UNIVERSITÀ
DEGLI STUDI
FIRENZE

Università degli Studi di Firenze
Dipartimento di Ingegneria dell'Informazione (DINFO)
Corso di Dottorato in Ingegneria dell'Informazione

Curriculum: Telecommunications and Telematics
SSD: ING-INF/03

Forensic methods for image and video integrity verification

Candidate
Yoshihisa Furushita

Supervisors
Prof. Alessandro Piva

Dr. Marco Fontani

PhD Coordinator
Prof. Fabio Schoen

ciclo XXXVII, 2021-2024

Università degli Studi di Firenze, Dipartimento di Ingegneria
dell'Informazione (DINFO).

Stay Hungry, Stay Foolish

Acknowledgments

I would like to express my gratitude to my supervisor, Prof. Alessandro Piva, and all my colleagues at the Signal Processing and Communications Laboratory (LESC) for their invaluable assistance during my research. In particular, I am deeply grateful to Dr. Daniele Baracchi and Dr. Dasara Shullani, who collaborated on the core aspects of my work. I also extend my thanks to Dr. Marco Fontani, forensic director at Amped Software, for his kind support, and to Prof. Gianni Ramponi for his insightful advice.

Abstract

The detection of double compression is an effective method for identifying tampered multimedia content. Tampering involves decoding, modifying, and re-encoding the original content. This re-encoding, known as double compression, leaves distinctive traces that do not exist in the original. By analyzing these traces, tampering can be detected. In recent years, research in image forensics has primarily focused on analyzing double-compressed images in the JPEG format. However, effective methods for detecting double compression in the High Efficiency Image File Format (HEIF), which has gained popularity for its ability to maintain high quality with reduced file size, remain limited. As HEIF employs H.265/HEVC encoding, traditional JPEG-based approaches cannot be directly applied. In response, we previously proposed a detection method based on coding ghost theory, although this method was restricted to cases where the quality of the first encoding was higher than that of the second. To overcome this limitation, we developed a lightweight image classifier capable of detecting double compression in HEIF images without relying on the quantization history. This model achieves higher detection accuracy than traditional methods while maintaining its lightweight characteristics. Similarly, the importance of double compression detection in video forensics has grown, particularly for the HEVC format, although research on double compression detection in video sequences with B-frames remains limited. Video tampering through double compression may cause frames to be re-encoded with different frame types from the original encoding, disrupting spatiotemporal consistency and potentially leading to anomalous behavior in intra-frame encoding elements. This effect leaves detectable traces in the encoding structure that were not present in the original video. In this study, we investigate double compression in both HEIF images and HEVC videos with B-frames, proposing a classifier based on changes in encoding information caused by re-encoding. Our method demonstrates superior detection accuracy compared to conventional techniques and proves effective in real-world scenarios.

Contents

Chapter 1

Introduction

In recent years, the demand for higher-resolution images and videos has increased due to the growing use of smartphones and tablets for the consumption of visual content and the widespread adoption of 4K and 8K TVs. To efficiently transmit large visual data over limited network bandwidth, new image and video compression technologies have been developed.

In particular, in 2013, H.265/HEVC (High Efficiency Video Coding) was introduced as the next-generation video coding standard to meet this demand. In 2017, Apple introduced HEIF (High Efficiency Image File Format), offering about twice the compression efficiency of JPEG. HEIF allows for significant file size reductions without sacrificing image quality, with HEVC being the most commonly used compression method within the format. As of January 2025, Apple's iOS holds about 30% of the global mobile OS market[1], while Android has supported HEIF since version 10. Adobe Photoshop has also integrated HEIF editing capabilities.

1.1 The objective

With the recent rise of accessible and inexpensive multimedia editing tools, it has become possible to easily alter original videos or images without technical expertise. Such malicious tampering can severely compromise the reliability and authenticity of digital media, especially in legal contexts where the content may be used as evidence.

[1] https://gs.statcounter.com/os-market-share/mobile/worldwide

To address these issues, multimedia forensics has been proposed, a research area whose studies are based on the idea that each stage in the lifecycle of a video or image—acquisition, compression, and editing—leaves unique traces that can be analyzed as a "digital fingerprint." By examining these traces, forensic experts can identify the source of the content or determine whether it has been tampered with. These traces include various types of artifacts:

- Physical traces: Imperfections in camera hardware, such as lens distortion [29] or photo-response non-uniformity (PRNU) [39], create unique patterns that can be analyzed to link an image or video to a specific device.

- Compression traces: Artifacts introduced by compression, such as block boundary discontinuities or double compression artifacts, can indicate tampering or re-encoding. [8, 12, 25, 33, 40]

- Camera processing traces: In-camera operations, such as the color filter array (CFA) demosaicing, introduce periodic patterns characteristic of specific camera models. Manipulations, such as splicing from a different camera, disrupt these patterns. [17, 42]

- Editing traces: Image alterations like resizing, rotation, or blurring leave detectable artifacts, particularly along object boundaries or in resampled areas. [2, 13, 30]

Although various methods have been developed to leverage these traces, many challenges remain unsolved.

One of the main research approaches arises from the observation that tampering with an image or video involves decoding, modifying, and re-encoding it, leading to double compression. Detecting this is crucial for verifying content authenticity. However, traditional video forensics methods are based on older standards like H.264/AVC, making them less effective for newer formats such as HEVC. Similar challenges also arise with HEIF images, where significantly less research exists compared to JPEG forensics.

This thesis addresses the challenge of detecting double compression in HEVC-encoded media. It proposes new methods tailored to these emerging standards and improves upon existing approaches.

1.2 Contributions

This thesis proposes two main contributions to the aforementioned multimedia forensic issues:

- Pioneering Research on **Double Compression Detection in HEIF images**: Previous research mainly focused on detecting double compression in JPEG images, but there were no effective methods for HEIF images. This study proposes the first method to detect double compression in HEIF images by leveraging compression characteristics. This study improves upon previous methods by introducing a detection method that does not heavily rely on the image's quantization history. This new approach enables accurate detection of double compression under various compression settings.
- Pioneering Research on **Double Compression Detection in HEVC Videos Containing B-frames**: While research on the double compression of HEVC video is ongoing, limited attention has been given to videos containing B-frames due to the added complexity these frames introduce into analysis methods. This study addresses this gap by proposing a novel classifier that extracts and analyzes encoding information from recompressed videos, effectively capturing the subtle changes caused by B-frame-based recompression. The proposed model not only achieves higher detection accuracy compared to previous methods but also proves to be practical in more realistic scenarios.

Chapter 2

Preliminaries

This chapter overviews the basic video encoding structure and foundational technologies behind the H.265/HEVC encoding standard. Like its predecessor, H.264/AVC, HEVC employs transform and predictive coding as its core framework while introducing new techniques to enhance coding efficiency. Specifically, this section examines predictive structures that improve compression efficiency and provides a detailed explanation of HEVC's key components: Coding Tree Units (CTU), Coding Units (CU), Prediction Units (PU), and Transform Units (TU). It is important to note that while HEVC encompasses all of the compression techniques discussed in this section, HEIF, which is designed for still images, only utilizes intra-prediction for compression and does not use inter-prediction techniques specific to video encoding.

2.1 Basic video encoding

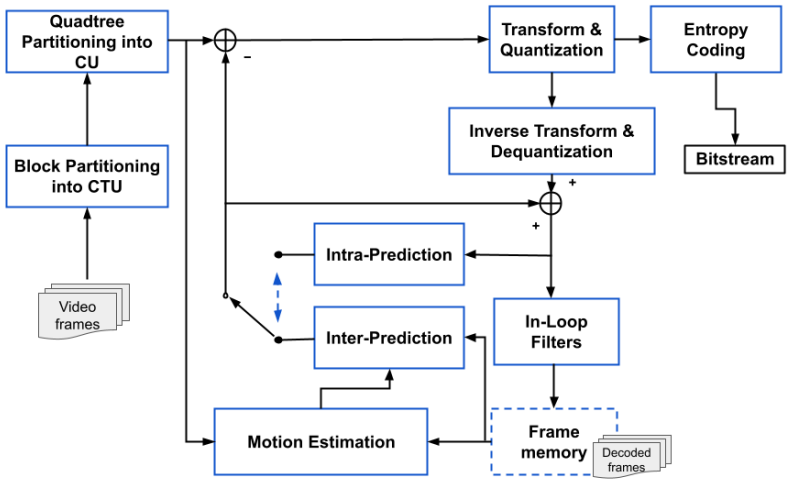


Figure 2.1: Block-based hybrid video encoding schematic diagram.

Video data consists of multiple frames categorized into three types: I-pictures, P-pictures, and B-pictures. An I-picture is typically the first frame and is compressed independently without relying on other frames. Due to this independence, it generally has the largest data size and is a crucial reference point for subsequent frames. A P-picture references a preceding I-picture or another P-picture, compressing only the differences, which reduces its data size. A B-picture references both preceding and succeeding frames, making it more efficient and smaller in data size than a P-picture.

Frames are grouped into GOP (Group of Pictures) structures, which contain a sequence of I-pictures, P-pictures, and B-pictures arranged in encoding order from one I-picture to the next. There are two types of GOPs: closed GOP and open GOP. In a closed GOP, all P-frames and B-frames within the GOP are decoded without referencing frames outside that GOP. In contrast, an open GOP allows frames within the GOP to reference frames from other GOP for decoding. This study focuses solely on closed GOP, where P-frames and B-frames follow an I-frame within the same GOP. Additionally, a sub-GOP refers to the sequence of consecutive B-frames between two reference frames within a GOP.

In HEVC, the prediction structure that determines compression efficiency and encoding/decoding speed can be flexibly configured. The evaluation of encoding performance for HEVC standardization was conducted based on Common Test Conditions (CTC), which define specific prediction structures. CTC defines four prediction structures: All Intra (AI), Random Access (RA), Low-Delay P Picture (LDP), and Low-Delay B Picture (LDB). AI encodes all pictures as I-pictures, while LDP and LDB aim for low delay by using only forward-referenced P-pictures or B-pictures. Because the decoding order matches the display order, these structures minimize delay.

Conversely, RA adopts a hierarchical B structure that uses forward and backward-referenced B pictures to achieve high coding efficiency. However, it may result in slower encoding due to the need for reordering after decoding. Our study used x265 [51] for encoding and configuring RA to not reference other B-frames for B-frame prediction, allowing for a more simplified setup.

2.1.1 Basic technology of H.265/HEVC

HEVC divides frames into blocks using four processing units for efficient encoding: Coding Tree Unit (CTU), Coding Unit (CU), Prediction Unit (PU), and Transform Unit (TU). CTU is the basic unit for partitioning an image, and each CTU consists of Coding Tree Blocks (CTB) for luminance and chrominance components. It is further divided into variable-sized CUs based on recursive quad-tree partitioning. Each CU consists of Coding Blocks (CB) for luminance and chrominance components and is further divided into variable-sized PUs and TUs responsible for prediction and transformation processing, respectively. The introduction of CTU, CU, PU, and TU allows the encoder to adjust processing according to the characteristics of the image and minimize prediction parameters, thereby reducing encoding costs. For example, in image regions with complex changes, many small CUs are allocated with more prediction parameters, such as motion vectors, to improve prediction performance. Conversely, large CUs are used for encoding regions with few changes, minimizing the number of prediction parameters.

2.2 Intra-Prediction

In HEVC, intra-prediction is performed on luminance and chrominance signals to reduce redundancy and increase compression efficiency by taking

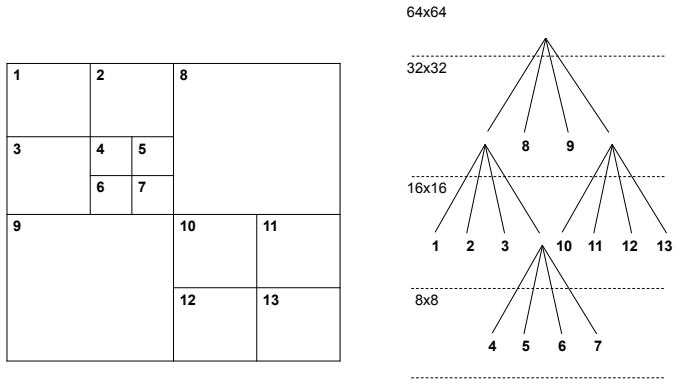


Figure 2.2: Example of the partitioning of 64×64 CTU into various size of CU.

Unit	**Max**	**Min**
CTU	64 × 64	16 × 16
CU	64 × 64	8 × 8
PU	64 × 64	4 × 4
TU	32 × 32	4 × 4

Table 2.1: The maximum and minimum sizes of the four processing units.

advantage of the high correlation between adjacent pixels in an image. The Prediction mode selection is performed at the PU level, while quantization and de-quantization are performed at the TU level. Intra-prediction for the luminance component uses two standard prediction operators (DC and Planar) and 33 directional operators. Directional operators (2-34) predict a target pixel by referring to an encoded pixel at specified angles. Planar prediction (0) uses interpolated values from four adjacent pixels, while DC prediction (1) uses the average value of surrounding pixels. Intra-prediction assigns a mode to each PU, prioritizing the horizontal and vertical patterns commonly found in natural images. Prediction modes closer to these directions have smaller angular displacements than others.

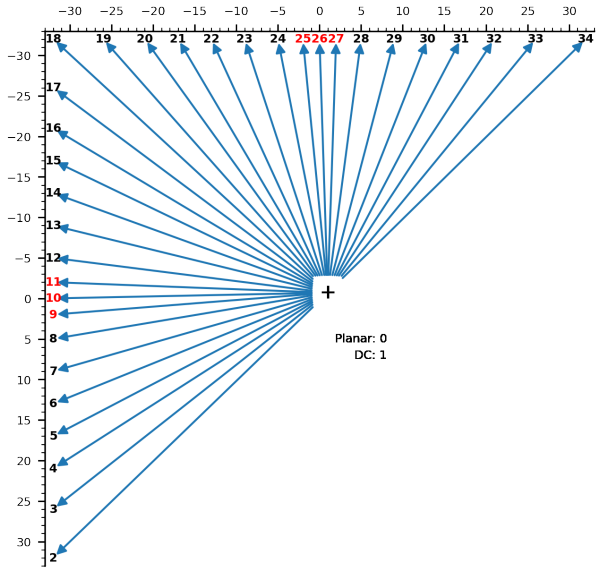


Figure 2.3: Intra-prediction mode for luminance components.

Intra-prediction for chrominance components employs Planar (0), DC (1), horizontal (10), vertical (26), and Intra-derived modes (36). The Planar, DC, horizontal, and vertical prediction modes are explicitly signaled, but if they match the luminance intra-prediction mode, the angular prediction mode (34) is applied instead. In the Intra-derived mode (36), chrominance intra-prediction uses the corresponding luminance intra-prediction mode to reduce signal overhead for encoding.

2.3 Inter-Prediction

In HEVC, as in H.264/AVC, motion compensation prediction is performed block-by-block. From reference frames stored in the frame memory, multiple motion vectors are searched from spatially and temporally adjacent PUs of the target PU for which the prediction vector will be determined. The motion vector with the smallest prediction error is selected from these options. While the expansion and diversification of block sizes have improved prediction efficiency, they have also introduced the drawback of over-segmenting the image compared to conventional methods, resulting in redundant sig-

naling and inefficient boundaries. For example, adjacent blocks often share similar movements in regions with fast-moving objects or areas with clear motion vectors. However, in HEVC, child blocks belonging to different parent blocks cannot share motion information. To address this issue, HEVC employs the merge mode. This technique reduces encoding costs by reusing the motion information (motion vectors, reference picture indices, and prediction directions) of multiple adjacent encoded PUs and transmitting only the index of the PU. Additionally, skip mode is used for smooth image regions where prediction residuals can be ignored.

2.4 Block-based hybrid video encoding

All video coding standards since H.261 have adopted a block-based hybrid video coding approach that combines transform and predictive coding, as previously mentioned. Figure 2.1 illustrates the schematic of the hybrid video coding method employed in HEVC. The input video signal consists of I, B, and P frames, which are divided into blocks. The encoder then selects either intra-prediction or inter-prediction to generate the prediction signal. Typically, prediction blocks are associated with prediction parameters such as motion vectors or intra-prediction modes. Next, the difference between the input video signal and the prediction signal is transformed and quantized, and compressed data is generated using entropy coding. The transformed and quantized difference signal is then inverse-quantized and inverse-transformed and added to the prediction signal. Subsequently, the in-loop filters remove distortions within the frame, which are stored in the frame memory and used for inter-prediction of subsequent frames.

2.5 Entropy Coding and In-Loop Filtering

HEVC employs Context-Adaptive Binary Arithmetic Coding (CABAC) as its entropy coding method. CABAC binarizes syntax elements based on probability distributions and applies arithmetic coding according to their context, adapting the model dynamically to improve compression efficiency. Since CABAC is applied after quantization, it does not influence the quantization parameter (QP) directly. However, it optimizes the encoded bitstream, contributing to compression efficiency and storage requirements. In

our study, we use a fixed QP setting, so CABAC's role is limited to improving compression efficiency without affecting QP selection.

HEVC also incorporates in-loop filters to reduce coding noise in the decoded frames, specifically, the deblocking filter and Sample Adaptive Offset (SAO). These filters mitigate compression artifacts and enhance image quality. The deblocking filter reduces blocking artifacts introduced during the prediction, transformation, and quantization processes by adaptively smoothing block boundaries based on neighboring pixels. SAO, applied after the deblocking filter, classifies image pixels and adjusts their values according to their classification, correcting ringing artifacts and pixel shifts introduced during decoding.

Idempotence in image and video compression refers to the property where repeated encoding with the same settings does not introduce significant changes. However, in-loop filters modify pixel values after decoding, which may affect idempotence. In particular, SAO applies offsets to pixel values, potentially introducing small variations over successive recompressions, even when all other encoding settings remain the same.

This study does not explicitly analyze the impact of in-loop filters on idempotence. Future research could further investigate these effects to better understand the role of idempotence in the compression process and its implications for forensic analysis.

A more detailed discussion on idempotence and its relevance to HEVC compression will be presented in Chapter 4.

Chapter 3

Literature review

This chapter briefly introduces related research on double compression detection for HEIF images and HEVC videos and compares it with studies on JPEG images and AVC videos. The first section provides an overview of JPEG image research, followed by HEIF image studies. The second section reviews research on AVC videos, concluding with studies on double compression detection in HEVC videos.

3.1 HEIF

As the JPEG format [49] has been widely used in most digital cameras and image processing tools for decades, much of the image forensics research has addressed this coding format. In particular, starting from the hypothesis that manipulation is obtained by reading a JPEG image, editing it, and saving it again in JPEG format, one of the most used solutions to detect tampering is examining the artifacts left during JPEG recompression. These artifacts can be classified into two categories: aligned double JPEG (A-DJPG) compression, in which the discrete cosine transform (DCT) grid of the first and second JPEG compressions align, and non-aligned double JPEG (NA-DJPG) compression, in which they do not align. Research in A-DJPG compression has explored the double quantization (DQ) effect that alters the histogram of DCT coefficients [4, 37], or Benford's law [1, 14, 31], and leveraged the idempotence of quantization [12]. On the other hand, studies on NA-DJPG compression observed changes in the regularity of block

artifacts [33, 40, 43, 54] and clustering patterns of DCT coefficients [7].

Although HEIF is considered a potential successor to JPEG, research on HEIF images in the context of digital forensics is limited [3, 41], and even more restricted on HEIF double compression detection [16]. This is probably because JPEG and HEIF standards use different encoding techniques, compression algorithms, and formats; therefore, directly applying JPEG double compression detection methods to HEIF images may not yield effective results.

On the other hand, research on double compression detection in HEVC videos, which use the same encoding technology as HEIF images, has attracted attention. A video consists of several frames, including P-frames predicted from past frames, B-frames predicted from both past and future frames, and I-frames that use spatial correlation to predict neighboring pixels and are coded independently of neighboring frames. A continuous group of frames starting with an I-frame and consisting of P- or B-frames is called a Group of Pictures (GOP).

In practical scenarios, an I-frame may be re-encoded using a GOP of a different length than the original and encoded as a P-frame. This frame, known as a relocated I-frame, breaks the temporal and spatial consistency in the GOP and serves as a clue for detecting double compression [20, 27, 53]. Other studies focus on changes in quantized DCT coefficients [24] and differences in encoded elements within P-frames [35] between single-compressed and double-compressed videos.

When a video is re-encoded using the same quality parameter, the encoding history is overwritten, making double compression detection difficult. Jiang et al. found that changes in encoded elements within I-frames are most significant between single-compressed and double-compressed videos and tend to remain stable with additional compressions, aiding the detection of double compression [28].

These previous works on double compression detection in HEVC videos can provide useful insights for research on double compression detection in HEIF images. However, still images contain only a single frame and cannot utilize temporal correlation like videos. Therefore, detecting double compression in HEIF images must rely on spatial correlation only. Additionally, there are limitations on the feature vectors that can be extracted from the dataset, resulting in fewer materials available for input into the double compression classifier.

3.2 HEVC

Wang *et al.* demonstrated that when a video is recompressed after frame deletion or insertion, periodic prediction errors occur because the video is encoded with a different Group of Pictures (GOP) structure than the original. [50] Based on this, Stamm *et al.* extended the research and proposed a method to detect frame deletion or insertion. [44] When the GOP structure changes between the first and second encoding processes, frames originally encoded as I-frames may be re-encoded as P or B-frames, resulting in significant differences from the original P or B-frames. [19] and [18] utilized the re-encoded frames, and they referred to them as "relocated I-frames (RI-frames)." Vázquez-Padín *et al.* analyzed anomalies in macroblock types within RI-frames in double-compressed videos and named this characteristic "Variation of Prediction Footprints (VPF)." They extended their research [46] to consider B-frames. [47] However, these studies focused on videos encoded with AVC or MPEG and may not apply to videos encoded with HEVC. To address this issue, new detection techniques that consider the specific characteristics of HEVC are currently being actively researched.

Research on detecting double compression in HEVC videos can be classified into two categories: "aligned GOP," where the GOP structure used in the first and second encoding processes is the same, and "non-aligned GOP," where the GOP structure differs between the two encodings. In the case of non-aligned GOP, it is necessary to consider RI-frames.

Researchers have proposed various methods for the case of aligned GOP to identify double-compressed videos. For instance, SVM-based methods have been developed that use the co-occurrence matrix of DCT coefficients or the distribution of Transform Unit (TU) size and DCT coefficients, which vary with the quantization parameter (QP), to detect double compression. [24, 32, 34] Other approaches include using compression idempotency to identify transcoded HEVC videos [9] and detecting double-compressed videos that have been re-compressed with the same QP. [11, 26] Additionally, histogram-based methods have been used to detect double compression by analyzing changes in the sizes of coding units, such as Prediction Unit (PU) and Transform Unit (TU). [35, 55] Some studies have observed that changes in encoding elements within I-frames are most pronounced between single and double-compressed videos and tend to converge when additional compression is applied. [28] Other researchers have extracted features from intra-prediction in I-frames and inter-prediction in P-frames and B-frames to detect fake

high-definition video. [56]

In research on non-aligned GOP, several studies have utilized spatial inconsistencies observed in RI-frames and temporal inconsistencies with adjacent frames as features for detecting double compression. [20, 23, 27, 53]. Additionally, other methods have been developed to analyze the quality degradation process of double compression in non-aligned GOP structures and the decision mode of in-loop filtering. [52] In real-world scenarios, the GOP size set by the device that recorded the original video and the editing tools often differs. Therefore, the re-encoded video is expected to use a different GOP structure.

Chapter 4

Double Compression Detection in HEIF images

This chapter presents two methods for detecting double compression in HEIF images, presented in a conference paper [16] and in an extended journal paper [15], discussing both the approaches and their outcomes. First, we provide an overview of the method that applies double compression detection techniques developed for JPEG images to HEIF images, focusing on the initial approach, which extends Farid's coding ghost theory in JPEG. We then present a new method involving a lightweight image classifier, proposed to address the limitations of the previous approach and discuss how this method enhances accuracy and efficiency compared to traditional techniques.

4.1 Introduction

Our research is inspired by previous studies on JPEG compression idempotency, which Farid proposed [12]. Compression idempotency means that when an original image is compressed repeatedly with the same encoding parameters, the resulting compressed image remains close to the original in terms of visual quality and characteristics.

Idempotency can be expressed as follows [5]: In JPEG compression, a color image transforms luminance (Y) and chrominance (Cb and Cr) channels, partitioning into 8×8-pixel blocks. These blocks are then subjected to

Discrete Cosine Transform (DCT), converting the image data from spatial to frequency domains. The frequency domain components (i.e., the DCT coefficients) of the input image (D) are quantized using a quantizer Q with step size Δ_1 (Q_{Δ_1}). This process involves dividing each frequency component by Δ_1 and rounding ($\lfloor . \rfloor$) the result to derive D'. While pixel rounding is inherent in the quantization process, its effect is negligible compared to the influence of the quantization step size Δ_1. Since rounding occurs consistently across all compression operations, it does not significantly impact the idempotency property of compression. Therefore, we do not explicitly consider pixel rounding in our analysis.

$$D' = Q_{\Delta_1}(D) = \Delta_1 \left\lfloor \frac{D}{\Delta_1} \right\rfloor, \tag{4.1}$$

Let us note that the quantization step size (Δ_1) is determined by a quality factor (QF). Higher QF values yield smaller Δ_1, preserving image quality, while lower QF values lead to larger Δ_1, facilitating image compression at the expense of quality. Suppose now that the compressed image is re-compressed, using the quantization parameter Q_{Δ_2}. The frequency domain components D'' are obtained. Idempotency consists in that if the same quantization parameter is used in the two compression processes (i.e., $\Delta_2 = \Delta_1$), D'' will be equal to D'.

$$D'' = Q_{\Delta_2}(D') = \Delta_2 \left\lfloor \frac{\Delta_1 \left\lfloor \frac{D'}{\Delta_1} \right\rfloor}{\Delta_2} \right\rfloor = \Delta_1 \left\lfloor \frac{\Delta_1 \left\lfloor \frac{D'}{\Delta_1} \right\rfloor}{\Delta_1} \right\rfloor = D'. \tag{4.2}$$

We applied this idea to HEIF images since the compression processes in JPEG and HEIF are based on similar concepts.

Consider D_0 as the collection of DCT coefficients from an image I_0, quantized with Q_0 as a QF. Let us assume that an uncompressed image, denoted by I, undergoes subsequent compression with a different QF, Q_1, resulting in the coefficients set D_1. Farid's research [12] demonstrated that the disparity between D_0 and D_1 is minimized when $Q_0 = Q_1$, owing to the idempotent nature of JPEG compression.

This idempotency forms the basis for verifying double compression. Starting with an uncompressed image (I) compressed initially at Q_0, followed by a second compression at Q_1 (assuming $Q_0 < Q_1$) to obtain image I_1, the resulting image data include DCT coefficients quantized with both Q_1 and

Q_0. For experimental verification, the image I_1 undergoes recompression using a quantization value Q_2, yielding the image I_2 with a corresponding set of DCT coefficients D_2. To analyze the effects of double compression, the sum of squared differences (SSD) is to quantitatively evaluate the discrepancy between D_1 and D_2. As previously discussed, the discrepancy between D_1 and D_2 is minimized when Q_2 is the same as Q_1 since the quantization effects remain consistent. However, considering that D_2 encompasses data initially quantized with Q_0, an additional minimum in SSD occurs when Q_2 is the same as Q_0. This additional minimum is commonly termed the "JPEG ghost."

The behavior of the JPEG ghost with varying quantization step sizes is illustrated in Figure 4.1. In the left figure, we observe the SSD among coefficients quantized using a step size of $Q_1 = 25$, followed by a subsequent quantization within the range $Q_2 \in [1, 30]$. The minimal difference occurs precisely when $Q_2 = Q_1 = 25$. In the right figure, we examine the SSD among coefficients initially quantized at $Q_0 = 10$, then $Q_1 = 25$, and further quantized within the same range $Q_2 \in [1, 30]$. Here, the minimum discrepancy arises at $Q_2 = Q_1 = 25$, accompanied by a local minimum at $Q_2 = Q_0 = 10$. In Farid's analysis, this comparison can also be conducted in the spatial domain using RGB pixel values. By identifying re-compressed versions with the smallest variances, we can detect double compression by observing the presence of the JPEG ghost.

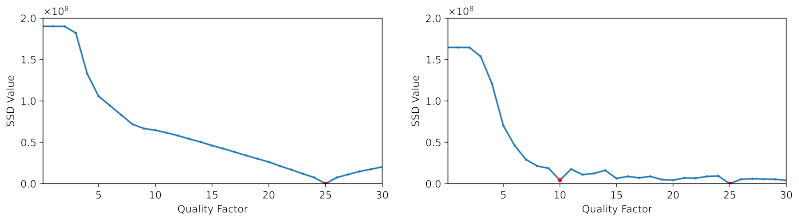


Figure 4.1: Sum of squared differences (SSD) vs. quality factor. (**Left**: Single-compressed image; **Right**: Double-compressed image)

The attack described in Farid's research involves an attacker copying tampered regions from another JPEG image and pasting them into a target image. Our study generalizes this approach by using double-compressed images rather than tampered ones. Since tampering implies that the image has been decoded and re-encoded, we focus on detecting entire images affected

by double compression rather than specific regions copied from other JPEG images.

4.2 Proposed Method

Ghost analysis is a method that involves recompressing an input image with varying quality levels and calculating the difference between the original and recompressed images. However, Farid's method primarily focuses on detecting forgeries and lacks an automated algorithm for identifying double compression. Additionally, the shape of the difference plot is influenced by the visual content of the image, making it difficult to capture local variations in flat images, such as the sky, where pixel differences tend to be smaller. This issue is especially pronounced in HEIF-compressed images, where local prediction is applied within each CTB, leading to stronger content dependency. To address this problem, we introduced a calibration technique to distinguish between features related to the image content and those related to the presence of double compression. Specifically, we shifted the pixel values of the input image by 15 pixels to the right in a circular manner, breaking the alignment of the compression grid and thus avoiding ghost artifacts. This calibration method has been proven effective in previous studies [6, 38].

Figure 4.2 shows that we calculated two mean absolute error (MAE) differences to detect double compression in HEIF images. One compares the input image with its recompressed version at varying QPs, while the other compares the calibrated image with its recompressed version. By comparing these MAE difference vectors and applying a detection rule, we successfully identified the presence of double compression. This method effectively minimizes the influence of the visual content of the image and accurately captures local MAE variations.

4.2.1 MAE difference extraction

Let I denote a single- or double-compressed input image. Our method starts by recompressing I with varying QP values from 1 to 51, resulting in 51 recompressed images labeled as $\widehat{I}_1, \widehat{I}_2, ..., \widehat{I}_{51}$. For each recompressed image, we compute its mean absolute error (MAE) against the original input image using the following equation, where (x, y) represents pixel coordinates within images of dimensions W (width) and H (height):

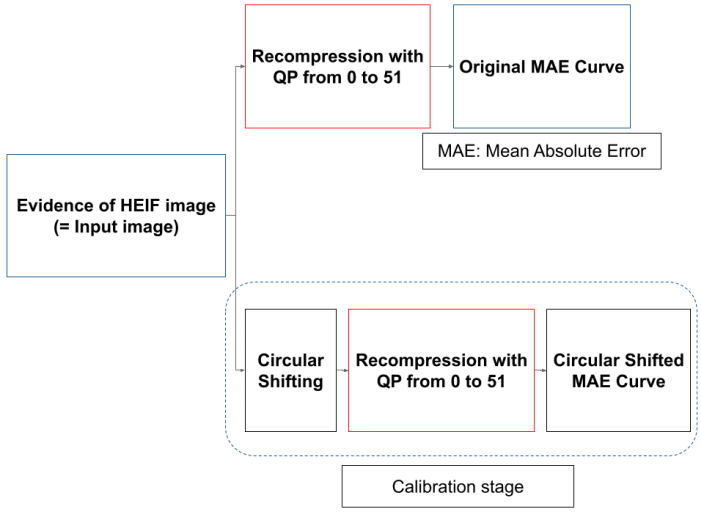


Figure 4.2: MAE calculation process

$$MAE(I, \widehat{I}_{QP}) = \frac{1}{WH} \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} |I(x,y) - \widehat{I}_{QP}(x,y)|. \tag{4.3}$$

Next, let CI represent the input image after circular shifting. We perform recompression on CI using QP values ranging from 1 to 51, resulting in 51 recompressed images labeled as $\widehat{CI}_1, \widehat{CI}_2, ..., \widehat{CI}_{51}$. Similarly, we compute the MAE between each recompressed circular shifted image and the original input image for each QP value. Figure 4.3 illustrates an example of the MAE curves and their MAE differences for a single- and double-compressed image. The upper figures show an image encoded with $QP = 24$, while the lower figures show that it is first encoded with $QP_1 = 35$ and then recompressed with $QP_2 = 24$. The left side of the figure displays the original MAE curve and the circular shifted MAE curve for the input image, while the right side presents the MAE differences at each QP value. This figure focuses on the case where $QP_1 > QP_2$, as it is the primary scenario in which HEIF ghosts appear. Experimental results confirm that when $QP_1 < QP_2$, no noticeable peaks emerge in the MAE difference plot, indicating the absence of double compression artifacts. Therefore, the proposed method does not apply to cases where $QP_1 < QP_2$. The upper image shows a

maximum MAE difference of 24, indicating a single compressed image. On the other hand, in the lower image, the MAE difference is observed to be larger at 24 and 35 than the other QP values on the x-axis, suggesting that the input image is double-compressed. This behavior can be attributed to the compression characteristics of HEIF. HEIF encoding exhibits a certain degree of idempotency, meaning that when recompressed at the same QP value, pixel differences remain minimal, resulting in a dip in the MAE curve. In contrast, when an image is recompressed at a slightly different QP value, the difference in pixel values becomes more pronounced, as the recompression process does not perfectly preserve the original encoding structure. This results in distinctive peaks in the MAE difference plot, highlighting that HEIF encoding is not entirely invariant to small changes in QP values but rather tends to minimize changes when recompressed at the same QP.

$$MAE(CI, \widehat{CI}_{QP}) = \frac{1}{WH} \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} |CI(x,y) - \widehat{CI}_{QP}(x,y)|. \tag{4.4}$$

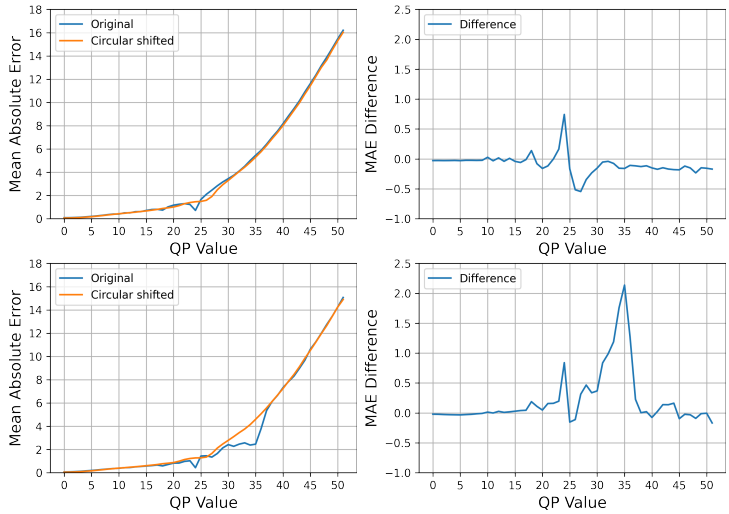


Figure 4.3: MAE curves and their MAE differences at each QP. (**Top**: Single-compressed image ($QP = 24$); **Bottom**: Double-compressed image ($QP_1 = 35$, $QP_2 = 24$))

4.2.2 MAE difference analysis

This section explains how we propose to detect HEIF double compression by analyzing the MAE difference plot. The proposed approach assumes that the input image's current quality factor (QP_2) is known. This assumption is easily satisfied, as the QP value used in the last encoding of an image can be readily determined by examining the file header. For the reasons presented in the previous section, when an image undergoes double compression, an additional peak should be visible to the right of QP_2, as shown in Figure 4.3. $X = [X_1, X_2, X_3, \ldots, X_{51}]$ represents an array to store the MAE difference for each QP value. Detecting double compression involves comparing the ratio of the overall MAE difference energy (denoted as $W = \sum_{i=1}^{51}(X_i)^2$) to the MAE difference energy in the right-hand portion of the QP_2 (denoted as $R = \sum_{i=QP_2+1}^{51}(X_i)^2$) against a specific threshold. We define the positive class as double-compressed images and the negative class as single-compressed images. The proposed method for detecting double compression is as follows:

$$\text{Double compression decision} = \begin{cases} 1, & \text{if } \frac{\mathrm{R}}{\mathrm{W}} > \text{Threshold} \\ 0, & \text{otherwise.} \end{cases} \tag{4.5}$$

The reason for focusing only on the right-hand portion of the MAE difference energy (from QP_2 to higher values) is that HEIF ghosts predominantly appear in the higher QP region due to the accumulation of residual errors caused by double compression. When an image is re-encoded with a lower QP value ($QP_2 < QP_1$), regions previously encoded with a high QP_1 retain larger residual errors. Since higher QP values cause greater information loss due to quantization, the effects of the first compression become more pronounced.

Including the entire QP range in the analysis may introduce unnecessary noise, as natural image characteristics and local contrast variations can lead to large MAE differences even in single-compressed images. Furthermore, in double-compressed images, the influence of regions re-encoded with a lower QP may spread, diminishing distinct compression artifacts. This increases the risk of misclassification, where double-compressed images may be misidentified as single-compressed or vice versa. By focusing only on the right-hand portion of the MAE difference, we effectively suppress such noise, reduce misclassification, and enhance detection sensitivity.

However, the proposed method works only when the QP used for the first encoding (QP_1) is larger than the QP used for the second encoding (QP_2). Theoretically, if $QP_1 = QP_2$, no coding ghosts occur. Experimental validation further confirms that when $QP_1 < QP_2$, HEIF ghosts do not emerge, rendering the proposed detection method ineffective in such cases.

4.3 Extended Proposed Method

To ensure that double-compressed images can be detected without depending heavily on the compression history of the input image, we constructed a feature vector that captures changes in encoding elements between the input image and its recompressed version. This feature vector includes statistical measures that highlight differences in encoding behavior caused by double compression. The flowchart of the proposed method is shown in Figure 4.4, which visually represents the feature extraction process involved in detecting double compression in HEIF images.

In the subsequent sections, after outlining our existing approach for detecting double compression in HEIF images, we discuss the changes occurring in encoding elements between the input image and its recompressed version. We also detail the new feature vector and the SVM classifier used in our research. Again, the proposed method assumes that the QP used in the last encoding of the input image is known. This assumption is easily fulfilled since the QP value used in the last encoding can be identified by examining the file header.

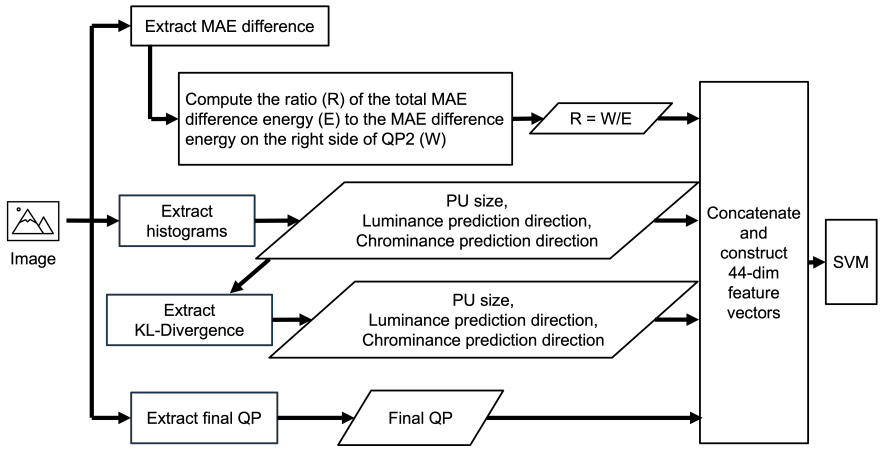


Figure 4.4: Flowchart of the extended proposed method.

According to the previous analysis, we detected double compression in HEIF images by analyzing the MAE difference plot. As explained earlier, when an image undergoes double compression, an additional peak should appear to the right of QP_2, as illustrated in Figure 4.3. In contrast, no additional peak appears to the right of the last QP value for single compression. We represent the MAE differences for each QP value as an array, denoted by $X = [X_1, X_2, X_3, \ldots, X_{51}]$. Detecting double compression involves comparing the ratio of the total MAE difference energy (denoted as $E = \sum_{i=1}^{51}(X_i)^2$) to the MAE difference energy in the right-hand portion of QP_2 (denoted as $W = \sum_{i=QP_2+1}^{51}(X_i)^2$), that is for all QP values higher than QP_2. In our new approach, the ratio ($R = \frac{W}{E}$) serves as one of the image's feature vectors without setting a specific threshold.

4.3.1 Statistical Analysis for Changes of Encoding Elements between Images

HEVC encodes images on a block-by-block basis, and the selection of the PU size and prediction mode can vary between compression cycles due to quantization errors and rate distortion (RD) cost optimization. Previous research on HEVC video suggests that the change in PU size between I-frames of single-compressed and double-compressed images is larger than that between double-compressed and triple-compressed images at the same

QP [28].

We utilized the Kullback-Leibler Divergence (KLD) to compare the changes in encoding elements between input images and their recompressed counterparts. KLD is particularly useful in this context as it measures the distance between two probability distributions, allowing us to quantify the difference in encoding element distributions caused by different compression cycles. This helps identify the compression artifacts and discrepancies that are more pronounced in double compression than in single compression.

To obtain the KLD values, we used the open-source bitstream converter heic2hevc [21] to convert the input and recompressed images into HEVC bitstream. The bitstream was then decoded using the HM 16.25 [22] to extract information on PU sizes (assuming values 64 × 64, 32 × 32, 16 × 16, 8 × 8, 4 × 4), luminance prediction directions (with possible values 0, 1, 9, 10, 11, 25, 26, 27), and chrominance prediction directions (with possible values 0, 1, 10, 26, 34, 36) for each 4 × 4 block. We computed histograms for each encoding element based on the information extracted from each 4 × 4 block. Laplace smoothing with a parameter $\alpha = 1$ was applied to avoid zero probabilities in the probability distribution.

In particular, the smoothed probabilities $\widehat{p}$ for an input image and $\widehat{q}$ for its recompressed image are calculated as follows:

$$\widehat{p_i} = \frac{C_{p,i} + \alpha}{\sum_{j=1}^{N} C_{p,j} + \alpha N}, \quad \widehat{q_i} = \frac{C_{q,i} + \alpha}{\sum_{j=1}^{N} C_{q,j} + \alpha N} \quad (i = 1, 2, 3, \ldots, N). \tag{4.6}$$

where $C_{p,i}$ and $C_{q,i}$ are the aggregate number of coding elements in each category, $\sum_{j=1}^{N} C_{p,j}$ and $\sum_{j=1}^{N} C_{q,j}$ are the corresponding total aggregate number of coding elements, and N is the number of categories.

The KL divergence between probability distributions P and Q is given by:

$$KL(P \parallel Q) = \sum_{i=1}^{N} \widehat{p_i} \log \left(\frac{\widehat{p_i}}{\widehat{q_i}} \right) \quad (i = 1, 2, 3, \ldots, N). \tag{4.7}$$

Thus, for PU sizes (64 × 64, 32 × 32, 16 × 16, 8 × 8, 4 × 4), luminance prediction directions (0, 1, 9, 10, 11, 25, 26, 27), and chrominance prediction directions (0, 1, 10, 26, 34, 36), the KL divergences are computed as follows:

1. PU sizes (five categories):

$$\widehat{p}_{PU_i} = \frac{C_{p_{PU,i}} + 1}{\sum_{j=1}^{5} C_{p_{PU,j}} + 5}, \quad (i = 1, 2, 3, 4, 5), \tag{4.8}$$

$$\widehat{q}_{PU_i} = \frac{C_{q_{PU,i}} + 1}{\sum_{j=1}^{5} C_{q_{PU,j}} + 5}, \quad (i = 1, 2, 3, 4, 5), \tag{4.9}$$

$$KL(P_{PU} \parallel Q_{PU}) = \sum_{i=1}^{5} \widehat{p}_{PU_i} \log \left(\frac{\widehat{p}_{PU_i}}{\widehat{q}_{PU_i}} \right). \tag{4.10}$$

2. Luminance prediction direction (eight categories):

$$\widehat{p}_{LUMA_i} = \frac{C_{p_{LUMA,i}} + 1}{\sum_{j=1}^{8} C_{p_{LUMA,j}} + 8}, \quad (i = 1, 2, \ldots, 8), \tag{4.11}$$

$$\widehat{q}_{LUMA_i} = \frac{C_{q_{LUMA,i}} + 1}{\sum_{j=1}^{8} C_{q_{LUMA,j}} + 8}, \quad (i = 1, 2, \ldots, 8), \tag{4.12}$$

$$KL(P_{LUMA} \parallel Q_{LUMA}) = \sum_{i=1}^{8} \widehat{p}_{LUMA_i} \log \left(\frac{\widehat{p}_{LUMA_i}}{\widehat{q}_{LUMA_i}} \right). \tag{4.13}$$

3. Chrominance prediction direction (six categories):

$$\widehat{p}_{CHROMA_i} = \frac{C_{p_{CHROMA,i}} + 1}{\sum_{j=1}^{6} C_{p_{CHROMA,j}} + 6}, \quad (i = 1, 2, \ldots, 6), \tag{4.14}$$

$$\widehat{q}_{CHROMA_i} = \frac{C_{q_{CHROMA,i}} + 1}{\sum_{j=1}^{6} C_{q_{CHROMA,j}} + 6}, \quad (i = 1, 2, \ldots, 6), \tag{4.15}$$

$$KL(P_{CHROMA} \parallel Q_{CHROMA}) = \sum_{i=1}^{6} \widehat{p}_{CHROMA_i} \log \left(\frac{\widehat{p}_{CHROMA_i}}{\widehat{q}_{CHROMA_i}} \right). \tag{4.16}$$

Figures 4.5–4.7 illustrate box-and-whisker plots showing the distribution of KL divergence between input images and their recompressed counterparts for PU size, luminance intra-prediction direction, and chrominance intra-prediction direction, respectively. The x-axis represents the QP values used to generate the input and recompressed images, plotted from left to right

in the order of single-compressed image (S) vs. double-compressed image (D) and double-compressed image (D) vs. triple-compressed image (T). The y-axis represents the KL divergence, computed by encoding 150 different images. Each plot presents the results for each QP scenario from left to right as follows: (1) QP_1 is larger than QP_2, (2) QP_1 is equal to QP_2, and (3) QP_2 is larger than QP_1. The bottom of the box indicates the first quartile (Q1), and the top indicates the third quartile (Q3). The yellow line inside the box represents the median (Q2) of the data. The whisker extending from the bottom of the box shows the range from Q1 to 1.5 times the interquartile range (IQR), and the whisker extending from the top of the box shows the range from Q3 to 1.5 times the IQR. Data points beyond these ranges are considered outliers. The results in scenarios (1) and (2) show that the variation in KL divergence between double-compressed and triple-compressed images is usually smaller than in KL divergence between single-compressed and double-compressed images. This is evident from the higher Q2 values and wider IQRs of the scenarios in most cases. On the other hand, scenario (3) does not show a clear difference in KL divergence, which makes it difficult for the classifier to identify the double-compressed images.

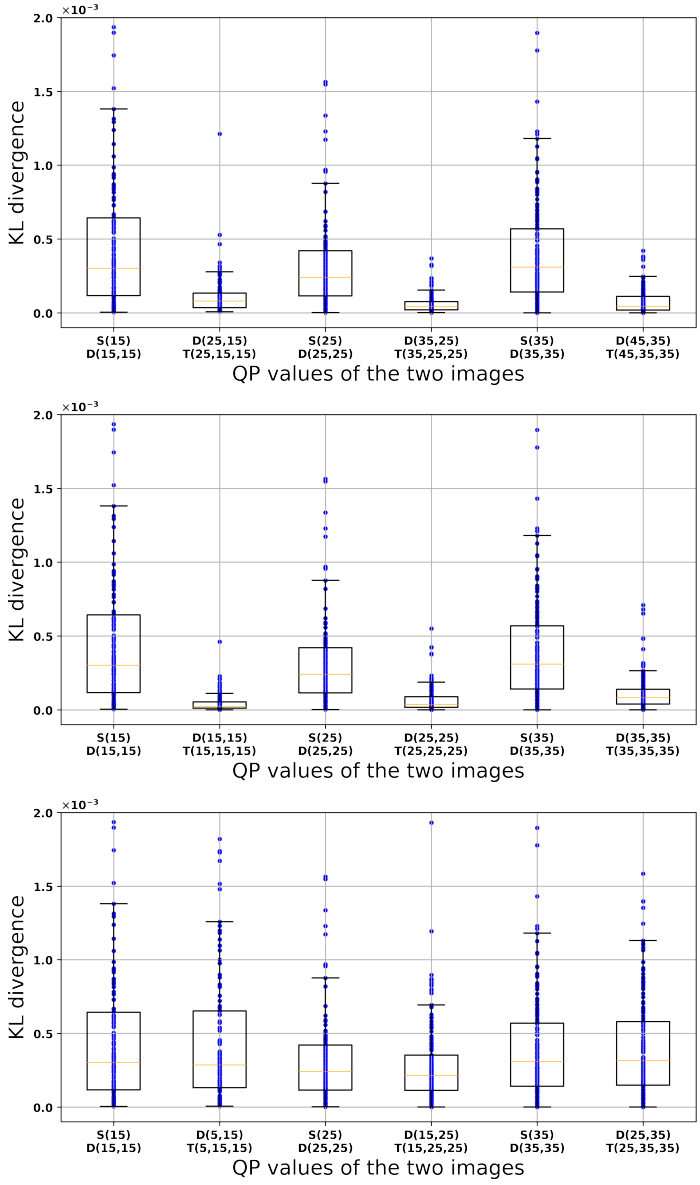


Figure 4.5: Box-and-whisker diagram of KL divergence for PU size. (**Top**: $QP_1 > QP_2$; **Middle**: $QP_1 = QP_2$; **Bottom**: $QP_1 < QP_2$)

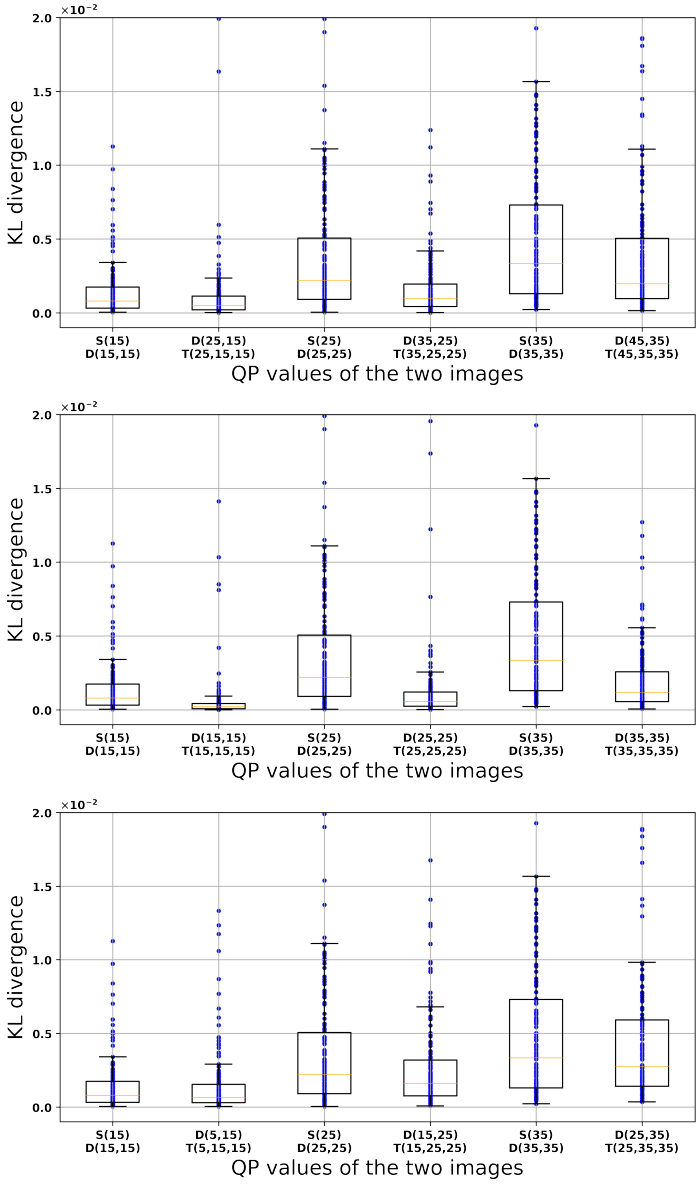


Figure 4.6: Box-and-whisker diagram of KL divergence for luminance intra-prediction direction.
(**Top**: $QP_1 > QP_2$; **Middle**: $QP_1 = QP_2$; **Bottom**: $QP_1 < QP_2$)

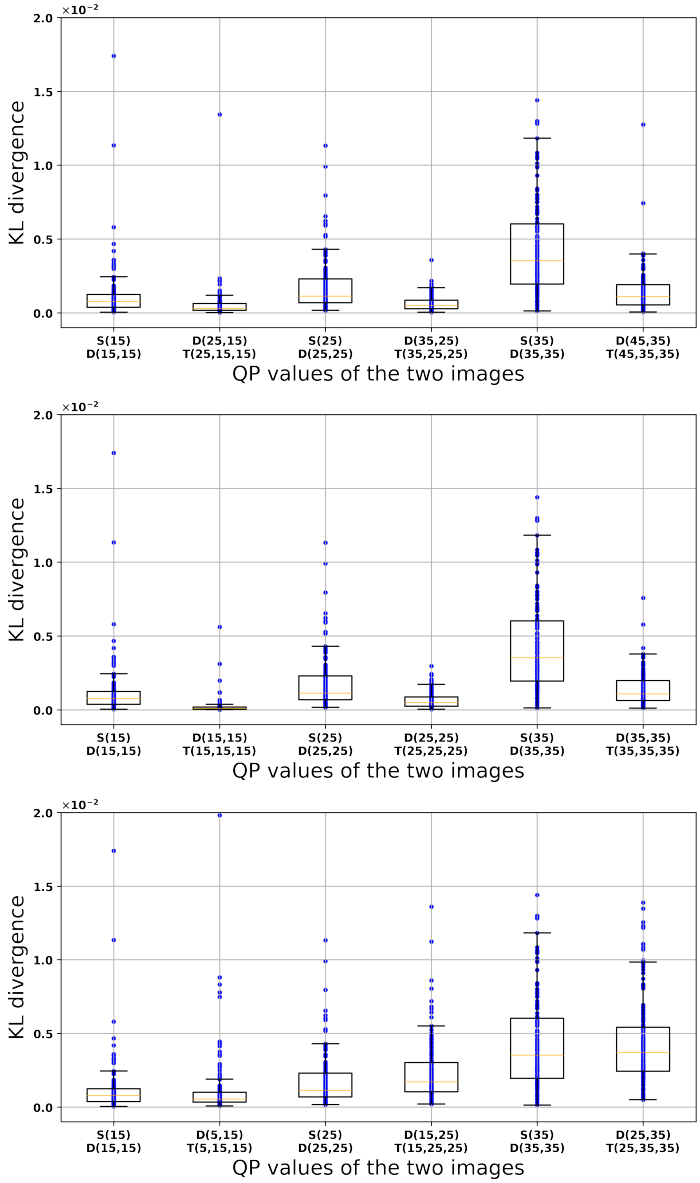


Figure 4.7: Box-and-whisker diagram of KL divergence for chrominance intra-prediction direction.
(**Top**: $QP_1 > QP_2$; **Middle**: $QP_1 = QP_2$; **Bottom**: $QP_1 < QP_2$)

4.3.2 Combining Feature Vectors

The above analysis compared the energy ratio of the entire MAE difference plot to the energy of the right-hand side of QP_2. By training the classifier to learn this ratio, we obtained an algorithm equivalent to the model used in [16] without setting a specific threshold. Furthermore, we overcame the model's limitations in [16] by observing that the more times images are compressed, the smaller the change in coding coefficients between them. The feature vectors reflecting the results of the analysis are as follows:

1. The ratio of the MAE difference energy to the total MAE difference energy on the right side of QP_2.
2. The histograms of PU size, luminance intra-prediction direction, and chrominance intra-prediction direction.
3. The KL divergence for the variation between images concerning PU size, luminance intra-prediction direction, and chrominance intra-prediction direction.
4. The QP value used for the last encoding.

From the information presented above, a 44-dimensional feature vector per image pair was created for a single-compressed image and its recompressed image or a double-compressed image and its recompressed image pair. This feature vector was input to an SVM classifier with a linear kernel to train and test a model for classifying single- and double-compressed images. We used Scikit-learn's Support Vector Classifier (SVC) with a linear kernel. Min-Max scaling was applied to the dataset to ensure that all features contributed equally to the SVM classifier. Each feature was scaled to a range between 0 and 1. To determine the optimal value of the regularization parameter C, we performed a grid search over the following set of C values: {0.01, 0.1, 1, 10, 100, 1000, 2000, 3000, 4000, 5000}. The model's performance was evaluated using cross-validation on the training set, and the C value that yielded the highest cross-validation accuracy was selected as the optimal parameter for our final model.

4.4 Experimental Results

This section describes the experimental validation of the proposed model. We outline the procedure for creating the dataset and compare the accuracy

of our double compression detector with conventional approaches in different scenarios. We also investigate the robustness of our method against images generated using different encoding tools.

In the evaluation process, single-compressed images are labeled as negative and double-compressed images as positive. There is no overlap in image content between the training and test data. For clarity, the single-compressed image and its recompressed version are collectively described as a single-compressed image pair, and the double-compressed image and its recompressed version are described as a double-compressed image pair.

4.4.1 Dataset

The dataset for our experiments comprises 300 TIF images featuring various indoor and outdoor scenes, including landscapes, buildings, objects, and nature. These images were captured using three camera models (Nikon D90, Nikon D40, and Nikon D7000) and were selected from the highly cited RAISE forensic dataset [10].

We included all 76 available images from the Nikon D40. The remaining 232 images were chosen from the Nikon D90 (116 images) and Nikon D7000 (116 images), making up a total of 308 images. From these 308 images, we selected 300 for our dataset. The overall breakdown of image content in the dataset is as follows: 70 buildings, 66 indoor scenes, 53 outdoor scenes, 41 objects, 40 nature scenes, and 38 landscapes. Some images contain multiple types of content. The image data can be accessed at [45].

While the primary focus of our experiment is on detecting double compression rather than specific content or camera models, we ensured a diverse selection to cover various scenarios and improve the generalizability of our results.

All images were cropped to a 3:2 aspect ratio and resized to 1200×800 pixels using the INTER-AREA interpolation algorithm from the OpenCV library to prevent aliasing. Finally, all images were saved in PNG format.

In this study, we encode and decode images using the open-source HEIF implementation, libheif [36]. Specifically, a PNG image was initially encoded at QP_2 to generate a single-compressed image. Subsequently, this single-compressed image was recompressed at QP_2 to produce a recompressed single-compressed image. Similarly, the process of generating double-compressed images involved encoding a PNG image at QP_1 and recompressing it at QP_2, resulting in a double-compressed image. This double-

compressed image was then recompressed at QP_2 to create a recompressed double-compressed image. The QP_1 values were selected from the set {10, 15, 20, 25, 30, 32, 35, 40, 45, 50}, while the QP_2 values were chosen from {5, 10, 16, 20, 24, 27, 32, 39, 42, 45}.

The maximum CTU size was set to 64. HEIF encoding utilized x265, a popular open-source HEVC encoder offering ten predefined preset options balancing encoding speed and image quality. This study employed the default (0) "medium" preset. It is important to note that x265 typically applies the input QP to the P-slice and adjusts the QP of the I-slice using an offset. To ensure the direct impact of the input QP on our HEIF still images, the offset value was adjusted to zero using the related libheif command (./heif-enc -p x265:ipratio = 1.0).

The above procedure generated 3000 single-compressed image pairs (3000 single-compressed images and 3000 recompressed single-compressed images.) The encoding process for the double-compressed image pairs was performed for each of three scenarios: (a) when QP_1 exceeds QP_2, (b) when QP_1 equals QP_2, and (c) when QP_2 exceeds QP_1. In scenario (a), 17,100 double-compressed and 17,100 recompressed double-compressed images were generated. In scenario (b), 3000 double-compressed and 3000 recompressed double-compressed images were generated. In scenario (c), 11,700 double-compressed and 11,700 recompressed double-compressed images were generated.

4.4.2 Performance Evaluation on Double Compression Detection for Mixed QP Scenario

To assess the performance of the double compression classifier on a test dataset containing images generated in all QP scenarios, we performed a 10-fold cross-validation on a dataset consisting of 3000 single-compressed image pairs and 3000 double-compressed image pairs. The double-compressed image pairs were selected equally from the three QP scenarios (1000 each) and added to the dataset. In every fold of the training process, 600 image pairs (300 single-compressed image pairs and 300 double-compressed image pairs) were selected from the entire dataset as a test dataset. The remaining 5400 image pairs were split into 4800 for training and 600 for validation. The best model was calculated by varying the candidate values of the regularization parameter C, and its generalization performance was evaluated using the best model and test data.

Figure 4.8 shows the average performance of ten evaluations. For evaluation, we considered the true positive rate (TPR), true negative rate (TNR), and accuracy (ACC). TPR, indicating the proportion of actual positive samples correctly predicted, was calculated as $\text{TPR} = \frac{\text{TP}}{\text{TP+FN}}$, while TNR, representing the proportion of actual negative samples correctly predicted, was calculated as $\text{TNR} = \frac{\text{TN}}{\text{TN+FP}}$. We also calculated accuracy as $\text{ACC} = \frac{\text{TPR+TNR}}{2}$. Standard deviations (SD) were also computed for these metrics. Our model achieved an accuracy of 81% and clearly outperformed the model in [16], which only considered coding ghosts.

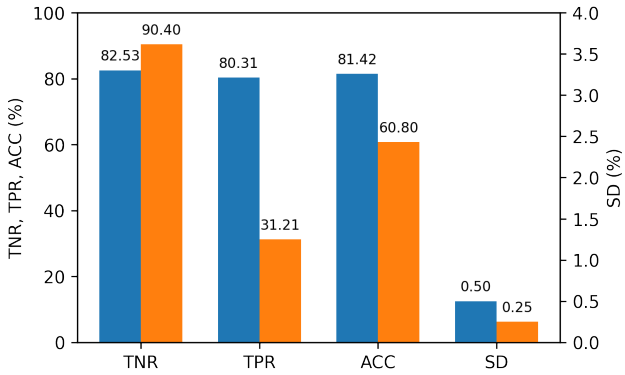


Figure 4.8: Performance for mixed QP scenarios.
(**Blue**: The proposed method; **Orange**: Ref. [16])

4.4.3 Performance Evaluation on Double Compression Detection for Each QP Scenario

To assess the performance of the double compression classifier for each QP scenario, we evaluated the test datasets containing the feature vectors of the double-compressed image pairs encoded in three different QP scenarios using a single classifier. Specifically, 300 single-compressed image pairs were randomly extracted from the entire dataset and combined with 300 double-compressed image pairs extracted for each of the three QP (quantization parameter) scenarios, creating test datasets of 600 image pairs for each QP scenario. The entire dataset of 5400 image pairs was divided into 4800 for training and 600 for validation, using 9-fold cross-validation. The best model was saved while varying the C-values, and its generalization performance was

evaluated on the three test datasets. To compare the performance of the proposed method and [16] clearly, we fixed the TNR at 90%, or as close to it as possible, and the TPR was calculated accordingly. This evaluation was repeated ten times, and the average performance was calculated. Figure 4.9–4.11 report the average performance for each QP scenario, respectively, showing that our proposed method outperformed Ref. [16] in all QP scenarios.

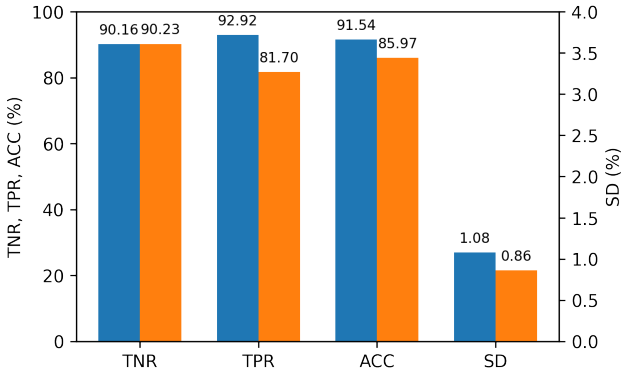


Figure 4.9: Performance for scenario ($QP_1 > QP_2$).
(**Blue**: The proposed method; **Orange**: Ref. [16])

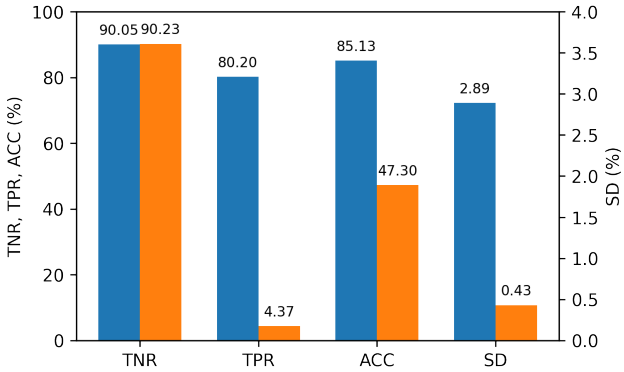


Figure 4.10: Performance for scenario ($QP_1 = QP_2$).
(**Blue**: The proposed method; **Orange**: Ref. [16])

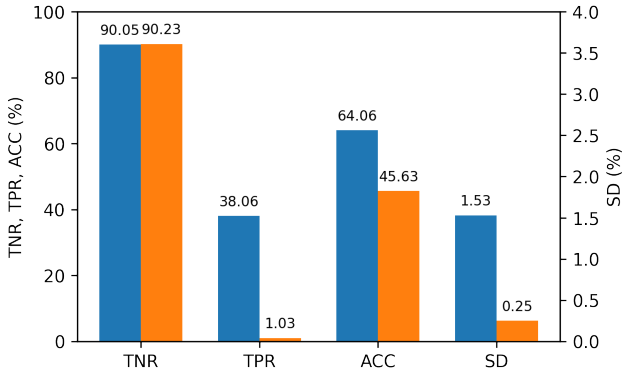


Figure 4.11: Performance for scenario ($QP_1 < QP_2$).
(**Blue**: The proposed method; **Orange**: Ref. [16])

4.4.4 Performance Evaluation on Double Compression Detection for Each QP Combination

We evaluated the performance of the double compression classifier for each QP combination of double-compressed images. Specifically, a test dataset was constructed by extracting 30 single-compressed image pairs and 30 double-compressed image pairs for each QP combination from the entire dataset. There were 106 QP combination patterns: 57 for scenario (a) when QP_1 exceeds QP_2, 10 for scenario (b) when QP_1 equals QP_2, and 39 for scenario (c) when QP_2 exceeds QP_1. To ensure fairness, the single-compressed images used for each test dataset were encoded using the final QP values applied in the double-compressed images. For instance, when evaluating double-compressed images with $QP_1 = 30$ and $QP_2 = 20$, the single-compressed images included in the test data were encoded with $QP = 20$. This approach ensures that the final QP value is consistent across both single- and double-compressed images, preventing any distinction based on QP values and maintaining fairness in the analysis.

The entire dataset of 5,400 image pairs was divided into 4800 for training and 600 for validation, using 9-fold cross-validation. The best model was saved while varying the C candidate values during the training phase, and its generalization performance was evaluated on each test dataset. This evaluation procedure was repeated 10 times to ensure robustness, and the average performance over these iterations was calculated.

We calculated the area under the curve (AUC) of the receiver operating characteristic (ROC) curve to compare the methods in [16] and our proposed method for each test dataset. AUC is a threshold-independent standard metric for evaluating a model's performance. Tables 4.1–4.3 compare the mean AUC of [16] and our new approach for each QP scenario. In scenario (a), the weakness in [16] was its inability to correctly identify double-compressed images with a QP difference of 5 or less. However, the new model compensated for this weakness. The model in [16] did not work well in scenarios (b) and (c). Our new approach achieved high performance in scenario (b) and also outperformed [16] in scenario (c).

		QP_2									
		5	10	16	20	24	27	32	39	42	45
QP_1	10	0.53	-	-	-	-	-	-	-		
	15	0.91	0.43	-	-	-	-	-	-	-	-
	20	0.94	0.84	0.41	-	-	-	-	-	-	-
	25	0.98	0.96	0.82	0.73	0.55	-	-	-	-	-
	30	1.0	1.0	0.99	0.99	1.0	0.91	-	-	-	-
	32	1.0	1.0	0.99	1.0	1.0	0.99	-	-	-	-
	35	1.0	1.0	1.0	1.0	1.0	1.0	0.95	-	-	-
	40	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.82	-	-
	45	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.93	-
	50	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.99
		QP_2									
		5	10	16	20	24	27	32	39	42	45
QP_1	10	0.98	-	-	-	-	-	-	-		
	15	1.0	0.9	-	-	-	-	-	-	-	-
	20	1.0	0.93	0.94	-	-	-	-	-	-	-
	25	1.0	0.96	0.98	0.98	0.95	-	-	-	-	-
	30	0.99	0.99	1.0	1.0	0.99	0.96	-	-	-	-
	32	0.99	0.98	1.0	1.0	1.0	0.99	-	-	-	-
	35	1.0	0.99	1.0	1.0	1.0	1.0	0.94	-	-	-
	40	1.0	0.99	1.0	1.0	1.0	1.0	1.0	0.93	-	-
	45	0.99	0.98	1.0	1.0	1.0	1.0	1.0	0.99	0.95	-
	50	0.99	0.98	0.99	1.0	1.0	1.0	1.0	1.0	0.99	0.96

Table 4.1: Average AUC for scenario ($QP_1 > QP_2$)
(Top: Ref. [16], Bottom: The proposed method)

		QP_2									
		5	10	16	20	24	27	32	39	42	45
QP_1	5	0.2	-	-	-	-	-	-	-		
	10	-	0.3	-	-	-	-	-	-	-	-
	16	-	-	0.29	-	-	-	-	-	-	-
	20	-	-	-	0.36	-	-	-	-	-	-
	24	-	-	-	-	0.43	-	-	-	-	-
	27	-	-	-	-	-	0.34	-	-	-	-
	32	-	-	-	-	-	-	0.42	-	-	-
	39	-	-	-	-	-	-	-	0.57	-	-
	42	-	-	-	-	-	-	-	-	0.59	-
	45	-	-	-	-	-	-	-	-	-	0.62

		QP_2									
		5	10	16	20	24	27	32	39	42	45
QP_1	5	0.99	-	-	-	-	-	-	-		
	10	-	0.93	-	-	-	-	-	-	-	-
	16	-	-	0.98	-	-	-	-	-	-	-
	20	-	-	-	0.98	-	-	-	-	-	-
	24	-	-	-	-	0.95	-	-	-	-	-
	27	-	-	-	-	-	0.92	-	-	-	-
	32	-	-	-	-	-	-	0.9	-	-	-
	39	-	-	-	-	-	-	-	0.93	-	-
	42	-	-	-	-	-	-	-	-	0.91	-
	45	-	-	-	-	-	-	-	-	-	0.89

Table 4.2: Average AUC for scenario ($QP_1 = QP_2$)
(Top: Ref. [16], Bottom: The proposed method)

		QP_2							
		16	20	24	27	32	39	42	45
QP_1	10	0.48	0.49	0.51	0.51	0.52	0.51	0.5	0.52
	15	0.36	0.49	0.49	0.51	0.53	0.52	0.5	0.53
	20	-	-	0.48	0.54	0.54	0.51	0.52	0.53
	25	-	-	-	0.54	0.59	0.55	0.53	0.53
	30	-	-	-	-	0.58	0.57	0.57	0.55
	32	-	-	-	-	-	0.62	0.58	0.56
	35	-	-	-	-	-	0.59	0.58	0.59
	40	-	-	-	-	-	-	0.69	0.62

		QP_2							
		16	20	24	27	32	39	42	45
QP_1	10	0.74	0.71	0.66	0.63	0.6	0.55	0.55	0.53
	15	0.98	0.84	0.81	0.76	0.64	0.62	0.56	0.54
	20	-	-	0.78	0.83	0.76	0.66	0.6	0.59
	25	-	-	-	0.88	0.8	0.74	0.65	0.61
	30	-	-	-	-	0.85	0.76	0.74	0.63
	32	-	-	-	-	-	0.82	0.73	0.65
	35	-	-	-	-	-	0.78	0.74	0.66
	40	-	-	-	-	-	-	0.83	0.67

Table 4.3: Average AUC for scenario ($QP_1 < QP_2$)
(Top: Ref. [16], Bottom: The proposed method)

4.4.5 Performance Evaluation on Double Compression Detection with Different Software

In this subsection, we assess the robustness of our proposed method when creating HEIF images using another image editing software, GIMP. Let S1 be the software used for single compression and S2 be the software used for double compression. We selected 40 TIF images from the entire dataset, converted them to PNG format, and used 30 of them to create 90 single-compressed images and the remaining 10 to create 90 double-compressed images for each of three different software combinations: (A) (S1, S2) = (GIMP, GIMP), (B) (S1, S2) = (libheif, GIMP), and (C) (S1, S2) = (GIMP, libheif). However, as GIMP uses quality factor (QF) instead of QP for encoding, experiments were conducted with new encoding parameters for each of (A), (B), and (C). PNG images were encoded with QP_2 or QF_2 to create single-compressed images, and PNG images were encoded with QP_1 or QF_1 and then recompressed with QP_2 or QF_2 to create double-compressed images. The encoding parameters in (A) were selected from {90, 85, 70} for QF_1 and {90, 85, 70} for QF_2. In (B), QF_2 was chosen from {90, 85, 70} and QP_1 from {2, 4, 12}. In (C), QP_2 was selected from {2, 4, 12} and QF_1 from {90, 85, 70}. Note that QF{90, 85, 70} is equivalent to QP{2, 4, 12}, and QP is used when comparing the encoding parameters in scenarios (B) and (C).

To evaluate the performance of the double compression classifier in different software combinations, we calculated AUC for each QP combination for each datum. A test dataset was constructed by extracting 10 single-compressed image pairs and 10 double-compressed image pairs for each software combination from the entire dataset. To ensure fairness, the single-compressed images used for each test dataset were only encoded with the final QP values used for the double-compressed images. We created a dataset consisting of 2400 single-compressed and 2400 double-compressed image pairs, excluding the images used to generate the test data, for 4800 image pairs. The dataset was split into 4320 for training and 480 for validation using a 10-fold cross-validation. The model with the best performance was saved while varying the C candidate values. Tables 4.4–4.6 display the average performance of 10 evaluations for software combinations (A), (B), and (C).

When the HEIF image was generated by (A), the double-compressed image was successfully detected if QF_1 was smaller or the same as QF_2. Using the same software ensures high consistency in the compression algorithm,

making the characteristics of double compression more distinct and, thus, easier to detect. On the other hand, the detection accuracy in (B) and (C) was better when QP_1 was larger than QP_2. Due to differences in compression algorithms between different software, the compression characteristics vary, making it more difficult to detect double compression, resulting in lower detection accuracy than (A). As a result of the observations, images encoded with libheif are more easily distinguishable in terms of MAE difference than those encoded with GIMP. Therefore, the accuracy of (C) is slightly higher than that of (B). Due to the different encoding algorithms of libheif and GIMP, the detection accuracy in (B) and (C) was better when QP_1 was larger than QP_2, while slightly lower than in (A) when they were equal.

		QF_2 (QP_2)		
QF_1 (QP_1)		90 (2)	85 (4)	70 (12)
	90 (2)	0.26	0.31	0.4
	85 (4)	0.29	0.3	0.39
	70 (12)	0.52	0.45	0.38

		QF_2 (QP_2)		
QF_1 (QP_1)		90 (2)	85 (4)	70 (12)
	90 (2)	0.91	0.59	0.6
	85 (4)	0.76	0.91	0.53
	70 (12)	0.96	0.97	0.96

Table 4.4: Average AUC for software combination (GIMP, GIMP)
(Left: Ref. [16], Right: The proposed method)

		QF_2 (QP_2)		
QP_1 (QF_1)		90 (2)	85 (4)	70 (12)
	2 (90)	0.24	0.33	0.42
	4 (85)	0.29	0.24	0.42
	12 (70)	0.75	0.67	0.35

		QF_2 (QP_2)		
QP_1 (QF_1)		90 (2)	85 (4)	70 (12)
	2 (90)	0.6	0.53	0.63
	4 (85)	0.81	0.54	0.67
	12 (70)	0.97	0.92	0.88

Table 4.5: Average AUC for software combination (libheif, GIMP)
(Left: Ref. [16], Right: The proposed method)

		QP_2 (QF_2)		
QF_1 (QP_1)		2 (90)	4 (85)	12 (70)
	90 (2)	0.18	0.22	0.36
	85 (4)	0.16	0.27	0.36
	70 (12)	0.33	0.42	0.33

		QP_2 (QF_2)		
QF_1 (QP_1)		2 (90)	4 (85)	12 (70)
	90 (2)	0.62	0.74	0.47
	85 (4)	0.68	0.87	0.56
	70 (12)	0.98	1.0	0.89

Table 4.6: Average AUC for software combination (GIMP, libheif) (Left: Ref. [16], Right: The proposed method)

4.5 Concluding remarks

In this chapter, we proposed two methods that detect double compression in HEIF images; in particular, a lightweight image classifier has been proposed, that by extracting and learning encoding information from the input image and its recompressed image is able to achieve an accuracy of 81% by clearly outperforming the first proposed model, which just reaches 60% of accuracy. To the best of our knowledge, our work is the first to address double compression detection in HEIF images. Our new approach addresses the limitations imposed by quantization history in double-compressed images. In future work, we will take into account the local image content to improve the performance of the detector. Indeed, our proposed method extracts encoding information from all regions in the image, including areas that do not contribute to double compression detection. Moreover, we will consider extending the analysis to identify forged areas in the image, for example, by running the double detection method in a block-wise fashion and detecting the presence of both double-compressed and single-compressed blocks within the same image.

Chapter 5

Double Compression Detection in HEVC Videos Containing B-frames

This chapter introduces a method for detecting double compression in HEVC videos containing B-frames, focusing on the approach and its effectiveness. First, we provide an overview of how re-encoding transforms original I-frames into P-frames or B-frames, introducing spatiotemporal inconsistencies. Our method detects double compression by analyzing changes in encoding information resulting from re-encoding. We then present a newly developed classifier designed to detect these artifacts, addressing the limitations of traditional methods and demonstrating improved accuracy and practical applicability in real-world scenarios.

5.1 Basic concept of RI-frames

This study assumes that the first encoding uses I-frames and P-frames while the second uses I-frames, P-frames, and B-frames. For example, modern devices such as Google Pixel (e.g., 3a, 7a) and iPhone (e.g., iPhone 15) typically use only I-frames and P-frames when recording live videos. This is likely because using B-frames requires a larger memory buffer and additional

computational resources. To perform real-time encoding at high resolution and frame rates, it is necessary to avoid an increase in computational load. This choice not only aligns with common real-world encoding practices but also simplifies the research process. This setup isolates the effects of B-frames introduced during re-encoding, making it easier to evaluate their influence on forensic detection. In this case, the original I-frame may be re-encoded as a P-frame or B-frame. Figure 5.1 illustrates how IP and IB frames are generated, where P' and B' indicate that they are influenced by the I-frame from the first encoding.

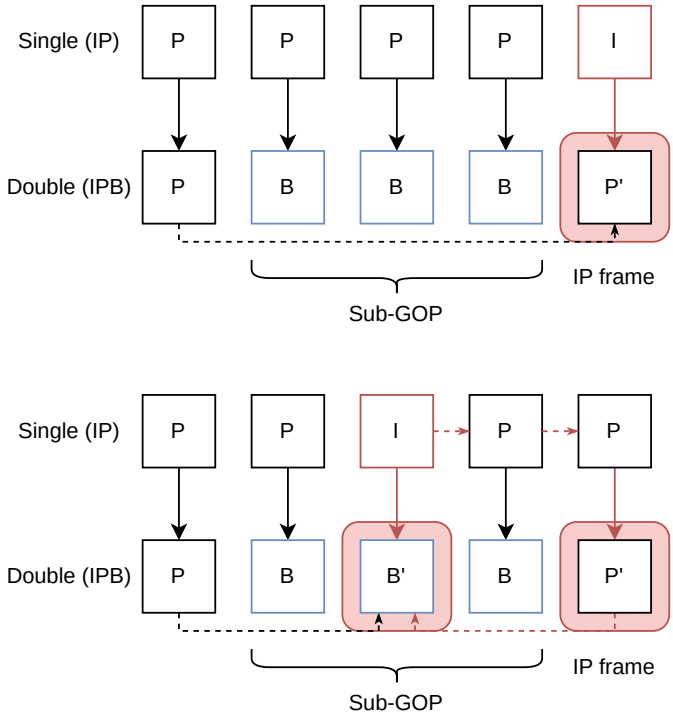


Figure 5.1: Basic concept of the RI-frames (Top: IP, Bottom: IB)

IP and IB frames retain many features from the original I-frame during the second encoding. As detailed in [47], the quantization noise introduced during the first compression expands the scope of intra-prediction, and

the selection of reference frames during re-encoding reduces inter-prediction accuracy. As a result, the proportion of intra-prediction increases, while inter-prediction decreases in IP and IB frames. In the case of IB frames, consecutive B-frames within a sub-GOP refer to the preceding P-frame and the subsequent P' frame for bidirectional prediction. However, in the first encoding, the P-frames following the I-frame were predicted by referencing the I-frame, and the elements of the I-frame propagated to the subsequent P-frames. The P' frame, immediately following the sub-GOP, is a re-encoded version of the P-frame containing elements of the I-frame, and IB frames tend to reference the motion information from this P' frame.

5.2 Proposed Method

This study proposes a method to detect double compression in HEVC videos by analyzing frame-level encoding information and using a bidirectional LSTM classifier. The framework includes dividing each frame into blocks, extracting encoding information, generating feature vectors, and feeding them into the LSTM model. Figure 5.2 visually represents the method's workflow, from input processing to classification.

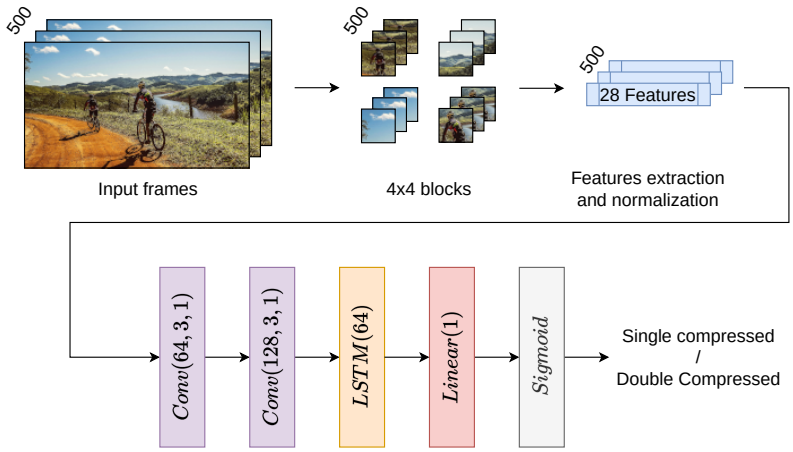


Figure 5.2: Flowchart of the proposed method

We performed feature extraction for each frame to identify abnormal behaviors in IP and IB frames in single- and double-compressed videos, as described in Section 5.1. The feature vector used in this analysis consists of elements shown in Table 5.1, represented as a 28-dimensional vector per frame, resulting in a final feature vector of $28 \times N$ dimensions, being N the number of frames analyzed in the video.

First, we examine the frame type of each frame, followed by analyzing CU size variations. For P-frames, CU types are classified as intra-coded (PI), skipped (PS), merged (PM), inter-coded with zero motion vectors (PVZ), and inter-coded with non-zero motion vectors (PVNZ). For IB frames, CUs are classified as intra-coded (BI), skipped (BS), and merged (BM), with inter-coded CUs further categorized as past (BP), bidirectional (BB), or future (BF). Subsequently, we extract luminance and chrominance prediction directions for intra-prediction.

Feature Category	**Feature Item**
Frame Type	I: 100, B: 010, P: 001
CU Size	64×64, 32×32, 16×16, 8×8
CU Type	Intra (PI/BI) Skip (PS/BS) Merge (PM/BM) Past MV (Zero) (PVZ/ -) Past MV (Non Zero) (PVNZ/BP) Bi-directional MV: BB Future MV: BF
Luminance Directions	0, 1, 9, 10, 11, 25, 26, 27
Chrominance Directions	0, 1, 9, 10, 34, 36

Table 5.1: Feature Vector Composition

Since the frame types (I, P, B) are categorical, it is converted into a one-hot encoding. Meanwhile, CU size, CU type, and prediction directions (Luminance and Chrominance) are represented as histograms, calculated as the total count of 4×4 CU blocks belonging to each category. All features are extracted for each frame and then normalized using Min-Max scaling to ensure equal contribution to the classifier. Further details on how these encoding elements are extracted and analyzed are provided in the next subsection.

In the following sections, we present a detailed statistical analysis of encoding characteristics, including IP and IB frame behaviors in both single- and double-compressed HEVC videos. Additionally, we provide an in-depth description of the LSTM model used for classification.

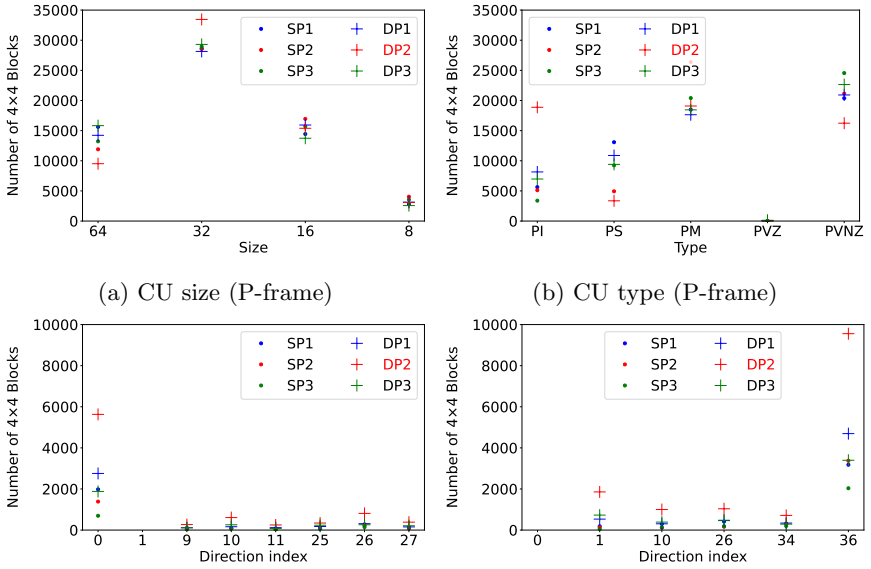
5.2.1 Analysis of Extracted Encoding Elements

Using HM 16.25 [22], each frame was divided into 4×4 blocks to extract encoding information on CU size, CU type, and luminance and chrominance prediction directions. Figure 5.3 and 5.4 compare the distribution of CU size, CU type, luminance intra-prediction directions, and Chrominance intra-prediction directions in three consecutive P-frames and three consecutive B-frames within a sub-GOP in single- and double-compressed videos. The IP and IB frames are positioned at the center of the three consecutive P-frames and the sub-GOP, respectively. Moreover, SP represents P-frames of single-compressed videos, DP represents P-frames of double-compressed videos, SB represents B-frames of single-compressed videos, and DB represents B-frames of double-compressed videos.

In this analysis, we constructed a dataset comprising 129 videos in total. Specifically, the dataset includes 43 original YUV videos, 43 videos processed with BLUR, and 43 videos processed with CLAHE [57]. The single-compressed videos were encoded using a GOP size of 25, a sub-GOP size of 3, and bitrates of 1000, 3000, and 5000 kbps. The initial encoding used only I and P frames for the double-compressed videos, with a GOP size of 12 and the same three bitrates. The subsequent compression employed the same parameters (GOP size, sub-GOP size, and bitrates) for the single-compressed videos. This setup resulted in 387 (3×129) single-compressed data points and 1161 (3×3×129) double-compressed data points for analysis.

In Figure 5.3, the top-left graph shows a decrease in 64×64 CUs and an increase in 32×32 CUs in double-compressed P-frames compared to single-compressed ones. The top-right graph shows an increase in intra-coded CUs (PI) and a decrease in skip-coded CUs (PS) and intra-coded CUs with non-zero motion vectors (PVNZ). These changes likely reflect the effect of increased intra-prediction, which results in higher luminance and chrominance values.

Figure 5.4 shows that, in double-compressed B-frames, larger CUs (64×64 and 32×32) increase while smaller CUs (16×16 and 8×8) decrease. After the IB frame, intra-coded CUs (BI) increase, while skip-coded (BS) and



(a) CU size (P-frame)

(b) CU type (P-frame)

(c) Luminance prediction directions (P-frame)

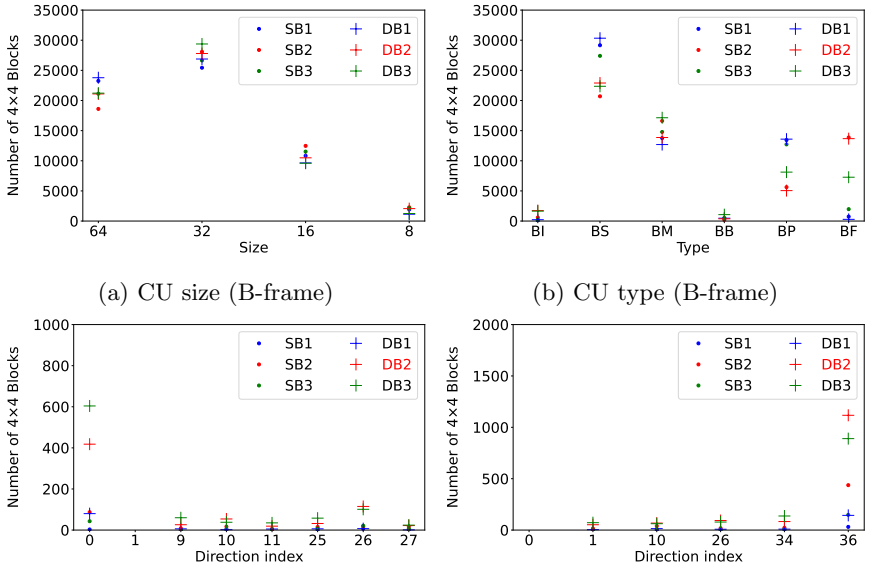
(d) Chrominance prediction directions (P-frame)

Figure 5.3: Encoding elements comparison between single- and double-compressed videos. IP frames ($DP2$) are depicted in red. The values represent averages across 387 single-compressed and 1161 double-compressed videos.

past motion vector inter-coded CUs (BP) decrease, and future motion vector inter-coded CUs (BF) increase. These trends indicate an increase in intra-components in re-encoded B-frames. Additionally, the P-frame following the sub-GOP exhibits characteristics similar to the IB frame, resulting in increased inter-prediction dependency, which leads to a decrease in BP and an increase in BF. Furthermore, increased intra-prediction leads to higher luminance and chrominance values.

5.2.2 Model Architecture

The feature vectors are fed into a bidirectional LSTM classifier, capturing forward and backward dependencies in sequential data. This architecture allows the LSTM to detect both spatial anomalies within and temporal anoma-



(a) CU size (B-frame)

(b) CU type (B-frame)

(c) Luminance prediction directions (B-frame)

(d) Chrominance prediction directions (B-frame)

Figure 5.4: Encoding elements comparison between single- and double-compressed videos. IB frames ($DB2$) are depicted in red. The values represent averages across 387 single-compressed and 1161 double-compressed videos.

lies between frames. Traditional neural networks, such as fully connected or convolutional networks, lack mechanisms to retain temporal information, making them less suited for sequential data modeling. In contrast, the bidirectional LSTM is well-suited for identifying compression-induced changes between frames, offering high robustness in detecting double compression. To ensure consistent scaling, Min-Max normalization was applied, adjusting each feature to a range of 0 to 1. Model performance was evaluated using cross-validation on the training set, and the model with the highest cross-validation accuracy was selected to evaluate the test data. The overall architecture of the proposed model, including layer configurations and parameter settings, is summarized in Table 5.2.

To implement our double compression detection, we designed a bidirectional LSTM classifier that combines CNN and LSTM layers for sequen-

tial analysis of video encoding data. First, the model utilizes two one-dimensional convolutional layers, where the input is processed to extract local features. The first convolutional layer (Conv1D-1) has 28 input channels and 64 output channels, with a kernel size of 3 and a stride of 1, while the second layer (Conv1D-2) has 64 input channels and 128 output channels, also with a kernel size of 3 and a stride of 1. Both convolutional layers are followed by a ReLU activation function, adding non-linearity and enhancing the model's capacity to capture complex feature patterns.

The extracted feature maps are then passed through a bidirectional LSTM (Bi-LSTM) composed of two layers designed to capture temporal dependencies in the encoding information. The LSTM has an input size of 128 and a hidden layer size of 64, allowing it to effectively analyze sequential frame data from both forward and backward directions.

Finally, the output from the LSTM is processed through a linear layer (Linear) that maps the 64-dimensional hidden state to a single output. A sigmoid activation function follows, restricting the output to the range [0, 1], which enables the model to perform binary classification with probabilistic output, identifying whether the input is double-compressed.

Normalized Feature Vectors: (500 frames, 28 dimensions)					
Layer	**Input Size**	**Output Size**	**Kernel Size**	**Stride**	**Activation**
Conv1D-1	(500, 28)	(500, 64)	3	1	ReLU
Conv1D-2	(500, 64)	(500, 128)	3	1	ReLU
Bi-LSTM	(500, 128)	(500, 64)	N/A	N/A	None
Linear	(500, 64)	(500, 1)	N/A	N/A	Sigmoid

Table 5.2: Network Layer Configuration

5.3 Experimental results

This section describes the experimental validation of the proposed model. We detail the dataset creation process and compare the accuracy of our double compression detector with the state-of-the-art methods. Bitrate and GOP size are crucial factors in determining video quality, so we constructed the dataset using various combinations of these parameters. Additionally, we analyzed how these two encoding parameters affect model performance and examined the impact of reducing the number of frames in the test data. For evaluation, single-compressed videos are labeled as negative, and double-

compressed videos are labeled as positive, ensuring no overlap in video content between the training and test datasets.

5.3.1 Dataset

We used 43 YUV videos[1] (720p, 1080p, 4K) [48]. Each video retained up to the first 500 frames, and the classifier was designed with a base resolution of 720p. However, because the classifier processes a 28-dimensional feature vector per frame, it is resolution-independent and can handle videos of any size. The scale of the feature vector remains consistent across resolutions due to normalization, enabling the processing of videos in various sizes without resizing or padding.

For data augmentation, we used FFmpeg[2] to extract frames from the YUV videos and applied CLAHE [57] and Blur processing using the albumentations library[3]. We set the probability parameter p=1.0, ensuring that blur is applied to every frame. The blur was implemented as a uniform box blur with a randomly selected square kernel size between 3 and 7 pixels, with larger kernels producing stronger blur effects. Each frame was processed sequentially with these augmentation methods and then recombined into YUV format using FFmpeg. This process resulted in three subsets of YUV videos for each original video content: the original (O), CLAHE-processed (C), and blur-processed (B) videos. These three subsets are derived from a single YUV video and are, therefore, grouped as a single dataset. This approach prevents content duplication by treating all subsets from the same source video as one dataset unit.

Table 5.3 summarizes the encoding parameters used for single and double compression. Single-compressed videos were encoded using x265 with bitrates B2 (1000, 3000, 5000 kbps), GOP sizes G2 (9, 25, 70), and sub-GOP sizes SG (3, 5, 7). For double compression, videos were first encoded

[1] **720p** ducks_take_off, FourPeople, in_to_tree, Johnny, KristenAndSara, mobcal, old_town_cross, park_joy, parkrun, shields, sintel_trailer, stockholm, vidyo1, vidyo3, vidyo4 **1080p** aspen, controlled_burn, crowd_run, dinner, factory, life, red_kayak, rush_field_cuts, rush_hour, snow_mnt, speed_bag, sunflower, touchdown_pass, tractor, west_wind_easy **4K** Netflix_Aerial, Netflix_BarScene, Netflix_DinnerScene, Netflix_Dancers, Netflix_DrivingPOV, Netflix_FoodMarket, Netflix_PierSeaside, Netflix_RitualDance, Netflix_RollerCoaster, Netflix_SquareAndTimelapse, Netflix_ToddlerFountain, Netflix_TunnelFlag, Netflix_WindAndNature

[2] https://www.ffmpeg.org/

[3] https://pypi.org/project/albumentations/

with only I- and P-frames at bitrates B1 (1000, 3000, 5000 kbps) and GOP sizes G1 (12, 30), then re-encoded with the same parameters (B2, G2, SG) as in single compression. HM 16.25 was used to extract the encoding information. For encoding with x265, closed GOP (–no-open-gop encoding flag) was used to ensure that each GOP referenced only its own frames, creating a uniform prediction structure. Additionally, B-frames were set to reference only I- or P-frames, disabling references to other B-frames (–no-b-pyramid encoding flag), simplifying prediction. Each dataset (OCB) contains 81 single-compressed videos (3×3×3×3) and 486 double-compressed videos (3×2×3×3×3). To ensure a fair evaluation, 81 double-compressed videos were randomly selected.

Parameters	single compression	Double compression
Encoder	x265	x265
Decoder	HM16.25	HM16.25
Resolution	1280x720	1280x720
YUV videos (O)	43	43
YUV videos (C)	43	43
YUV videos (B)	43	43
Dataset (O, C, B)	43	43
1st Bitrate	B2{1000, 3000, 5000}	B1{1000, 3000, 5000}
1st GOP	G2{9, 25, 70}	G1{12, 30}
2nd Bitrate	-	B2{1000, 3000, 5000}
2nd GOP	-	G2{9, 25, 70}
Sub-GOP	SG{3, 5, 7}	SG{3, 5, 7}
Videos per Dataset	81 (=3×3×3×3)	486 (=3×3×2×3×3×3)

Table 5.3: Encoding parameters for dataset creation

For all performance evaluations, 40 video datasets were randomly selected from the original 43 datasets. We applied 10-fold cross-validation, using 32 sets for training, 4 for validation, and 4 for testing.

5.3.2 Performance Evaluation on Double Compression Detection

The proposed model, implemented in PyTorch, was optimized using the Adam algorithm with momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. A

mini-batch size of 64 and an initial learning rate of 0.005 (decayed by $\gamma = 0.5$ every 6 epochs) were employed, up to a maximum of 50 epochs. Two bidirectional LSTM layers enhanced feature extraction, and early stopping was applied if the validation loss did not improve for 10 consecutive iterations.

We considered two well-known approaches to the double compression detection problem in the evaluation and reported the accuracy performances in Table 5.4. The first one, [47], was originally developed for H.264/AVC videos, where it utilizes macroblock (MB) types as input and analyzes MB behavior based on multiple rules. Since HEVC uses coding units (CU) instead, we modified the input to utilize CU data extracted from each frame. The second one proposed in [20], although designed for H.265 videos, extracts variations in CU size and CU type over three consecutive P frames but does not consider B frames.

The proposed method achieved an overall accuracy of 80.06%, outperforming other methods. Moreover, the proposed approach showed a high True Positive Rate (TPR), indicating its superior reliability in identifying double-compressed samples compared to existing methods. In contrast, the method by He et al. [20] reported the smallest accuracy, likely due to the limited number of IP frames available across the GOP size combinations used in our experimental setup. Additionally, the results reported by [47] demonstrate the approach's clear capability in identifying single-compressed videos, although the implemented rules proved insufficient for detecting double-compressed ones.

Method	**TNR**	**TPR**	**Acc**	**SD**
The proposed method	78.07	**82.05**	**80.06**	1.16
Vazquez et al. [47]	88.77	33.37	61.06	1.43
He et al. [20]	57.06	60.60	58.85	0.49

Table 5.4: Performance for double compression detection (%).
The evaluation metrics used are true positive rate (TPR), true negative rate (TNR), accuracy (ACC), and the standard deviation (SD).

5.3.3 Performance Evaluation On Double Compression Detection for each bitrate

To assess the impact of different bitrate combinations, we evaluated the classifier on a test dataset of double-compressed videos encoded with 9 distinct bitrate scenarios. Each scenario included 108 single- and double-compressed videos, with the second bitrate consistent across all samples. Figure 5.5 displays test accuracy for each bitrate scenario, with the X-axis representing bitrate scenarios and the Y-axis showing accuracy.

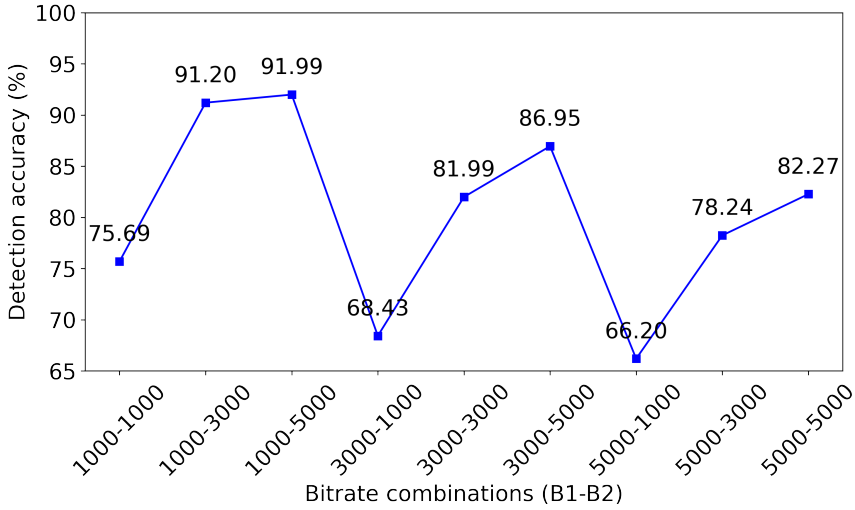


Figure 5.5: Detection accuracy for each bitrate combination

When the first bitrate ($B1$) is smaller than the second ($B2$), the first compression is stronger, leaving residual anomalies in the RI-frames. A larger bitrate difference ($B1$-$B2$) improved the accuracy of double-compression detection. On the other hand, when ($B1$) is larger than ($B2$), the second compression is stronger, overwriting the anomalies in the RI-frames, and a larger bitrate difference ($B1$-$B2$) tended to reduce detection accuracy. When ($B1$) and ($B2$) are the same, lower compression strength in ($B2$) resulted in higher accuracy. These results are generally consistent with the findings discussed in [47], where similar trends were observed in double-compression detection methods.

5.3.4 Performance Evaluation On Double Compression Detection for each GOP size

We evaluated the classifier's performance across six different GOP size scenarios to analyze the impact of GOP configuration on double-compression detection. The test dataset consisted of subsets for each GOP size containing 108 single- and double-compressed videos, with a consistent second GOP size. Figure 5.6 illustrates the results, with the X-axis showing GOP scenarios and the Y-axis indicating test accuracy.

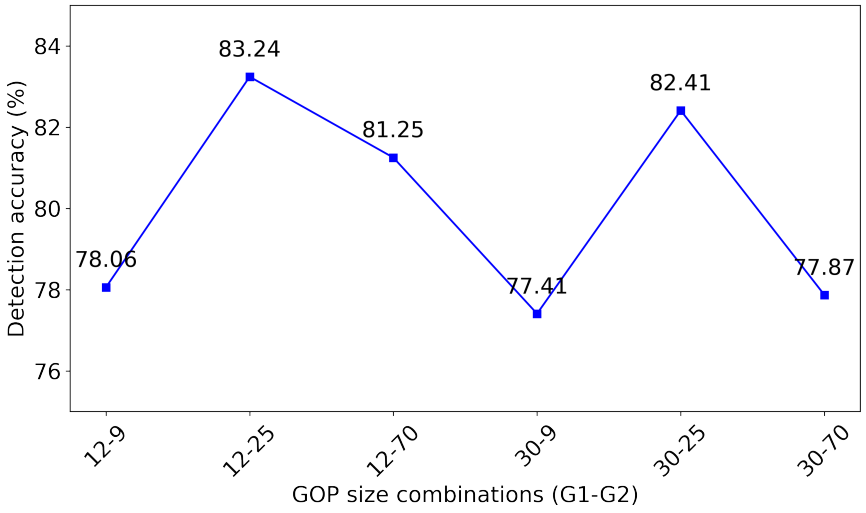


Figure 5.6: Detection accuracy for each GOP size combination

According to the Figure, the scenario where the first GOP size ($G1$) is 12 and the second GOP size ($G2$) is 25 achieved the highest accuracy. In contrast, the scenario with $G1$=30 and $G2$=9 showed the lowest accuracy. The variation in accuracy is related to the consistent extraction of IP and IB frames, regardless of sub-GOP size.

Table 5.5 shows the number of IP and IB frames extracted from 500 video frames for each sub-GOP across the six GOP scenarios, along with their sum and standard deviation. In the high-accuracy scenarios of ($G1$, $G2$) = (12,25) and ($G1$, $G2$) = (30,25), the standard deviations of the extracted IP and IB frames are smaller, indicating stable feature vector extraction regardless of sub-GOP size.

(G1,G2)	SG	IP	IB	(G1,G2)	SG	IP	IB
(12,9)	3	0	28	(30,9)	3	0	11
	5	15	13		5	5	6
	7	0	28		7	0	11
SUM		15	69	SUM		5	28
SD		8.66	8.66	SD		2.89	2.89
(12,25)	3	8	32	(30,25)	3	3	10
	5	8	32		5	0	13
	7	5	35		7	0	13
SUM		21	99	SUM		3	36
SD		1.73	1.73	SD		1.73	1.73
(12,70)	3	22	18	(30,70)	3	7	7
	5	16	24		5	6	8
	7	10	30		7	2	12
SUM		48	72	SUM		15	27
SD		6.00	6.00	SD		2.65	2.65

Table 5.5: The number of RI-frames and the Standard Deviation for Each GOP Size and Sub-GOP.

5.3.5 Performance Evaluation On Double Compression Detection for each frame number

A test dataset was prepared with double-compressed videos encoded in five frame sizes (500, 400, 300, 200, and 100 frames) to explore how varying frame counts affect classifier performance. Each subset included 324 single- and double-compressed videos. Table 5.6 shows that test accuracy slightly improved with increased frame size. The classifier achieved an accuracy of 79.72% with 500 frames, compared to 76.56% with 100 frames, suggesting minimal impact of frame size on overall performance.

This suggests that our classifier is robust to variations in frame size, and its performance does not significantly degrade even when the frame size is greatly reduced. Therefore, while having more frames is ideal, the classifier still maintains adequate performance with fewer frames, making it applicable in scenarios where the number of frames is limited.

Frame size	TNR	TPR	Acc
500	78.12	81.33	79.72
400	76.26	79.82	78.04
300	76.48	79.32	77.90
200	77.81	79.48	78.64
100	72.75	80.37	76.56

Table 5.6: Performance for double compression detection for each frame size (%)

5.4 Concluding remarks

This study proposes a novel classifier to detect double compression in HEVC videos by extracting and analyzing encoding information from re-compressed videos that utilize B-frames. By focusing on these re-encoded videos, the classifier effectively captures the subtle changes caused by B-frame-based recompression, significantly improving detection accuracy. The proposed method achieved an accuracy of 80%, notably outperforming existing models. This is the first study addressing the challenge of detecting double compression in HEVC videos containing B-frames, filling a crucial gap in the current literature. In future research, we plan to extend our investigation to more realistic scenarios, such as detecting double compression in videos where both the first and second compressions include B-frames.

Chapter 6

Conclusions

This chapter summarizes the contribution of the thesis and discusses future research.

6.1 Summary of contribution

In this thesis, we proposed new methods for detecting double compression in multimedia encoded with the High Efficiency Video Coding (HEVC) standard. The main contributions of this research are summarized as follows:

- **Research on Double Compression Detection in HEIF Images**: Previous studies mainly focused on detecting double compression in JPEG images, with no effective methods for HEIF images. This study proposes a new detection method based on compression idempotency, which extracts encoding information from the input image and its re-compressed version without relying on quantization history. This method enables high-accuracy detection of double compression across various compression settings, achieving approximately 20% higher detection accuracy than traditional methods.
- **Research on Double Compression Detection in HEVC Videos Containing B-Frames**: Although research on double compression in HEVC videos has gained attention, studies on videos containing B-frames remain limited due to the spatial and temporal complexity that B-frame encoding introduces. To bridge this gap, we propose a new classifier that effectively captures the spatial and temporal encoding

feature changes between single-compressed and B-frame-based recompressed videos by extracting and analyzing encoding information from re-encoded videos. Our model achieves approximately 20% higher detection accuracy than previous methods and has proven effective in real-world scenarios.

6.2 Directions for future work

- **Research on Double Compression Detection in HEIF Images:** To further improve detection accuracy, we plan to incorporate localized image content into the analysis to enhance the classifier's sensitivity to areas indicative of double compression. Currently, encoding information is uniformly extracted from the entire image, including regions that may not contain compression artifacts. By focusing on specific areas, we expect to improve detection performance. We also aim to develop a block-wise approach to detect single-compressed and double-compressed blocks within an image, enabling more detailed detection of tampered regions.

- **Research on Double Compression Detection in HEVC Videos Containing B-Frames:** In future research, we aim to extend the classifier's applicability to more realistic scenarios, specifically to detect double compression in HEVC videos where both the first and second compressions include B-frames. This extension will further enhance forensic applicability by identifying encoding inconsistencies even in cases where both compressions contain B-frames. Furthermore, we will consider robustness across various video qualities and encoding settings, ensuring adaptability to changes in bitrate and other compression parameters.

Appendix A

Publications

This research activity has led to several publications in international journals and conferences. These are summarized below. [1]

International Conference

1. **Y. Furushita**, M. Fontani, S. Bianchi, A. Piva, and G. Ramponi. "Double Compression Detection of HEIF Images Using Coding Ghosts", in *International Congress on Information and Communication Technology*, London, 2024, *Springer*, pp. 305–315, [DOI:10.1007/978-981-97-3562-4_25]

International Journal

1. **Y. Furushita**, M. Fontani, S. Bianchi, A. Piva, and G. Ramponi. "A Lightweight Double Compression Detector for HEIF Images Based on Encoding Information", *Sensors*, vol. 24, no. 16, p. 5103, 2024, *This article belongs to the Special Issue Image/Video Coding and Processing Techniques for Intelligent Sensor Nodes: 2nd Edition*, [DOI:10.3390/s24165103]

Under Review

1. **Y. Furushita**, D. Baracchi, M. Fontani, D. Shullani, and A. Piva. "Detection of Double Compression in HEVC Videos Containing B-frames", *Journal of Imaging*, 2025.

[1] The author's bibliometric indices are the following: H-index = 1, the total number of citations = 3 (source: Google Scholar on Feb 26, 2025).

Bibliography

[1] I. Amerini, R. Becarelli, R. Caldelli, and A. Del Mastio, "Splicing forgeries localization through the use of first digit features," in *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2014, pp. 143–148.

[2] K. Bahrami, A. C. Kot, L. Li, and H. Li, "Blurred image splicing localization by exposing blur type inconsistency," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 5, pp. 999–1009, 2015.

[3] D. Baracchi, M. Iuliani, A. G. Nencini, and A. Piva, "Facing image source attribution on iphone x," in *Digital Forensics and Watermarking: 19th International Workshop, IWDW 2020, Melbourne, VIC, Australia, November 25–27, 2020, Revised Selected Papers 19*. Springer, 2021, pp. 196–207.

[4] M. Barni, L. Bondi, N. Bonettini, P. Bestagini, A. Costanzo, M. Maggini, B. Tondi, and S. Tubaro, "Aligned and non-aligned double jpeg detection using convolutional neural networks," *Journal of Visual Communication and Image Representation*, vol. 49, pp. 153–163, 2017.

[5] P. Bestagini, S. Milani, M. Tagliasacchi, and S. Tubaro, "Video codec identification extending the idempotency property," in *European Workshop on Visual Information Processing (EUVIP)*. IEEE, 2013, pp. 220–225.

[6] T. Bianchi, A. De Rosa, M. Fontani, G. Rocciolo, and A. Piva, "Detection and classification of double compressed mp3 audio tracks," in *Proceedings of the first ACM workshop on Information hiding and multimedia security*, 2013, pp. 159–164.

[7] T. Bianchi and A. Piva, "Detection of nonaligned double jpeg compression based on integer periodicity maps," *IEEE transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 842–848, 2011.

[8] ——, "Image forgery localization via block-grained analysis of jpeg artifacts," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 1003–1017, 2012.

[9] A. Costanzo and M. Barni, "Detection of double avc/hevc encoding," in *2016 24th European Signal Processing Conference (EUSIPCO)*. IEEE, 2016, pp. 2245–2249.

[10] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato, "Raise: A raw images dataset for digital image forensics," in *6th ACM multimedia systems conference*, 2015, pp. 219–224.

[11] A. A. Elrowayati, M. F. L. Abdullah, A. Abd Manaf, and A. S. Alfagi, "Tampering detection of double-compression with the same quantization parameter in hevc video streams," in *2017 7th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*. IEEE, 2017, pp. 174–179.

[12] H. Farid, "Exposing digital forgeries from jpeg ghosts," *IEEE transactions on information forensics and security*, vol. 4, no. 1, pp. 154–160, 2009.

[13] J. Fridrich, D. Soukal, J. Lukas *et al.*, "Detection of copy-move forgery in digital images," in *Proceedings of digital forensic research workshop*, vol. 3, no. 2. Cleveland, OH, 2003, pp. 652–63.

[14] D. Fu, Y. Q. Shi, and W. Su, "A generalized benford's law for jpeg coefficients and its applications in image forensics," in *Security, Steganography, and Watermarking of Multimedia Contents IX*, vol. 6505. SPIE, 2007, pp. 574–584.

[15] Y. Furushita, M. Fontani, S. Bianchi, A. Piva, and G. Ramponi, "A lightweight double compression detector for heif images based on encoding information," *Sensors*, vol. 24, no. 16, p. 5103, 2024.

[16] Y. Furushita, M. Fontani, M. Bressan, S. Bianchi, A. Piva, and G. Ramponi, "Double compression detection of heif images using coding ghosts," in *International Congress on Information and Communication Technology*. Springer, 2024, pp. 305–315.

[17] A. C. Gallagher and T. Chen, "Image authentication by detecting traces of demosaicing," in *2008 IEEE computer society conference on computer vision and pattern recognition workshops*. IEEE, 2008, pp. 1–8.

[18] P. He, X. Jiang, T. Sun, and S. Wang, "Double compression detection based on local motion vector field analysis in static-background videos," *Journal of Visual Communication and Image Representation*, vol. 35, pp. 55–66, 2016.

[19] P. He, X. Jiang, T. Sun, S. Wang, B. Li, and Y. Dong, "Frame-wise detection of relocated i-frames in double compressed h. 264 videos based on convolutional neural network," *Journal of Visual Communication and Image Representation*, vol. 48, pp. 149–158, 2017.

[20] P. He, H. Li, H. Wang, S. Wang, X. Jiang, and R. Zhang, "Frame-wise detection of double hevc compression by learning deep spatio-temporal represen-

tations in compression domain," *IEEE Transactions on Multimedia*, vol. 23, pp. 3179–3192, 2020.

[21] "heic2hevc," https://github.com/yohhoy/heic2hevc.

[22] "HM software," https://hevc.hhi.fraunhofer.de.

[23] J. H. Hong, Y. Yang, and B. T. Oh, "Detection of frame deletion in hevc-coded video in the compressed domain," *Digital Investigation*, vol. 30, pp. 23–31, 2019.

[24] M. Huang, R. Wang, J. Xu, D. Xu, and Q. Li, "Detection of double compression for hevc videos based on the co-occurrence matrix of dct coefficients," in *International Workshop on Digital Watermarking*. Springer, 2015, pp. 61–71.

[25] C. Iakovidou, M. Zampoglou, S. Papadopoulos, and Y. Kompatsiaris, "Content-aware detection of jpeg grid inconsistencies for intuitive image forensics," *Journal of Visual Communication and Image Representation*, vol. 54, pp. 155–170, 2018.

[26] R.-S. Jia, Z.-H. Li, Z.-Z. Zhang, and D.-D. Li, "Double hevc compression detection with the same qps based on the pu numbers," in *ITM Web of Conferences*, vol. 7. EDP Sciences, 2016, p. 02010.

[27] X. Jiang, P. He, T. Sun, and R. Wang, "Detection of double compressed hevc videos using gop-based pu type statistics," *IEEE access*, vol. 7, pp. 95 364–95 375, 2019.

[28] X. Jiang, Q. Xu, T. Sun, B. Li, and P. He, "Detection of hevc double compression with the same coding parameters based on analysis of intra coding quality degradation process," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 250–263, 2019.

[29] M. K. Johnson and H. Farid, "Exposing digital forgeries through chromatic aberration," in *Proceedings of the 8th workshop on Multimedia and security*, 2006, pp. 48–55.

[30] M. Kirchner, "Fast and reliable resampling detection by spectral analysis of fixed linear predictor residue," in *Proceedings of the 10th ACM workshop on Multimedia and security*, 2008, pp. 11–20.

[31] B. Li, Y. Q. Shi, and J. Huang, "Detecting doubly compressed jpeg images by using mode based first digit features," in *2008 IEEE 10th Workshop on Multimedia Signal Processing*. IEEE, 2008, pp. 730–735.

[32] Q. Li, R. Wang, and D. Xu, "Detection of double compression in hevc videos based on tu size and quantised dct coefficients," *IET Information Security*, vol. 13, no. 1, pp. 1–6, 2019.

[33] W. Li, Y. Yuan, and N. Yu, "Passive detection of doctored jpeg image via block artifact grid extraction," *Signal Processing*, vol. 89, no. 9, pp. 1821–1829, 2009.

[34] Z.-H. Li, R.-s. Jia, Z.-Z. Zhang, X.-Y. Liang, and J.-W. Wang, "Double hevc compression detection with different bitrates based on co-occurrence matrix of pu types and dct coefficients," in *ITM Web of Conferences*, vol. 12. EDP Sciences, 2017, p. 01020.

[35] X. Liang, Z. Li, Y. Yang, Z. Zhang, and Y. Zhang, "Detection of double compression for hevc videos with fake bitrate," *IEEE Access*, vol. 6, pp. 53 243–53 253, 2018.

[36] "libheif," https://github.com/strukturag/libheif.

[37] Z. Lin, J. He, X. Tang, and C.-K. Tang, "Fast, automatic and fine-grained tampered jpeg image detection via dct coefficient analysis," *Pattern Recognition*, vol. 42, no. 11, pp. 2492–2501, 2009.

[38] J. Lukáš and J. Fridrich, "Estimation of primary quantization matrix in double compressed jpeg images," in *Proc. Digital forensic research workshop*, 2003, pp. 5–8.

[39] J. Lukáš, J. Fridrich, and M. Goljan, "Detecting digital image forgeries using sensor pattern noise," in *Security, steganography, and watermarking of multimedia contents VIII*, vol. 6072. SPIE, 2006, pp. 362–372.

[40] W. Luo, Z. Qu, J. Huang, and G. Qiu, "A novel method for detecting cropped and recompressed image block," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, vol. 2. IEEE, 2007, pp. II–217.

[41] S. McKeown and G. Russell, "Forensic considerations for the high efficiency image file format (heif)," in *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*. IEEE, 2020, pp. 1–8.

[42] A. C. Popescu and H. Farid, "Exposing digital forgeries in color filter array interpolated images," *IEEE Transactions on Signal Processing*, vol. 53, no. 10, pp. 3948–3959, 2005.

[43] Z. Qu, W. Luo, and J. Huang, "A convolutive mixing model for shifted double jpeg compression with application to passive image authentication," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2008, pp. 1661–1664.

[44] M. C. Stamm, W. S. Lin, and K. R. Liu, "Temporal forensics and anti-forensics for motion compensated video," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 4, pp. 1315–1329, 2012.

[45] "TIFF images used for experiment," https://git.lesc.dinfo.unifi.it/yoshihisa/TIFF_image.

[46] D. Vazquez-Padin, M. Fontani, T. Bianchi, P. Comesaña, A. Piva, and M. Barni, "Detection of video double encoding with gop size estimation," in *2012 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2012, pp. 151–156.

[47] D. Vázquez-Padín, M. Fontani, D. Shullani, F. Pérez-González, A. Piva, and M. Barni, "Video integrity verification and gop size estimation via generalized variation of prediction footprint," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1815–1830, 2019.

[48] "video data," https://media.xiph.org/video/derf/.

[49] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[50] W. Wang and H. Farid, "Exposing digital forgeries in video by detecting double mpeg compression," in *Proceedings of the 8th workshop on Multimedia and security*, 2006, pp. 37–47.

[51] "x265," https://www.videolan.org/developers/x265.html.

[52] Q. Xu, X. Jiang, T. Sun, and A. C. Kot, "Detection of hevc double compression with non-aligned gop structures via inter-frame quality degradation analysis," *Neurocomputing*, vol. 452, pp. 99–113, 2021.

[53] Q. Xu, T. Sun, X. Jiang, and Y. Dong, "Hevc double compression detection based on sn-pupm feature," in *Digital Forensics and Watermarking: 16th International Workshop, IWDW 2017, Magdeburg, Germany, August 23-25, 2017, Proceedings 16*. Springer, 2017, pp. 3–17.

[54] S. Ye, Q. Sun, and E.-C. Chang, "Detecting digital image forgeries by measuring inconsistencies of blocking artifact," in *2007 IEEE International Conference on Multimedia and Expo*. Ieee, 2007, pp. 12–15.

[55] L. Yu, Y. Yang, Z. Li, Z. Zhang, and G. Cao, "Hevc double compression detection under different bitrates based on tu partition type," *EURASIP Journal on Image and Video Processing*, vol. 2019, no. 1, pp. 1–12, 2019.

[56] Y. Yu, H. Yao, R. Ni, and Y. Zhao, "Detection of fake high definition for hevc videos based on prediction mode feature," *Signal Processing*, vol. 166, p. 107269, 2020.

[57] K. Zuiderveld, "Contrast limited adaptive histogram equalization," in *Graphics gems IV*, 1994, pp. 474–485.