Multimedia Tools and Applications
https://doi.org/10.1007/s11042-024-18666-2

iHELP: a model for instant learning of video coding in VR/AR real-time applications

Yousef O. Sharrab[1,2] · Mohammad A. Alsmirat[3,4] · Mohammad Ali H. Eljinini[1] · Nabil J. Sarhan[2]

Received: 8 September 2023 / Revised: 7 January 2024 / Accepted: 15 February 2024

Abstract
Virtual and augmented reality (VR/AR), teleoperation, and telepresence technologies heavily depend on video streaming and playback to enable immersive user experiences. However, the substantial bandwidth requirements and file sizes associated with VR/AR and 360-degree video content present significant challenges for efficient transmission and storage. Modern video coding standards, including HEVC, AV1, VP9, VVC, and EVC, have been designed to address these issues by enhancing coding efficiency while maintaining video quality on par with the H.264 standard. Nonetheless, the adaptive block structures inherent to these video coding standards introduce increased computational complexity, necessitating additional intra-prediction modes. The integration of AI in video coding has the potential to substantially improve video compression efficiency, reduce file sizes, and enhance video quality, making it a crucial area of research and development within the video coding domain. As AI systems can execute a wide array of tasks and adapt to new challenges, their incorporation into video coding may result in even more advanced compression techniques and innovative solutions to meet the ever-evolving demands of the industry. In this study, we introduce a state-of-the-art adaptive instant learning-based model, named iHELP, developed to address the computational complexity arising from encoders' adaptive block structures. The iHELP model achieves outstanding coding efficiency and quality while considerably improving encoding speed. iHELP model has been tested on HEVC, but it applies to other encoders with similar adaptive block structures. iHELP model employs entropy-based block similarity to predict the splitting decision of the LCU, determining whether to divide the block based on the correlation between the block content and previously adjacent encoded blocks in both spatial and temporal dimensions. Our methodology has been rigorously evaluated using the HEVC standard's common test conditions, and the results indicate that iHELP serves as an effective solution for efficient video coding in bandwidth-constrained situations, making it suitable for real-time video applications. The proposed method achieves an 80% reduction in encoding time while maintaining comparable PSNR performance relative to the RDO approach. The exceptional potential of the iHELP model calls for further exploration, as no other existing methods have demonstrated such a high level of performance.

✉ Mohammad A. Alsmirat
malsmirat@sharjah.ac.ae; masmirat@just.edu.jo

Extended author information available on the last page of the article

Published online: 05 March 2024

Keywords Artificial intelligence in video coding · High efficiency video coding (HEVC) · Instant high efficiency learning-based prediction (iHELP) · Fast HEVC encoding · Entropy-based LCU partitioning · Adaptive instant learning-based approach · Entropy-based block similarity · Entropy-based block similarity

Abbreviations

Throughout this research paper, we will be using abbreviations to refer to terms that are frequently mentioned. To help you better understand the material presented. In Table 1, we have provided a list of these abbreviations along with their meanings.

1 Introduction

The rapid advancement in VR, AR, teleoperation, and telepresence technologies has sparked a revolution in video communication and interaction. These applications, known for their immersive qualities, hinge on the efficient transmission of high-quality video. However, traditional video coding methods struggle with the immense file sizes and bandwidth demands of modern, high-resolution content [1]. Artificial Intelligence (AI) stands as a transformative force in video coding, heralding significant enhancements in compression efficiency and latency reduction, crucial for delivering detailed experiences over constrained bandwidths [2]. This paradigm shift is vital for seamless VR and AR streaming, enhancing teleoperation, and enriching telepresence experiences.

Optimizing AI for high-resolution video encoding brings challenges, particularly in balancing encoding speed and efficiency and mitigating error propagation issues [3–5]. Our novel approach, iHELP, addresses these challenges by integrating AI into the video encoding process, particularly focusing on the adaptive quad-tree coding in HEVC [6]. iHELP employs machine learning to predict the optimal block size in HEVC, analyzing the content correlation of the current block with its spatial and temporal neighbors. This method

Table 1 List of abbreviations and their meanings

Abbreviation	Meaning
VR/AR	Virtual and Augmented Reality
HEVC	High-Efficiency Video Coding
AV1	AOMedia Video 1
VP9	A video coding format developed by Google
VVC	Versatile Video Coding
EVC	Essential Video Coding
AI	Artificial Intelligence
iHELP	Instant High-Efficiency Learning-based Prediction
LCU	Largest Coding Unit
PSNR	Peak Signal-to-Noise Ratio
RDO	Rate-Distortion Optimization
ABT	Adaptive Binary Tree
QTBT	Quadtree-plus-Binary-tree

Table 1 continued

Abbreviation	Meaning
FMO	Flexible Macroblock Order
SVM	Support Vector Machines
MSE	Mean Square Error
CU	Coding Unit
PU	Prediction Unit
TU	Transform Unit
TP	Termination Probability
TPA	Termination Probability Average
ESE	Encoding Speed Enhancement
BD-PSNR	Bjontegaard Delta-Peak Signal-to-Noise Ratio
BD-Rate	Bjontegaard Delta-Rate
ETR	Encoding Time Reduction
QP	Quantization Parameters
HM	HEVC Test Model
GOP	Group of Pictures

significantly reduces computational complexity, achieving an 80% reduction in encoding time while preserving video quality, as verified through extensive testing against common HEVC test conditions [7–9]. Moreover, iHELP's entropy-based content conditions prevent error propagation, ensuring accurate predictions across various video contents. This feature not only accelerates iHELP's processing speed but also solidifies its reliability and efficiency for diverse applications [10]. iHELP's revolutionary potential extends beyond immersive experiences in VR and AR; it is poised to redefine the streaming and interaction of high-resolution video content. By significantly lowering bandwidth requirements and latency, iHELP opens new avenues for the broader adoption and accessibility of immersive technologies [11]. This paper explores the nuances of iHELP, its innovative design, and its performance impact. We demonstrate iHELP's superiority over existing methods in speed and efficiency, setting a new standard in AI-powered video coding. Our journey through this paper will unveil the potential of iHELP in reshaping the future of video communication technology and its applications in immersive experiences.

Paper organization

The paper is structured as follows: Section 2 provides background information and a review of related work. Section 3 discusses the novelty of the iHELP model. Section 4 details our proposed algorithm, iHELP. Section 5 elaborates on the performance evaluation methodology used. In Section 6, we present and analyze our research outcomes. Finally, Section 7 concludes the paper, summarizing our findings and the implications of iHELP in video coding.

2 Background

Video communication has become an essential tool in both personal and professional contexts, providing an efficient and effective way to connect with others, regardless of their

location. It has revolutionized the way we work, learn, and communicate and will continue to play a significant role in the future. Video communication is used in a wide range of settings, including business meetings, remote work, telemedicine, education, and personal communication. In business, video communication is used for virtual meetings, conferences, and interviews, allowing participants to connect from anywhere in the world. Remote work has also become more prevalent in recent years, and video communication has played a significant role in enabling remote teams to collaborate effectively [12, 13].

2.1 Efficiency video coding in live distance learning application

Video encoding plays a critical role in live distance learning, which involves the real-time transmission of video and audio content between instructors and learners who are physically located in different places. Live distance learning platforms require high-quality video streaming and playback to provide an immersive and engaging experience for learners. However, the high bandwidth requirements of video content present significant challenges for efficient transmission and storage, especially in bandwidth-limited scenarios.

To overcome these challenges, advanced video encoding standards such as HEVC, AV1, and VP9 are used to compress video content to a smaller size without sacrificing image quality. This allows for more efficient transmission and storage of video content, enabling live distance learning platforms to deliver high-quality video content to learners without overwhelming network bandwidth.

Moreover, live distance learning platforms require real-time video encoding for seamless interaction between instructors and learners. The encoding speed and efficiency of the video encoder are critical in ensuring a smooth and uninterrupted learning experience. Techniques like the iHELP Model, which has been primarily tested on HEVC but is also applicable to other encoders with similar adaptive block structures such as AV1 and VP9, can significantly enhance encoding speed while maintaining video quality. By predicting the optimal size of the block based on the content correlation of the current block with its spatial and temporal neighbors, iHELP achieves high coding efficiency and quality, making it ideal for real-time video applications like live distance learning [14].

2.2 Video encoding

Video encoding is converting raw video data into a compressed format that can be easily stored, transmitted, and played back on various devices. The primary reason for video encoding is to reduce video file size while maintaining the original video's quality. This is particularly important for high-definition (HD) and ultra-high definition (UHD) video, which has much larger file sizes than standard-definition video. HD video has a resolution of 1280x720 pixels or 1920x1080 pixels, while UHD video has a resolution of 3840x2160 pixels or 7680x4320 pixels. The larger resolution of HD and UHD video means that they have more detail and require more storage space. In order to transmit or store such large amounts of data, compression is necessary.

Video encoding for HD and UHD video typically uses advanced video compression standards such as H.264, H.265, and VP9. These standards use various compression techniques such as motion estimation, intra-prediction, and inter-prediction to reduce the size of the video files. Motion estimation involves analyzing the movement of objects in the video and only transmitting the differences between frames. Intra-prediction uses the information from neighboring pixels within the same frame to predict the value of a pixel, and inter-prediction

uses information from neighboring frames to predict the value of a pixel. H.264 is a widely used video compression standard for HD video, and H.265 (also known as HEVC) is a newer standard that provides even better compression for UHD video. VP9 is another compression standard developed by Google that is used for streaming video on platforms such as YouTube. These standards have become essential for delivering high-quality video content over the internet, as they allow for faster streaming and reduced buffering times [15].

In addition to compression standards, video encoding also involves setting various parameters, such as bit rate, frame rate, and color space. These parameters can affect the quality of the video and the amount of storage space required. Choosing the right parameters for video encoding is important to maintain the quality of the original video while also reducing the size of the file. Video encoding is essential for storing, transmitting, and playing back HD and UHD video content. Advanced video compression standards such as H.264, H.265, and VP9 are used to reduce the size of video files while maintaining the quality of the video. Video encoding is an important technology that has enabled the delivery of high-quality video content over the internet, and it will continue to be a crucial aspect of video technology as video resolutions continue to increase [16].

Video encoding plays a critical role in live distance learning, which involves real-time transmission of video and audio content between instructors and learners who are physically located in different places. Live distance learning platforms require high-quality video streaming and playback to provide an immersive and engaging experience for learners. However, the high bandwidth requirements of video content present significant challenges for efficient transmission and storage, especially in bandwidth-limited scenarios.

Various video encoding techniques and standards have been developed to address these challenges:

2.3 Evolution of video encoders: a chronological overview

In this section, we present a brief overview of video encoders, arranged chronologically from oldest to newest, along with a description of each.

- MPEG-2, developed in the 1990s, is an older video compression standard still used for broadcast television and DVD video. It provides good video quality but requires a higher bitrate than H.264 and other newer compression standards.
- MPEG-4, developed in the early 2000s, offers better compression efficiency than MPEG-2 and is employed in various applications such as video conferencing, streaming video, and mobile video. It also serves as a container format for audio and video data.
- JPEG 2000, developed in the early 2000s, is primarily used for image compression but can also be employed for video compression. It provides both lossless and lossy compression, making it suitable for applications like medical imaging and digital cinema.
- Theora, developed in the early 2000s, is an open and royalty-free video compression standard based on VP3, an earlier codec from On2 Technologies. Theora is primarily used for web-based video streaming and provides reasonable video quality at a lower bitrate than H.264. However, it has been largely superseded by newer standards such as VP8 and VP9.
- H.264, introduced in 2003, is the most commonly used video compression standard. It strikes a balance between video quality and file size, making it appropriate for a wide range of applications such as video streaming, video conferencing, and video surveillance.
- Dirac, developed by BBC Research in the mid-2000s, is an open and royalty-free video compression standard designed for a wide range of applications, including internet video

streaming, video conferencing, and digital television broadcasting. It offers high-quality video at a low bitrate and supports lossless as well as lossy compression.

- VP8, introduced by Google in 2010, is an open and royalty-free video compression standard widely used for web-based video streaming. It offers improved compression efficiency compared to earlier standards, enabling smoother online video streaming experiences.
- H.265/HEVC, developed in 2013, provides better video quality than H.264 at the same file size, making it particularly useful for high-resolution videos such as 4K and 8K. However, it requires more processing power to encode and decode video compared to H.264.
- VP9, developed by Google in 2013, is primarily used for web-based videos such as YouTube. It provides high-quality video at a lower bitrate than H.264, making it a suitable choice for online video streaming.
- Daala, developed by the Xiph.Org Foundation in the 2010s, is an experimental open-source video compression standard that aims to provide better compression efficiency than existing standards while avoiding patent encumbrances. It employs novel techniques such as lapped transforms and perceptual vector quantization to achieve improved video quality at lower bitrates. However, its development has been put on hold in favor of the AV1 standard.
- AV1, developed by the Alliance for Open Media (AOMedia) in 2018, is a newer open-source video compression standard that offers better compression efficiency than H.265 and VP9. It is ideal for streaming high-quality video over the internet, although it requires more processing power to encode and decode video compared to other compression standards.
- VVC, also known as H.266, is a video compression standard developed by the Joint Video Experts Team (JVET) and finalized in 2020. It aims to provide a significant improvement in compression efficiency compared to H.265/HEVC, particularly for high-resolution and high-dynamic-range (HDR) video content.
- EVS, developed by the Moving Picture Experts Group (MPEG) and finalized in 2020, is a video compression standard designed to provide better compression efficiency than H.265/HEVC while minimizing patent licensing costs. It features a modular design with a royalty-free baseline profile and an enhanced profile that includes additional patent-encumbered tools for higher compression efficiency.

These video encoders play a crucial role in various applications, enabling efficient video storage and transmission while maintaining video quality. As technology advances and higher-resolution video formats become more prevalent, the development of new video compression standards will continue to be an important area of research and innovation.

2.4 Real-time video encoding

Real-time video encoding refers to the process of compressing video data in real-time as it is captured, transmitted, or displayed. This process is essential for a variety of applications, including video conferencing, live streaming, gaming, virtual reality, and augmented reality. Real-time video encoding enables these applications to deliver high-quality video content while minimizing latency and bandwidth requirements [17].

In real-time video encoding, video data is compressed using a codec, which stands for encoder/decoder. A codec is a software or hardware algorithm that compresses and decompresses video data. There are two types of codecs: lossless and lossy. Lossless codecs preserve

all the original data in the video stream, while lossy codecs discard some data to achieve higher compression ratios. The choice of codec depends on the application requirements and available resources.

Real-time video encoding is a challenging task that requires significant computational resources. The encoding process must be completed within a tight timeframe to avoid latency and ensure a smooth viewing experience. To achieve this, real-time video encoders use specialized hardware and software that optimize the encoding process for speed and efficiency.

One of the most important factors in real-time video encoding is the tradeoff between video quality and bandwidth requirements. Higher-quality video requires more data, which means that more bandwidth is needed to transmit or store the video stream. However, bandwidth is often limited, especially in applications such as video conferencing and live streaming. Real-time video encoders must balance these tradeoffs to ensure that the video quality is sufficient for the application while minimizing bandwidth requirements.

Real-time video encoding is a critical component of many applications, including those in the fields of entertainment, education, healthcare, and communication. As the demand for high-quality video content continues to grow, real-time video encoding will become even more important for delivering a seamless and immersive experience for users.

2.5 Applications of real-time video encoding

Real-time video encoding plays a crucial role in various applications, as it enables instantaneous communication and interaction. In video conferencing, real-time video encoding is indispensable, allowing remote participants to see and hear each other without any significant delay. If real-time encoding were not employed, the video and audio streams would suffer from latency, leading to disjointed conversations. Live streaming of events, concerts, and sports games relies on real-time video encoding. This ensures that the audience can view the unfolding action and that latency is minimized to prevent any substantial delays in the stream.

Real-time video encoding is utilized in gaming to capture and stream gameplay as it occurs, letting viewers watch and interact with the game in real-time. Virtual and augmented reality applications demand real-time video encoding to provide immersive experiences. The encoding synchronizes the virtual or augmented elements with the real-world environment, resulting in a seamless and engaging experience for the user. Telemedicine leverages real-time video encoding to enable remote consultations between patients and doctors. This real-time encoding allows doctors to see and diagnose patients promptly, irrespective of their geographical locations.

2.6 VR/AR real-time video encoding

Virtual reality and augmented reality are two rapidly growing fields that require high-quality video content to provide a truly immersive experience. In VR, users are transported to a virtual environment, while in AR, virtual content is overlaid in the real world. Both VR and AR rely heavily on real-time video coding to ensure a smooth and seamless experience for the user. This is where the need for efficient encoders comes into play [1].

Efficient encoders are necessary for VR and AR because they can compress video data in real time while maintaining high quality. This is particularly important for VR and AR applications, which require high frame rates and low latency to create a sense of presence

and immersion. Real-time video coding is essential for ensuring that the video content is delivered to the user in a timely and uninterrupted manner.

One of the key challenges in VR and AR video encoding is the need to maintain high quality while keeping the file size small. This is because the file size directly impacts the amount of data that needs to be transmitted or stored, which can affect the overall performance of the VR or AR system. Efficient encoders can help reduce the file size while maintaining high quality, which is crucial for delivering a seamless VR or AR experience [11].

Another challenge in VR and AR video encoding is the need to support a wide range of devices and platforms. VR and AR content can be viewed on a variety of devices, including smartphones, tablets, and VR headsets, each with different processing capabilities and screen resolutions. Efficient encoders can provide the flexibility needed to support these devices and platforms while also ensuring that the video content is optimized for each device.

Overall, the need for efficient encoders for VR and AR is crucial for creating a truly immersive experience for the user. Real-time video coding is essential for ensuring that the video content is delivered in a timely and uninterrupted manner, while efficient encoding can help to reduce the file size while maintaining high quality. As VR and AR continue to grow in popularity, efficient encoders will become even more important for delivering high-quality video content that provides a truly immersive experience for users.

2.7 Teleoperation and telepresence real-time video encoding

Teleoperation and telepresence are two related concepts that involve the use of technology to allow a person to remotely control or interact with a physical environment or object. Teleoperation typically involves the use of robotics or other machinery to perform tasks remotely, while telepresence is focused more on providing a sense of being physically present in a remote location. One way to enhance the sense of telepresence is through the use of 360-degree video, which allows the viewer to see a full view of the remote environment in all directions. This can be especially important for teleoperation in complex or dynamic environments, where the operator needs to have a comprehensive understanding of the environment to make effective decisions.

For example, in telemedicine applications, a surgeon might use a remote-controlled robot to perform a procedure on a patient in a different location. By using a 360-degree video camera on the robot, the surgeon can get a complete view of the patient and their surroundings, allowing for more accurate and efficient control of the robot. Similarly, in teleconferencing or remote collaboration scenarios, the use of 360-degree video can help to create a more immersive and engaging experience for participants. By providing a full view of the remote environment, participants can feel more connected to the discussion and have a greater sense of presence in the remote location. The use of 360-degree video can play an important role in enhancing teleoperation and telepresence experiences, by providing a more complete and immersive view of the remote environment.

HEVC is a key technology that can greatly improve the quality and efficiency of 360-degree video, making it a more viable option for teleoperation and telepresence applications. With HEVC, video content can be compressed to a smaller size without sacrificing image quality, allowing for more efficient transmission and storage of 360-degree video. By using HEVC in combination with 360-degree video, teleoperation can be greatly enhanced, as operators can receive a full view of the remote environment in high quality while also benefiting from reduced bandwidth requirements and improved transmission speeds. Similarly, telepresence applications can benefit from HEVC's improved compression capabilities, allowing

for a more immersive experience that can be transmitted with lower latency and bandwidth requirements. The combination of 360-degree video and HEVC can greatly enhance the effectiveness of teleoperation and telepresence applications, providing a more complete and immersive view of remote environments while also enabling more efficient transmission and storage of video content.

2.8 Comparing AV1, HEVC, VP9, VVC, and EVC

When comparing AV1, HEVC, VP9, VVC, and EVC, several factors need to be considered, including compression efficiency, processing power, licensing, and adoption.

As illustrated in Fig. 1, EVC, Av1, and VVC outperform HEVC in required bandwidth. AV1 provides better compression efficiency than both HEVC and VP9, which means that it can produce the same video quality at a lower bitrate. This makes AV1 an ideal choice for streaming high-quality video over the internet. However, HEVC also offers improved compression efficiency compared to VP9, making it a suitable option for many applications. VVC is a more recent video coding standard that provides even better compression efficiency than AV1, HEVC, and VP9. EVC is a newer video coding standard that aims to provide improved compression efficiency over HEVC while being backward-compatible with existing video codecs [18].

All five codecs, AV1, HEVC, VP9, VVC, and EVC, require more processing power to encode and decode video compared to older compression standards like H.264. However, AV1 and VVC require even more processing power than HEVC and VP9, which can make them more challenging to use on older devices or with limited computing resources. HEVC and VVC are patented and require licenses to use, which can make them more expensive to implement in commercial products. On the other hand, both AV1 and VP9 are open-source standards and can be used without any licensing fees. EVC is a standard developed by several companies and organizations, including Apple, Huawei, and Qualcomm, and is expected to have licensing fees lower than those of HEVC and VVC. HEVC has been around longer and is supported by a wider range of devices and software, including many popular video streaming services. VP9 is also widely adopted, particularly by Google's ecosystem, including YouTube. However, AV1 is gaining popularity and is supported by a growing number of devices and software, particularly for online video streaming. VVC and EVC, being more recent standards, have not yet achieved widespread adoption but are expected to gain more support as their performance advantages become more apparent.

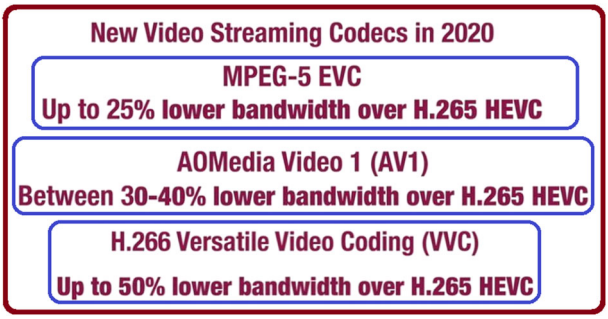


Fig. 1 Comparison of bandwidth requirements for new codecs relative to HEVC [19]

AV1, HEVC, VP9, VVC, and EVC are all block-based video compression standards, which means that they use a similar structure of dividing the video frames into blocks and applying compression techniques to each block separately. However, the specific structures used by each codec differ in some ways. For example, HEVC uses a structure called a Coding Tree Unit (CTU) to divide the frames into blocks, while AV1 uses a structure called a Superblock, VP9 uses a 64x64 pixel block structure, VVC employs a Treeblock structure, and EVC uses a flexible block structure that can adapt to the content of the video frame [20]. In HEVC, CTUs are further divided into smaller blocks called Large Coding Units (LCUs), which are similar in structure to AV1's Superblocks, VP9's partitioning system, VVC's Treeblocks, and EVC's flexible block structure. LCUs, Superblocks, VP9's block structures, Treeblocks, and EVC's flexible block structure are all used to apply advanced compression techniques such as inter-prediction, intra-prediction, and transform coding to individual blocks of video frames [21]. Therefore, while the specific names and structures used by AV1, HEVC, VP9, VVC, and EVC may differ, all five standards rely on block-based compression techniques to achieve high-quality video compression [22].

2.9 HEVC adaptive quad tree structure

The HEVC standard incorporates novel features, including a versatile data structure composed of the CU, PU, and transform unit. Initially, the RDO algorithm was employed to partition the LCU into CUs, which proved to be computationally demanding and unsuitable for real-time applications. As a result, sub-optimal LCU partitions were generated, leading to prolonged encoding times.

To address this challenge, an adaptive quad-tree structure was implemented in HEVC. As reported by [23], each LCU has a fixed size of 64×64 and can be divided into four CUs of sizes 32×32 when necessary. Moreover, each 32×32 CU can be further subdivided into four CUs with dimensions of 16×16, and each 16×16 CU can be partitioned into four additional CUs with sizes of 8×8. These partitions are designated as depth 0, 1, 2, and 3 for 64×64, 32×32, 16×16, and 8×8, respectively. The quad-tree CTU structure is depicted in Fig. 2.

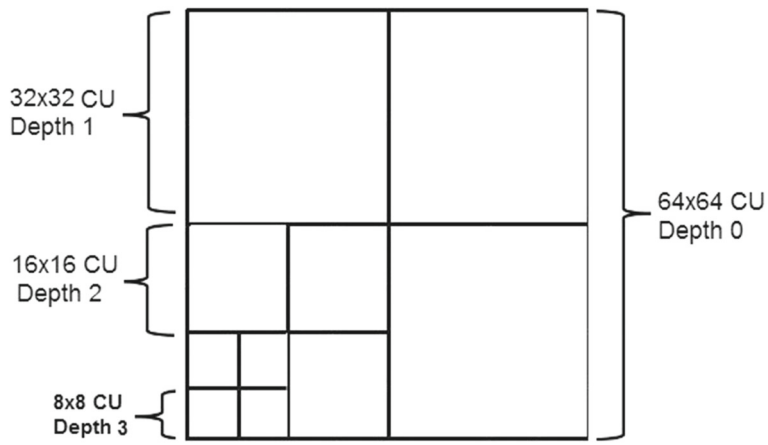


Fig. 2 LCUQuad-tree structure

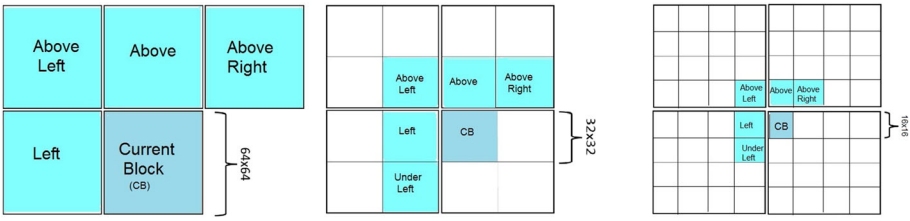


Fig. 3 Depths of 0, 1, and 2 of spatial neighbors CUs

Figure 3 depicts the spatial neighbors at depths 0, 1, and 2. The block being processed, designated as $CU_{current}$, is surrounded by spatial neighbors of equal size, as evident in the figure. The HEVC frame encoding adheres to a z-order from left to right and top to bottom. Furthermore, Fig. 4 illustrates the temporally co-located neighbor from the preceding frame, which shares the same size and spatial z-order as $CU_{current}$ in the current frame.

The partitioning scheme in HEVC facilitates the division of video frames into a hierarchical tree structure of rectangular blocks called CUs. CUs can subsequently be partitioned into smaller rectangular blocks known as PUs and TUs.

The partitioning scheme in HEVC offers numerous advantages, encompassing enhanced compression efficiency and superior video quality. By dividing video frames into smaller blocks, HEVC can more precisely represent intricate visual details, such as texture and motion, resulting in higher-quality video with fewer discernible compression artifacts.

Furthermore, the adaptive partitioning scheme in HEVC enables improved encoding efficiency by adjusting to the specific attributes of each video frame. For instance, the partitioning scheme can be tailored to encode smoother areas of the image with larger blocks, while employing smaller blocks for more complex regions.

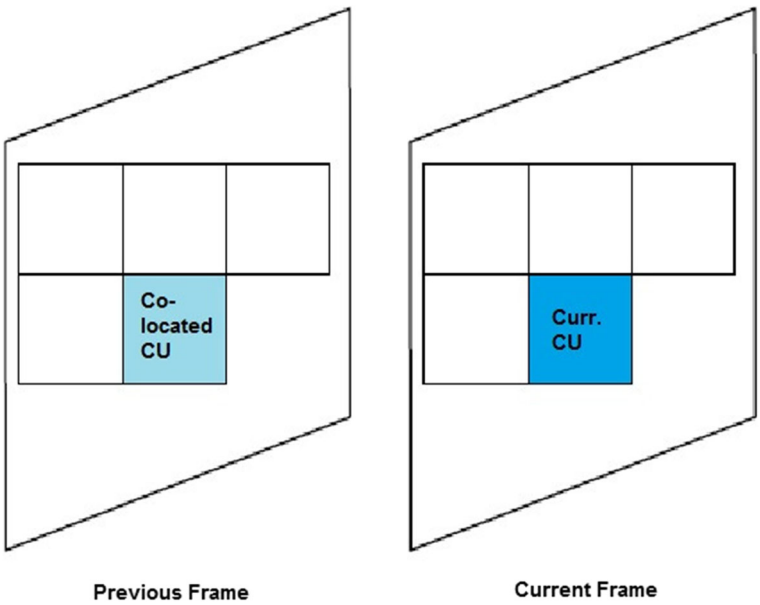


Fig. 4 Temporal neighbor CU

2.10 AV1 Adaptive Binary Tree (ABT) coding structure

AV1's ABT coding structure is a hierarchical approach that adaptively partitions CUs into sub-CUs of different sizes based on the content of the video frame being encoded. The ABT structure starts with the entire frame as a single CU and recursively divides it into sub-CUs until a minimum block size is reached. The block sizes that can be used for sub-CUs are defined by the flexible block structure, which is hierarchical and ranges from 4x4 pixels to 128x128 pixels. The decision of whether to partition a CU or not is based on various factors, such as the spatial and temporal activity in the CU, the amount of texture in the CU, and the rate-distortion tradeoff. The ABT structure also allows for rectangular CUs to be used, which can be useful in situations where the content of the video frame has an aspect ratio that is different from the standard square block sizes. The ABT structure is adaptive in that the encoder can change the CU size and structure on a frame-by-frame basis, based on the characteristics of the video content.

The ABT structure in AV1 provides a flexible and adaptive way to partition CUs into sub-CUs of different sizes, which can improve coding efficiency and video quality. The flexibility in block sizes allows for greater efficiency in coding video frames, as the encoder can choose the block size that best fits the content being encoded. The block sizes that can be used in AV1 are defined as powers of 2, ranging from 4x4 pixels to 128x128 pixels. The block structure is hierarchical, meaning that smaller blocks are grouped to form larger blocks. The block sizes used in AV1 are as follows:

4x4 pixels, 4x8 pixels, 8x4 pixels, 8x8 pixels, 8x16 pixels, 16x8 pixels, 16x16 pixels, 16x32 pixels, 32x16 pixels, 32x32 pixels, 32x64 pixels, 64x32 pixels, 64x64 pixels, 64x128 pixels, 128x64 pixels, 128x128 pixels. Furthermore, AV1 allows for rectangular blocks to be used, with block sizes such as 2x8, 8x2, 4x16, 16x4, and so on. This flexibility in block sizes allows for greater efficiency in coding video frames, as the encoder can choose the block size that best fits the content being encoded.

2.11 VVC Quadtree-plus-Binary-Tree (QTBT) coding structure

VVC's QTBT coding structure is a hierarchical approach that adaptively partitions CUs into sub-CUs of different sizes based on the content of the video frame being encoded. The QTBT structure combines the quadtree partitioning used in HEVC with an additional binary tree splitting for more refined partitioning. The QTBT structure starts with the entire frame as a single CU and recursively divides it into sub-CUs using both quadtree and binary tree partitioning until a minimum block size is reached. The block sizes that can be used for sub-CUs are defined by the flexible block structure, which is hierarchical and ranges from 4x4 pixels to 128x128 pixels [24].

The decision of whether to partition a CU or not is based on various factors, such as the spatial and temporal activity in the CU, the amount of texture in the CU, and the rate-distortion tradeoff. The QTBT structure also allows for rectangular CUs to be used, which can be useful in situations where the content of the video frame has an aspect ratio that is different from the standard square block sizes. The QTBT structure is adaptive in that the encoder can change the CU size and structure on a frame-by-frame basis, based on the characteristics of the video content.

The QTBT structure in VVC provides a flexible and adaptive way to partition CUs into sub-CUs of different sizes, which can improve coding efficiency and video quality. The

flexibility in block sizes allows for greater efficiency in coding video frames, as the encoder can choose the block size that best fits the content being encoded.

The block sizes that can be used in VVC are defined as powers of 2, ranging from 4x4 pixels to 128x128 pixels. The block structure is hierarchical, meaning that smaller blocks are grouped to form larger blocks. The block sizes used in VVC are as follows:

4x4 pixels, 4x8 pixels, 8x4 pixels, 8x8 pixels, 8x16 pixels, 16x8 pixels, 16x16 pixels, 16x32 pixels, 32x16 pixels, 32x32 pixels, 32x64 pixels, 64x32 pixels, 64x64 pixels, 64x128 pixels, 128x64 pixels, 128x128 pixels.

Furthermore, VVC allows for rectangular blocks to be used, with block sizes such as 2x8, 8x2, 4x16, 16x4, and so on. This flexibility in block sizes allows for greater efficiency in coding video frames, as the encoder can choose the block size that best fits the content being encoded.

2.12 EVC flexible coding structure

The EVC standard also uses a block-based coding structure, similar to HEVC, AV1, and VVC. However, EVC employs a more flexible and adaptive approach to partitioning CUs into sub-CUs of different sizes.

EVC's flexible coding structure allows for CUs to be partitioned into sub-CUs of sizes ranging from 4x4 to 64x64 pixels, with rectangular blocks also being supported. The partitioning decision is based on various factors, such as the texture complexity and motion activity in the CU, and the rate-distortion tradeoff. This adaptiveness allows for greater coding efficiency and video quality.

EVC's coding structure also includes a hierarchical FMO that allows for more efficient coding of video frames with complex motion or texture patterns. FMO partitions the video frame into slices and partitions the slices into macroblocks, which can be further subdivided into CUs. This hierarchical approach allows for more efficient coding of complex video frames, as the encoder can adaptively choose the best coding structure based on the content of the video.

EVC also includes various advanced coding tools, such as intra-prediction, inter-prediction, and transform coding, to improve compression efficiency and video quality. In addition, EVC supports features such as sub-pixel motion estimation and loop filtering to reduce visual artifacts in compressed video.

In terms of licensing, EVC is a royalty-free video coding standard developed by a consortium of companies, including Huawei, Qualcomm, and Samsung. This makes EVC a more accessible option for companies and individuals who may not have the resources to pay licensing fees for other advanced video coding standards.

Overall, EVC's flexible and adaptive coding structure allows for greater coding efficiency and video quality compared to older compression standards like H.264. Additionally, its royalty-free licensing model makes it a more accessible option for companies and individuals looking to implement advanced video coding technology.

2.13 RDO in block partitioning for HEVC and AV1

Rate-distortion optimization is an essential technique employed in modern video codecs, such as HEVC and AV1, to determine the optimal block partitioning for efficient video compression. RDO aims to find the optimal balance between the amount of compressed data

(bitrate) and the quality of the reconstructed video (distortion). This section discusses RDO's role in block partitioning for both HEVC and AV1 video codecs.

2.13.1 RDO in HEVC

In HEVC, video frames are divided into a hierarchical tree structure of rectangular blocks called CUs. The adaptive quad-tree structure in HEVC allows for recursive block partitioning, where each CU can be further divided into four equally sized smaller CUs. The encoding process involves selecting the optimal CU size and partitioning scheme that minimizes the rate-distortion cost.

RDO plays a crucial role in this decision-making process. During the encoding process, HEVC evaluates various block partitioning schemes and calculates the rate-distortion cost for each possible partition. The partition with the lowest rate-distortion cost is selected as the optimal choice. In this way, RDO helps HEVC to efficiently adapt the partitioning scheme to the specific characteristics of each video frame, enabling better encoding efficiency and improved video quality.

2.13.2 RDO in VP9

Like HEVC and AV1, VP9 also utilizes RDO techniques to optimize encoding decisions. In VP9, video frames are divided into rectangular blocks called superblocks, which have a size of 64x64 pixels. These superblocks can be further partitioned into smaller coding units.

During the encoding process, RDO is used to make various decisions, such as selecting the best mode for each block, determining optimal motion vectors, and choosing the most suitable quantization parameters. RDO in VP9 involves calculating the rate-distortion cost for each possible encoding decision and selecting the one with the lowest cost. This process helps to achieve a high compression ratio while maintaining good video quality, which is especially important for applications with high-resolution video or limited bandwidth availability.

However, it is important to note that RDO introduces computational complexity to the encoding process, which can result in longer encoding times. Despite this drawback, the benefits of RDO in terms of improved video quality and reduced bitrate often outweigh the increased computational requirements.

In summary, RDO plays a significant role in optimizing the encoding process for various video codecs, including VP9. By employing RDO techniques, VP9 can efficiently compress video content and deliver high-quality video at reduced bitrates.

2.13.3 RDO in AV1

Similar to HEVC, AV1 also employs an adaptive quad-tree structure in its partitioning scheme. In AV1, video frames are divided into rectangular blocks called superblocks, which can be hierarchically partitioned into smaller blocks using a quad-tree structure. The partitioning process in AV1 aims to determine the optimal block size and partitioning scheme for different regions of the frame based on their content complexity.

RDO is integral to this process in AV1. During the encoding process, AV1 evaluates different block partitioning schemes by calculating the rate-distortion cost for each possible partition. The partition with the lowest rate-distortion cost is selected as the optimal choice. This allows AV1 to adapt its partitioning scheme to various video content and achieve more

efficient encoding by using larger blocks for smooth regions and smaller blocks for complex, detailed areas.

In conclusion, RDO is a critical component in block partitioning for both HEVC and AV1 video codecs, guiding the selection of optimal partitioning schemes to minimize rate-distortion costs. By employing RDO, both HEVC and AV1 achieve better compression efficiency and improved video quality, enabling the delivery of high-quality video content at reduced bitrates.

2.14 AI in video coding

AI has demonstrated the potential to significantly enhance video encoding processes, such as HEVC and AV1, by accelerating the encoding process while preserving compression efficiency. Several approaches can be employed to achieve these improvements.

AI and machine learning techniques have been utilized in video coding for various purposes, including augmenting compression efficiency, improving video quality, and decreasing computational complexity. Machine learning algorithms, capable of analyzing data patterns and learning from them, can accurately predict the content of future frames, leading to superior compression performance and heightened video quality. Furthermore, deep learning techniques, such as convolutional neural networks (CNNs), can identify and eliminate redundancies within video frames, resulting in more efficient compression, reduced file sizes, and preserved video quality.

Transformers, a type of deep learning model, have demonstrated great success in natural language processing tasks, such as language translation and text classification. Recently, researchers have started exploring the use of Transformers in video coding, which is the process of compressing video data for efficient storage and transmission. These deep learning models can replace conventional block-based motion estimation and compensation methods with attention-based motion estimation. This approach employs a self-attention mechanism to pinpoint relevant frames within a video sequence containing similar features to the current frame. By concentrating on these frames, the model can more precisely estimate and compensate for motion, leading to enhanced compression efficiency. Transformers can also be utilized for intra-frame prediction, which involves predicting the pixel values of a frame based on previously coded frames. Traditional video coding typically employs block-based prediction methods, which can result in block artifacts and reduced compression efficiency. Using a transformer model for intra-frame prediction can yield smoother and more accurate predictions, culminating in superior compression efficiency. The use of Transformers in video coding is a promising area of research that has the potential to significantly improve compression efficiency and video quality. However, there are still many challenges that need to be addressed, such as the high computational cost of training and inference with large Transformer models, as well as the need for efficient implementation in real-world video coding applications.

AI can analyze existing encoding algorithms and propose optimizations or entirely novel algorithms that maintain or enhance compression efficiency while diminishing computational complexity. This process could lead to expedited encoding times without sacrificing quality. AI can also aid in the design of specialized hardware or suggest optimizations for current hardware to expedite video encoding processes. This may involve developing application-specific integrated circuits (ASICs) or optimizing graphics processing units (GPUs) for video encoding tasks. Moreover, AI can devise techniques to improve the parallelization of encoding tasks, making them more compatible with multi-core processors and distributed computing

environments. This approach would facilitate faster encoding by distributing the workload across multiple cores or machines. Additionally, AI can employ machine learning techniques to create adaptive encoding methods that more efficiently allocate bits to different parts of a video based on its content. These methods would adapt to each video's unique characteristics, leading to more efficient compression and potentially faster encoding times.

AI holds the potential to substantially improve video compression efficiency, reduce file sizes, and enhance video quality in video encoding processes like HEVC and AV1. However, realizing these improvements necessitates significant research and development endeavors.

2.15 Shannon entropy

The partitioning of the LCU has been the subject of much research in the field of video encoding, as it can have a significant impact on the efficiency and quality of the encoding process. Different partitioning strategies have been proposed, each with its own strengths and weaknesses.

One common approach is to use a quad-tree structure to recursively divide the LCU into smaller blocks until a predetermined minimum block size is reached. This approach is used in the current standard for video encoding, HEVC, and is effective in achieving high compression ratios while maintaining good image quality. Our research compares with and uses the block entropy that was introduced by Shannon [25].

Shannon entropy is a measure of the amount of uncertainty or randomness in a given set of data. It was first introduced by Claude Shannon in the field of information theory and has since found wide application in various fields, including image processing.

In the context of image processing, Shannon entropy can be used to analyze the information content of an image or a block within an image. The entropy of an image is a measure of the amount of information contained in the image, while the entropy of a block within an image is a measure of the amount of information contained in that block.

The entropy of an image can be calculated using the following formula:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

where X is the set of pixel values in the image, n is the total number of unique pixel values in the image, $p(x_i)$ is the probability of pixel value x_i occurring in the image, and $\log_2$ is the base 2 logarithm.

The more uniform the distribution of pixel values in the image, the higher the entropy will be, indicating that the image contains a large amount of information. Conversely, if the pixel values are concentrated around a few values, the entropy will be lower, indicating that the image contains less information.

Similarly, the entropy of a block within an image can be calculated using the same formula, but considering only the pixel values within that block. This can be useful in identifying regions of an image that contain more information than others and can be used for tasks such as image segmentation and compression.

In summary, the Shannon entropy of an image or a block within an image provides a measure of the amount of information contained in that image or block. It can be used for a wide range of image processing tasks and is an important tool for analyzing and understanding the information content of digital images.

2.16 Entropy-based CU splitting and termination algorithm

Several studies have proposed techniques to increase the encoding speed of the HEVC encoder. Some studies have focused solely on the partitioning process, as evidenced by studies such as [7, 10, 26, 27]. In particular, the study by [26] suggested an approach based on the amount of information in a block, measured using Shannon entropy. This algorithm is referred to as the "entropy-based algorithm," and Shannon entropy is referred to as "ShannonEntropy" in this paper.

The amount of information needed to encode an image or video by a compression algorithm is represented by the entropy of a CU, as per the definitions of Shannon entropy [28] and information [29].

To calculate the entropy value of a Coding Unit (CU), one can use the Shannon Entropy Equation [28, 29]. This equation was originally introduced by Claude E. Shannon in his 1948 paper "A Mathematical Theory of Communication" [30] and is expressed as follows:

$$E_{cu} = -\sum_{i=0}^{N} \frac{f(i)}{N} \times \log_2 \frac{f(i)}{N}, \tag{1}$$

The CU's total number of pixels and the total number of occurrences of a pixel value i (frequency of i), denoted by N and f respectively, are used in (1) to evaluate its entropy. To increase pixel redundancy in the CU, noise is eliminated by applying quantization before computing the entropy. The authors of [26] proposed a method that uses this entropy-based algorithm to partition the CU, and Fig. 5 shows its pseudocode. This technique achieves a significantly faster encoding speed than other existing methods, with an average bitrate that is 3.5 times faster than the original RDO algorithm. Additionally, the average degradation in quality, as measured by PSNR, is negligible.

These three papers propose different methods for improving the encoding efficiency of video compression through early termination or rapid determination of CUs.

In [10], an adaptive threshold for the MSE of prediction residuals is used to terminate CU splitting early, resulting in an average reduction in encoding time of up to 34.83

In [31], the size of CUs is rapidly determined based on the depth of spatial and temporal neighbors, and CUs are split or terminated based on the size of neighboring CUs. The authors claim a 43% reduction in encoding time compared to the original HM5.0 encoder.

```
1. For each frame in the video:
    2. Initialize an empty list to store the encoded LCU data
    3. For each LCU in the frame:
        4. Initialize a variable to store the total entropy of all possible CUs in the LCU
        5. For each CU in the LCU:
            6. Calculate the entropy value of the CU and store it
            7. Add the entropy value of the CU to the total entropy of the LCU
        8. Calculate the average entropy of all CUs in the LCU
        9. For each CU in the LCU:
            10. If the entropy of the CU is greater than 3.5, split to the next CU
            11. If the entropy of the CU is less than 1.2, terminate encoding
            12. If the entropy of the CU is within ±0.15 of the average entropy, terminate encoding
            13. If the entropy of the CU is not within ±0.15 of the average entropy, split the CU
        14. Store the encoded LCU data
    15. Output the encoded frame data
\end{algorithmic}
\end{algorithm}
```

Fig. 5 Shannon entropy algorithm [ShannonEntropy]

In [7], CU splitting and termination are modeled as a binary classification problem using SVM, resulting in a time saving of approximately 41.9% under the low delay profile setting compared to the HEVC reference software.

Overall, these papers highlight the importance of efficient CU splitting and termination in video compression and demonstrate different methods for achieving this.

Recent technological advancements have led to significant research across various domainsin AI, real time and communications. Jangade and Babulal [32] explored deep learning models for human pose estimation, emphasizing its real-time applications [32]. In the feild of wireless sensor networks (WSN), Bairagi et al. [33] proposed an energy-efficient protocol enhancing routing performance [33].

Significant contributions in 2023 include Al-Ghuwairi et al.'s work on intrusion detection in cloud computing using machine learning [34], Sharrab et al.'s deep neural networks in social media forensics [35], and Parikh et al.'s program on cooperative adaptive cruise control [36]. Sharrab et al. [15] compared deep learning-based object detection algorithms [16], while Sharrab et al. [1] discussed AI, VR, and 6G in smart, immersive classrooms [1]. Tarabin et al. [37] focused on detecting distracted drivers [37], and Al-Ghuwairi et al. [34] visualized software refactoring [38].

3 Novelty of the iHELP model

The iHELP model represents a significant advancement in the field of video encoding, distinguishing itself from previous studies through its innovative approach and unique features. Unlike traditional methods, iHELP employs a novel adaptive instant learning mechanism that enables real-time adaptability and immediate applicability, avoiding the extensive training phases commonly associated with machine learning or deep learning models. This distinct approach allows iHELP to rapidly adjust encoding strategies based on the current video content, leading to a more efficient encoding process.

A key aspect of iHELP's novelty lies in its ability to predict the optimal size of coding units (CUs) by leveraging the correlation of a current block with its spatial and temporal neighbors. This predictive capability is realized through a weighted average calculation of Termination Probabilities (TPs) of adjacent processed blocks, a method not typically found in conventional video encoding algorithms. Furthermore, iHELP integrates entropy-based conditions to evaluate the correlation between blocks, enhancing partitioning decisions, and effectively preventing error propagation.

The iHELP model also sets itself apart by employing empirically fine-tuned thresholds and heuristics, specifically tailored to handle high-definition video characteristics and the complexities of modern video coding standards. These customized parameters contribute to iHELP's robustness and adaptability across various video resolutions and formats.

Moreover, iHELP's design is inherently lightweight and responsive, making it particularly suitable for environments with rapid processing needs and lower computational loads. This aspect of the model contrasts with many existing AI frameworks in video encoding that rely heavily on complex data training and extensive computational resources.

One of the key innovations of iHELP lies in its adaptability to dynamic scene changes. Unlike traditional video encoding methods that struggle with rapid content variations, iHELP employs a combination of adaptive instant learning and entropy-based conditions. This enables it to efficiently handle dynamic environments by accurately predicting optimal block sizes. Such adaptability is crucial for applications with high scene variability, like

live streaming or surveillance, ensuring consistent encoding efficiency and video quality irrespective of scene complexity. This feature distinctly sets iHELP apart in the realm of video encoding technologies.

In summary, the iHELP model introduces a groundbreaking approach to video encoding, marked by its adaptive instant learning techniques, predictive capabilities, entropy-based conditions, and tailored empirical thresholds. These features collectively contribute to the model's novelty, setting it apart from existing studies and making it a promising solution for efficient and effective video encoding in diverse applications.

4 Proposed work

Our approach utilizes artificial learning to improve the encoding process of new blocks by leveraging knowledge gained from previously encoded blocks. To achieve this, we propose a new algorithm named iHELP that prioritizes reducing encoding time while maintaining high coding efficiency and video quality. This leads to not only faster encoding but also lower power consumption. iHELP predicts the coding unit (CU) size of the current block being processed by examining the partitioning of same-sized CUs in both spatial and temporal (co-located) neighborhoods that have already undergone processing. Additionally, to prevent error propagation, the algorithm applies a condition based on the entropy of the current block ($CU_{current}$).

The iHELP model predicts the partitioning decision for $CU_{current}$ by calculating a weighted average of the Termination Probability (TP) for all neighboring blocks in both spatial and temporal domains. Based on this information, the algorithm decides whether to divide the block into four sub-blocks or stop searching for the optimal partition of $CU_{current}$. The average of the TPs of spatially and temporally co-located processed blocks is known as the Termination Probability Average (TPA), which determines the probability of either splitting or terminating $CU_{current}$ based on the partitioning decision of neighboring blocks. Neighboring blocks that share an edge or corner with $CU_{current}$ and are of the same size are considered spatially or temporally co-located processed blocks. The decision to split or terminate the current block $CU_{current}$ is based on the TPA factor, which is determined by considering the partition choices of neighboring blocks and the entropy measure of $CU_{current}$. By taking into account the partition decisions of spatially and temporally co-located processed blocks, iHELP calculates a weighted average of the Termination Probability (TP) values, resulting in the TPA value. This TPA value then determines the likelihood of either splitting the block into smaller sub-blocks or terminating the search for the optimal partition of $CU_{current}$.

The formal definition of TPA is as follows:

$$TPA = \frac{1}{\sum_{i=1}^{N} W_i} \times \sum_{i=1}^{N} W_i, \times TP_i, \tag{2}$$

where N represents the count of both spatial and temporal neighbors. The weight, denoted as W_i, is defined in (4). Additionally, TP_i denotes the Termination Probability of the i^{th} neighbor as displayed in (3). The neighbors can be either spatial or temporal and are categorized as Co-located, Left, Above Left, Above, Above Right, and Under Left, as illustrated in Figs. 3 and 6.

$$TP_i = \left\{ \begin{array}{l} 0 \text{ if } CU_i \text{ has smaller CUs} \\ 1 \text{ else} \end{array} \right\} \tag{3}$$

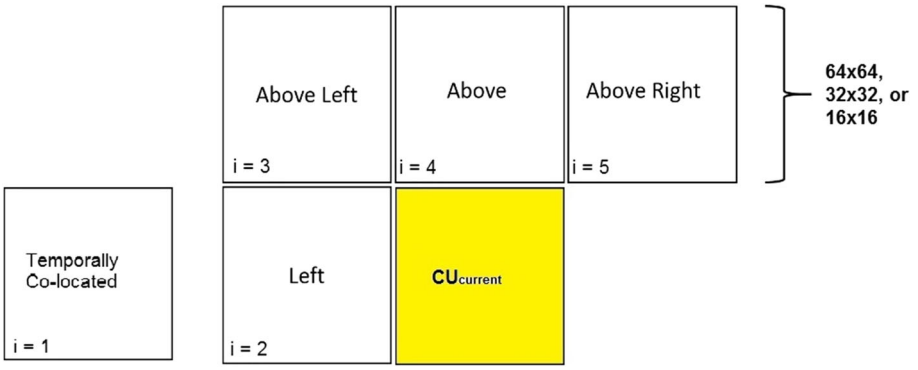


Fig. 6 Temporal and spatial neighbors

where, i refers to the neighboring elements that are the same size as $CU_{current}$, both spatially and temporally.

$$W_i = \frac{1}{1 + |Entropy_{current} - Entropy_i|} \tag{4}$$

The weight, denoted by W_i, between the current block's entropy, $Entropy_{current}$, and its i^{th} neighbor block's entropy, $Entropy_i$, is computed as $1/(1 + |Entropy_{current} - Entropy_i|)$. Here, $|Entropy_{current} - Entropy_i|$ is the absolute value of the difference between the two entropy values, and the weight represents the strength of correlation between the two blocks. The weight is bounded in the closed real interval [0, 1], where the maximum weight $W_{\max}$ corresponds to the highest correlation (i.e., $W_{\max} = 1/(1+0) = 1$), and the minimum weight $W_{\min}$ represents no correlation (i.e., $W_{\min} = 1/(1 + \infty) = 0$).

For example, suppose the entropy value of the current block is 2, and the entropy value of the neighbor block at location i is 5. In that case, the weight for that neighbor block is $W_i = 1/(1 + |2 - 5|) = 1/4$. To calculate the entropy value of a CU ($Entropy_{cu}$), (1) in Section 2 can be used.

Figure 7 illustrates the concept of TP_i. The figure shows a 64 × 64 coding unit (CU) that has been partitioned into four CUs. Three of the CUs are 32 × 32, while the fourth CU, positioned at the bottom left, is subdivided into four CUs, three of which are 16 × 16. Likewise, the fourth CU located at the top left is partitioned into four CUs, each with a size of 8 × 8.

In calculating the weighted average of all the neighbors' TPs, the algorithm takes into account not only the individual values of TP for each neighbor but also the weight (W) between that neighbor and $CU_{current}$. A neighbor with a higher weight has a greater impact on the termination decision based on its TP.

As an example, assume the weights for the CUs shown in Fig. 7 are as follows: [W2 = 1.0, W3 = 0.3, W4 = 0.1, W5 = 0.1]. Then, the computation of $TPA_{CUcurrent}$ is as follows: $TPA_{CUcurrent} = [1/(W2 + ... + W5)] \times [W2 \times TP2 + ... + W5 \times TP5] = (1/1.5) \times (1.0 \times 0 + 0.3 \times 1 + 0.1 \times 1 + 0.1 \times 1) = 0.33$. In this scenario, the neighbor with the highest correlation ($W = 1$) dominates the decision. Therefore, the algorithm decides to split if the error propagation condition is met, despite three neighbors not having smaller CUs but having a lower correlation with $CU_{current}$.

The iHELP model considers multiple factors, such as TPA, the entropy value of the current coding unit (CU) relative to the average entropy of all potential partitions within the LCU,

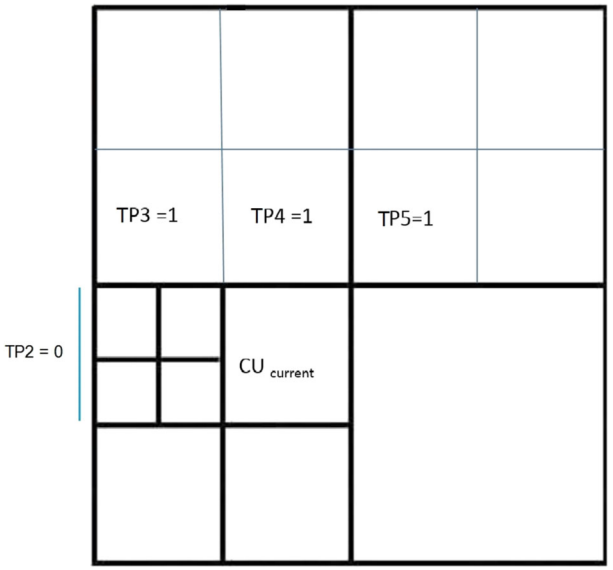


Fig. 7 An example of how to calculate TP_i, assuming the current size of CU is 16×16, is shown below

the entropy value of the temporally adjacent block, and the entropy value of each neighboring block, to partition the LCU. The pseudocode of the iHELP model can be found in Fig. 8.

Figure 8 demonstrates the utilization of a series of conditions in the iHELP model to decide whether to split or terminate a current coding unit (CU) while recursively partitioning the depth. The algorithm is triggered when the current CU's depth is below 3 and has over 4 adjacent CUs. To calculate the likelihood of termination, the algorithm begins by initializing various factors such as depthFactor, tpFactor, and dtFactor.

The iHELP model employs multiple conditions to determine whether to split or terminate the CU. The primary condition requires the Total Propagated Accuracy (TPA) to be at least 0.70. After this condition is met, the algorithm assesses if the entropy of the CU being examined is less than or equal to 1.2 plus dtFactor or if the difference between the entropy of the CU and the average entropy of neighboring CUs is less than or equal to 0.15 times the average entropy plus dtFactor. The algorithm also checks if the difference between the entropy of the CU and the entropy of the co-located CU is less than or equal to 0.5 plus dtFactor or if the sum of the entropy of the CU being examined and the average entropy of neighboring CUs is less than or equal to 3.5 plus dtFactor.

```
Input: A video
Output: Encoded video data

1. For each frame in the video:
    1.1 Initialize an empty list to store the encoded LCU data
    1.2 For each LCU in the frame:
        1.2.1 Calculate the entropy value of each possible CU in the LCU and store it
        1.2.2 Calculate the total average entropy of all possible CUs in the LCU
        1.2.3 Begin at depth 0 with a CU size of 64 x 64
        1.2.4 For each CU in the LCU:
            1.2.4.1 Use the stored entropy value of the CU to make a decision:
                1.2.4.1.1 If the entropy of the CU is greater than 3.5, split to the next CU
                1.2.4.1.2 If the entropy value of the CU is less than 1.2, terminate encoding
                1.2.4.1.3 If the entropy value of the CU is within ±0.15 of the average entropy, terminate encoding
                1.2.4.1.4 If the entropy value of the CU is not within ±0.15 of the average entropy, split the CU
        1.2.5 Store the encoded LCU data
    1.3 Output the encoded frame data
```

Fig. 8 iHELP model [iHELP]

The algorithm will stop if the main condition is satisfied along with any of the above conditions. However, if the main condition isn't met, the algorithm will examine another condition, which is TPA less than or equal to 0.30. If this is true, the algorithm will check if the entropy of the analyzed CU is greater than or equal to 3.0 plus dtFactor, or if the absolute difference between the entropy of the analyzed CU and the average entropy of neighboring CUs is greater than or equal to 0.15 times the average entropy plus dtFactor. Additionally, the algorithm will determine whether the sum of the entropy of the analyzed CU and the average entropy of neighboring CUs is greater than 6.0 plus dtFactor. If any of these conditions are fulfilled, the algorithm will divide the current CU into four CUs.

If the aforementioned conditions are not met, the original RDO method is utilized by the algorithm. The algorithm adjusts the depth factor, TPA, and entropy thresholds by utilizing a set of constants that are based on previous studies' statistics and fine-tuned by the authors.

The constants used in the algorithm are derived from statistical analysis as presented in [26]. We fine-tuned these constants to account for high-definition (HD) video characteristics, depth factor variations, and Total Propagated Accuracy (TPA) values specific to our iHELP model. The value of 0.15 indicates that if the entropy of the current coding unit (CU_{current}) and the entropy of potential smaller coding units within it are similar, then dividing CU_{current} into smaller units will not significantly decrease the overall entropy. Conversely, values of 3.0 and 1.2 represent high and low entropy thresholds, respectively. A high entropy value (e.g., 3.0) suggests that splitting CU_{current} will likely result in a substantial reduction in entropy, advocating for further partitioning. On the other hand, a low entropy value (e.g., 1.2) signals that further partitioning is unlikely to yield substantial entropy reduction, thereby indicating termination of the partitioning process. Additionally, the threshold values of 3.5 and 6.0, of entropy values under specific conditions, were also obtained from the statistical analysis in [26]. These values have been meticulously adjusted and integrated into our iHELP model to enhance its decision-making efficacy in various video coding scenarios.

iHELP deviates from the approach proposed in [31] by considering same size neighbors instead of 64×64 block neighbors.

Additionally, iHELP considers both spatial and temporal neighbors, unlike the entropy-based algorithm described in [26], which only considers conditions based on the entropy values of blocks.

While iHELP's conditions are based on previous studies such as [26], they have been adapted to account for depth and TPA values. In iHELP, termination or splitting is only allowed if it meets the TPA threshold and satisfies one of the entropy conditions to prevent error propagation. Furthermore, unlike [26], which uses entropy as the primary criterion for making decisions, iHELP uses entropy as a secondary condition alongside other criteria.

In conclusion, it is important to clarify the specific nature of AI techniques employed within the iHELP model. iHELP diverges from traditional AI frameworks, which often heavily rely on machine learning or deep learning algorithms. Instead, iHELP integrates a unique form of adaptive instant learning. This approach is specifically designed for the dynamic and efficient encoding of video data, focusing on real-time adaptability and computational efficiency.

Our implementation of AI in iHELP is centered around the analysis of previously encoded blocks and empirical data, rather than the complex data training models typical in deep learning. This design choice ensures that iHELP remains lightweight, responsive, and more suitable for environments where rapid processing and lower computational loads are prioritized. By distinguishing these aspects of AI utilization in iHELP, we aim to provide a clear understanding of our model's capabilities and its innovative approach to video encoding.

4.1 Adaptive learning in the iHELP model

The iHELP model introduces a novel approach to video encoding that employs adaptive instant learning techniques. This model deviates from conventional AI methods, which rely heavily on extensive training phases with machine learning or deep learning models. Instead, iHELP is designed for real-time adaptability and immediate applicability, avoiding the need for pre-training phases.

The adaptive learning mechanism in iHELP operates through the following steps:

1. *Extraction of Localized Patterns*: By examining spatial and temporal neighbors of the current Coding Unit (CU), iHELP identifies partitioning patterns of these processed blocks.
2. *Termination Probability Calculation*: A Termination Probability (TP) for the current CU is computed as a weighted average of the TPs of adjacent processed blocks, reflecting the model's ability to instantly learn and predict partitioning outcomes.
3. *Entropy-Based Condition*: An entropy measure is used to evaluate the correlation between the current block and its neighbors, contributing to the partitioning decision and preventing error propagation.
4. *Adaptive Instant Learning*: iHELP's core philosophy is its on-the-fly learning capability, which adjusts encoding strategies in real time based on the current video content, unlike traditional AI models that depend on pre-learned data.
5. *Balancing Efficiency and Quality*: iHELP strives to maintain a balance between encoding speed, computational efficiency, and video quality, which is crucial for real-time encoding scenarios.
6. *Empirical Thresholds and Heuristics*: The model employs statistically fine-tuned empirical thresholds and heuristics to navigate partitioning decisions, tailored to high-definition video attributes, depth factors, and Total Propagated Accuracy (TPA) values specific to the iHELP model.

This adaptive learning strategy signifies a shift from traditional AI techniques, positioning iHELP as an optimal solution for environments where prompt processing and adaptability are essential. It is optimized for on-the-spot decision-making rather than relying on historical data models, showing promise for future advancements in real-time video encoding applications.

4.2 Adaptability to dynamic content

iHELP's design incorporates advanced techniques to efficiently handle dynamic scene changes. It utilizes adaptive instant learning and entropy-based conditions to analyze and predict optimal block sizes, even in rapidly changing environments. This ensures high coding efficiency and quality, irrespective of scene complexity. Such adaptability makes iHELP particularly suitable for applications like live streaming or surveillance, where scene variability is high.

4.2.1 Performance in dynamic environments

Our performance evaluation demonstrates that iHELP maintains consistent encoding efficiency across various dynamic content scenarios. This adaptability is a testament to its robust design, tailored to meet the demands of real-time video applications.

iHELP's ability to adapt to dynamic scene changes sets it apart in the field of video encoding. This feature, coupled with its computational efficiency, positions iHELP as a

versatile tool for a broad spectrum of video applications, especially those involving high variability and motion intensity.

4.3 Processing strategies in iHELP for VR/AR applications

The iHELP (Instant High-Efficiency Learning-based Prediction) model is tailored to meet the real-time processing demands of VR/AR applications. It employs several strategies to ensure efficient and rapid video encoding:

- **Content correlation-based prediction**: iHELP predicts optimal block sizes by analyzing the content correlation between the current block and its spatial and temporal neighbors, reducing computational load.
- **Adaptive instant learning**: This technique adapts encoding strategies in real-time based on current video content, optimizing processing resources.
- **Entropy-based conditions**: iHELP uses entropy-based conditions for efficient and accurate encoding decisions, preventing error propagation.
- **Parallelization capability**: The model's structure allows parallel processing of different coding units, enhancing processing speed.
- **Optimized for HEVC**: iHELP is specifically optimized for the HEVC standard, making it ideal for high data rates and resolutions in VR/AR applications.
- **Tailored empirical thresholds and heuristics**: These are designed to navigate partitioning decisions quickly while maintaining a balance between encoding speed, efficiency, and quality.

These strategies collectively enable iHELP to effectively manage the high processing requirements of real-time VR/AR applications, delivering high-quality video encoding with low latency and high throughput.

4.4 Limitations and challenges of the iHELP model

While the iHELP model presents significant advancements in video encoding, it is crucial to acknowledge its limitations and the challenges associated with its implementation in real-world scenarios. This subsection aims to provide a transparent and comprehensive understanding of these aspects.

- **Dependency on localized data**: iHELP's effectiveness is contingent on the characteristics of previously encoded blocks within the same video sequence. This reliance means that the model may face challenges in accurately predicting partitioning strategies for video content with high variability or unique patterns not previously encountered.
- **Real-time processing constraints**: Given that iHELP is designed for real-time adaptability, its performance is heavily dependent on the processing capabilities of the system. In environments with limited computational resources, the model might not achieve the desired efficiency or may necessitate a trade-off between speed and video quality.
- **Complexity in highly dynamic scenes**: The model may struggle to maintain optimal performance in highly dynamic or fast-paced video scenes where rapid changes occur. This could lead to less efficient encoding or a potential decrease in video quality.
- **Scalability issues**: The scalability of iHELP across various video resolutions and formats is another consideration. Adapting the model to different video coding standards and resolutions may require significant modifications or additional optimizations.

- **Integration with existing systems**: Integrating iHELP into existing video encoding frameworks and systems could pose challenges, particularly regarding compatibility and the need for additional modifications to accommodate the model's unique learning-based approach.
- **Empirical threshold adjustments**: The model's reliance on empirical thresholds and heuristics, while beneficial for specific scenarios, may limit its generalizability. Tailoring these parameters for diverse video contents and applications remains a challenge.
- **Future technological advancements**: As video encoding technologies continue to evolve, the iHELP model may require ongoing updates and refinements to stay relevant and effective in the face of new advancements and standards in the field.

4.5 Latency management strategies in the iHELP model

The iHELP model addresses latency issues, which are critical for real-time applications like VR/AR, through several key strategies:

- **Predictive analysis**: By predicting optimal block sizes using content correlation between current blocks and their spatial and temporal neighbors, iHELP reduces computational processing, lowering encoding time and minimizing latency.
- **Adaptive instant learning**: This feature enables swift adjustment of encoding strategies based on current video content, enhancing real-time responsiveness.
- **Entropy-based conditions**: These conditions ensure efficient and accurate partitioning decisions, contributing to faster processing and lower latency.
- **Optimized for HEVC**: iHELP's optimization for the HEVC standard aids in handling high-resolution videos with reduced latency.
- **Empirical thresholds and heuristics**: Fine-tuned for specific video characteristics, these thresholds enable quick decision-making, reducing encoding time and thereby minimizing latency.

These strategies ensure that iHELP effectively handles the stringent low-latency requirements of real-time video applications.

4.6 Comparative analysis of iHELP and traditional video encoding methods

In this section, we provide a detailed comparative analysis of the proposed iHELP model against traditional video encoding methodologies, highlighting differences in encoding speed, efficiency, and quality metrics...

5 Experimental setup and performance evaluation

We evaluated various algorithms by comparing their performance using multiple metrics, such as *Encoding Speed Enhancement* (ESE), *Bjontegaard Delta-PSNR* (BD-PSNR), *Bjontegaard Delta-Rate* (BD-Rate), *Peak Signal to Noise Ratio* ($PSNR$), *Encoding Time Reduction* (ETR), average bitrate, and average *Peak Signal-to-Noise Ratio* (PSNR).

When assessing HEVC enhancement algorithms, encoding speed is the primary factor to consider since HEVC can achieve a bitrate reduction of approximately 50% while maintaining the same quality. To determine the proposed algorithm's relative improvement over RDO,

we introduced the *Encoding Speed Enhancement* (ESE) metric, which can be calculated as follows:

$$ESE = \frac{EA_{codingSpeed} - RDO_{codingSpeed}}{RDO_{codingSpeed}}, \tag{5}$$

In the previously given formula, ESE denotes the speed improvement in encoding achieved by the evaluated algorithm (EA) over the reference algorithm, RDO. Algorithm iHELP can be compared to RDO using this formula.

The quality and coding efficiency of encoding algorithms are often evaluated using metrics such as PSNR. However, there are limitations to using PSNR for this purpose, as highlighted in the literature. To address these limitations, the Bjøntegaard model is widely recognized as a more suitable metric for codec evaluation. The Bjøntegaard Delta-PSNR (BD-PSNR) metric calculates the average difference in PSNR, in decibels, between two rate-distortion (RD) curves generated by encoding videos at different bitrates. It measures the average PSNR difference for the same bitrate. On the other hand, the Bjontegaard Delta-Rate (BD-Rate) calculates the average percentage difference in bitrate required to achieve the same PSNR between two RD curves. BD-Rate is a useful metric for evaluating the compression efficiency of different codecs and is widely used in the video coding industry [39].

To calculate BD-PSNR, we can approximate the difference between the integrals of the RD curves of the two algorithms being compared, and then divide by the integration interval. This approximation method has been employed in previous studies [40].

To reduce the bitrate, we use the BD-Rate measure, which calculates the average difference in bitrate between two R-D curves. BD-PSNR is computed by finding the disparity in the area under the two curves and dividing it by the logarithm of the bitrate range. The equation for BD-PSNR is expressed mathematically as:

$$BD - PSNR \approx \frac{1}{r_H - r_L} \int_{r_L}^{r_H} (D_2(r) - D_1(r))dr, \tag{6}$$

When calculating BD-PSNR, we examine two fitted Rate-Distortion (RD) curves labeled as $D_1(r)$ and $D_2(r)$. Additionally, we establish the upper and lower bounds of the two curves as r_H and r_L correspondingly. To calculate these values, the following equations are utilized:

$$r_L = \max(\min(r_{1,1}, ..., r_{1,N_1}), \min(r_{2,1}, ..., r_{2,N_1})), \tag{7}$$

and

$$r_H = \min(\max(r_{1,1}, ..., r_{1,N_1}), \max(r_{2,1}, ..., r_{2,N_1})). \tag{8}$$

In the equations provided above, the logarithm of the bitrate is represented by the variable r, which is equal to $\log(R)$. To depict the computation process of BD-PSNR (BD_{PSNR}).

To estimate the Bjontegaard Delta-Rate (BD-Rate), which refers to the average difference in bitrates between two RD curves, the horizontal area under the curves is divided by the PSNR interval. The formula is expressed as follows:

$$BD - Rate \approx 10^E - 1, \tag{9}$$

where

$$E = \frac{1}{D_H - D_L} \int_{D_L}^{D_H} (r_2(D) - r_1(D))dD, \tag{10}$$

The equation given calculates the BD-Rate, which uses two fitted Rate-Distortion (RD) curves, denoted as $r_1(D)$ and $r_2(D)$, respectively. Here, D represents the distortion measured

in terms of PSNR. The starting and ending PSNR values for the two curves are determined by D_L and D_H. To calculate these values, follow the steps below:

$$D_L = \max(\min(D_{1,1}, ..., D_{1,N_1}), \min(D_{2,1}, ..., D_{2,N_1})), \tag{11}$$

and

$$D_H = \min(\max(D_{1,1}, ..., D_{1,N_1}), \max(D_{2,1}, ..., D_{2,N_1})). \tag{12}$$

The BD-Rate is used to measure the video codec's efficiency. The RD curves represent the trade-off between bit rate and distortion in video compression. The first curve is the original encoder's RD curve, while the second curve is for the compressed video codec being tested. The BD-Rate measures the percentage change in bit rate needed to achieve the same level of distortion for the compressed video codec as the original encoder. This metric is useful for comparing the performance of different codecs in terms of their compression efficiency. A lower BD-Rate indicates better compression performance.

The following equation can be used to calculate the $PSNR$ value between the original frame A and its encoded counterpart B:

$$PSNR(dB) = 10 \times \log \frac{MAX^2}{MSE}, \tag{13}$$

assuming an 8-bit representation for each pixel, MAX corresponds to the highest possible pixel value in the image, which is 255. Here, MSE stands for Mean-Square Error.

The equation for computing the MSE value is as follows:

$$MSE = \frac{1}{n \times m} \sum_{i=1}^{n} \sum_{j=1}^{m} (A_{ij} - B_{ij})^2, \tag{14}$$

where m and n represent the width and height of the image in pixels, respectively.

Evaluating HEVC partitioning algorithms involves considering the encoding time as a critical attribute. The metric used to quantify the performance of a proposed algorithm is known as Encoding Time Reduction (ETR). ETR is computed as the difference between the encoding time of the proposed algorithm and RDO relative to the RDO encoding time. The formula for calculating ETR is:

$$ETR = \frac{EA_{encodingTime} - RDO_{encodingTime}}{RDO_{encodingTime}}, \tag{15}$$

Here, EA represents the algorithm being compared to RDO, such as iHELP, and ETR denotes the encoding time reduction of RDO over EA.

We primarily employ a low-delay approach utilizing the IBBB pattern configuration. This configuration involves using a GOP that consists of all B frames, except for the first frame which is an I frame [41]. This approach is particularly useful for applications that require low latency, such as real-time video communication and gaming. By using only B frames in the GOP, we can reduce the amount of data that needs to be transmitted or stored, resulting in lower latency and higher efficiency. However, this approach can also result in lower video quality, especially when there is a lot of motion in the scene. To mitigate this issue, we can employ various techniques such as motion estimation and compensation, rate control, and adaptive quantization. Overall, the IBBB pattern configuration is a powerful tool for achieving low delay and high efficiency in video applications, but it requires careful consideration of various factors such as scene complexity, motion, and available bandwidth. With the right configuration and techniques, we can achieve high-quality video with low latency, enabling a wide range of real-time applications.

We implemented the algorithm in the HEVC Test Model, specifically, HEVC HM 13.0 [42], for our experiments. To minimize the impact of other processes during the experiments, we ensured that the computer ran with a minimal set of processes and drivers. Moreover, we repeated each experiment three times on each of the three computers.

The encoding speed results of the M4800 computer are presented in all experiments, except for when comparing the encoding speed on different computers (refer to Fig. 18). It's worth mentioning that the quality and bitrate of the results are consistent and do not vary across different experiments on the same computer or on different computers.

The experimental setup employed three computers with the following specifications:

(1) A Dell Precision M4700 with an x64-based PC, featuring an Intel(R) Core(TM) i7-3840QM CPU @ 2.80GHz, 4 cores, 8 logical processors, 16.0 GB installed physical memory (RAM), and 64-bit Microsoft Windows 8.1 Pro.
(2) A Dell Precision M4800 with an x64-based PC, featuring an Intel(R) Core(TM) i7-4810QM CPU @ 2.80GHz, 4 cores, 8 logical processors, 32.0 GB installed physical memory (RAM), and 64-bit Microsoft Windows 7 Enterprise.
(3) An HP EliteBook 820 G1 with an x64-based PC, featuring an Intel(R) Core(TM) i5-4310U CPU @ 2.00GHz, 2 cores, 4 logical processors, 8.0 GB installed physical memory (RAM), and 64-bit Microsoft Windows 7 Enterprise.

5.1 Dataset

The video sequences used in this research are summarized as follows:

- "crop_Traffic" is a video sequence that shows vehicles moving on a busy road from a stationary camera. It contains varying levels of traffic density and involves different types of vehicles.
- "Basket_ball_Drill" is a basketball practice session recorded with a stationary camera. It shows basketball players performing drills, passing the ball, and shooting baskets.
- "RaceHorses_832x480_30" is a horse racing sequence that captures a close-up view of the horses as they run on the track. It is shot at 30 frames per second and has a resolution of 832x480.
- "Basket_ball_Pass" is a video sequence that captures basketball players passing the ball to each other during a practice session. It features various types of passes, including chest passes, bounce passes, and overhead passes.
- "BlowingBubbles_416x240_50" is a video sequence that shows a young girl blowing bubbles in a garden. The video is shot at a resolution of 416x240 and has a frame rate of 50 frames per second.
- "Tennis" is a video sequence that shows a tennis match being played on a court. It features two players competing against each other, hitting the ball back and forth over the net.
- "ducks_take_off_420_720p50" is a video sequence that captures a group of ducks taking off from a pond. It is shot at a resolution of 720p and has a frame rate of 50 frames per second.
- "ducks_take_of_1080p50" is the same video sequence as "ducks_take_off_420_720p50" but shot at a higher resolution of 1080p.
- "ducks_take_off_2160p50" is the same video sequence as "ducks_take_off_420_720p50" but shot at an even higher resolution of 2160p.
- "park_joy_off_420_720p50" is a video sequence that shows people enjoying themselves in a park. It is shot at a resolution of 720p and has a frame rate of 50 frames per second.

- "park_joy_1080p50" is the same video sequence as "park_joy_off_420_720p50" but shot at a higher resolution of 1080p.
- "park_joy_2160p50" is the same video sequence as "park_joy_off_420_720p50" but shot at an even higher resolution of 2160p.
- "720p50_mobcal_ter" is a video sequence that shows people walking on a city street. It is shot at a resolution of 720p and has a frame rate of 50 frames per second.
- "720p50_parkrun_ter" is a video sequence that shows people running in a park. It is shot at a resolution of 720p and has a frame rate of 50 frames per second.
- "elephants_dream_720p24" is a short animated film that tells the story of two characters, Emo and Proog, as they explore a surreal world. It is shot at a resolution of 720p and has a frame rate of 24 frames per second.
- "life_1080p30" is a video sequence that captures various scenes from nature, including plants, animals, and landscapes. It is shot at a resolution of 1080p and has a frame rate of 30 frames per second.
- "big_buck_bunny_1080p" is a short animated film that tells the story of a rabbit named Big

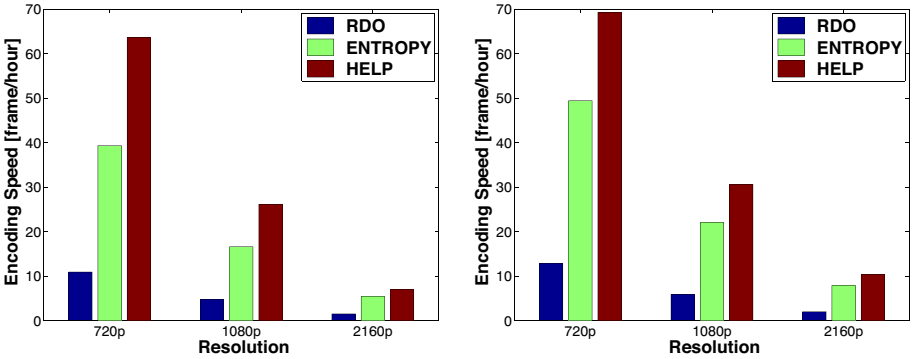
The key features of the sequences used in our algorithm testing, as reported in [43, 44], are summarized in Table 2. To establish the thresholds of our algorithms, we focused on a subset of these sequences: crop_Traffic, Basket_ball_Drill, and Basket_ball_Pass.

Table 2 Properties of the video sequences in use

Name of resolution	Number of frames	Sequence	Quantization Parameter (QP)	Description
HEVC test sequences				
crop_Traffic	150f	$2560w \times 1600h$	[22, 27, 32, 37, 42, 47]	class A-HEVC test sequence
Basket_ball_Drill	150f	$832w \times 480h$	[22, 27, 32, 37, 42, 47]	class C-HEVC test sequence
RaceHorses_832x480_30	97f	$416w \times 240h$	[32, 37, 42, 47]	class C-HEVC test sequence
Basket_ball_Pass	150f	$416w \times 240h$	[22, 27, 32, 37, 42, 47]	class D-HEVC test sequence
BlowingBubbles_416x240_50	97f	$416w \times 240h$	[32, 37, 42, 47]	class D-HEVC test sequence
Test sequences for other encoders				
Tennis	150f	$1920w \times 1080h$	[22, 27, 32, 37, 42, 47]	Full High Definition
ducks_take_off_420_720p50	50f	$1280w \times 720h$	[32]	High Definition
ducks_take_off_1080p50	50f	$1920w \times 1080h$	[32]	Full High Definition
ducks_take_off_2160p50	6f	$3840w \times 2160h$	[32]	Ultra High Definition
park_joy_off_420_720p50	50f	$1280w \times 720h$	[32]	High Definition
park_joy_1080p50	50f	$1920w \times 1080h$	[32]	Full High Definition

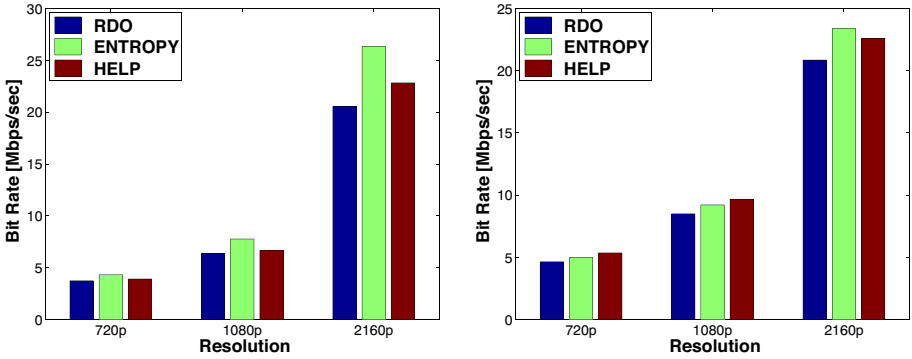
Table 2 continued

Name of resolution	Number of frames	Sequence	Quantization Parameter (QP)	Description
park_joy_2160p50	6f	$3840w \times 2160h$	[32]	Ultra High Definition
720p50_mobcal_ter	60f	$1280w \times 720h$	[37]	High Definition
720p50_parkrun_ter	60f	$1280w \times 720h$	[37]	High Definition
elephants_dream_720p24	60f	$1280w \times 720h$	[37]	High Definition
life_1080p30	60f	$1920w \times 1080h$	[37]	Full High Definition
big_buck_bunny_1080p24	60f	$1920w \times 1080h$	[37]	Full High Definition



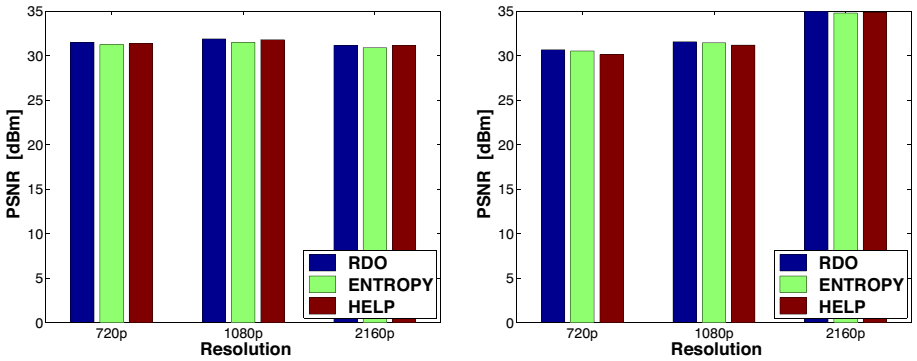
(a) Ducks_take_off (b) Park_joy

Fig. 9 Comparing different models in terms of encoding speed vs. resolution



(a) Ducks_take_off (b) Park_joy

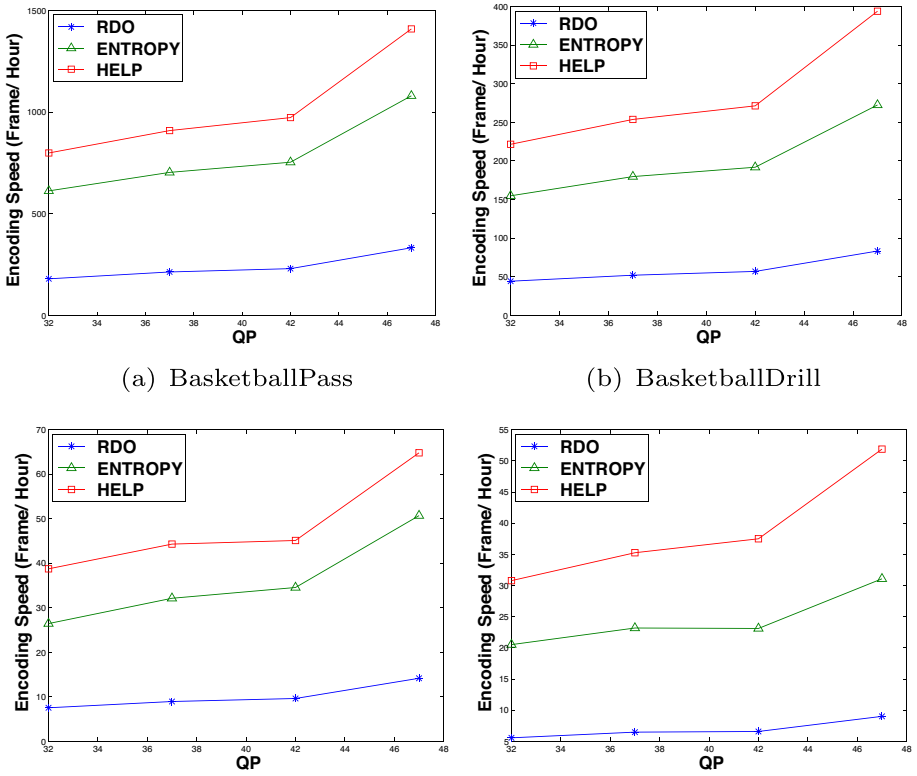
Fig. 10 Comparing different models in terms of bitrate vs. resolution



(a) Ducks_take_off (b) Park_joy

Fig. 11 Comparing different models in terms of PSNR vs. resolution

Our algorithm testing utilized sequences with key features summarized in Table 2, as reported in [43, 44]. To determine our algorithm's thresholds, we selected a subset of these sequences, namely crop_Traffic, Basket_ball_Drill, and Basket_ball_Pass.



(a) BasketballPass (b) BasketballDrill

(c) Tennis (d) Traffic

Fig. 12 Assessing encoding speed vs. QP for various algorithms

6 Results and analysis

In this section, we assess the performance of three algorithms, RDO, ShannonEntropy, and iHELP, on two video sequences, Ducks_take_off and Park joy, at resolutions of 720p, 1080p, and 2160p. Throughout the results, we use the abbreviation ShannonEntropy or Entropy interchangeably to refer to the entropy-based algorithm proposed by Zhang et al. [26]. The encoding speed, bitrate, and PSNR results are presented in Figs. 9, 10, and 11, respectively.

The findings indicate that the iHELP model offers a significant speed improvement over RDO, with an average speed enhancement of 4.31 (equivalent to 5.31 times faster), compared to 2.72 for ShannonEntropy. However, there is no considerable increase in bitrate when utilizing the iHELP model, and the decrease in PSNR is negligible. Overall, the iHELP model delivers better coding efficiency and quality than ShannonEntropy.

Figure 12 illustrates the encoding speed for different quantization parameters ($QP = 32, 37, 42, 47$). The figure shows that iHELP's encoding speed is five times faster than RDO's for all sequences and all QPs, on average. ShannonEntropy's encoding speed is 3.5 times faster than RDO's. However, iHELP and ShannonEntropy have higher ESEs than RDO, with iHELP being 4 times higher and ShannonEntropy 2.5 times higher.

Figure 13 illustrates the difference in coding efficiency between iHELP and ShannonEntropy, as seen in their respective bitrates at different quantization parameters ($QP =$

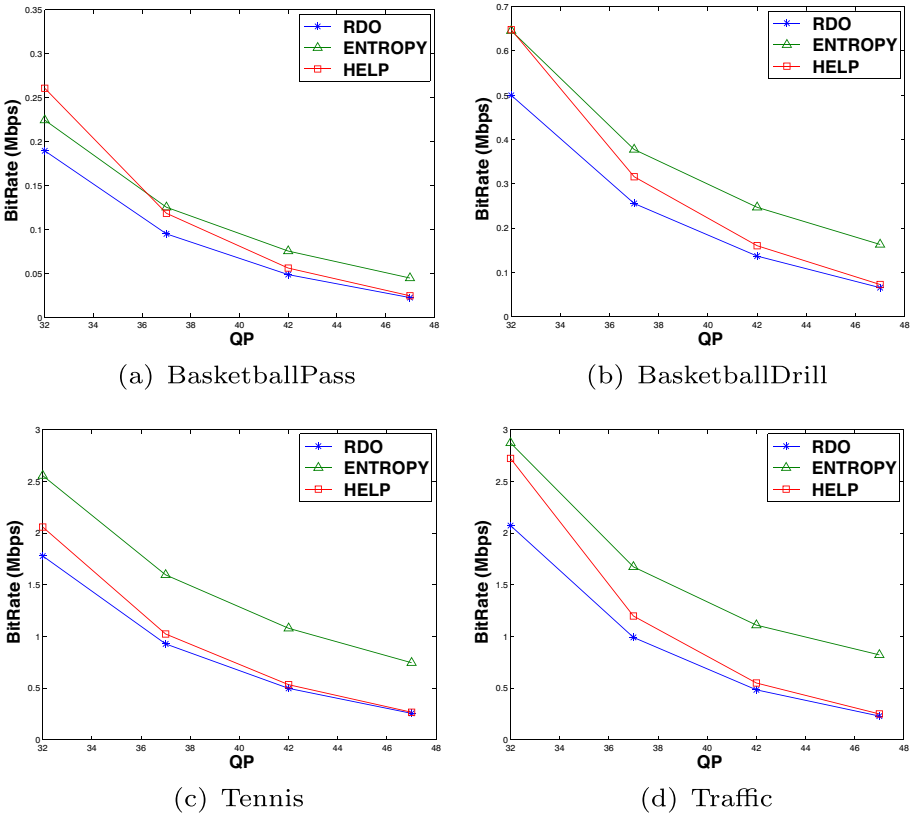


Fig. 13 Comparing different models in terms of bitrate vs. QP

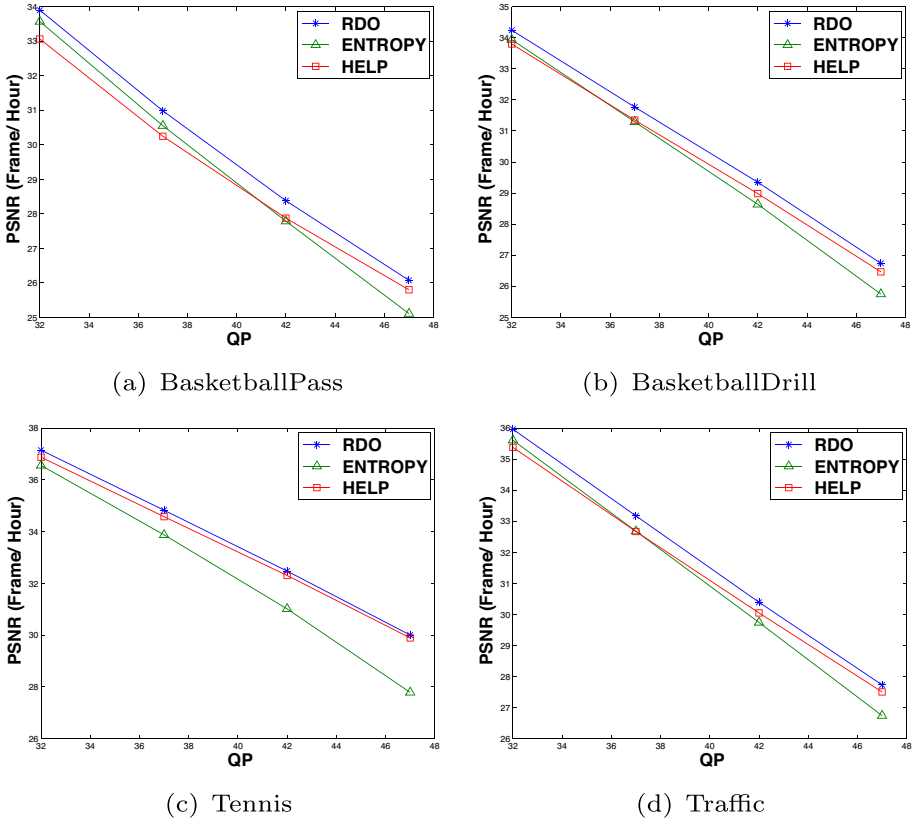


Fig. 14 Evaluating PSNR versus QP across various algorithms

(32, 37, 42, 47)). The figure clearly demonstrates that iHELP outperforms ShannonEntropy in terms of coding efficiency.

Figure 14 illustrates a comparison of the quality of iHELP and ShannonEntropy algorithms at various quantization parameters. The results indicate that iHELP generally outperforms ShannonEntropy in terms of quality across most sequences and quantization parameters. Specifically, at most QPs and in most sequences, iHELP produces higher PSNR values than ShannonEntropy.

We assessed multiple partitioning algorithms, comparing their encoding speed for varying QP values. Figure 15 illustrates the encoding speed versus bitrate for $QP = (32, 37, 42, 47)$. Our analysis shows that, on average, the iHELP model exhibits an encoding speed that is roughly 1.5 times faster than ShannonEntropy.

Figure 16 shows the quality versus bitrate of four sequences with different quantization parameters ($QP = (32, 37, 42, 47)$). The results suggest that the iHELP model outperforms ShannonEntropy across all sequences and most of the QP values, in terms of quality.

Figure 17 displays the Y-PSNR, U-PSNR, V-PSNR, and BD-PSNR values of the ShannonEntropy to RDO, iHELP to RDO, and iHELP to ShannonEntropy algorithms for the Tennis sequence, with QP values of (32, 37, 42, 47). The figure illustrates that iHELP maintains comparable quality and bitrate to RDO, and also outperforms ShannonEntropy in both quality and coding efficiency.

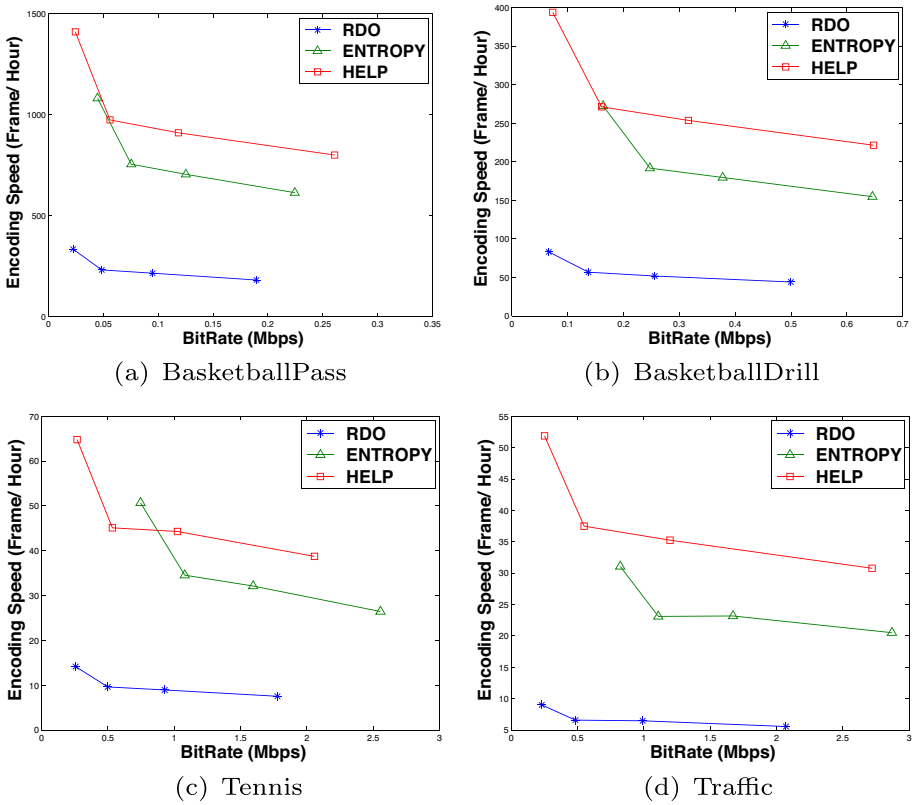


Fig. 15 Different algorithms comparison in terms of encoding speed vs. bitrate @ QP = 32, 37, 42, 47

Figure 18 presents a comparison of the encoding speed of the iHELP, ShannonEntropy, and RDO algorithms for the Tennis sequence, on three different computers. The results suggest that, in terms of encoding speed, iHELP outperforms ShannonEntropy on all three machines. However, the figure does not display the bitrate and quality, which remained consistent across all three computers.

Based on a combination of 46 different sequences and QP values, iHELP achieves an Encoding Speed Enhancement (ESE) of 4, indicating that its encoding speed is approximately five times faster than that of the RDO algorithm, on average. Similarly, ShannonEntropy achieves an ESE of approximately 2.5 over RDO.

Additionally, in terms of Bjøntegaard Delta-PSNR (BD-PSNR), iHELP outperforms ShannonEntropy by an average of 1.416, and it achieves a lower average Bjøntegaard Delta-rate (BD-rate) of 34.25. Equation (5) is used to calculate the encoding speed enhancement..

6.1 Comparative analysis

We conducted a comparative analysis of the proposed iHELP methodology against traditional methodologies (TnB, Hybrid2, and RDO). The analysis focused on key metrics: Encoding Time, Bitrate, Y-PSNR, and Encoding Speed Enhancement (ESE).

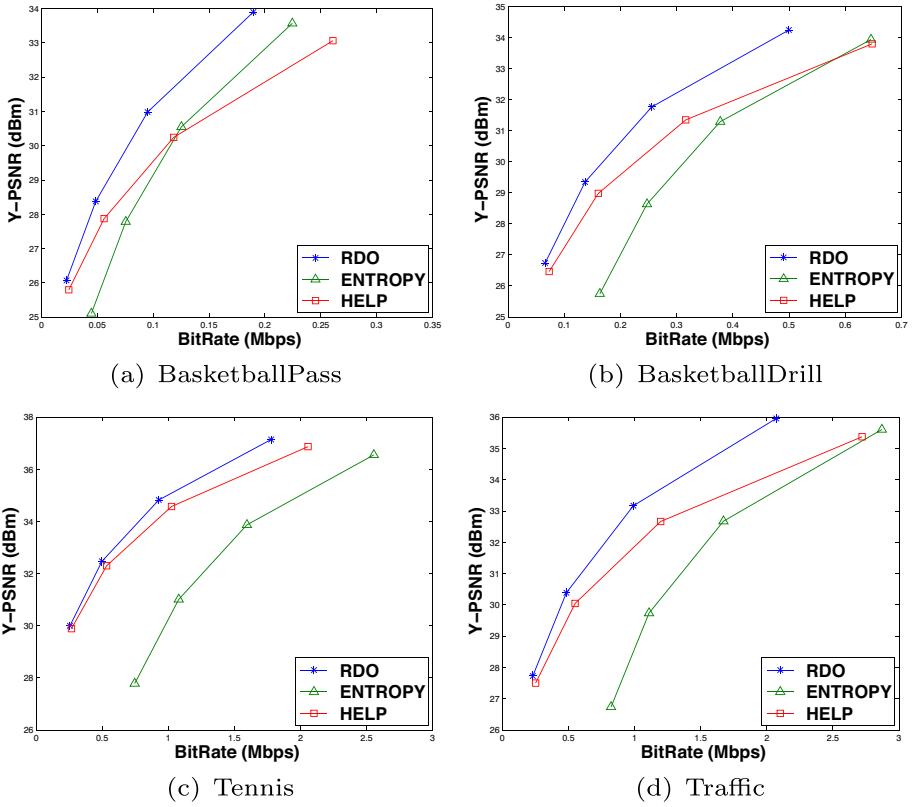


Fig. 16 Different algorithms comparison in terms of PSNR vs. bitrate with @ QP = 32, 37, 42, 47

Encoding speed iHELP consistently demonstrates a higher encoding speed compared to TnB, Hybrid2, and RDO across various sequences and QPs, as reflected in its higher ESE values.

Bitrate While iHELP tends to have a slightly increased bitrate compared to RDO, it maintains good bitrate efficiency relative to TnB and Hybrid2.

Y-PSNR (Quality) iHELP generally maintains comparable video quality to RDO, TnB, and Hybrid2, even with its faster encoding speed, as indicated by Y-PSNR values.

Encoding time reduction iHELP shows substantial improvements in encoding time reduction, surpassing both TnB and Hybrid2.

Summary table The following table summarizes the comparative analysis across different metrics (Table 3):

In conclusion, iHELP emerges as a highly efficient video encoding methodology, offering significant enhancements in encoding speed without substantial sacrifices in bitrate or video quality.

7 Conclusions

iHELP: Pioneering the future of video coding with adaptive instant learning

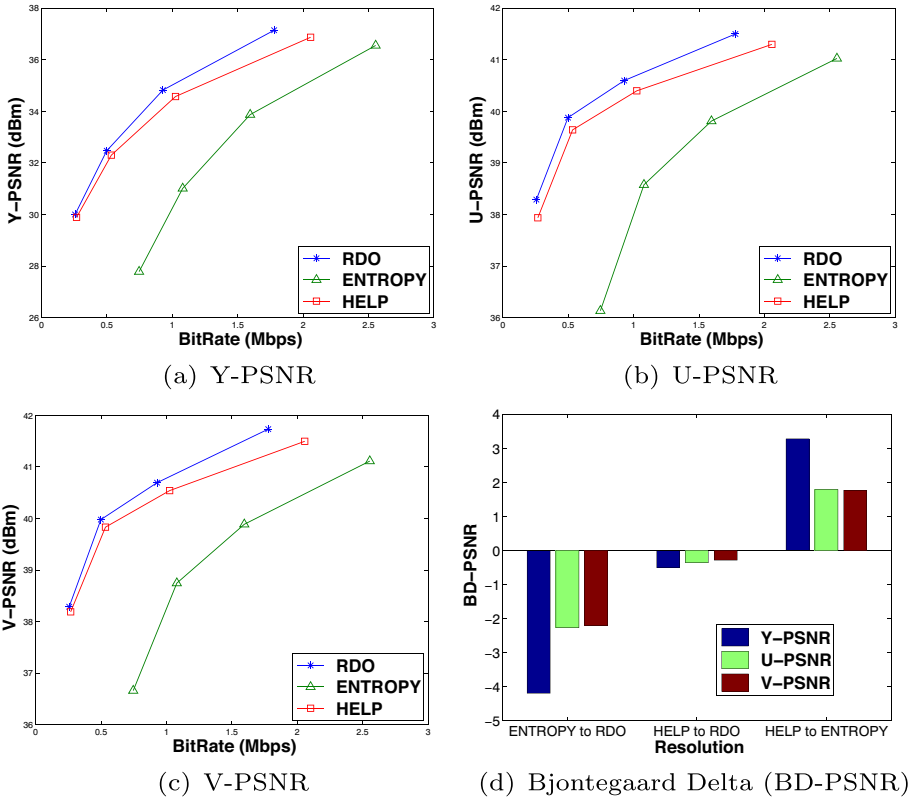


Fig. 17 Comparing different models in terms of YUV-PSNR vs. bitrate at QP = 32, 37, 42, 47

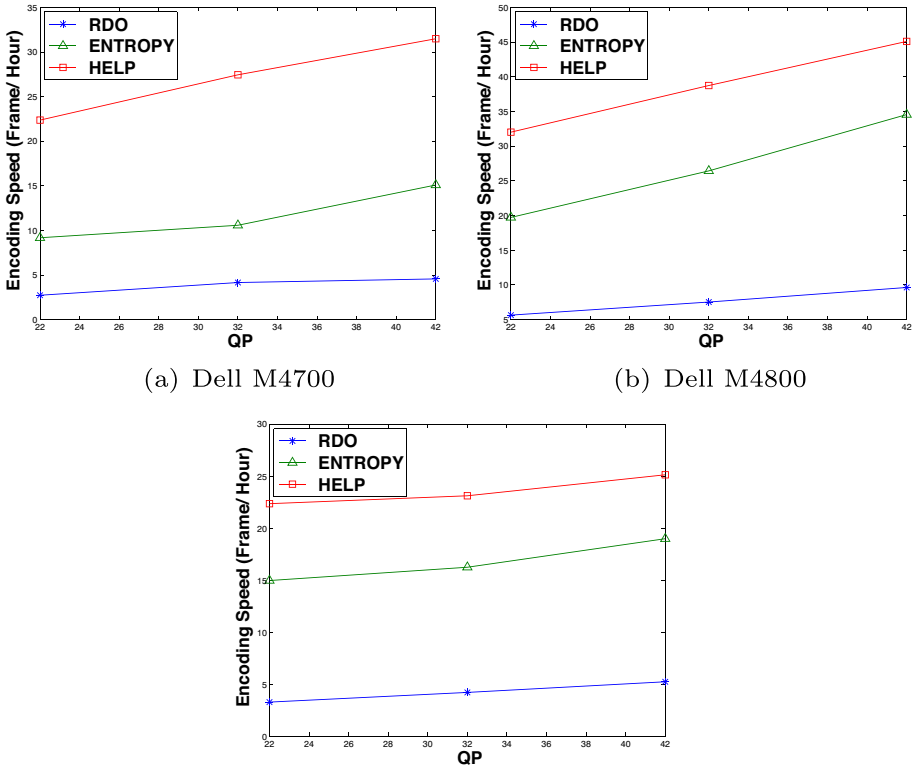
In the dynamic realm of video coding, iHELP stands out as a revolutionary approach, particularly in addressing the complexities associated with adaptive block structures. Our proposed technique, iHELP, signifies a major leap forward, achieving an astounding 80% reduction in encoding time while preserving exceptional video quality.

The essence of iHELP lies in its ingenious utilization of content correlations among spatial and temporal neighbors within video blocks. This methodology facilitates the precise prediction of block sizes, thereby circumventing the need for exhaustive search processes and significantly expediting the encoding procedure without degrading the visual output.

Beyond its remarkable speed, iHELP brings to the table an ingenious implementation of entropy-based conditions. These act as a robust mechanism to forestall error propagation, ensuring consistently accurate predictions across a wide array of video content. This combination of speed and reliability establishes iHELP as a transformative solution in the sphere of video coding efficiency.

As we gaze into iHELP's future, we are excited about its potential applications and enhancements:

- **Integration with deep learning:** The fusion of iHELP with deep neural networks promises to further refine its prediction accuracy and speed, all while maintaining a low computational load.



(a) Dell M4700 (b) Dell M4800

(c) HP EliteBook

Fig. 18 Different algorithms comparisons in terms of encoding speed vs. QP using three computers with different specifications, QP = 22, 32, 42

- **Adaptation to various standards:** Plans are in place to expand iHELP's capabilities to other encoding standards such as AV1, VP9, VVC, and EVC, customizing its approach to suit the unique attributes of each standard and broadening its application scope.
- **Empowering immersive technologies:** iHELP has the potential to revolutionize VR and AR experiences, paving the way for real-time video coding that transcends the current limits of visual immersion.

In conclusion, iHELP is not merely an advancement in video coding; it represents a paradigm shift towards greater efficiency and broader applicability. Its innovative design, outstanding performance, and versatile potential mark it as a foundational technology for the

Table 3 Comparative analysis summary

Metric	iHELP	TnB	Hybrid2	RDO
Encoding Speed (ESE)	Highest	Moderate	Low	Lowest
Bitrate	Slightly increased	Moderate	Moderate	Baseline
Y-PSNR	Comparable	Comparable	Comparable	Baseline
Encoding time reduction	Highest	Moderate	Low	None

future of video communication, leading us into an era of enhanced, efficient, and accessible video experiences.

Author Contributions Yousef Sharrab has performed most of the experiments and the writing. Mohammad Alsmirat collaborated with Yousef to finalize the implementations and the writing of the paper. Mohammad Ali H. Eljinini revised the writing of the paper, Finally, Nabil Sarhan was the PhD supervisor of Yousef and he gave the idea and reviewed the writing.

Funding This work is partially funded by the University of Sharjah grant number 2202150230

Data Availability Some of the data used in this research is publicly available, as referenced within this paper. Newly generated data is not public as it is still under study by the authors.

Declarations

Informed consent All authors are informed about this submission.

References

1. Sharrab Y, Almutiri NT, Tarawneh M, Alzyoud F, Al-Ghuwairi A-RF, Al-Fraihat D (2023) Toward smart and immersive classroom based on AI, VR, and 6G. Int J Emerg Technol Learn (Online) 18(2):4
2. Lee J-H, Lee Y-W, Jun D, Kim B-G (2020) Efficient color artifact removal algorithm based on high-efficiency video coding (HEVC) for high-dynamic range video sequences. IEEE Access 8:64099–64111
3. Li J, Li B, Xu J, Xiong R, Gao W (2018) Fully connected network-based intra prediction for image coding. IEEE Trans Image Process 27(7):3236–3247
4. Liu Y, Liu S, Wang Y, Zhao H (2020) Video coding and processing: a survey. Neurocomputing 408:331–344
5. Alonso JB, Cabrera J, Shyamnani R, Travieso CM, Bolaños F, García A, Villegas A, Wainwright M (2017) Automatic anuran identification using noise removal and audio activity detection. Expert Syst Appl 72:83–92
6. Elrowayati AA, Alrshah MA, Abdullah MFL, Latip R (2020) HEVC watermarking techniques for authentication and copyright applications: challenges and opportunities. IEEE Access 8:114172–114189
7. Shen X, Yu L (2013) CU splitting early termination based on weighted SVM. EURASIP J Image Vid Process 1:1–11
8. Li X, Gong N (2020) Run-time deep learning enhanced fast coding unit decision for high efficiency video coding. J Circ Sys Comput 29(03):2050046
9. Liu D, Li Y, Lin J, Li H, Wu F (2020) Deep learning-based video coding: a review and a case study. ACM Comput Surv (CSUR) 53(1):1–35
10. Yu Q, Zhang X, Wang S, Ma S (2012) Early termination of coding unit splitting for HEVC. IEEE, pp 1–4
11. Sharrab YO, Alsmadi I, Sarhan NJ (2022) Towards the availability of video communication in artificial intelligence-based computer vision systems utilizing a multi-objective function. Clust Comput 25(1):231–247
12. Yan L, Shi Y, Wei M, Wu Y (2023) Multi-feature fusing local directional ternary pattern for facial expressions signal recognition based on video communication system. Alex Eng J 63:307–320
13. Pavlič J, Tomažič T, Kožuh I (2022) The impact of emerging technology influences product placement effectiveness: a scoping study from interactive marketing perspective. J Res Interact Mark 16(4):551–568
14. Mystakidis S (2023) Sustainable engagement in open and distance learning with play and games in virtual reality: playful and gameful distance education in VR. In: Research anthology on virtual environments and building the metaverse, pp 297–312. IGI Global
15. Sharrab YO, Alsmirat M, Hawashin B, Sarhan N (2021) Machine learning-based energy consumption modeling and comparing of H. 264 and Google VP8 encoders. Int J Electr & Comput Eng (2088-8708) **11**(2)
16. Sharrab YO, Alsmira M, Dwekat Z, Alsmadi I, Al-Khasawneh A et al (2021) Performance comparison of several deep learning-based object detection algorithms utilizing thermal images. In: 2021 Second international conference on intelligent data science technologies and applications (IDSTA). IEEE, pp 16–22

17. Sharrab YO, Sarhan NJ (2017) Modeling and analysis of power consumption in live video streaming systems. ACM Trans Multimed Comput Commun Appl (TOMM) 13(4):1–25
18. Wang M, Li J, Zhang L, Zhang K, Liu H, Wang S, Kwong S, Ma S (2019) Extended coding unit partitioning for future video coding. IEEE Trans Image Process 29:2931–2946
19. Brownlee M (2020) H.266, AV1 & MPEG-5 Explained - New Video Codecs for 2020. Retrieved from https://youtu.be/rCS39ibUN-Y
20. Punchihewa A, Bailey D (2020) A review of emerging video codecs: challenges and opportunities. In: 2020 35th International conference on image and vision computing New Zealand (IVCNZ). IEEE, pp 1–6
21. Minopoulos G, Psannis KE, Kokkonis G, Ishibashi Y (2020) QoE assessment of video codecs for video streaming over 5G networks. In: 2020 3rd World symposium on communication engineering (WSCE). IEEE, pp 34–38
22. Minopoulos G, Memos VA, Psannis KE, Ishibashi Y (2020) Comparison of video codecs performance for real-time transmission. In: 2020 2nd International conference on computer communication and the internet (ICCCI). IEEE, pp 110–114
23. Shen L, Zhang Z, Liu Z (2014) Effective CU size decision for HEVC intracoding. IEEE Trans Image Process 23(10):4232–4241
24. Timmerer C, Wien M, Yu L, Reibman A (2021) Special issue on open media compression: overview, design criteria, and outlook on emerging standards. Proc IEEE 109(9):1423–1434
25. Shannon CE (1951) Prediction and entropy of printed English. Bell Syst Tech J 30(1):50–64
26. Zhang M, Qu J, Bai H (2013) Entropy-based fast largest coding unit partition algorithm in high-efficiency video coding. Entropy 15(6):2277–2287
27. Choi K, Jang E (2011) Coding tree pruning based cu early termination. document JCTVC-F092 of JCT-VC, Torino, IT
28. Shannon CE (2001) A mathematical theory of communication. ACM SIGMOBILE Mob Comput Commun Rev 5(1):3–55
29. Tsai D-Y, Lee Y, Matsuyama E (2008) Information entropy measure for evaluation of image quality. J Digit Imaging 21(3):338–347
30. Shannon CE, Weaver W (2015) The Mathematical Theory of Communication. University of Illinois press
31. Hsu W-J, Hang H-M (2013) Fast coding unit decision algorithm for HEVC. In: Signal and information processing association annual summit and conference (APSIPA):2013 Asia-Pacific. IEEE, pp 1–5
32. Jangade J, Babulal KS (2023) Study on deep learning models for human pose estimation and its real time application. In: 2023 6th International conference on information systems and computer networks (ISCON). IEEE, pp 1–6
33. Bairagi PP, Dutta M, Babulal KS (2023) An energy-efficient protocol based on recursive geographic forwarding mechanisms for improving routing performance in WSN. IETE J Res pp 1–13
34. Al-Ghuwairi A-R, Sharrab Y, Al-Fraihat D, AlElaimat M, Alsarhan A, Algarni A (2023) Intrusion detection in cloud computing based on time series anomalies utilizing machine learning. J Cloud Comput 12(1):127
35. Sharrab Y, Al-Fraihat D, Alsmirat M (2023) Deep neural networks in social media forensics: unveiling suspicious patterns and advancing investigations on twitter. In: 2023 3rd Intelligent cybersecurity conference (ICSC). IEEE, pp 95–102
36. Parikh J, Abuchaar O, Haidar E, Kailas A, Krishnan H, Nakajima H, Maile M, Meier J, Rajab S, Sharrab Y et al (2015) Vehicle-to-infrastructure program cooperative adaptive cruise control
37. Tarabin M, Alketbi MM, Alfalasi HR, Alsmirat M, Sharrab Y (2023) Detecting distracted drivers using convolutional neural networks. In: 2023 Fourth international conference on intelligent data science technologies and applications (IDSTA). IEEE, pp 59–66
38. Al-Ghuwairi A-R, Al-Fraihat D, Sharrab Y, Alrashidi H, Almujally N, Kittaneh A, Ali A (2023) Visualizing software refactoring using radar charts. Sci Rep 13(1):19530
39. Sheikh HR, Sabir MF, Bovik AC (2006) A statistical evaluation of recent full reference image quality assessment algorithms. IEEE Trans Image Process 15(11):3440–3451
40. Bjontegaard G (2008) Improvements of the BD-PSNR model. In: ITU-T SG16/Q6, 35th VCEG Meeting, Berlin, Germany, July, 2008
41. Bossen F (2012) Common test conditions and software reference configurations. document JCTVC-H1100 of JCT-VC, San Jose, CA, USA
42. Radicke S, Hahn J-U, Wang Q, Grecos C (2016) A parallel HEVC intra prediction algorithm for heterogeneous CPU+ GPU platforms. IEEE Trans Broadcast 62(1):103–119
43. Martin R, Lina SPFFK, Tatjana M. Video trace library. http://trace.eas.asu.edu/yuv/
44. Wong KS. Video traces. http://web.fsktm.um.edu.my/koksheik

Authors and Affiliations

Yousef O. Sharrab[1,2] · Mohammad A. Alsmirat[3,4] · Mohammad Ali H. Eljinini[1] · Nabil J. Sarhan[2]

Yousef O. Sharrab
sharrab@iu.edu.jo

Mohammad Ali H. Eljinini
ma.eljinini@iu.edu.jo

Nabil J. Sarhan
nabil@wayne.edu

1 Department of Data Science and Artificial Intelligence, Isra University, Amman, Jordan

2 Deep Learning Lab, ECE Department, Wayne State University, Detroit, MI, USA

3 Department of Computer Science, University of Sharjah, Sharjah, UAE

4 Department of Computer Science, Jordan University of Science and Technology, Ar-Ramtha, Jordan