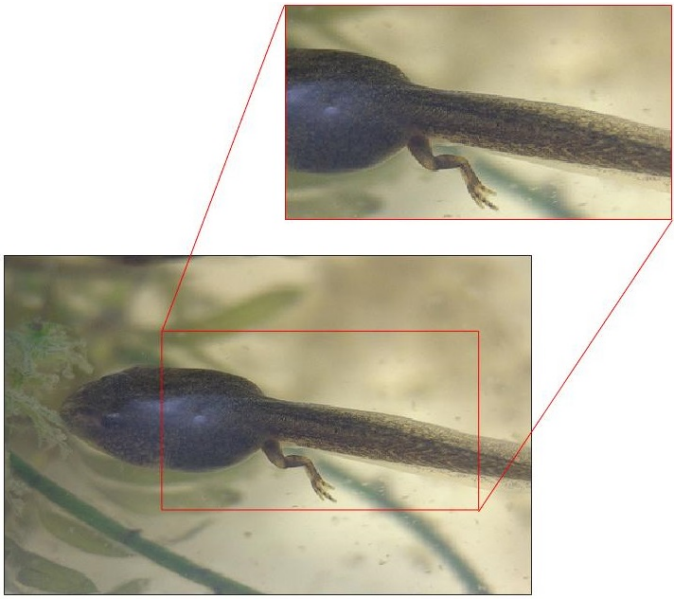
Figure 5.10: Feature maps extracted by Layer 3 of Inception V3 from the original Image#651 with GT label "Brambling" from the ImageNet validation dataset and its JPEG compressed version with QF = 10. Best viewed in electronic format

V3 and ResNet-50 V2 became 96.348%, and 95.262%, respectively. These classification accuracy results are improved with respect to the results in Table 5.1 due to the increased granularity of the QF range, which gives more freedom to the HRS selector. Table 5.3 also shows the contributions of the QF values selected by HRS for Inception V3 and ResNet-50 V2, respectively. This table confirms in both cases that a JPEG compressed version with a lower QF is selected by HRS more often than its counterpart with a higher QF.

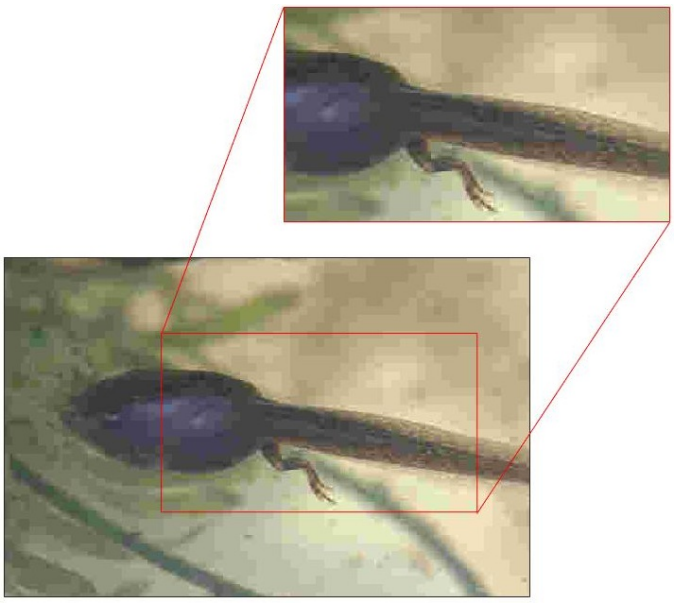
Table 5.3: Percentages of QF values selected by HRS with the Original Image and its 21 QF Versions using Inception V3 and ResNet-50 V2.

Image Version	Inception V3 (%)	ResNet-50 V2 (%)
ORG	0.084	0.058
100	0.134	0.126
95	0.338	0.490
90	0.334	0.346
85	0.472	0.552
80	0.426	0.456
75	0.386	0.048
70	0.432	0.298
65	0.470	0.318
60	0.484	0.464
55	0.450	0.114
50	0.542	0.510
45	0.668	0.480
40	0.742	0.684
35	1.084	2.612
30	1.328	1.552
25	2.016	2.224
20	3.266	3.304
15	6.476	6.360
10	20.666	18.476
5	32.250	26.820
0	26.952	33.708

To summarize, we have shown that if a JPEG compressed version of an image is selected on an individual image base as an input to an underlying DNN and the GT label of the original image is known to the selector (but unknown to the underlying DNN), the

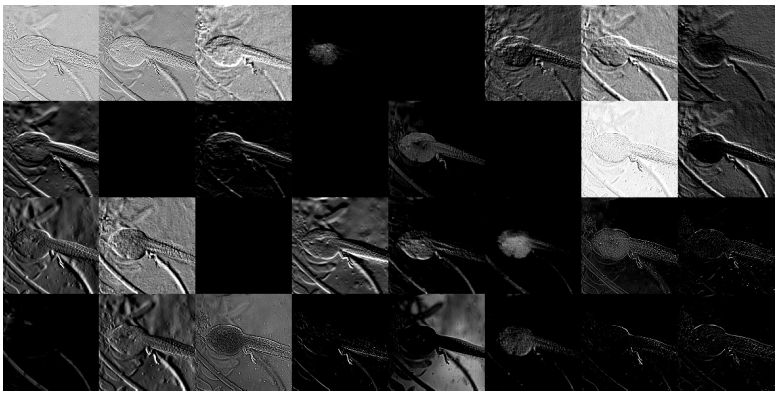


(a) Original Image

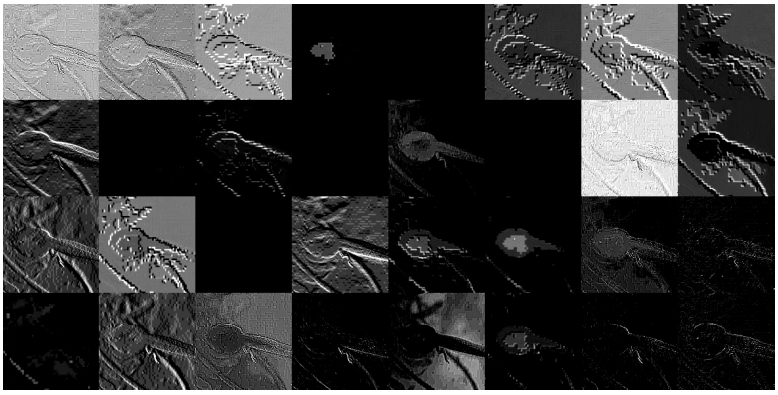


(b) QF=10 Image

Figure 5.11: Image #37 from the ImageNet Validation dataset with its GT label "Tailed frog": (a) the original image; (b) the JPEG compressed version with QF = 10. Best viewed in electronic format.

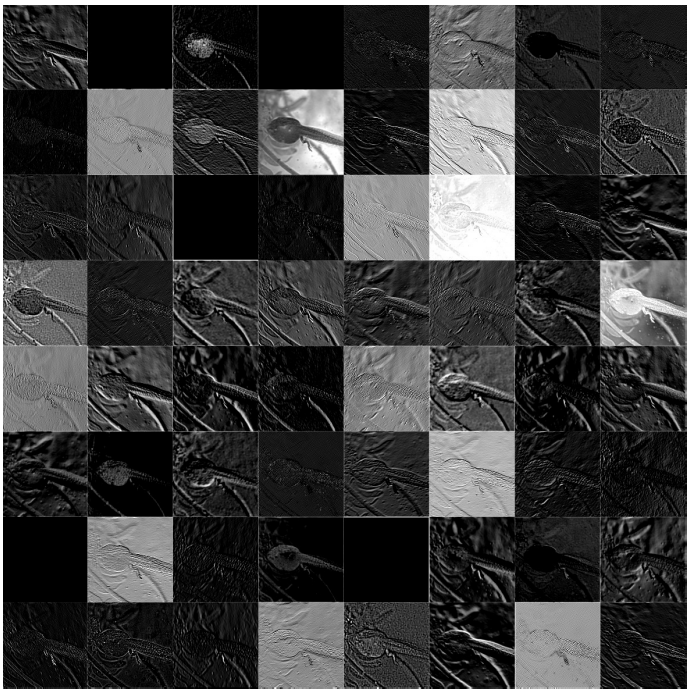


(a) Original Image Feature Maps

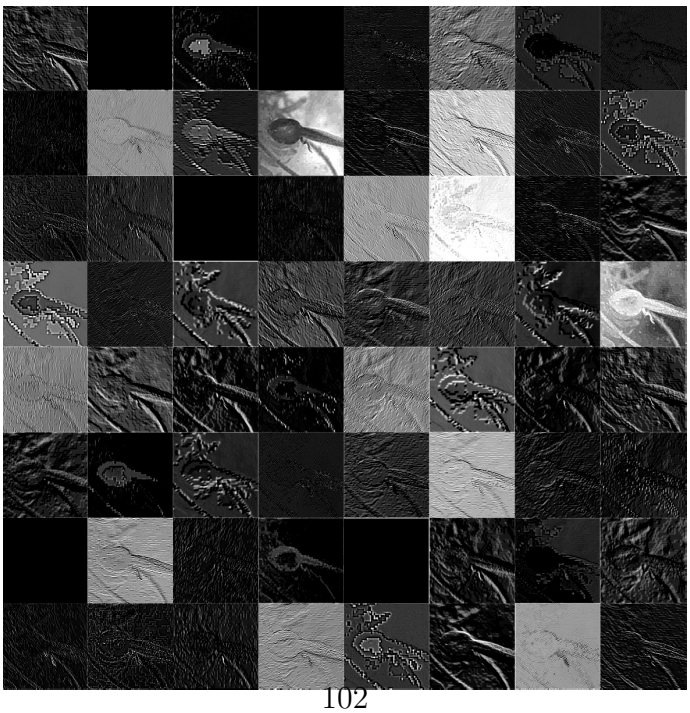


(b) Compressed Image Feature Maps

Figure 5.12: Feature maps extracted by Layer 2 of Inception V3 from the original Image#37 with GT label "Tailed Frog" from the ImageNet validation dataset and its JPEG compressed version with QF = 10. Best viewed in electronic format



(a) Original Image Visualization



(b) Compressed Image Visualization

Figure 5.13: Feature maps extracted by Layer 3 of Inception V3 from the original Image#37 with GT label "Tailed Frog" from the ImageNet validation dataset and its JPEG compressed version with QF = 10. Best viewed in electronic format

classification performance of the underlying DNN can be improved significantly while the size in bits of the input image can be reduced dramatically. In addition, in our paper [133], we trained Inception V3 and ResNet-50 V2 with a strategy inspired by the HRS selector. This strategy assumes that the GT label is unknown to the selector as well as the underlying DNN and yields an average increase of classification accuracy of 0.4%. Therefore, in contrast to conventional understanding, JPEG compression indeed helps DL in image classification. In the next section, we will show other selectors that maintain the accuracy of DNNs, but while significantly reducing the size in bits of the input images for long-term storage purposes.

5.4 Selectors Maintaining Classification Accuracy While Reducing Input Size

Let us now go back to Figure 5.2 and assume that the GT label of the original image is unknown to the selector therein. In this section, we present three selectors which maintain the same top-1 accuracy, the same top-5 accuracy, and the same top-1 accuracy and top-5 accuracy as those of the underlying DNN, respectively, while reducing the size in bits of the input image to the underlying DNN to some degree. These three selectors are referred to as Top-1 Keeper (T1K), Top-5 Keeper (T5K), and Top-1 and Top-5 Keeper (TTK), respectively.

For any original image I, T1K selects I_j as an input to the underlying DNN if and only if $0 \leq j \leq 10$ is the largest integer such that the top-1 label in the sorted P_j is the same as that in the sorted P_0. Similarly, for any original image I, T5K selects I_j as an input to the underlying DNN if and only if $0 \leq j \leq 10$ is the largest integer such that the set of top-5 labels within the sorted P_j is the same as that in the sorted P_0. Likewise, TTK selects I_j as an input to the underlying DNN if and only if $0 \leq j \leq 10$ is the largest integer such that both the top-1 label in and the set of top-5 labels within the sorted P_j are the same as those in the sorted P_0, respectively. It is clear that on any set of images, T1K achieves the same top-1 accuracy as that of the underlying DNN, T5K achieves the same top-5 accuracy as that of the underlying DNN, and TTK achieves the same top-1 accuracy and top-5 accuracy as those of the underlying DNN.

Table 5.4 shows the top-1 accuracy and top-5 accuracy of T1K and T5K on the whole ImageNet validation dataset when the underlying DNN is Inception V3 and ResNet-50 V2 pre-trained with the original images in the training set of the ILSVRC 2012 dataset, respectively. As seen from this table, T1K degrades the top-5 accuracy by up to 1.5%, and

T5K reduces the top-1 accuracy by up to 1.26%. However, the advantage is the dramatic reduction in the input size in bits. Tables 5.5 and 5.6 show CR results of T1K, T5K, and TTK for the whole ImageNet validation dataset when the underlying DNN is Inception V3 and ResNet-50 V2, respectively. In Tables 5.5 and 5.6, the default size is the total size in GB of all original images in ImageNet validation dataset, while the new size is the total size of all selected input images by T1K, T5K, or TTK as the case may be. As seen in these tables, the compression ratios achieved by T1K, T5K, and TTK, are on average 8.8, 3.3, and 3.1, respectively.

Table 5.4: Top-1 and top-5 accuracy results of T1K and T5K on the whole ImageNet validation dataset.

Selector	Inception V3 Top-1	Inception V3 Top-5	ResNet-50 V2 Top-1	ResNet-50 V2 Top-5
Default	77.6%	93.8%	75.58%	92.8%
T1K	77.6%	92.5%	75.58%	91.3%
T5K	76.7%	93.8%	74.32%	92.8%

Table 5.5: Compression ratio results of T1K, T5K, and TTK for the whole ImageNet validation dataset with Inception V3 as the underlying DNN.

Selector	Default size (GB)	New size (GB)	CR
T1K	6.7	0.76	8.8x
T5K	6.7	2.1	3.1x
TTK	6.7	2.3	2.9x

Table 5.6: Compression ratio results of T1K, T5K, and TTK for the whole ImageNet validation dataset with ResNet-50 V2 as the underlying DNN.

Selector	Default size (GB)	New size (GB)	CR
T1K	6.7	0.76	8.8x
T5K	6.7	1.9	3.5x
TTK	6.7	2.0	3.3x

These results demonstrate the advantage of selectors in Figure 5.2 in terms of input storage savings while roughly maintaining the classification accuracy of the underlying DNN. Applications that require long-term storage of multimedia such as image surveillance will benefit from these selectors.

5.5 Chapter Summary

We have formulated, in this chapter, a new framework to investigate the impact of JPEG compression on deep learning (DL) in image classification. Fix an underlying deep neural network (DNN) pre-trained with pristine ImageNet images. For any original image, the framework allows one to select, among many JPEG compressed versions of the original image including possibly the original image itself, a suitable version as an input to the underlying DNN. It has been demonstrated that within the framework, a selector can be designed so that the classification accuracy of the underlying DNN can be improved significantly while the size in bits of the selected input is, on average, reduced dramatically in comparison with the original image. Therefore, compression, if used in a right manner, helps DL in image classification, which is in contrast to the conventional understanding that JPEG compression generally degrades the classification accuracy of DL.

Specifically, in the case where the ground truth label of the original image is known to the selector but unknown to the underlying DNN, a selector called Highest Ranking Selector (HRS) has been presented and shown to be optimal in the sense of achieving the highest top-k accuracy on any set of images for any k among all possible selectors. When the selection is made among the original image and its 10 JPEG compressed versions with their quality factor (QF) values ranging from 100 to 10 with a step size of 10, HRS improves, on average, the top-1 accuracy and top-5 accuracy of Inception V3 and ResNet-50 on the whole ImageNet validation set by 5.6% and 1.9%, respectively while reducing the input size in bits dramatically—the compression ratio (CR) between the size of the original images and the size of the selected input images by HRS is 8 for the whole ImageNet validation dataset.

Selectors without the knowledge of the ground truth label of the original image have also be proposed. They either maintain the top-1 accuracy, top-5 accuracy, or top-1 and top-5 accuracy of the underlying DNN. It has been shown that when applied to Inception V3 and ResNet-50, these selectors achieve CRs of 8.8, 3.3, and 3.1, respectively for the whole ImageNet validation dataset.

Although only JPEG compression has been considered, the proposed framework can be applied to any other codecs such as HEVC and H.264 [119], [131] as well. For example, we

applied our HRS selector to the ImageNet validation set under HEVC compression with the following QP set $\{0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 51\}$. Experimental results show that using HRS selector improved the default accuracy of Inception V3 and ResNet-50 V2: (1) the top-1 accuracy of Inception V3 and ResNet-50 V2 became 86.56%, and 84.384%, respectively, (2) the top-5 accuracy of Inception V3 and ResNet-50 V2 became 96.77%, and 95.624%, respectively. It is worth noting that HEVC's least quality level, QP = 51, was the most selected with percentages equals to 28.706% and 29.6% in the case of Inception V3 and ResNet-50 V2, respectively.

Therefore, our results also motivate the development of new compression theory and algorithms for DL, which have to achieve good trade-offs among the compression rate, compression distortion, and classification accuracy, where the compression distortion is for human, and the classification accuracy is for DL machines. Developing these new and hopefully universal compression theory and algorithms is left open for future research. We also believe that compression will be a vital component in the pre-processing stage of DNN design to improve performance.

Chapter 6

Conclusion and Future Work

This chapter concludes the thesis with a summary of contributions and presents a few thoughts on future research.

6.1 Conclusion

This thesis has proposed a variety of novel image/video compression algorithms to serve today's multimedia system that should serve each of human and machine visions. For human vision, our goal was focused on video compression to improve the trade-off between compression rate, compression distortion, and time complexity, while our goal for machines was to show that compression, if used in the right manner, can help improve the classification accuracy for DNN machines, while reducing the size in bits of the input image.

Towards the human vision perspective, we proposed adaptive frame-level QP selection methods for the LD HEVC by integrating the coding propagation effect into RDO through a notion called the temporal propagation length, which is defined as the impact length of the current frame on its future frames. Based on a linear model to model the inter-frame distortion dependency and analysis for the temporal propagation length, we adaptively determined the QP value and the Lagrangian Multiplier of each frame. Experimental results demonstrated that our methods can achieve BDBR savings for the LD configurations of 5.0% and 4.9% for the $\varepsilon-$prediction method, and 4.9% and 4.9% for the $\mu-$prediction method. All these improvements at the expense of an insignificant average increase of 1% time overhead. In addition, the results also show that our proposed method can provide

considerable coding efficiency improvements for video conferencing sequences in specific. These improvements can go up to -12.9% and -12.2%, respectively. It is worth nothing that our algorithm and results piqued the serious interest of Google in adopting the proposed LD adaptive QP algorithm in their VP9 codec. Hence, we plan to push our adaptive QP algorithm in their systems.

Along the same line of the human vision perspective, we presented a new CU partition prediction method equipped with hierarchical fully connected NN models and features from LPTCM to minimize the computational intensity in the HEVC CU partition process, while controlling the trade-off between the compression rate and compression distortion. Also, we highlighted challenges to equip NN online learning with HEVC and proposed adaptive training strategies to these challenges. Experimental results demonstrated that our technique is among the best NN methods with controlled BD-rate loss and of comparable performance to others. Our technique achieves 32% TS average with 1.6% BD-rate average, and attains time savings that can go up to 60% for high resolution sequences at low bitrates.

In the proposed CU partition algorithm, a fully connected NN machine 'saw' manually extracted LPTCM features to make a classification decision to help reduce the computational intensity of compression at a controlled trade-off between compression rate and compression distortion.

Turning to the machine vision perspective where convolutional neural network directly 'see' the input JPEG image, we formulated a new framework to investigate the impact of JPEG compression on deep learning (DL) in image classification. Fix an underlying deep neural network (DNN) pre-trained with pristine ImageNet images. For any original image, the framework allows one to select, among many JPEG compressed versions of the original image including possibly the original image itself, a suitable version as an input to the underlying DNN. It is demonstrated that within the framework, a selector can be designed so that the classification accuracy of the underlying DNN can be improved significantly while the size in bits of the selected input is, on average, reduced dramatically in comparison with the original image. Therefore, compression, if used in the right manner, helps deep learning in image classification. With the proposed framework, we demonstrate for the first time that JPEG image compression helps improve the classification accuracy of the underlying DNN while the size in bits of the selected input is, on average, reduced dramatically in comparison with the input original image. This demonstration is on the contrary of the conventional understanding that JPEG compression generally hurts the classification accuracy of DL. We also proposed other selectors to maintain the top-1 accuracy, top-5 accuracy, top-5 accuracy, while reducing the size in bits of the input images.

6.2 Future Work

Needless to say, the research in this thesis can be extended in several ways. That being said, we discuss some of these ways in the following that can be pictured in Figure 1.1 as mentioned in the introduction section.

6.2.1 Human Vision Perspective: CTU-based Adaptive QP Selection and Inter-dependency Aware Rate Control

In this thesis, we have proposed an adaptive frame-level QP selection algorithm for the hierarchical coding structure in LD HEVC by characterizing the coding propagation effect through a notion called temporal propagation length. This algorithm achieved overall BDBR savings that go up to 5.0% and 4.9% for LDP and LDB HEVC, respectively. However, this work could be extended in several ways. To tackle one way, as reported in Section 3.7, our algorithm do not achieve the same improvements with sequences with complex scenes and large motion leading to poor leverage of the frame-level temporal dependency. As part of future work, we believe that extending our algorithm to the CTU-level while taking into consideration the utilization percentages of all past frames in the encoding order as well as adaptive temporal propagation lengths can enhance the current global RDO model leading to an increase in the coding efficiency improvements. To tackle another way, the R-λ based rate control scheme has been adopted in HEVC due to its coding efficiency improvements of 15.9% in LD coding over the previous rate control techniques [80]. The R-λ involves two main steps: bit allocation and quantisation parameter selection, where bit allocation is an important key to coding efficiency improvements. We believe that our proposed RDO model can be combined with the bit allocation step in the current R-λ method under the LD HEVC configuration; thus, more potential coding efficiency improvements can be realized [55].

6.2.2 Machine Vision Perspective: Design of Compression targeting Machine Vision

To date, all image/video data has been typically consumed by humans, but there will be an increasing demand for this video data to be consumed and processed by machines. According to Cisco's visual networking index published in February 2019, it has been estimated that global machine-to-machine (M2M) traffic will increase more than sevenfold from 2017 to 2022. In a few years, imagine that you are driving your self-driving car

and you approach a reckless motorcycle. Your car should recognize this motorcycle, slow down, pull to the right, and communicate with other cars of different models about this potential accident; all of this should happen, while you are checking your email, watching a movie or even reading a book. This example among others implies a strong need in optimizing image/video for machine vision [113, 43].

Neural networks were built and inspired based on human brains. In particular, human visual system solves the image recognition task through hierarchical processing along the ventral pathway of the visual cortex. Given this hierarchy, the visual preference of neurons gradually increases from oriented bars in primary visual cortex (V1) to complex objects in inferotemporal cortex (IT), where neural activity provides a robust, invariant, and linearly-separable object representation [38, 39]. Along the same line, DNNs are widely-used because of their ability in extracting the required features for classification from the raw pixels of images [49]. To extract these features, DNNs learn the parameters of non-linear activation functions using a backpropagation learning algorithm. These functions progressively transform raw pixels of the input image to produce the output predicted label. These non-linear functions provide multi-layer representations to the input image, where each representation is a feature that amplifies certain aspects of the raw pixels required for classification and suppress irrelevant information. Typically, the first few representations project primitive features such as existence of edges, their orientations, and their arrangements. Subsequent layers combine representations from previous layers to classify familiar objects [73].

Since late 1980s, a range of image and video lossy codecs from JPEG to HEVC and beyond have been introduced to obtain better trade-off between compression rate and compression distortion, while paying little attention to the DNN machine's classification accuracy [127, 98, 131, 119]. In the literature, some compression methods targeting machine vision have been proposed. For example, [90] proposed a three-level quantization step size method based on the importance of the frequency coefficients to specific DNN. Their results show that they can reduce the size in bits of the input image, while only maintaining the accuracy of DNNs.

This thesis showed that the compression, if used in the right manner, can improve the accuracy of DNNs and reduce the size in bits of the input image. However, our selectors in this thesis were based on JPEG compression and utilized, for each original image, 11 compressed versions including the original itself. The HRS optimal selector with the most gains in terms of accuracy and compression ratio required the knowledge of the ground truth label. This discussion culminates in the following two questions: Can we design a selector that can achieve the gains of the HRS selector without the need of the ground truth label? How can we develop new compression theory and algorithms for DL, which

have to achieve good trade-offs among the compression rate, compression distortion, and classification accuracy, where the compression distortion is for human, and the classification accuracy is for DL machines?

To take some steps towards these two questions, one way is to model JPEG's uniform mid-tread dead-zone quantization function as a summation of rectifier linear unit functions with trainable parameters. Then, within the JPEG framework, we should design a differentiable loss function that optimizes between compression rate, compression distortion, and DNN's accuracy, and in turn we can jointly train the compression system with machine vision via mini-batch learning algorithm.

It is worthwhile mentioning that the topic of video compression targeting machine vision is recently creating a strong research traction. In July 2019, MPEG has created a group called the Video Compression for Machines (VCM) group to create standards for "compression coding for machine vision as well as compression for human-machine hybrid vision." This standard will be dedicated for broad use with any video-related Internet of Things (IoT) devices. According the MPEG Chairman, the main focus of this group is to answer the following question: "So far, video coding 'descriptors' were designed to achieve the best visual quality—as assessed by humans—at a given bitrate. The question asked by video coding for machines is: What descriptors provide the best performance for use by a machine at a given bitrate?"

References

[1] Cisco visual networking index: Forecast and trends. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html.

[2] Elecard videos download page. www.elecard.com/en/download/videos.html.

[3] FFmepg. http://www.ffmpeg.org.

[4] HM. https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/.

[5] Inception v3 Tensorflow model. http://download.tensorflow.org/models/image/imagenet/inception-2015-12-05.tgz.

[6] JSVM, H.264/SVC joint scalable video coding model. ftp://garcon.ient.rwth-aachen.de/.

[7] Resnet-50 v2 Tensorflow model. http://download.tensorflow.org/models/resnet_v2_50_2017_04_14.tar.gz.

[8] Standard hevc test sequences. ftp://ftp.tnt.uni-hannover.de.

[9] VP9 Standard, howpublished = http://www.webmproject.org/vp9/.

[10] Web technology surveys for JPEG usage statistics. https://w3techs.com/technologies/details/im-jpeg/all/all.

[11] X.265. https://x265.readthedocs.io/en/default/cli.html#mode-decision-analysis.

[12] Xiph.org video test media page. https://media.xiph.org/video/derf/.

[13] *Subjective quality evaluation of the upcoming HEVC video compression standard*, volume 8499, 2012.

[14] *Comparison of compression efficiency between HEVC/H.265 and VP9 based on subjective assessments*, volume 9217, 2014.

[15] Martín Abadi et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016.

[16] H. Amer, A. Rashwan, and E. Yang. Fully connected network for HEVC cu split decision equipped with Laplacian transparent composite model. In *2018 Picture Coding Symposium (PCS)*, pages 189–193, June 2018.

[17] H. Amer and E. H. Yang. Adaptive quantization parameter selection for low-delay HEVC via temporal propagation length estimation. *Signal Processing: Image Communication*, 70:114–130, 2020.

[18] H. Amer and E. H. Yang. Scene change detection techniques based on adaptive and sequential outlier detection for real-time video coding applications. *IEEE Transactions on Circuits and Systems Video Technology*, In Preparation.

[19] H. Amer and En-Hui Yang. Scene-based low delay HEVC encoding framework based on transparent composite modeling. In *2016 IEEE International Conference on Image Processing (ICIP)*, September 2016.

[20] H. Amer and En-Hui Yang. Low-delay HEVC adaptive quantization parameter selection through temporal propagation length estimation. In *2018 IEEE International Conference on Image Processing, ICIP 2018, Athens, Greece, October 7-10*, pages 211–215, 2018.

[21] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[22] Yoshua Bengio, Frédéric Bastien, Arnaud Bergeron, Nicolas Boulanger-Lewandowski, Thomas Breuel, Youssouf Chherawala, Moustapha Cisse, Myriam Côté, Dumitru Erhan, Jeremy Eustache, et al. Deep learners benefit more from out-of-distribution examples. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 164–172, 2011.

[23] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[24] Alessandro Bianchi, Moreno Raimondo Vendra, Pavlos Protopapas, and Marco Brambilla. Improving image classification robustness through selective CNN-filters fine-tuning. *arXiv preprint arXiv:1904.03949*, 2019.

[25] Tejas S Borkar and Lina J Karam. Deepcorrect: Correcting DNN models against image distortions. *IEEE Transactions on Image Processing*, 2019.

[26] F. Bossen. Common HM test conditions and software reference configurations. *JCTVC-L1100*, April 2013.

[27] B. Bross, W.-J. Han, J.-R Ohm, G. J. Sullivan, Y.-K. Wang, and T. Wiegand. High efficiency video coding (hevc) text specification draft 10 (for fdis & consent). *JCT-VC, Doc. JCTVC-L1003*, 10, Jan 2013.

[28] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[29] Zhuo Chen, Weisi Lin, Shiqi Wang, Long Xu, and Leida Li. Image quality assessment guided deep neural networks training. *arXiv preprint arXiv:1708.03880*, 2017.

[30] J. C. Chiang et al. A fast h.264/avc-based stereo video encoding algorithm based on hierarchical two-stage neural classification. *IEEE Journal of Selected Topics in Signal Processing*, 5(2):309–320, April 2011.

[31] Sandeep P Chinchali, Eyal Cidon, Evgenya Pergament, Tianshu Chu, and Sachin Katti. Neural networks meet physical networks: Distributed inference between edge devices and the cloud. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, pages 50–56. ACM, 2018.

[32] Rémi Cogranne. Determining JPEG Image Standard Quality Factor from the Quantization Tables. *arXiv e-prints*, page arXiv:1802.00992, Feb 2018.

[33] G. Cote, B. Erol, M. Gallant, and F. Kossentini. H.263+: video coding at low bit rates. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(7):849–866, Nov 1998.

[34] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[35] Yuanying Dai et al. A convolutional neural network approach for post-processing in HEVC intra coding. *CoRR*, abs/1608.06690, 2016.

[36] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[37] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[38] James J DiCarlo and David D Cox. Untangling invariant object recognition. *Trends in cognitive sciences*, 11(8):333–341, 2007.

[39] James J DiCarlo, Davide Zoccolan, and Nicole C Rust. How does the brain solve visual object recognition? *Neuron*, 73(3):415–434, 2012.

[40] Samuel Dodge and Lina Karam. Understanding how image quality affects deep neural networks. In *2016 eighth international conference on quality of multimedia experience (QoMEX)*, pages 1–6. IEEE, 2016.

[41] Samuel Dodge and Lina Karam. A study and comparison of human and deep learning recognition performance under visual distortions. In *2017 26th international conference on computer communication and networks (ICCCN)*, pages 1–7. IEEE, 2017.

[42] Samuel F Dodge and Lina J Karam. Quality robust mixtures of deep neural networks. *IEEE Transactions on Image Processing*, 27(11):5553–5562, 2018.

[43] Ling-Yu Duan, Jiaying Liu, Wenhan Yang, Tiejun Huang, and Wen Gao. Video Coding for Machines: A Paradigm of Collaborative Compression and Intelligent Analytics. *arXiv e-prints*, page arXiv:2001.03569v2, Jan 2020.

[44] F. Duanmu et al. Fast cu partition decision using machine learning for screen content compression. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 4972–4976, Sept 2015.

[45] Y. Gao, C. Zhu, S. Li, and T. Yang. Temporally dependent rate-distortion optimization for low-delay hierarchical video coding. *IEEE Transactions on Image Processing*, 26(9):4457–4470, Sept 2017.

[46] Y. Gao, C. Zhu, S. Li, and T. Yang. Source distortion temporal propagation analysis for random-access hierarchical video coding optimization. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(2):546–559, Feb 2019.

[47] Robert Geirhos, David HJ Janssen, Heiko H Schütt, Jonas Rauber, Matthias Bethge, and Felix A Wichmann. Comparing deep neural networks against humans: object recognition when the signal gets weaker. *arXiv preprint arXiv:1706.06969*, 2017.

[48] Herbert Gish and John Pierce. Asymptotically efficient quantizing. *IEEE Transactions on Information Theory*, 14(5):676–683, 1968.

[49] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. The MIT Press, 2016.

[50] Klemen Grm, Vitomir Štruc, Anais Artiges, Matthieu Caron, and Hazım K Ekenel. Strengths and weaknesses of deep learning models for face recognition against image degradations. *IET Biometrics*, 7(1):81–89, 2017.

[51] D. Grois, D. Marpe, A. Mulayoff, B. Itzhaky, and O. Hadar. Performance comparison of h.265/mpeg-hevc, vp9, and h.264/mpeg-avc encoders. In *Picture Coding Symposium (PCS), 2013*, pages 394–397, Dec 2013.

[52] Lionel Gueguen, Alex Sergeev, Ben Kadlec, Rosanne Liu, and Jason Yosinski. Faster neural networks straight from jpeg. In *Advances in Neural Information Processing Systems*, pages 3933–3944, 2018.

[53] E. h. Yang and X. Yu. Soft decision quantization for H.264 with main profile compatibility. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(1):122–127, Jan 2009.

[54] J. He, E. H. Yang, F. Yang, and K. Yang. Adaptive quantization parameter selection for H.265/HEVC by employing inter-frame dependency. *IEEE Transactions on Circuits and Systems for Video Technology*, PP(99):1–1, September 2017.

[55] Jing He and Fuzheng Yang. Efficient frame-level bit allocation algorithm for h. 265/hevc. *IET Image Processing*, 11(4):245–257, 2017.

[56] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[57] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[58] P. Helle, S. Oudin, B. Bross, D. Marpe, M. O. Bici, K. Ugur, J. Jung, G. Clare, and T. Wiegand. Block merging for quadtree-based partitioning in HEVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1720–1731, Dec 2012.

[59] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[60] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[61] Md Tahmid Hossain, Shyh Wei Teng, Dengsheng Zhang, Suryani Lim, and Guojun Lu. Distortion robust image classification with deep convolutional neural network based on discrete cosine transform. *arXiv preprint arXiv:1811.05819*, 2018.

[62] Nan Hu and En-Hui Yang. Fast mode selection for HEVC intra-frame coding with entropy coding refinement based on a transparent composite model. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(9):1521–1532, Sept 2015.

[63] S. Hu, H. Wang, S. Kwong, T. Zhao, and C. C. J. Kuo. Rate control optimization for temporal-layer scalable video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(8):1152–1162, Aug 2011.

[64] III Hugh Everett. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11(3):399–417, 1963.

[65] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[66] Samil Karahan, Merve Kilinc Yildirum, Kadir Kirtac, Ferhat Sukru Rende, Gultekin Butun, and Hazim Kemal Ekenel. How image degradations affect deep cnn-based face recognition? In *2016 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pages 1–5. IEEE, 2016.

[67] Il-Koo Kim, Junghye Min, T. Lee, Woo-Jin Han, and JeongHoon Park. Block partitioning structure in the HEVC standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1697–1706, Dec 2012.

[68] J. Kim, H. Zeng, D. Ghadiyaram, S. Lee, L. Zhang, and A. C. Bovik. Deep convolutional neural models for picture-quality prediction: Challenges and solutions to data-driven image quality assessment. *IEEE Signal Processing Magazine*, 34(6):130–141, Nov 2017.

[69] Alex Krizhevsky and Geoffery Hiton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[70] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[71] J. Lainema, F. Bossen, W. J. Han, J. Min, and K. Ugur. Intra coding of the HEVC standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1792–1801, Dec 2012.

[72] T. Laude and J. Ostermann. Deep learning-based intra prediction mode decision for hevc. In *2016 Picture Coding Symposium (PCS)*, pages 1–5, Dec 2016.

[73] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[74] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[75] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

[76] B. Li, H. Li, L. Li, and J. Zhang. λ domain rate control algorithm for High Efficiency Video Coding. *IEEE Transactions on Image Processing*, 23(9):3841–3854, Sept 2014.

[77] B Li, GJ Sullivan, and J Xu. Comparison of compression performance of HEVC draft 9 with avc high profile and performance of hm9.0 with temporal scalability characteristics. In *JCTVC-L0322, 12th JCT-VC meeting, Geneva, Switzerland*, 2013.

[78] B. Li, J. Xu, D. Zhang, and H. Li. Qp refinement according to Lagrange multiplier for High Efficiency Video coding. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, pages 477–480, May 2013.

[79] Bin Li, Dong Zhang, Houqian Li, and J Xu. Qp determination by lambda value. In *JCT-VC of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 9th Meeting, Geneva, Switzerland, Doc. JCTVC-I0426*, 2012.

[80] Li Li, Bin Li, Houqiang Li, and Chang Wen Chen. λ-domain optimal bit allocation algorithm for high efficiency video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(1):130–142, 2016.

[81] S. Li, C. Zhu, Y. Gao, Y. Zhou, F. Dufaux, and M. T. Sun. Lagrangian multiplier adaptation for rate-distortion optimization with inter-frame dependency. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(1):117–129, Jan 2016.

[82] Xiang Li et al. Rate-complexity-distortion evaluation for hybrid video coding. In *2010 IEEE International Conference on Multimedia and Expo (ICME)*, pages 685–690, July 2010.

[83] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John E Hopcroft. Convergent learning: Do different neural networks learn the same representations?

[84] Dong Liu, Yue Li, Jianping Lin, Houqiang Li, and Feng Wu. Deep learning-based video coding: A review and a case study. *arXiv preprint arXiv:1904.12462*, 2019.

[85] J. Liu, Y. Cho, Z. Guo, and J. Kuo. Bit allocation for spatial scalability coding of H.264/SVC with dependent rate-distortion analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(7):967–981, July 2010.

[86] Shan Liu and C. C. J. Kuo. Joint temporal-spatial bit allocation for video coding with dependency. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(1):15–26, Jan 2005.

[87] Y. C. Liu et al. Svm-based fast intra cu depth decision for hevc. In *2015 Data Compression Conference*, pages 458–458, April 2015.

[88] Z. Liu et al. Cu partition mode decision for hevc hardwired intra encoder using convolution neural network. *IEEE Transactions on Image Processing*, 25(11):5088–5103, Nov 2016.

[89] Z. Liu, X. Yu, S. Chen, and D. Wang. Cnn oriented fast hevc intra cu mode decision. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2270–2273, May 2016.

[90] Zihao Liu, Tao Liu, Wujie Wen, Lei Jiang, Jie Xu, Yanzhi Wang, and Gang Quan. DeepN-JPEG: A deep neural network favorable JPEG-based image compression framework. In *Proceedings of the 55th Annual Design Automation Conference*, page 18. ACM, 2018.

[91] Siwei Ma, Xinfeng Zhang, Chuanmin Jia, Zhenghui Zhao, Shiqi Wang, and Shanshe Wanga. Image and video compression with neural networks: A review. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[92] D. Marpe, H. Schwarz, and T. Wiegand. Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):620–636, July 2003.

[93] D. Mukherjee, J. Bankoski, A. Grange, Jingning Han, J. Koleszar, P. Wilkins, Yaowu Xu, and R. Bultje. The latest open-source video codec vp9 - an overview and preliminary results. In *Picture Coding Symposium (PCS), 2013*, pages 390–393, Dec 2013.

[94] Mark Nelson. *The Data Compression Book*. Henry Holt and Co., Inc., New York, NY, USA, 1991.

[95] J. R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand. Comparison of the coding efficiency of video coding standards —including high efficiency video coding (hevc). *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1669–1684, Dec 2012.

[96] A. Ortega and K. Ramchandran. Rate-distortion methods for image and video compression. *IEEE Signal Processing Magazine*, 15(6):23–50, Nov 1998.

[97] Edouard Oyallon, Eugene Belilovsky, Sergey Zagoruyko, and Michal Valko. Compressing the input for cnns with the first-order scattering transform. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 301–316, 2018.

[98] William B Pennebaker and Joan L Mitchell. *JPEG: Still image data compression standard*. Springer Science & Business Media, 1992.

[99] T Recommendation. CCITT T. 81. 1993.

[100] Shaoqing Ren et al. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[101] Iain E. Richardson. *The H.264 Advanced Video Compression Standard*. Wiley, 2nd edition, August 2010.

[102] I.E. Richardson. *Video Codec Design: Developing Image and Video Compression Systems*. John Wiley & Sons, 2002.

[103] I.E. Richardson. *H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia*. Wiley, 2003.

[104] Oren Rippel and Lubomir Bourdev. Real-time adaptive image compression. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2922–2930. JMLR. org, 2017.

[105] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[106] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[107] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120, Sept 2007.

[108] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Analysis of hierarchical b pictures and mctf. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 1929–1932. IEEE, 2006.

[109] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[110] Y. Shan and En-Hui Yang. Fast hevc intra coding algorithm based on machine learning and laplacian transparent composite model. In *2017 International Conference on Acoustics, Speech, and Signal Processing*, March 2017.

[111] X. Shang et al. Fast cu size decision and pu mode decision algorithm in hevc intra coding. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 1593–1597, Sept 2015.

[112] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[113] Qiu Shen, Juanjuan Cai, Linfeng Liu, Haojie Liu, Tong Chen, Long Ye, and Zhan Ma. Codedvision: Towards joint image understanding and compression via end-to-end learning. In Richang Hong, Wen-Huang Cheng, Toshihiko Yamasaki, Meng Wang, and Chong-Wah Ngo, editors, *Advances in Multimedia Information Processing – PCM 2018*, pages 3–14, Cham, 2018. Springer International Publishing.

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

[115] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[116] G. J. Sullivan and T. Wiegand. Rate-distortion optimization for video compression. *IEEE Signal Processing Magazine*, 15(6):74–90, Nov 1998.

[117] G.J. Sullivan, J. Ohm, Woo-Jin Han, and T. Wiegand. Overview of the High Efficiency Video Coding (HEVC) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, Dec 2012.

[118] C. Sun and E. Yang. An efficient dct-based image compression system based on laplacian transparent composite model. *IEEE Transactions on Image Processing*, 24(3):886–900, March 2015.

[119] Budagavi Madhukar Sullivan Gary J. Sze, Vivienne. *High Efficiency Video Coding (HEVC)*. Springer International Publishing, 2014.

[120] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, June 2016.

[121] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[122] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[123] TK Tan, Madhukar Budagavi, and Jani Lainema. Summary report for te5 on simplification of unified intra prediction.

[124] George Toderici et al. Full resolution image compression with recurrent neural networks. *CoRR*, abs/1608.05148, 2016.

[125] Matej Ulicny and Rozenn Dahyot. On using cnn with dct based image data. In *Proceedings of the 19th Irish Machine Vision and Image Processing conference IMVIP*, 2017.

[126] J. Vanne, M. Viitanen, T. D. Hamalainen, and A. Hallapuro. Comparative rate-distortion-complexity analysis of hevc and avc video codecs. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1885–1898, Dec 2012.

[127] G. K. Wallace. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, Feb 1992.

[128] Cheng Wang, Yifei Han, and Weidong Wang. An end-to-end deep learning image compression framework based on semantic analysis. *Applied Sciences*, 9(17):3580, 2019.

[129] S. Wang, S. Ma, D. Zhao, and W. Gao. Lagrange multiplier based perceptual optimization for High Efficiency Video Coding. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*, pages 1–4, Dec 2014.

[130] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[131] T. Wiegand et al. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, July 2003.

[132] Mai Xu, Tianyi Li, Zulin Wang, Xin Deng, and Zhenyu Guan. Reducing complexity of HEVC: A deep learning approach. *CoRR*, abs/1710.01218, 2017.

[133] E. H. Yang, H. Amer, and Y. Jiang. Compression helps deep learning in image classification. *Transactions on Image Processing*, Under Review.

[134] E.-H. Yang et al. Transparent composite model for dct coefficients: Design and analysis. *IEEE Transactions on Image Processing*, 23(3):1303–1316, March 2014.

[135] E. H. Yang and X. Yu. Rate distortion optimization for H.264 interframe coding: A general framework and algorithms. *IEEE Transactions on Image Processing*, 16(7):1774–1784, July 2007.

[136] E.-H. Yang and X Yu. Transparent composite model for large scale image/video processing. In *2013 IEEE International Conference on Big Data*, pages 38–44, Oct 2013.

[137] R. Yang, M. Xu, and Z. Wang. Decoder-side hevc quality enhancement with scalable convolutional neural network. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 817–822, July 2017.

[138] T. Yang, C. Zhu, X. Fan, and Q. Peng. Source distortion temporal propagation model for motion compensated video coding optimization. In *2012 IEEE International Conference on Multimedia and Expo*, pages 85–90, July 2012.

[139] Jonghwa Yim and Kyung-Ah Sohn. Enhancing the performance of convolutional neural networks on quality degraded datasets. *arXiv preprint arXiv:1710.06805*, 2017.

[140] X. Yu, Z. Liu, J. Liu, Y. Gao, and D. Wang. Vlsi friendly fast cu/pu mode decision for hevc intra encoding: Leveraging convolution neural network. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 1285–1289, Sept 2015.

[141] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[142] H. Zeng, K. N. Ngan, and M. Wang. Perceptual adaptive Lagrangian multiplier for High Efficiency Video Coding. In *2013 Picture Coding Symposium (PCS)*, pages 69–72, Dec 2013.

[143] H. Zhang and Z. Ma. Fast intra mode decision for high efficiency video coding (hevc). *IEEE Transactions on Circuits and Systems for Video Technology*, 24(4):660–668, April 2014.

[144] T. Zhao, Z. Wang, and C. W. Chen. Adaptive quantization parameter cascading in HEVC hierarchical coding. *IEEE Transactions on Image Processing*, 25(7):2997–3009, July 2016.

[145] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4480–4488, 2016.