A Research on Enhancing Reconstructed Frames in Video Codecs

PHAM DO, Kim Chi

(開始ページ / Start Page)
1
(終了ページ / End Page)
122
(発行年 / Year)
2022-09-15
(学位授与番号 / Degree Number)
32675甲第554号
(学位授与年月日 / Date of Granted)
2022-09-15
(学位名 / Degree Name)
博士（工学）
(学位授与機関 / Degree Grantor)
法政大学 (Hosei University)
(URL)
https://doi.org/10.15002/00025871

Doctoral Dissertation Reviewed by
Hosei University

A Research on Enhancing Reconstructed Frames in Video Codecs

PHAM Do Kim Chi

Abstract

A series of video *codecs*, combining encoder and decoder, have been developed to improve the human experience of video-on-demand: higher quality videos at lower bitrates. Despite being at the leading of the compression race, the High Efficiency Video Coding (HEVC or H.265), the latest Versatile Video Coding (VVC) standard, and compressive sensing (CS) are still suffering from lossy compression. Lossy compression algorithms approximate input signals by smaller file size but degrade reconstructed data, leaving space for further improvement. This work aims to develop hybrid codecs taking advantage of both state-of-the-art video coding technologies and deep learning techniques: traditional non-learning components will either be replaced or combined with various deep learning models. Note that related studies have not made the most of coding information, this work studies and utilizes more potential resources in *both encoder and decoder* for further improving different codecs.

In the *encoder*, motion compensated prediction (MCP) is one of the key components that bring high compression ratios to video codecs. For enhancing the MCP performance, modern video codecs offer interpolation filters for fractional motions. However, these handcrafted fractional interpolation filters are designed on ideal signals, which limit the codecs in dealing with real-world video data. This proposal introduces a deep learning approach for all Luma and Chroma fractional pixels, aiming for more accurate motion compensation and coding efficiency.

One extraordinary feature of CS compared to other codecs is that CS can recover multiple images at the *decoder* by applying various algorithms on the one and only coded data. Note that the related works have not made use of this property, this work enables a deep learning-based compressive sensing image enhancement framework using multiple reconstructed signals. Learning to enhance from multiple reconstructed images delivers a valuable mechanism for training deep neural networks while requiring no additional transmitted data.

In the encoder and decoder of modern video coding standards, in-loop filters (ILF) dedicate the most important role in producing the final reconstructed image quality and compression rate. This work introduces a deep learning approach for improving the handcrafted ILF for modern video coding standards. We first utilize various coding resources and present novel deep learning-based ILF. Related works perform the rate-distortion-based ILF mode selection at the coding-tree-unit (CTU) level to further enhance the deep learning-based ILF, and the corresponding bits are encoded and transmitted to the decoder. In this work, we move towards a deeper approach: a reinforcement-learning based autonomous ILF mode selection scheme is presented, enabling the ability to adapt to different coding unit (CU) levels. Using this approach, we require no additional bits while ensuring the best image quality at local levels beyond the CTU level.

While this research mainly targets improving the recent video coding standard VVC and the sparse-based CS, it is also flexibly designed to adapt the previous and future video coding standards with minor modifications.

Acknowledgements

This dissertation presents the research conducted during the years of 2019-2022 at the Graduate School of Science and Engineering, Hosei University. During the time of carrying out this research, I have received great support from the beloved people I would like to thank.

First and foremost, I would like to express my sincere gratitude to Professor Jinjia Zhou for her enthusiasm supervising, and encouragement from day one. I am very grateful and proud of the knowledge and skills I have learned from her. There is no doubt that her scientific feedback and recommendations have brought my research to a higher level. I also thank her for supporting and being at my side in many aspects of life so that I could fully focus on the research.

I would like to thank Professor Koichi Ogawa and Professor Hitoshi Iyatomi for the feedback they provided as part of my dissertation committee. Your suggestions brought my dissertation to a higher level than it would have been.

I will forever be thankful to Professor Kazuo Yana. He is one of the most humble and kind-hearted people I know and you never forget once you meet him. My research and life in Japan would never be that nice without his kind help and support. I would also thank the Graduate School Section of Hosei University, Koganei campus for providing the quick answer and kind assistance.

I would also send my special thanks to my laboratory members and alumni: Muchen Li (who is friendly, helpful and knowledgeable in video coding and Japanese life-related stuff), Xu Jiayao (aka Yao, who is cute, warm-hearted and I am so proud of you for the one you have become), Man M. Ho, Hoang M. Trinh, Xiao Mingjie and Ho Tan Nguyen (friendly and funny friends who lighten my boring days), Thuy T. T. Tran and Huyen T. T. Bui (my fashionable friends), Jian Yang, Sato and Morita (nice guys who have always provide kind help and listen to my problems, good luck at research!).

I thank all of my friends (too many to be listed out but you are here) and special

people who were so fun to be around: Mr. Quan (who is smart, gentle, and supports me in everything), Ms. Vy (who always acts quickly when I need help, who is also smart and has solid knowledge, so proud of her!), Ms. Tham and Mr. Duy (who have sympathetic ears). Their wise counsel apparently motivate me a lot.

Last but not least, I thank my wonderful mom, dad, and sister for their unconditional and endless love. You are always the reason I keep going on every day. Special thanks to my best friend, soul-mate and boyfriend Nam for being there all the way. I could not make it without their backup. I am so blessed for being a child, a student, and a friend of you!

List of Publications

Journal papers

[1] **Chi Do-Kim Pham**, Jian Yang and Jinjia Zhou, "CSIE-M: Compressive Sensing Image Enhancement Using Multiple Reconstructed Signals for Internet of Things Surveillance Systems," in IEEE Transactions on Industrial Informatics, vol. 18, no. 2, pp. 1271-1281, Feb. 2022, DOI: 10.1109/TII.2021.3082498.

[2] **Chi Do-Kim Pham** and Jinjia Zhou, "Deep Learning-Based Luma and Chroma Fractional Interpolation in Video Coding," in IEEE Access, vol. 7, pp. 112535-112543, 2019, DOI: 10.1109/ACCESS.2019.2935378.

Conference papers

[1] **Chi Do-Kim Pham**, Chen Fu and Jinjia Zhou, "Deep Learning based Spatial-Temporal In-loop filtering for Versatile Video Coding," 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2021, pp. 1861-1865, DOI: 10.1109/CVPRW53098.2021.00206.

[2] **Chi Do-Kim Pham**, Jinjia Zhou, "A Convolutional Neural Network for Fractional Interpolation in Video Coding," The International Symposium on Artificial Intelligence and Robotics (ISAIR 2019), Daegu, Korea, Aug. 2019.

[3] Thuy Thi Thu Tran, Jirayu Peetakul, **Chi Do-Kim Pham**, Jinjia Zhou, "Bi-directional intra prediction based measurement coding for compressive sensing images", 2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP), DOI: 10.1109/MMSP48831.2020.9287074, Sep. 2020.

[4] Ho Tan Nguyen, **Chi Do-Kim Pham**, Jinjia Zhou, "SpeedDeblur: A Framework to speed up CNN-based Deblurring for HEVC compressed video," 2021 IEEE 23rd

International Workshop on Multimedia Signal Processing (MMSP).

[5] Jian Yang, **Chi Do-Kim Pham**, Jinjia Zhou, "JVCSR: Video Compressive Sensing Reconstruction with Joint In-loop Reference Enhancement and Out-loop Super-resolution," The 28th International Conference on Multimedia Modeling (MMM), Qui Nhon, Vietnam, April 2022.

[6] Jiayao Xu, **Chi Do-Kim Pham**, Chen Fu, Jinjia Zhou, "A 81.92Gpixels/S Fast Reconstruction of Images from Compressively Sensed Measurements", 2022 IEEE International Symposium on Circuits & Systems (ISCAS).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since the late twentieth century, the world has witnessed a significant change in visual communication. Video, one of the most costly multimedia data, is now occupying 82% internet traffic [3]. The most noticeable elements within a video sequence are video frame rate and resolution. Video frame rates have been growing from 24 frames per second (FPS) to up to 300 FPS performed by modern monitors. Video resolution refers to the number of pixels on display, has been significantly increased from Standard Definition (SD) to High Definition (HD). It is now reported that up to 8K resolution has been widely supported by the modern displays. Along with the growth of frame rate and display resolutions, modern visual communication systems require higher effective video compression algorithms. Video codec, a computing technology comprised of compression and decompression algorithms, offers a smaller data volume for storage and transmission while delivering a same video quality of raw video (as shown in Fig. 1.1).

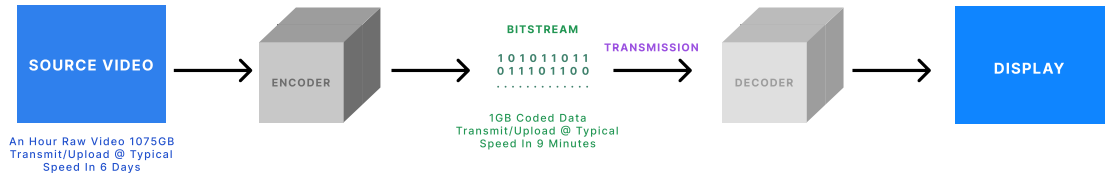


Figure 1.1: Image and video coding and transmission scenario. Raw images and videos sensed by cameras are encoded into smaller data files which are more friendly for storage and transmission through the internet.

Following the success of image coding standard by the Joint Photographic Experts Group (JPEG) [4] and JPEG 2000 [5], a series of video coding standards [2, 6–12] has been established on both spatial and temporal compression. Common video cod-

ing standards and their performance have been shown in Fig. 1.2. Advanced Video Coding (AVC, as also known as H.264 or MPEG-4 Part 10) [11], which was developed by the Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC JTC1 Moving Picture Experts Group (MPEG), is the commonly used video coding standard that can be played on almost any device since 2003. AVC can provide a compression ratio up to 2000:1, corresponding to 1Mbps after compressing a 2Gbps video. In 2013, High Efficiency Video Coding (HEVC) [12], also known as H.265 and MPEG-H Part 2, is standardized and achieves up to 50% better coding performance compared to its predecessor H.264/AVC. Compared to AVC, remarkable changes and updates have been made. In AVC, the processing units are macroblocks while they are coding tree units (CTUs) and coding units (CU) in HEVC. In spatial-based coding, HEVC increases intra-prediction modes to 35 [13] from nine modes supported by AVC. In temporal coding, more complex inter-prediction has been proposed. The JCT-VC has improved the fractional interpolation filters for 1/4 pixels, aiming for better motion compensation [14]. At the decoder, HEVC adopts Sample Adaptive Offset (SAO) [15] besides the deblocking filters [16] inherited from AVC. Later on, Google developed VP9 [9] and AV1 [10] relied on the VP8 [8], aiming to compete HEVC and supporting open-source and royalty-free. Released in 2020, the video coding standard Versatile Video Coding (H.266/VVC) [2] reduces the compressed file size of the HEVC by 50% at the same image quality. This high coding efficiency comes from the following main improvements. First, the number of intra-prediction modes has now supported up to 81 direction modes [17]. Second, 1/16 pixel Luma motion vectors (MV) are supported, compared to 1/4 MV in HEVC. Decoder-side MV refinement and Affine transform are also applied for better inter-coding. In VVC in-loop filter, luma mapping with chroma scaling (LCMS) and adaptive loop filtering (ALF) are additionally introduced beside HEVC Deblocking filters and SAO [18].

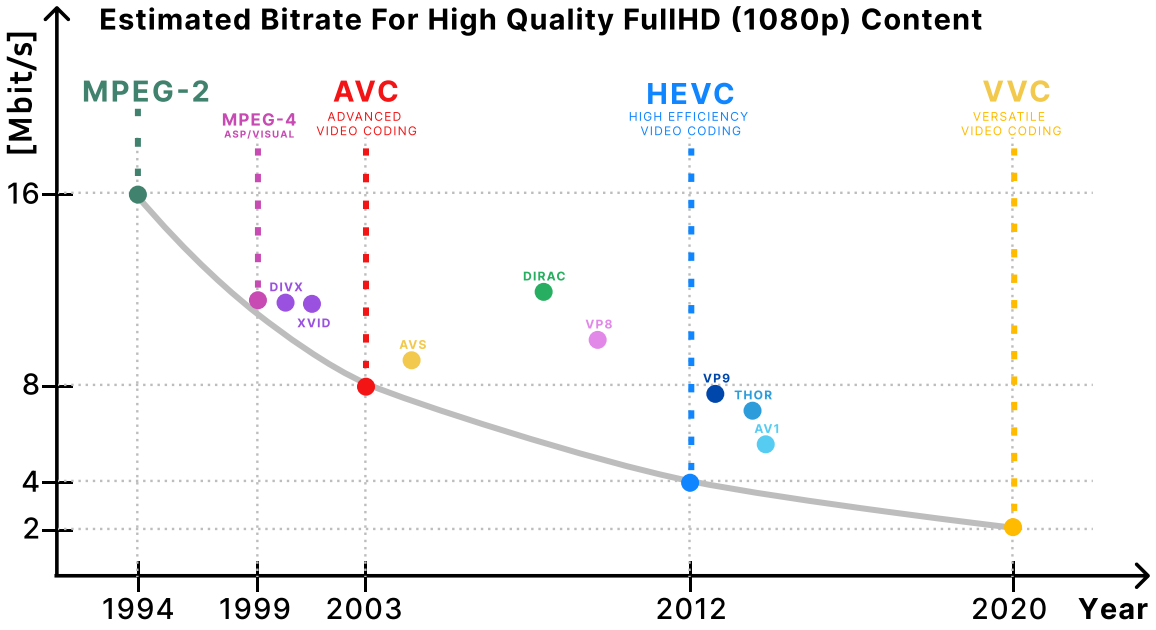


Figure 1.2: Coding performance comparison of different coding standards over the years.

Although obtaining high coding efficiency, current video coding standards suffer from high complexity, which mainly extends encoding speeds. In 2006, Donoho introduced the term compressive sensing (CS) [19] based on sparsity and incoherence. Building upon the fact the signals are sparse in some transformed or original domain, CS can efficiently compress and approximate nearly original signals from a few non-zero samples. CS simply performs matrix multiplications in the encoder, which is computing-friendly to be integrated into low-resource devices. Sampling data from the encoder, also known as measurements, is transmitted to the decoder with a much lower amount than the original signal. In the decoder, reconstructing the input signal is to solve the underdetermined problem, which requires very high computational complexity. Investigating CS reconstruction algorithms is a continuing concern within the field of signal processing. Current CS recovery strategies are mainly divided into three approaches: greedy algorithms, convex optimization, and gradient-based algorithms. Greedy algorithms solve the CS reconstruction problem by minimizing the least-square error iteratively. Commonly used greedy algorithm-based CS recovery includes Or-

thogonal e Matching Pursuit (OMP) [20], Sparsity adaptive matching pursuit (SAMP) [21], Compressive Sampling Matching Pursuit (CoSaMP) [22]. In the convex optimization approach, ℓ_p-(quasi)-norm is relaxed with p$\geq$0. The common implementations for solving this problem includes Basis Pursuit [23], Basis Pursuit De-Noising (BPDN) [23], modified BPDN [24], Least Absolute Shrinkage and Selection Operator (LASSO) [25], L1 equality constraints via primal-dual algorithm (L1EQPD) [26] and Spectral Projected-Gradient for L1 (SPGL1) [27].

1.1 Motivations

Despite the fact that coding standards and algorithms have been continuously enhanced, *there is always space for further improving these lossy coding approaches*. Deep learning and its outstanding representative Convolutional Neural Network (CNN) have been receiving significant attention in recent years. CNNs have been widely used and obtained remarkable results in image and video processing. Convolutional Neural Network (CNN), a most representative model of deep learning, well improves the performance of the traditional method in high-level computer vision such as classification [28], detection[29, 30] to low-level computer vision tasks like image denoising [31], and super-resolution. SRCNN [32], a very first CNN model in learning-based super-resolution, learns the mapping between an input of low resolution-image and output of the high-resolution image, outperforms the traditional method bicubic. The work [33] aims to learn image detail on a 20-layer CNN model VDSR to improve the quality of low-resolution input. Following these successes of CNN-based image super-resolution, Zhang *et al.* have demonstrated the effectiveness of CNN in common image restoration tasks, including image denoising, enhancement, artifact removal [31]. Dong *et al.* proposes four-layer Artifacts Reduction Convolutional Neural Network (ARCNN) [34] for improving quality of JPEG images. In [35], the authors leverage JPEG coding information to improve the reconstructed image in both pixel and transform domains.

Pointing out that PSNR does not fully reflect the performance of reconstructed images, the work [36] proposes a one-to-many network trained by a combination of perceptual loss, naturalness loss, and JPEG loss for producing a series of artifact-free candidates. Current end-to-end deep video coding tools, where deep neural networks are used as video coding frameworks [37, 38], are not widely used in visual applications. First, the performance of deep coding tools mostly depends on training data. Second, the encoder and decoder in video coding are very complicated and need to be well designed for tiny blocks. The deep networks are insufficient to represent the complex encoding and decoding components. Finally, their coding performance cannot exceed the recent video coding standards such as VVC and AV1 in various coding configurations/conditions. Regarding to video codec components, deep learning technologies have been applied to intra and inter predictions, probability prediction, transforming, and coding optimizations. Li *et al.* [39] predict the pattern of an $N \times N$ intra block given the already reconstructed neighboring blocks. Cui *et al.* [40] propose a CNN architecture for refining the HEVC intra-predicted block, obtaining 0.70% bitrate reduction compared to HEVC. Referring to inter-coding, inter-predicted block refinement is also one of the active topics. After motion compensated prediction (MCP), several CNN architectures [41, 42] have been introduced for further improving the inter-predicted blocks. In these works, the predicted blocks are input to the proposed networks with the already reconstructed neighboring blocks, fully exploiting the spatial and temporal information.

In video *encoder*, motion compensation is one of the key components that significantly reduce the temporal redundancy during the inter-coding. In motion compensation, fractional interpolation filters are introduced for enhancing the reference frames for better predictions of the current frame. Several deep learning-based fractional interpolations [43–49] have been proposed for enhancing the Discrete Cosine Transform-based interpolation filters (DCTIF) in HEVC. To handle the problem of

changing integer pixel after convolution, [43] first proposes three CNN models CNNIF_H, CNNIF_V, and CNNIF_D for horizontal, vertical, and diagonal half-pixel positions, respectively. By producing three half-pixel independently, they keep integer pixels for the later process of HEVC. Later on, Zhang *et al.* introduces a CNN model followed by a Constrained mask with different weights for the integer pixels and three half pixels[44]. Similar to [43], Yan *et al.* trains 15 models for three half samples and 12 quarter samples in [47]. In [45], half and quarter pixels share nine convolution layers and be separately produced in group variational transformation in GVTCNN. Similarly, the work [49] designs switch mode based fractional interpolation to reduce the drawback of the motion shift in [45]. Liu *et al.* implements GVCNN that supports sub-pixel (half-pixel or quarter-pixel) under different QPs in a model [48]. Although prior research generally confirms that CNN-based fractional interpolation improves coding performance, there are some drawbacks that could be improved in these approaches: only half-pixel are supported [43, 44], many models need to be trained for fractional positions [44], or predicting fractional pixel from integer pixel may not good because the motion shift between integer and fractional pixel are not always stable [43, 45, 48, 49].

In video *decoder*, deep learning-based reconstructed image enhancement has received much attention. Yang *et al.* propose Quality Enhancement Convolutional Neural Networks, including QE-CNN-I and QE-CNN-P [50] that are able to handle intra-coding distortions and inter-coding distortions. Jin *et al.* propose a Multi-level Progressive Refinement Networks through an adversarial training approach (MPRGAN) [51] that focuses on enhancing intra-coding frames. During feed-forward, MPRGAN generates multi-level residual in a coarse-to-fine fashion, allowing adjustments to be suitable for different resource-aware applications. In [52], Wand *et al.* propose Multi-Scale Deep Decoder (MSDD), making use of multi-scale similarity from the decoder to enhance the coding efficiency. He *et al.* [53] propose a partition-masked Convolution Neural Network based on an attention learning scheme to guide to quality enhance-

ment process. The authors of [54] propose a residual-based video restoration network (Residual- VRN) making use of the coded prediction residual to enhance the decoded frames. Tsai *et al.* predict residuals between AVC compressed videos and original videos. Huffman coding [55] is then applied for compressing the predicted residual in a lossless manner. At the decoder, AVC reconstructed videos and decoded residuals are combined for the final output video. Apart from the general video codecs, CS approximates the original signal by a small number of non-zero entries. Decoding CS coded data is to solve an underdetermined problem where a set of reconstructed signals can be obtained given different algorithms. This property enables a potential resource in training CNN models, pointed out by none of the existing works.

In improving video *encoder and decoder*, CNN-based in-loop is the most active fields. It is reported that more challenges are required in designing ILF because reference frames are changed in-looply. Meng *et al.* [56] propose a multi-channel long-short-term dependency residual network (MLSDRN) that builds upon the idea of information storage and information update function of the human memory cell for improving HEVC ILF. Kang *et al.* [57] introduce multi-modal/multi-scale convolutional neural network (MMS-net) replacing the DBF and SAO in HEVC. In [58], the authors introduce a practical convolutional neural network filter (CNNF) in which inputs are both decoded frame and QP for intra-frame loop filter. In [59], two QP ranges of 20 to 29 and 30 to 39 are trained in different CNN models for replacing HEVC DBF and SAO. In [60], a residual highway CNN (RHCNN) are trained in different QP ranges. RHCNN is also trained for different frame types, including Intra (I), Predicted (P), and Bidirectional predicted (B) frames trading off the quality enhancement performance and training complexities. Although high-coding performance are recorded, the existing deep learning-based ILF works do not make full use of spatial and temporal information in the encoder and decoder, leaving rooms for further improvements.

1.2 Objective and scope of the research

The overall goal of this research is to study deep learning methods for enhancing reconstructed frames, therefore, improving the performance of video codecs including VVC [2], HEVC [12], and the sparsity-based coding CS [19]. Moving towards this goal, this research conducts a comprehensive review and proposes improving video encoder and decoder using different deep learning techniques. Our work focuses on improving the quality of reconstructed images and minimizing the transmitted rates, uncovering an intelligent encoder-decoder for modern visual-enabled applications.

1.3 Enhancing reconstructed frames in video codecs: the overall scheme

Addressing problems in section 1.1, this work focuses on improving reconstructed frames, including reference frames and the current frame generated by the modern codecs, and solves remaining problems in related works, aiming for better coding efficiency. In this dissertation, we propose a novel deep learning-based *enhancing reconstructed frames for video codecs*. The proposal (Fig. 1.3) consists of three main parts, and each is specifically designed for either encoder or decoder.

First, a deep learning-based fractional interpolation filter is proposed, aiming for enhancing the reference frames in video encoders. Inter-coding techniques such as motion estimation and motion compensation are the keys to reducing temporal redundancy in video encoding. During encoding, they search and predict current frames from already reconstructed frames in a list of reference frames. Since H.264, a series of fractional interpolation filters were introduced to generate sub-sample positions from the reference frames. These sub-sample positions are ideal candidates for predicting current frames. Although more accurate motions can be obtained, the designed filters

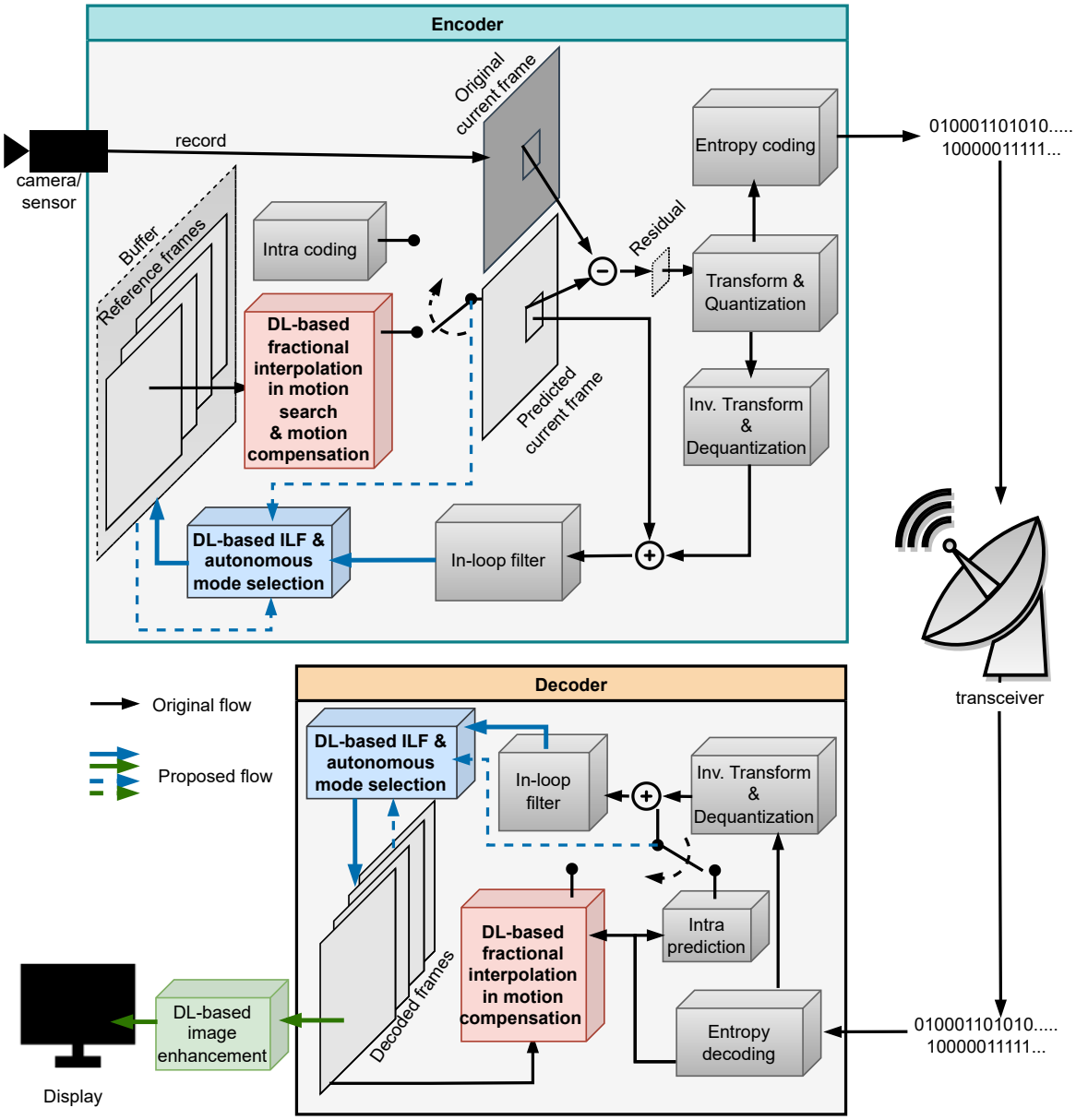


Figure 1.3: The proposals (in color) in video coding system includes three main parts. First, a deep learning-based fractional interpolation (in pink) is proposed in video encoder, aiming to enhance the motion search and motion compensation of inter-coding. Second, a deep learning-based in-loop filtering (in blue) with autonomous mode selection is introduced to enhance the coding efficiency of both encoder and decoder. Finally, a deep learning-based image enhancement (in green) is proposed for enhancing reconstructed images in the decoder.

are restricted in the filtered area and limited in dealing with diverse signals. This work introduces a CNN approach for both Luma and Chroma fractional interpolation, offering high-quality sub-sample candidates for compensating the current frame in both

brightness and color space. In addition, a switch mode-based fractional interpolation filters is proposed, optimizing rate and distortion to CU levels.

Second, we investigate deep into different CS reconstruction algorithms and propose a deep learning approach for enhancing reconstructed frames in CS-decoder systems. This proposal takes advantage of the CS sparsity feature: the ability to reconstruct multiple signals given the same coded measurement. The diversity of reconstructed images is a potential resource for CNN in generating higher quality images, addressed by none of the current works. By this approach, this work is the first to fully use the spatial information generated by different algorithms while maintaining the lowest amount of data to be transmitted to the decoder.

Third, we introduce a deep learning-based in-loop filtering (ILF) enhancement for improving the reconstructed image quality, hence, increasing coding performance in the video encoder and decoder. The existing deep learning-based ILF enhancement works mainly focus on learning the one-to-one mapping between the reconstructed and the original video frame, ignoring the potential resources at encoder and decoder. This work introduces a deep learning-based Spatial-Temporal In-Loop filtering (STILF) that takes advantage of the coding information to improve VVC in-loop filtering. In STILF, we propose a self-enhancement Convolutional neural network with coding unit (CU) map (SECUM) and the reference-based enhancement CNN with the optical flow (REOF). Besides minimizing the distortion, this research also targets optimizing the rate during compression, bringing a comprehensive coding scheme for video coding standards in general and the latest video coding standard VVC in particular. A reinforcement learning-based technique is proposed to further enhance reconstructed images to the local levels, offering the optimal rates and distortion for storage and transmission.

1.4 Main results and contributions

This research focuses on enhancing reconstructed frames of the three most common video codecs, including compressive sensing, HEVC, and VVC video coding. The manuscript is comprised of three main proposals presented in the following chapters, in which the contributions for each work are presented. The rest of this section gives brief introductions and a summary of the contributions and results of each proposal.

1.4.1 Enhancing reference-frame interpolation for video encoder

Motion-compensated prediction is one of the essential methods to reduce temporal redundancy in inter coding. In video encoder, the target of motion-compensated prediction is to predict the current frame from the list of reference frames. Recent video coding standards commonly use interpolation filters to generate sub-pixels of the reference frame. These sub-pixels are candidates for motion compensation. However, the fixed filters are not flexible enough to adapt to the variety of natural video contents. Inspired by the success of Convolutional Neural Network (CNN) in super-resolution, we propose CNN-based fractional interpolation for Luminance (Luma) and Chrominance (Chroma) components in motion-compensated prediction to improve the coding efficiency. Moreover, two syntax elements indicate interpolation methods for the Luminance and Chrominance components, have been added to *bin-string* and encoded by CABAC using regular mode. As a result, our proposal gains 2.9%, 0.3%, 0.6% Y, U, V BD-rate reduction, respectively, under low delay P configuration.

1.4.2 Compressive sensing image enhancement at video decoder

Artificial Intelligence of Things (AIoT) has brought artificial intelligence (AI) to the cutting-edge Internet of Things (IoT). In recent years, Compressive sensing (CS), which relies on sparsity, has been widely embedded and expected to bring more energy efficiency and a longer battery lifetime to IoT devices. Different from the other image compression standards, CS can get various reconstructed images by applying different reconstruction algorithms on coded data. Using this property, it is the first time to propose a deep learning-based compressive sensing image enhancement framework using multiple reconstructed signals (CSIE-M). Firstly, images are reconstructed by different CS reconstruction algorithms. Secondly, reconstructed images are assessed and sorted by a No-reference quality assessment module before being inputted to the quality enhancement module by order of quality scores. Finally, a multiple-input recurrent dense residual network is designed to exploit and enrich useful information of the reconstructed images. Experimental results show that CSIE-M obtains 1.88-8.07dB PSNR improvement while the state-of-the-art works achieve a 1.69-6.69 dB PSNR improvement under sampling rates from 0.125 to 0.75. On the other hand, using multiple reconstructed versions of the signal can improve 0.19-0.23 dB PSNR, and only 4% reconstructing time is increasing compared to using a reconstructed signal.

1.4.3 In-loop filtering image enhancement for video encoder-decoder

The existing deep learning-based in-loop filtering (ILF) enhancement works mainly focus on learning to restore the original video frame from the reconstructed frame(s) without utilizing various potential resources at encoder and decoder. This work proposes a deep learning-based Spatial-Temporal In-Loop Filtering (STILF) that takes

advantage of the coding process to improve the performance of VVC in-loop filtering with CTU precision. The CTU-adaptive STILF has three main functions: VVC default in-loop filtering, self-enhancement with CU map (SECUM) for exploiting the spatial information inside a frame, and the reference-based enhancement with the optical flow (REOF) that extracts temporal information for enhancing the current frame. Experimental results show that 3.78%, 6.34%, 6%, and 4.64% BD-rate reductions are respectively obtained under common test conditions of All Intra, Low Delay P, Low Delay B, and Random Access configurations.

Moreover, this work enhances the spatial-temporal in-loop filtering (STILF) using a reinforcement-learningbased autonomous mode selection (AMS) that takes advantage of the coding information to improve VVC in-loop filtering. STILF filters accurately perform on the coding unit (CU) level, and the best filtering mode is selected by calculating the corresponding rate-distortion costs. In order to avoid the extra bits indicating the ILF mode of each CU, we further propose a reinforcement learning-based autonomous mode selection (AMS) method. We formulate the ILF mode selection as a decision-making problem. An agent network is proposed to predict a new CU partition and ILF mode for each predicted CU. Our method relies on absolutely no bits for ILF mode while offering more accurate pixel compensation. As a result, the proposed STILF-AMS obtains 4.13%, 7.1%, 6.93%, and 5.5% bitrate saving under All Intra, Low Delay P, Low Delay B, and Random Access configurations. Up to 18% bitrate saving can be obtained compared to the state-of-the-art video coding standard VVC.

1.5 Dissertation outline

This dissertation is composed of four main chapters. Each chapter is a complete and independent research that could be read separately by the readers. This chapter introduces video coding and the related deep learning-based video coding improvements. The key objective of this chapter is first to give readers a comprehensive review of

video coding, then an analysis of the drawbacks followed by the current related works and SOTAs.

In Chapter 2, the research on deep learning-based fractional interpolation for enhancing video encoder is presented. Our filters are designed to complete hand-crafted interpolation filters of video coding standards, extending the ability to deal with the diversity of video content. In this work, only one model is trained for all the fractional positions, enabling the flexibility to deal with any other video coding standard with the least modifications.

Chapter 3 presents and discusses the proposed deep learning-based compressive sensing enhancement using multiple reconstructed images for enhancing video decoder. In this chapter, we analyze and use the feature of CS compared to other coding techniques: to have multiple reconstructed images given the one and only compressed data. Using this unique feature, we propose a novel CNN that effectively enhances the CS reconstructed images without sending additional data, enabling the ability to reconstruct higher-quality images.

Chapter 4 presents a deep learning-based framework for VVC in-loop filtering enhancement for video encoder and decoder. In this work, we first propose two deep CNN architectures for enhancing VVC reconstructed images. Go over the switch mode-based ILF selection at CTU level, we introduce a novel technique for splitting and enhancing the coding efficiency of traditional video coding standards. Second, we propose an autonomous mode selection for deciding ILF for each CU in the reconstructed image using a deep reinforcement learning approach. This research uncovers a new mode selection technique for state-of-the-art codecs without storing additional bits.

Chapter 5 presents the conclusion of this dissertation.

Chapter 2

Enhancing reference-frame interpolation for video encoder

2.1 Introduction

H.265/High Efficiency Video Coding (HEVC) [61] had outperformed its predecessor H.264/AVC [62] to become the state-of-the-art video coding standard. Compared to H.264/AVC, H.265/HEVC has been improved its coding techniques and achieves a 25-50% better data compression at the same image quality [63]. One of the critical technologies that significantly contributes to the high coding performance of HEVC is motion compensated prediction (MCP). MCP (as shown in Fig. 2.1) aims to predict the current frame from the reference frames which are previously reconstructed and store the residual along with the motion vector between the corresponding blocks, which benefits for reducing the temporal redundancy in inter coding. However, the converting signals from the analog domain to the digital domain may omit some data, which could make prediction worse. Therefore, if the best matching block does not fall into integer samples, fractional pixels and fractional motion vector are required for these movements. Widely used, MCP applies interpolation filters on the reference frame, considered as integer samples, to obtain fractional samples. For interpolation filters, H.265/HEVC offers 7-tap quarter and 8-tap half Discrete Cosine Transform interpolation filters (DCTIF) for fractional interpolation while they are the 6-tap filter for half-pixel and average filters for quarter-pixel interpolation in H.264/AVC.

Refer to fractional interpolation in video coding, there have been many works that focus on improving fixed filters [14], or designing adaptive filters [64, 65], or hardware

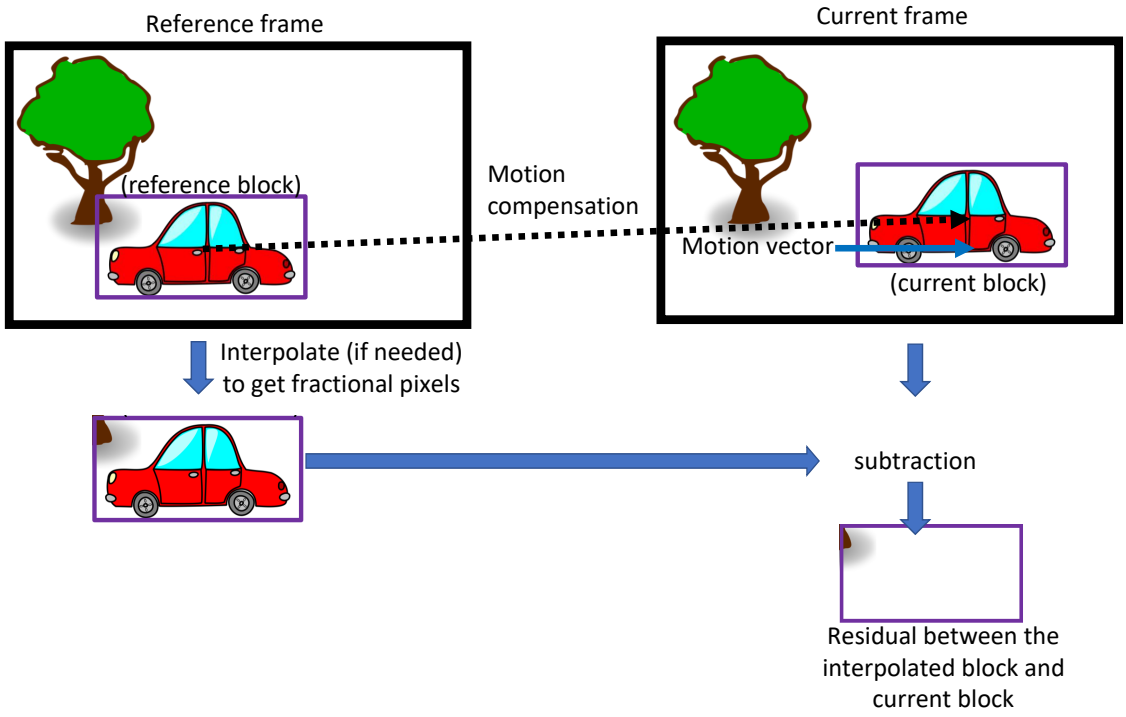


Figure 2.1: The concept of motion compensated prediction. In modern video coding standards, MCP first searches for the best matching of block from a list of reference frames. Reference block then will be interpolated to get more accurate motion compensation. Among the interpolated blocks, fractional motion which performs the smallest sum of absolute difference (SAD) will be chosen. Finally, corresponding motion vector and residual will be encoded and send to decoder.

design [66] for fractional interpolation. Due to the covered area is limited, the correlation between neighboring pixels may not be fully exploited. Moreover, the input signal is not always ideal and stable for these handcrafted filters. For these reasons, designed filters may not be able to adapt to the diversity of natural video content. In some aspects, fractional interpolation in MCP can be considered as a specific task of super-resolution where a high-resolution image is reconstructed from a low-resolution image (images).

Recently, deep learning-based methods have been widely used and obtained remarkable results in image and video processing. Convolutional Neural Network (CNN), a most representative model of deep learning, well improves the performance of the traditional method in high-level computer vision such as classification [28], detection[29,

30] to low-level computer vision tasks like image denoising [31], and mostly super-resolution. SRCNN [32], a very first CNN model in learning-based super-resolution, learns the mapping between an input of low resolution-image and output of the high-resolution image, outperforms traditional method bicubic. The work [33] aims to learn image detail on a 20-layer CNN model VDSR to improve the quality of low-resolution input. Despite the robust of CNN in improving super-resolution, they can not be directly applied for fraction interpolation in video coding because of two main problems. First, CNN-based super-resolution may change integer pixel after convolution. Second, and more importantly, the training sets of super-resolution and fractional interpolation in video coding are different. While the former aims to recover the high-resolution image by enhancing quality of the low-resolution image, the latter focuses on producing fractional samples that close to the current block to be encoded from the reference frames.

Inspired by the contributions of deep learning in video coding, recent studies have implemented diverse approaches to deep learning-based fractional interpolation works to improve the performance of MCP. To handle the problem of changing integer pixel after convolution, [43] first proposes three CNN models CNNIF_H, CNNIF_V, and CNNIF_D for horizontal, vertical, and diagonal half-pixel positions, respectively. By producing three half pixel independently, they keep integer pixels for the later process of HEVC. Later on, Zhang *et al.* introduces a CNN model followed by a Constrained mask with different weights for the integer pixels and three half pixels[44]. Similar to [43], Yan *et al.* trains 15 models for three half samples and 12 quarter samples in [47]. In [45], half and quarter pixels share nine convolution layers and be separately produced in group variational transformation in GVTCNN. Similarly, the work [49] designs switch mode based fractional interpolation to reduce the drawback of the motion shift in [45]. Liu *et al.* implements GVCNN that supports sub-pixel (half-pixel or quarter-pixel) under different QPs in a model [48].

For the second problem, besides the issue of different training sets, another difficulty that needs to be solved is fractional pixel does not exist in the real image. Generally, existing works assume integer and fractional pixels in the original frame, encode integer video, and learn the mapping between the reconstructed integer and fractional pixels [43, 45, 48, 49] or the mapping between the interpolated frame of the reconstructed reference frame and the original reference image [44]. Another way is encoding the original video and extracting the inter-coding block and its reference block to be ground-truth label and input of CNN [47].

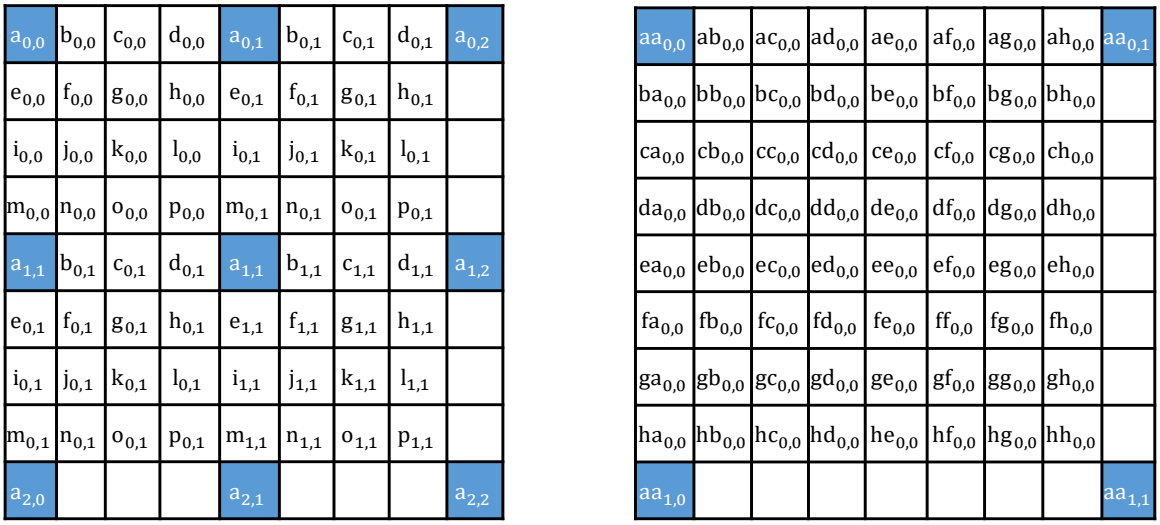
Although prior research generally confirms that CNN-based fractional interpolation improves coding performance, there are some drawbacks that could be improved in these approaches: only half-pixel are supported [43, 44], many models are required for fractional positions [47], or predicting fractional pixel from integer pixel may not good because the motion shift between integer and fractional pixel are not always stable [43, 45, 47, 48]. In paper [46], although the one models for all fractional samples and the prepossessing helps to reduce the motion shift problem, Chroma components are not well treated. In this work, we take the next step towards the CNN-based fractional interpolation in video coding: all components can be processed by CNN. In our proposal, the Y, U and V components of the reference frame is interpolated by the Discrete Cosine Transform-based interpolation filter (DCTIF) before feeding into CNN to avoid the motion shift problem. To take full advantage of CNN and DCTIF, we implement a rate-distortion optimization (RDO)-based interpolation method selection for each CU: Luma and Chroma components are interpolated by either DCTIF or CNN. Note that U and V components share one model for all fractional interpolation. For this selection, we encode two flags that indicate the interpolation method for Luma and Chroma components. As a result, we archive 2.9%, 0.3%, 0.6% BD-rate reduction compared to the anchor HEVC under low delay P configuration. Our work makes the following three contributions:

1. We present two CNN models for Luma and Chroma fractional pixels interpolation in video coding. A reconstructed frame is first interpolated by DCTIF to get 15 fractional samples. Our models take an input of a fractional sample and output corresponding fractional sample. Only one model is trained for 15 fractional samples at each QP. We use one model for Y component interpolation and a shared Chroma model for U and V components interpolation. Totally, we train eight models for four QPs in Luma and Chroma components.

2. We investigate a dataset generation method for our Y, U, V fractional interpolation training. As commonly, we generate our training set by assuming integer and fractional pixel in each video frame and encoding integer video. Each reconstructed video frame is interpolated by DCTIF to be CNN input and the fractional pixels extracted from the original frame are ground-truth labels for CNN.

3. We implement an RDO-based selection for Luma and Chroma fractional interpolation. In this selection, we define two new syntax elements to HEVC bin-string and encode them under CABAC regular mode. Each syntax element indicates the interpolation method for Luma or Chroma components for each CU that chooses inter coding with the fractional motion vector.

The proposal can be integrated to the existing video coding standards that support fractional motion search such as H.262, MPEG-4 Part 10 (as known as H.264/AVC), H.265/HEVC, AV1, etc., to further improve the coding efficiency. Moreover, network architecture also can be utilized for half and quarter-pixel interpolation for the coming video coding standard VVC/H.266. Since the different coding standards provide different noises, re-train our network on other standards reconstructed frames is required. The interpolation CNN output can replace the output of the interpolation filters in motion compensation to further improve the prediction of interpolation filters.



(a) Luma sub-samples

(b) Chroma sub-samples

Figure 2.2: Luma (a) and Chroma (b) sub-samples in HEVC. In (a) Luma sub-samples, $a_{i,j}$ present for integer pixel, $c_{i,j}$, $i_{i,j}$,$k_{i,j}$ are the half pixels and others are quarter pixels. Half and quarter pixels in the Luma component are interpolated by a 7-tap quarter and an 8-tap half DCTIF. At (b) Chroma sub-samples, $aa_{i,j}$ presents for integer samples and other pixels are fractional samples which are interpolated by four-tap filters.

2.2 Related works

In HEVC, motion search searches for the best matching fractional Luma samples in the list of previously reconstructed reference frames and stores the motion vector points to the best fractional pixel. Fig. 2.2 presents the positions of integer and fractional samples of Luma and Chroma components in HEVC. In Luma sub-samples, $a_{i,j}$ presents integer pixels, $c_{i,j}$, $i_{i,j}$, and $k_{i,j}$ are half pixel, and others are quarter pixels. In Luma fractional interpolation, HEVC applies horizontal filters on integer samples for $b_{i,j}$, $c_{i,j}$, and $d_{i,j}$. For $e_{i,j}$, $i_{i,j}$, and $m_{i,j}$, vertical filters are applied on integer samples. For the other samples, the vertical filter of its row is applied on the fractional sample at its column in the top row. For example, to obtain $g_{0,0}$, vertical filter of $e_{0,0}$ is applied on $c_{0,0}$.

HEVC supports fractional samples and motion vector for the Luma component up to quarter accuracy for the common used YUV format 420. Chroma component, whose

resolution is equal to the half of the Luma component's, holds a motion vector that accurate to one-eighth samples. Therefore, eight samples are interpolated using 4-tap filters [61] in motion compensation. Motion vector at Chroma component is calculated as:

$$fracMV_{Chroma} = MV \bmod 8 \tag{2.1}$$

Fig. 2.2 (b) presents fractional-sample in Chroma component. If the fractional MV points to $ab_{0,0}$, $ab_{0,0}$ and $ab_{i,j}$ are interpolated with the resolution is equal to Chroma resolution of the current block to be encoded. In Luma and Chroma component fractional interpolating, the applied area is restricted in the tap size of the interpolation filters, which limits the predicted block quality.

Recently, a learning based method has been applied to fractional interpolation to further improve the fractional interpolation filter of the video coding standard. Since fractional pixels do not exist in the real image, existing works assume and extract them from the original image and simulate the encoding method. A common method is to learn the mapping between encoded integer pixel and the fractional pixels extracted from the original image [43–45, 47–49]. Although they improve the encoding performance, there are some drawback could be improved: directly learning the fractional pixels from the integer pixels [43, 45, 48] may cause bad prediction since the motion between them are not always stable, or only half-pixel are trained [43, 44], or one CNN model for one fractional pixel [47] is not suitable for the real application.

2.3 Methods

As mentioned above, despite the fact that CNN-based techniques have acquired outstanding performance compared to traditional super-resolution methods, they can not be directly applied to fractional interpolation in video coding. In this work, we de-

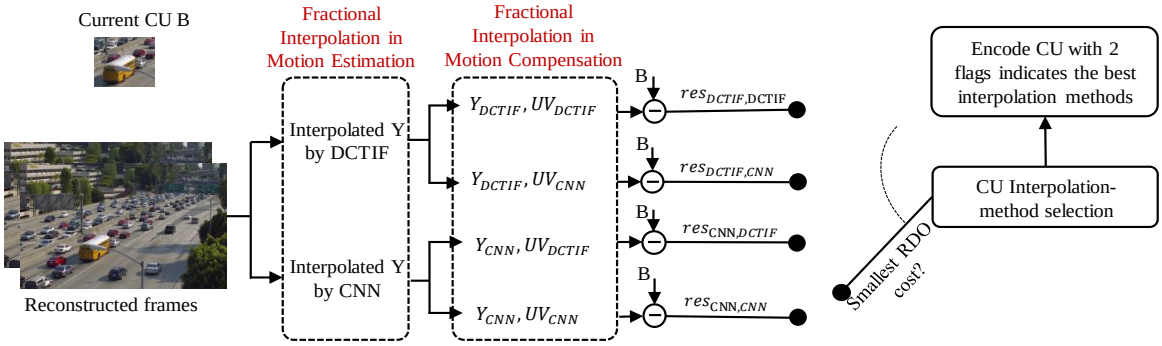


Figure 2.3: Visualization of our proposal. In Motion Search, the Y component of the reference frame is interpolated by DCTIF and CNN. In Motion compensation, U and V components of the reconstructed frame are interpolated by both CNN and DCTIF, and the Y component is interpolated by the method in motion search. The residual between current CU B and predicted CU are calculated and encoded with 2 bits indicate interpolation methods for Luma and Chroma components. Finally, an RDO-based fractional interpolation selection is implemented to decide which interpolation method should be used for Luma and Chroma fractional interpolation.

sign a training set and propose two Convolutional Neural Networks for Luma and Chroma fractional interpolation in video coding. We offer a rate-distortion optimization (RDO)-based selection for choosing interpolation mode of DCTIF and CNN for both Luma and Chroma components of a CU. In Fig. 2.3, we present our proposal in integrating CNN-based fractional interpolation and the interpolation method for Luma and Chroma Components to HEVC. In searching for the best fractional pixels at encoding, Y component of the reference frame is interpolated in both DCTIF and CNN. At motion compensation, we interpolate Y component by the method used in motion search, U and V components are separately interpolated by CNN and DCTIF. At encoding residual and calculating RDO cost for CU that chooses inter coding with a fractional motion vector, we also encode two flags indicating fractional interpolation methods for Luma and Chroma components by choosing the smallest RDO cost between four possible interpolation methods: DCTIF and DCTIF, DCTIF and CNN, CNN and CNN, and CNN and DCTIF for Luma and Chroma, respectively.

This section outlines our training set generation followed by our network architec-

ture and ends with Luma and Chroma interpolation selection.

2.3.1 Training set generation

In training any network, the training set plays a vital role for network behaviors. For CNN-based super-resolution, a popular training set includes a low-resolution image as input and the corresponding original high-resolution image as the ground-truth label. The prevailing method for creating input of CNN is doing down-sample the original image and up-sample the result for a low-resolution image [32, 33]. Due to the goal of reducing bitrate, MCP predicts the current image from the fractional-pixel images of the reference frames, which make training set of super-resolution could not be directly applied for fractional interpolation tasks. Moreover, integer pixels could be changed after convolution, and fractional pixels do not exist in the real image are also the main problems in creating a training set for applying the idea CNN-based super-resolution for fractional interpolation in video coding.

Frequently implemented, prior works encode the integer-position image and learn the mapping between reconstructed integer-position video and the fractional-position video. To solve the problem of integer pixel change after convolution, the work [44] generates a double-resolution image including integer and half pixels and feeds it to a very deep convolutional neural network for super-resolution (VDSR) followed by a mask that restricts integer pixels. The works [43, 45, 47, 48] separately generate half and quarter pixels based on integer pixels rather than generate double- or four-time-resolution image that includes integer and fractional pixels and keeps the integer pixel for MCP.

We implement an experiment to answer the question of what should be the ground-truth label for our training set. The target of MCP is to predict the current block to be encoded from the reference frames. However, the pair of the current block and reference block are achieved only in the encoding process and could be changed under

Table 2.1: Bjøtegaard-Delta bit-rate (BD)-rate reduction (%) of the replacing interpolated frame by the original frame when encoding down-sampled video compared to the anchor HEVC.

Class	Y	U	V
Average B	-39.3	-16.2	-16.5
Average C	-19.9	-8.4	-8.1
Average E	-23.7	-5.5	-5.6
Average all sequences	-27.6	-10.0	-10.0

different encoding configuration. Therefore, a training set of the current block and reference block may be restricted. In testing for ground-truth, we assume and extract integer and fractional pixels from the original video frames, do the encoding for integer images and set fractional images extracted from the original image as the interpolated image for encoding the down-sampled video. For this experiment, we test the first five frames of every sequence in class B, C, and E. For sequences in class B and E; the down-sampled videos do not fit the CU partition where HEVC cannot encode the down-sampled video, we then duplicate some rows and columns at the original video before downsampling. Experimental settings will be sufficiently described in 2.4.1. The results in Table 2.1 shows our BD-rate compared to the anchor HM under low delay P configuration.

Following the successful of our experiment in finding a training set for our CNN, we create a training set that takes the reconstructed integer video as input and the extracted fractional pixel as ground-truth, integer and fractional pixels are assumed from the original image. The process of training set generating visualized in Fig. 2.4 can be described as follow:

1. We extract the integer and fractional-position video by assuming integer and fractional pixels in every 4-by-4 non-overlapping blocks of each frame. Integer, half and quarter positions are pixels at similar positions to fractional samples in Fig. 2.2. We then obtain a low-resolution video of integer pixels (integer-position

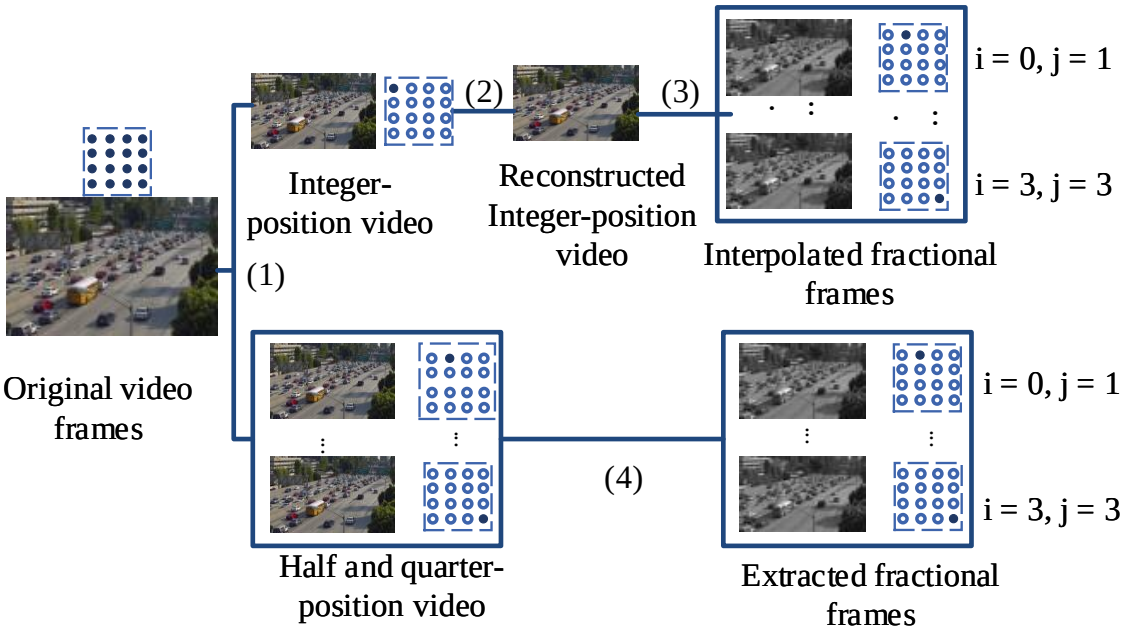


Figure 2.4: Our training set generation for learning fractional interpolation in video coding.

video) and 15 low-resolution videos including three half- and 12 quarter-position videos corresponding to 15 fractional samples.

2. Encode low-resolution video under low delay P configuration with QPs of 22, 27, 32 and 37 to get reconstructed down-sampled video.

3. Extract the Y component from reconstructed frames and interpolate them to 15 fractional samples by DCTIF. These 15 fractional samples are used as training input for our CNN.

4. Extract the Y component from each fractional-position video frame to be the CNN ground-truth label for training. Each pair of the fractional sample interpolated by DCTIF and the fractional sample extracted from the original frame is considered as a training sample.

For generating the training set of the Chroma component, we do the down-sampling to one eighth for all components because Chroma components require one eighth fractional

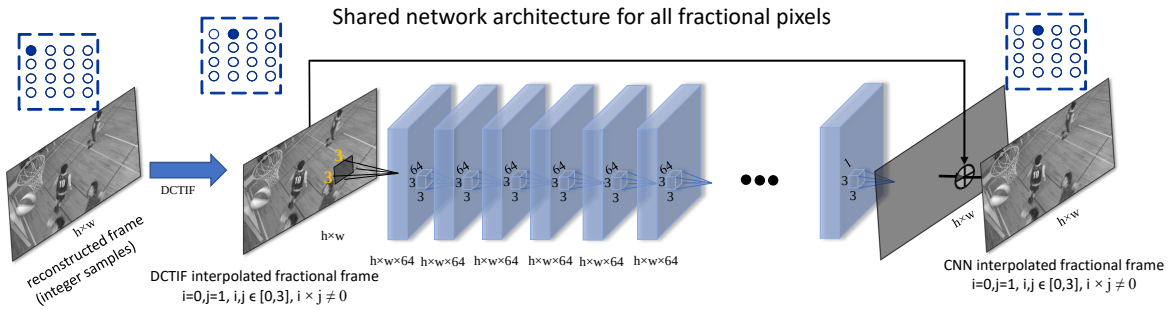


Figure 2.5: Our CNN architecture for learning all the fractional samples in Luma and Chroma components. For each feeding forward, one fractional sample is fed into our CNN. 15 fractional Luma samples and 63 fractional samples are supported by this architecture, respectively.

samples. All the processes for generating the Chroma training set are the same as the Luma Y component, except we have 63 fractional positions for Chroma components. Note that we use the default interpolation method DCTIF for the Y component when we generate a training set for Chroma components and vice versa.

We note that training a deep network with a small training set can cause overfitting which is not good for predicting test data. We then do the data augmentation for our training set. During training, input and its corresponding ground-truth label can be vertically, horizontally flipped, or none of them. We randomly rotated images to 0°, 90°, 180°, or 270° to avoid training biases.

2.3.2 Network architecture

In inter coding, MCP interpolates reconstructed frames to fractional accuracy and finds a fractional pixel that is closest to the current frame to be encoded. To further improve the interpolation filter DCTIF in HEVC, in this work, we design two CNN models for fractional pixel interpolation in Luma and Chroma components. As mentioned above, despite the facts that both of super-resolution and fractional interpolation in video coding need to increase the resolution of the input and CNN has remarkable

results in super resolution, we cannot directly apply CNN-based super resolution for fractional interpolation in video coding. In this subsection, we introduce our network architecture, inspired by the Very Deep Super Resolution VDSR[33] for Luma and Chroma fractional interpolation. The network architecture (Fig. 2.5) includes 20 convolutional layers. Each layer does convolution by applying 64 3×3 filters with the stride of one. Padding is set as one to keep the size of the input image after convolution. For each convolutional layer, we set a ReLU activation layer after convolution except for the final layer. We set a learning rate that starts at 0.1 and decreases it by ten after ten epochs. Training stops at 50 epochs with a batch size of 128.

At training, our network takes an input of interpolated fractional sample and an ground-truth label of real fractional sample extracted from original frame, which are described in 2.3.1. Given the reconstructed image x contains the integer pixels and the fractional-position frames y_j assumed and extracted from the original video frames, we apply DCTIF on reconstructed image x to obtain x'_j fractional samples where j is from one to 15 in Luma and one to 63 in Chroma components.

Each interpolated image x' by DCTIF are fed into CNN for the output y'. Our network aims to learn the mapping between x'_j and y'_j by minimizing the loss function:

$$L(\theta) = \frac{1}{2} \sum_{i=1}^{n} ||y'_i - y_i||^2 \tag{2.2}$$

where n is number of training samples obtained from training set generation 2.3.1. At testing, reconstructed images are interpolated to 15 fractional-samples image in Luma component and 63 fractional-samples in Chroma components before feeding into CNN.

2.3.3 Rate-distortion optimization (RDO)-based interpolation mode selection

In video coding, selection between different encoding modes is an efficient tool to improve the image quality, decrease bitrate, or reduce the computational complexity of video coding standard such as HEVC. We note that DCTIF and CNN have the abilities to deal with different signals. To take advantage of DCTIF and CNN, we define two context models for a Luma and Chroma interpolation method selection. Rate-distortion optimization (RDO) allows the encoder to find the best coding mode for a coding level. HEVC defines the rate-distortion (RD) cost RD_m for a block coded in mode m as follows:

$$RD_m = D_m + \lambda R_m \tag{2.3}$$

where, D_m indicates the distortion between the predicted block generated by mode m and the original block, λ is applicable weighting factor depends on configuration and set by the video coding standards. R_m indicates the total rate to encode the current block. Our selection is based on RDO cost and be described as follows. In motion estimation, we do the interpolation for the Y component by DCTIF and CNN. For each interpolation method, we choose the best fractional sample and send the best fractional MV. In motion compensation, we then do the interpolation for all components by both DCTIF and CNN. There are four possibilities: DCTIF and DCTIF, DCTIF and CNN, CNN and DCTIF, CNN and CNN for Luma and Chroma component, respectively. After calculating residual and encoding all parameters including two bits for Luma and Chroma interpolation method flags, an RDO-based selection will choose among four possibilities the interpolation methods for each CU coded with fractional motion vector. Each and every sub-CU in the same CU, if coded with fractional motion vector, share the same interpolation methods.

2.4 Experiments

2.4.1 Experimental settings and evaluation method

For our experiments, we focus on two main parts: training and coding. HEVC Test Model (HM) version 16.18 is used for training set generation and demonstration of CNN-based fractional interpolation in video coding. The format of all the test sequences is YUV 4:2:0.

In our training, we use PyTorch 1.0.0 with the support of the NVIDIA Tesla V100 GPU. As mentioned earlier, we have trained two models for Luma and Chroma components for each quantization parameter (QP) value. Eight models have been trained for four QPs: 22, 27, 32, and 37. Training set for QP 32 and 37 models are acquired from three sequences *Pedestrian*, *Traffic* and *PeopleOntheStreet* [67]. For QP 22 and 27's models, we produce training set from *Traffic* and *PeopleOnTheStreet*. These sequences are obtained from HEVC standard test sequences. For training set generation, we encode our training sequences under Low Delay P configuration with full search is enable and an Intra period of 16 for a stable training set. Other parameters are set as default. We benefit from the residual training, which takes 10 hours to train Luma and eight hours for Choma component models. In generating fractional pixels for the Chroma component, some pixels are duplicated to match with the HM requirements. For training, the input image and ground-truth label from 2.3.1 are split into 41x41 patches with a stride of 16.

Figure 2.6 illustrates the reference types of four difference configurations defined by HEVC. Among the configurations, motion compensated prediction is not performed on Intra configuration. Therefore, the following experiments will be conducted under Low Delay P, Low Delay B and Random Access configurations. We set all parameters to default except full search is enabled. Because Bitrate and Peak signal-to-noise ratio (PSNR) are all need to be considered in evaluating our method, we use a well-

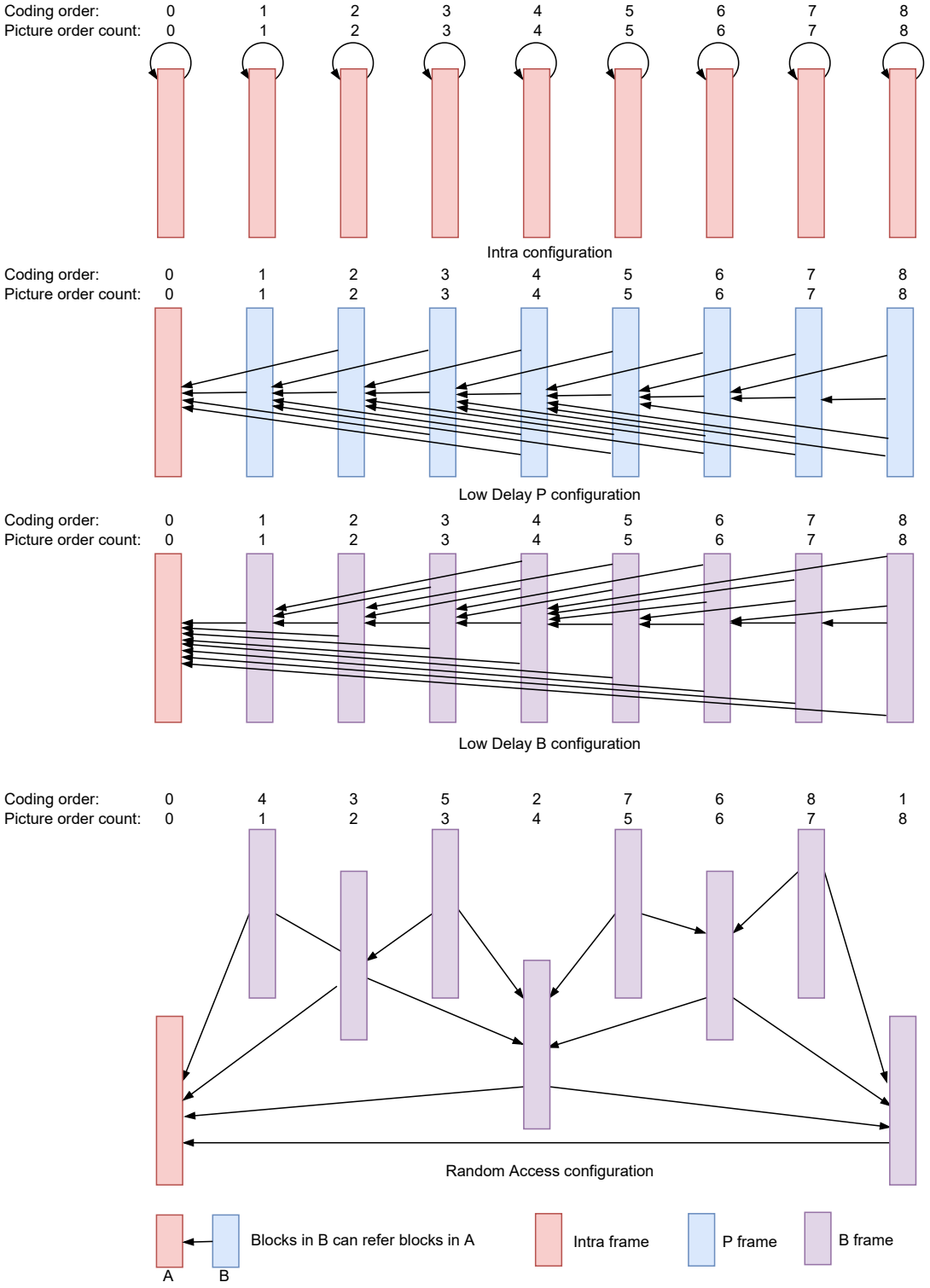


Figure 2.6: Frame referencing in four main configurations.

known measuring metric, Bjøtegaard-Delta bit-rate (BD-rate) [68]. BD-rate takes at least four samples from different video coding techniques and tells how many bits are reduced between them at the same video quality. In our case, we compare our proposals obtained from four QPs. Testing experiments are conducted on HEVC standard test sequences shown in Fig. 2.7.

2.4.2 Experimental results

2.4.3 RDO-based interpolation method selection result

In our proposal, RDO cost-based interpolation-method selection for each CU has been implemented in encoding. When a CU is coded with a motion vector that has the fractional part, two bits indicate the flags for Luma and Chroma interpolation methods are encoded. Fig. 2.8 shows our visualization on the first P frame of RaceHorseC under the Low Delay P configuration. In our visualization, cyan blocks indicate CU that choose DCTIF for interpolating all components, magenta blocks indicate CUs that choose DCTIF for Y and CNN for UV components, yellow blocks indicate CUs that choose CNN for Y and DCTIF for UV components, and red blocks indicate CUs that choose CNN for interpolating all components. The rest parts are CUs coded with integer motion vector or intra coding. As our visualizations, CUs at the static background tend to choose DCTIF for fractional interpolation and CUs at moving object tend to choose CNN for fractional interpolation.

On another hand, we measure the ratio of choosing interpolation methods for each CU. In HEVC, CU size is variety, which makes the calculating hitting ratio should be on the area than on count. Since these two flags are dependent, we calculate the hitting ratio on two flags as one than separately counting. In Table 2.2, we show the hitting ratio of using CNN and DCTIF for Luma and Chroma components at CU-level. Class F, where Luma and Chroma components rarely choose CNN for fractional interpolations, obtains the lowest bitrate saving compare to other classes.

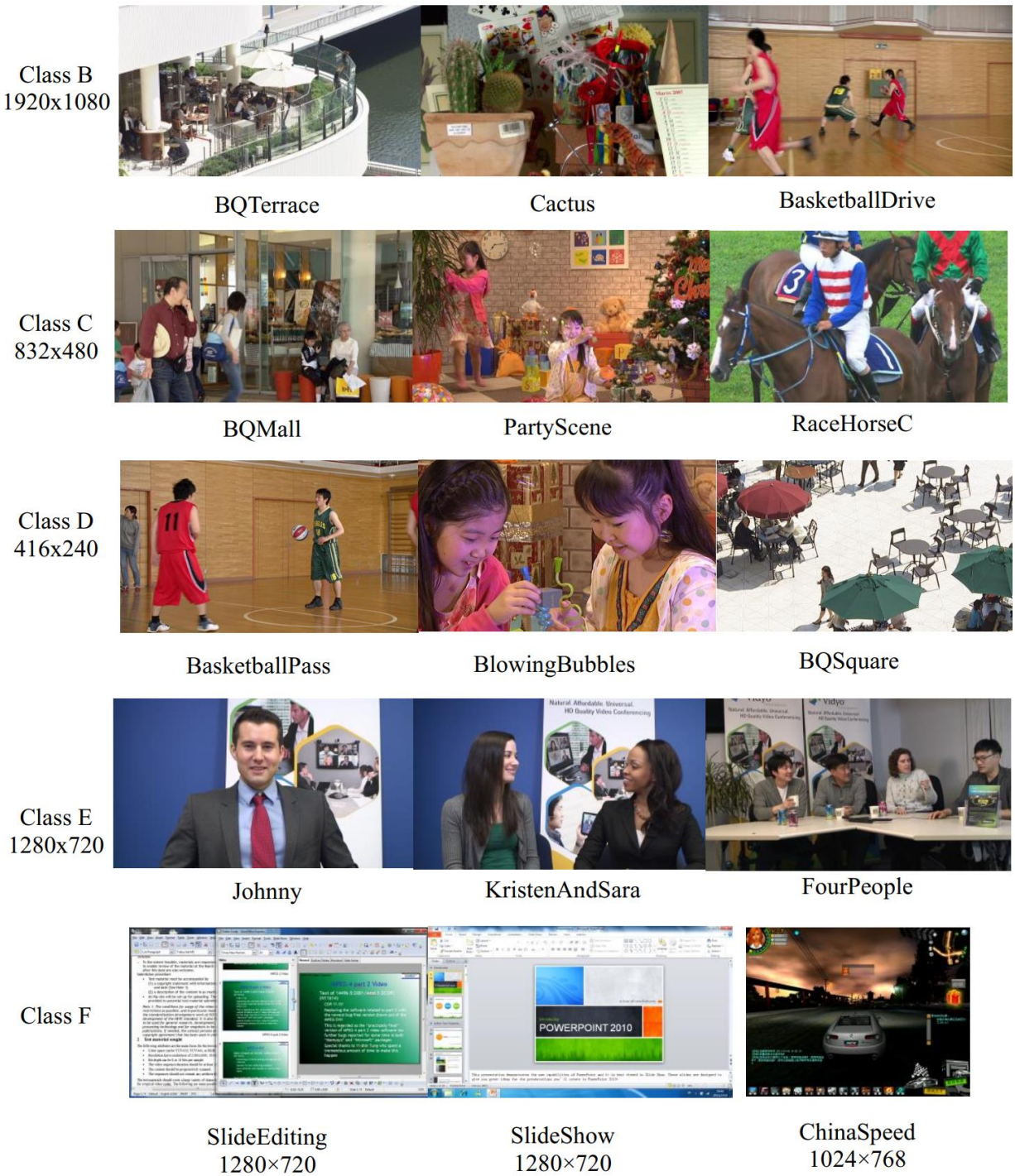


Figure 2.7: Example on HEVC test sequences .

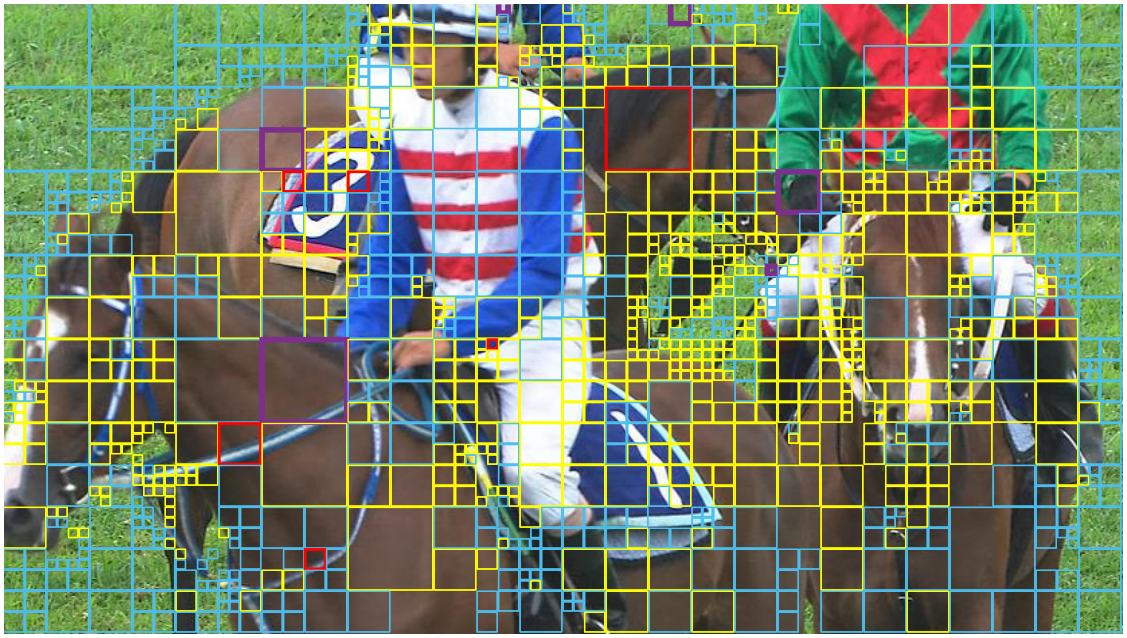


Figure 2.8: Visualization of our CU selection on *RaceHorseC*. In this figure, cyan, magenta, yellow, and red blocks indicate CUs that choose DCTIF for YUV interpolation, DCTIF for Y and CNN for UV interpolation, CNN for Y and DCTIF for UV interpolation, and CNN for YUV interpolation, respectively. The rest parts are CUs that choose inter coding with integer motion vector or intra coding.

Table 2.2: Hitting ratio (%) of two interpolation methods for Luma and Chroma component under Low Delay P configuration.

Class	$DCTIF_Y$ $DCTIF_{UV}$	$DCTIF_Y$ CNN_{UV}	CNN_Y $DCTIF_{UV}$	CNN_Y CNN_{UV}
B	54.51	0.51	44.36	0.62
C	53.18	0.78	45.07	0.97
D	57.75	0.47	41.34	0.44
E	64.18	0.32	34.71	0.79
F	71.84	1.51	26.14	0.51
All	60.29	0.72	38.32	0.67

2.4.4 Comparison with existing works

We also experiment to compare our proposal to existing CNN-based fractional interpolation works. For a fair comparison, we reimplement our proposal on HM 16.7 and

Table 2.3: BD-rate (%) comparison between CNN-based fractional interpolation and our proposal under Low Delay P configuration.

Class	[45]	[49]	Ours
Average B	-3.0	-3.0	-3.8
Average C	-1.7	-2.7	-3.0
Average D	-1.5	-2.5	-3.5
Average E	-1.9	-2.8	-4.6
Average all sequences	-2.1	-2.8	-3.7

disable the CNN-based fractional interpolation and the context model for the Chroma component. In this experiment, only Y component is interpolated by CNN, and DCTIF is used for interpolating Chroma components. The selection of CNN/DCTIF fractional interpolation is also implemented for the Y component, and the flag for the interpolation method is encoded in CABAC regular mode. In this comparison, we use the reimplemented results from paper [49].

The results (Table 2.3) shows that our proposal surpasses the GVTCNN [45] and the switch mode-based fractional interpolation [49] on HM-16.7. In general, we achieve a 3.7 BD-rate reduction on average and rank first all Classes from B to E.

We also compare our work with the works [43, 44]. In these works, only half pixels are interpolated by CNN, quarter pixels are interpolated by DCTIF. In this comparison, we also re-implement our work on HM 16.7 and only half pixels are interpolated by CNN. The results (Table 2.4) shows that our half-pixel interpolation outperforms CNNIF[43] and *Zhang*'s work [44] in average.

2.4.5 Overall results

Our experiments under Low Delay P, Low Delay B, and Random Access configurations are shown in Table 2.5. Generally, we obtain the highest BD-rate reduction on Low Delay P configuration and the lowest saving bitrate belongs to the test under Random

Table 2.4: Comparison of CNN-based half-pixel interpolation and our proposal under Low Delay P configuration (anchor: HM 16.7).

Class	Sequence	CNNIF [43]	Zhang *et al.* [44]	Ours (half)	Ours (half and quarter)
B	Kimono	-1.1	-	-4.4	-4.9
	ParkScene	-0.4	-	-0.8	-0.1
	Cactus	-0.8	-	-3.2	-4.6
	BasketbalDrive	-1.3	-	-3.4	-3.7
	BQterrace	-3.2	-	-3.9	-5.7
C	BasketballDrill	-1.2	-0.9	-3.7	-4.8
	BQMall	-0.9	-0.3	-1.8	-2.5
	PartyScene	0.2	-0.1	-1.2	-1.8
	RacehorsesC	-1.5	-0.4	-2.4	-3.0
D	BasketballPass	-1.3	-1.2	-2.3	-3.5
	BQSquare	1.2	0.3	-1.8	-3.9
	BlowingBubbles	-0.3	-0.2	-2.6	-3.1
	RaceHorses	-0.8	-0.7	-2.9	-3.7
E	FourPeople	-1.3	-0.6	-3.8	-4.3
	Johnny	-1.2	-0.7	-4.7	-5.4
	KristenAndSara	-1.0	-0.4	-3.6	-3.9
F	BasketballDrillText	-1.4	-0.4	-3.4	-4.2
	ChinaSpeed	-0.6	-0.6	-0.6	-0.8
	SlideEditing	0.0	-0.0	-0.2	-0.1
	SlideShow	-0.7	-0.2	-0.2	-0.6
Average B		-1.4	-	-3.2	-3.8
Average C		-0.9	-0.4	-2.3	-3.0
Average D		-0.3	-0.5	-2.4	-3.5
Average E		-1.2	-0.6	-4.0	-4.6
Average F		-0.7	-0.3	-1.1	-1.4
Average all sequence$^{(*)}$		-0.91	-0.44	-2.56	-3.22

$^{(*)}$: Average calculated on numbers of existing sequences.

Table 2.5: BD-rate (%) of our proposal compared to HEVC under Low Delay P, Low Delay B and Random Access configurations.

Class	Sequence	Low Delay P			Low Delay B			Random Access		
		Y	U	V	Y	U	V	Y	U	V
B	Kimono	-4.7	1.1	0.6	-1.1	1.3	0.6	-0.7	0.6	0.0
	ParkScene	-1.3	1.1	0.2	-0.4	0.7	0.4	-0.5	0.2	0.0
	Cactus	-4.2	-1.6	-1.9	-3.2	-1.1	-0.6	-2.3	-0.5	-0.9
	BasketbalDrive	-4.2	-1.6	-1.9	-1.4	0.1	-0.8	-1.7	-0.1	0.3
	BQterrace	-6.5	-2.1	-2.9	-2.5	-0.2	-0.6	-1.6	-0.2	-0.2
C	BasketballDrill	-4.0	-0.2	-1.4	-3.1	0.5	-0.2	-1.5	-0.6	-0.9
	BQMall	-2.0	0.0	-0.3	-1.7	-0.1	-0.4	-0.9	-0.2	-0.4
	PartyScene	-1.8	-0.9	-0.6	-1.1	-0.2	-0.1	-0.6	-0.2	-0.5
	RacehorsesC	-2.6	-0.7	-0.3	-2.1	-0.1	-0.5	-1.6	-0.6	-1.5
D	BasketballPass	-2.6	-0.8	-0.7	-2.2	-1.6	-1.8	-1.1	0.2	-0.2
	BQSquare	-4.2	-1.6	-1.3	-2.2	-0.7	-0.4	-0.9	0.4	0.1
	BlowingBubbles	-2.5	-0.5	-1.3	-2.7	-0.8	-0.6	-1.0	-0.9	0.1
	RaceHorses	-2.9	0.3	-1.0	-2.8	-0.1	-1.4	-1.4	-0.8	-0.9
E	FourPeople	-4.3	-0.3	-0.4	-3.8	-0.2	-0.7	-2.9	-0.1	-0.4
	Johnny	-5.7	-2.3	-1.5	-2.6	-0.5	-0.5	-2.1	-0.1	-0.4
	KristenAndSara	-4.1	-0.9	-1.6	-3.4	-1.1	-0.8	-2.2	-0.3	-0.5
F	BasketballDrillText	-3.3	1.1	-0.1	-3.0	0.7	0.4	-1.0	-0.5	-1.2
	ChinaSpeed	1.2	1.6	1.4	-0.6	-0.2	-0.4	-1.0	-0.9	-0.8
	SlideEditing	1.3	1.0	1.0	-0.2	-0.2	-0.2	0.2	0.4	0.2
	SlideShow	1.0	0.1	1.6	-0.5	-1.6	-2.0	-0.4	0.2	4.7
Average B		-4.3	-0.5	-1.1	-1.7	0.2	-0.2	-1.4	0.0	-0.2
Average C		-2.6	-0.5	-0.6	-2.0	0.0	-0.3	-1.1	-0.4	-0.8
Average D		-3.0	-0.6	-1.1	-2.5	-0.8	-1.0	-1.1	-0.3	-0.2
Average E		-4.7	-1.1	-1.2	-3.3	-0.6	-0.7	-2.4	-0.2	-0.5
Average F		0.1	1.0	1.0	-1.1	-0.3	-0.5	-0.6	-0.2	0.7
Average all sequences		-2.9	-0.3	-0.6	-2.0	-0.3	-0.5	-1.2	-0.2	-0.2

Access configuration.

We obtain a 2.9, 0.3, and 0.6 % Y, U, and V BD-rate saving compared to the original HM under Low Delay P configuration and up to 6.5%, 2.1%, 2.9 % Y, U, V BD-rate reduction on sequence *BQTerrace*. Results show that the proposal can deal with high-resolution videos such as sequences in class A and E where average BD-rate reductions for Y component are over 4%. We can obtain a high and stable BD-rate reduction for all components of class E sequences where backgrounds are static. Although Y, U, and V components are trained, and an RDO-based interpolation method selection has been implemented, there is some space that our methods cannot improve. For example, it can be seen that our proposal does not work well on screen-content sequence *ChinaSpeed*, *SlideEditing*, and *SlideShow* under Low Delay P configuration since no data for these content has been trained. The future work may include training for the screen- content video data.

For Low Delay B configuration results, the best performance belongs to class E, where 3.3%, 0.6%, and 0.7% Y, U, and V BD-rate reductions are obtained, respectively. Although our models do not work well on screen-content videos of class F under Low Delay P configuration, they acquire the average BD-rate reduction of 0.4%, 0.6%, and 0.8% on *ChinaSpeed*, *SlideEditing*, and *SlideShow* under Low Delay B configuration.

For Random Access configuration, we achieve 1.2%, 0.2%, and 0.2% BD-rate reduction on Y, U, and V components, respectively. This experiment obtains a Y BD-rate reduction up to 2.9% at sequence *FourPeople*. Along with the configurations that we do experiments, Random Access obtains the lowest bit rate saving at all components.

To further investigate the quality of our method compare to HEVC, we visualize RD-curves (Fig. 2.9) of our method on Y, U, and V components of sequence *BQTerrace*. It can be seen that our proposal can significantly increase the PSNR at the high bit rates more than at lower bit rates.

We also experiment to figure the effect of the separate network on Chroma component. In this experiment, we train four models for each Chroma component at the

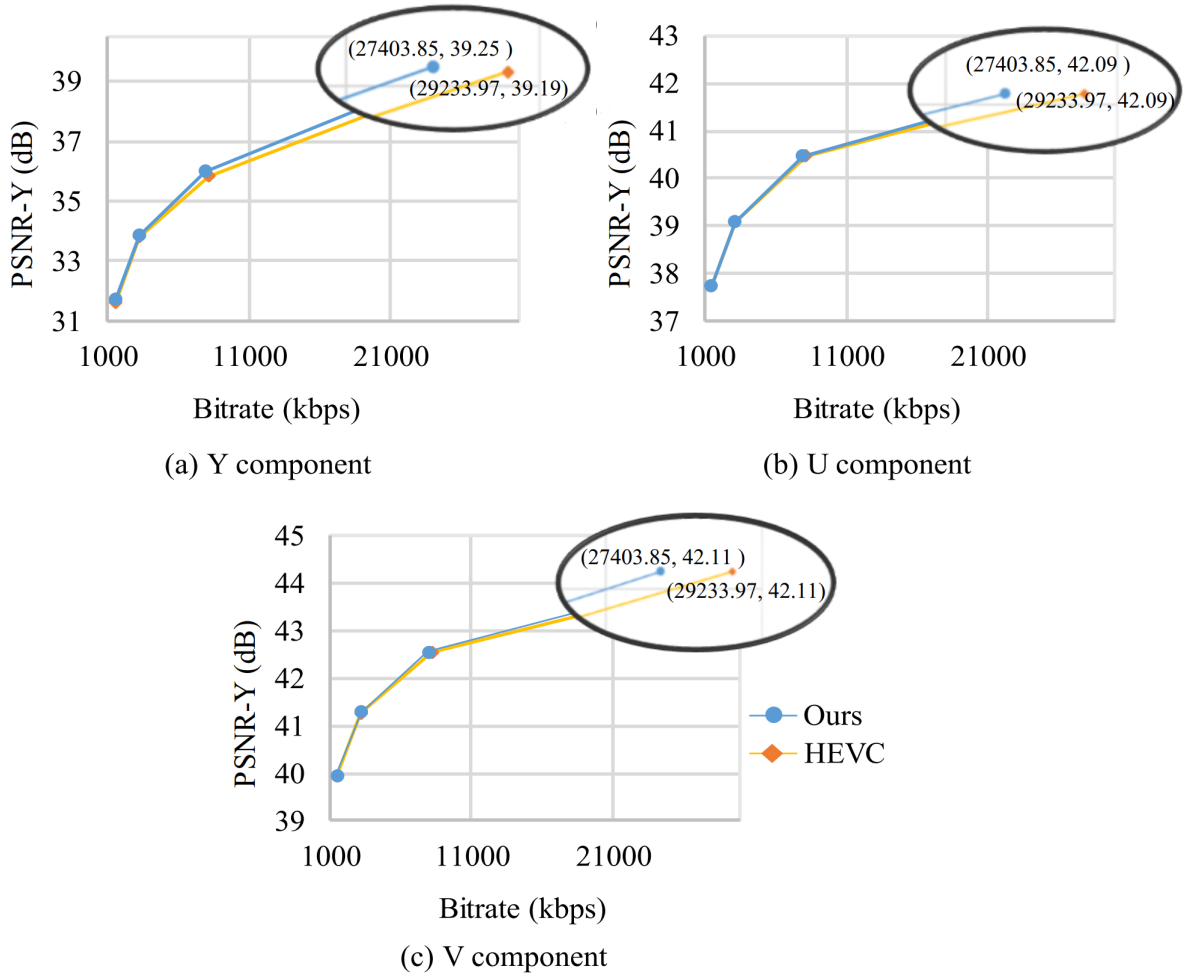


Figure 2.9: R-D curves of sequence BQTerrace under Low Delay P configuration on (a) Y component, (b) U component and (c) U component.

Table 2.6: BD-rate (%) of our proposals in seperating models for U and V compared to HEVC under Low Delay P configuration.

Class	Combining models			Separate models		
	Y (%)	U (%)	V (%)	Y (%)	U (%)	V (%)
Average B	-4.3	-0.5	-1.1	-4.3	-1.1	-3.2
Average C	-2.6	-0.5	-0.6	-2.4	-0.2	-2.6
Average D	-3.0	-0.6	-1.1	-3.0	-1.0	-3.0
Average E	-4.7	-1.1	-1.2	-4.7	-1.2	-2.8
Average F	0.1	1.0	1.0	0.0	0.7	0.0
Average all sequences	-2.9	-0.3	-0.6	-2.8	-0.6	-2.3

separate test. Eight models have been trained in the combining experiment, and twelve models have been trained on the separate experiment. The result of testing training models (Table 2.6) shows that our separate test archives better BD-rate reduction on U and V components than the combining experiment. However, Y BD-rate is decreased by 0.1% at separate models even models for Y component are the same. For more detail, training separate models can archive up to 6.6% for Y, 3.0% for U, and 5.1% for V BD-rate reduction in sequence *BQTerrace*.

2.5 Chapter conclusions

In this chapter, we present a deep learning-based method for fractional interpolation in video coding and design a training set for our CNN models. In our proposal, Luma and Chroma components are first interpolated by DCTIF before feeding to CNN. This approach enables the ability to deal with any video codec regardless of how many fractional samples are supported. Finally, an RDO cost-based interpolation method selection is performed for choosing the best fractional interpolation method at CU level. As a result, we obtain an average BD-rate reduction of 2.9%, 0.3%, and 0.6% on Y, U, and V component, respectively, under low Delay P configuration. We also show the

effects of training the separate models or combining models for U and V components and a comparison of our method compared to the existing works.

Chapter 3

Compressive sensing image enhancement at video decoder

3.1 Introduction

Internet of Thing (IoT) interconnects numerous devices including sensors, cameras, smart home products, and smart city products in an environment. It provides a fundamental way to communicate, store, transmit, and process sensed data, giving better productivity and more efficient solutions for improving the quality of human life. Surveillance systems have been pointed out as one of the most necessary but challenging solutions in urban developments due to large data storage requirements and the high computational complexity in processing images and videos sensed by cameras. Therefore, compression methods that can adapt the requirements of (1) saving the power consumption and prolonging the battery lifetime of IoT devices, (2) securing the data, and (3) balancing the traffic load when traveling throughout the network are preferred in designing sensing devices for surveillance systems. Traditional Shannon-Nyquist theorem states that signals need to be sampled at twice the bandwidth to be recoverable. In IoT systems such as remote surveillance and astronomy satellites, the Shannon Nyquist rate is costly, requires ample storage space and wide bandwidth for transmission. Compressive sensing (CS), which requires only a few compressive measurements to contain nearly all the useful information, breaks the limitation stated by Shannon-Nyquist's theory. CS has adapted the requirements and becomes one of the effective lossy compression methods that are considered when designing devices for IoT applications [69–71]. First, in the CS encoder, only matrices multiplication is

performed. The matrices multiplication is simple to be embedded in resource-limited devices and ensures energy saving for IoT self-powered devices. Second, the measurement matrix is only shared between the encoder and decoder makes the network secure. Third, the amount of sent measurement is fixed throughout the time, ensuring the traffic for transmission. Unlike the other compression standards, reconstructing CS signals is to solve a nonlinear inverse problem [19]. Since different CS reconstruction algorithms dissimilarly model the recovery solution, choosing a CS reconstruction algorithm for an application is challenging. On the other hand, there is always space to further improve the signals reconstructed by lossy compression methods. In this work, a study on enhancing CS reconstructed images is proposed to improve the qualities of CS reconstructed signals without changing the encoder and the decoder, keeping the requirements of IoT solutions.

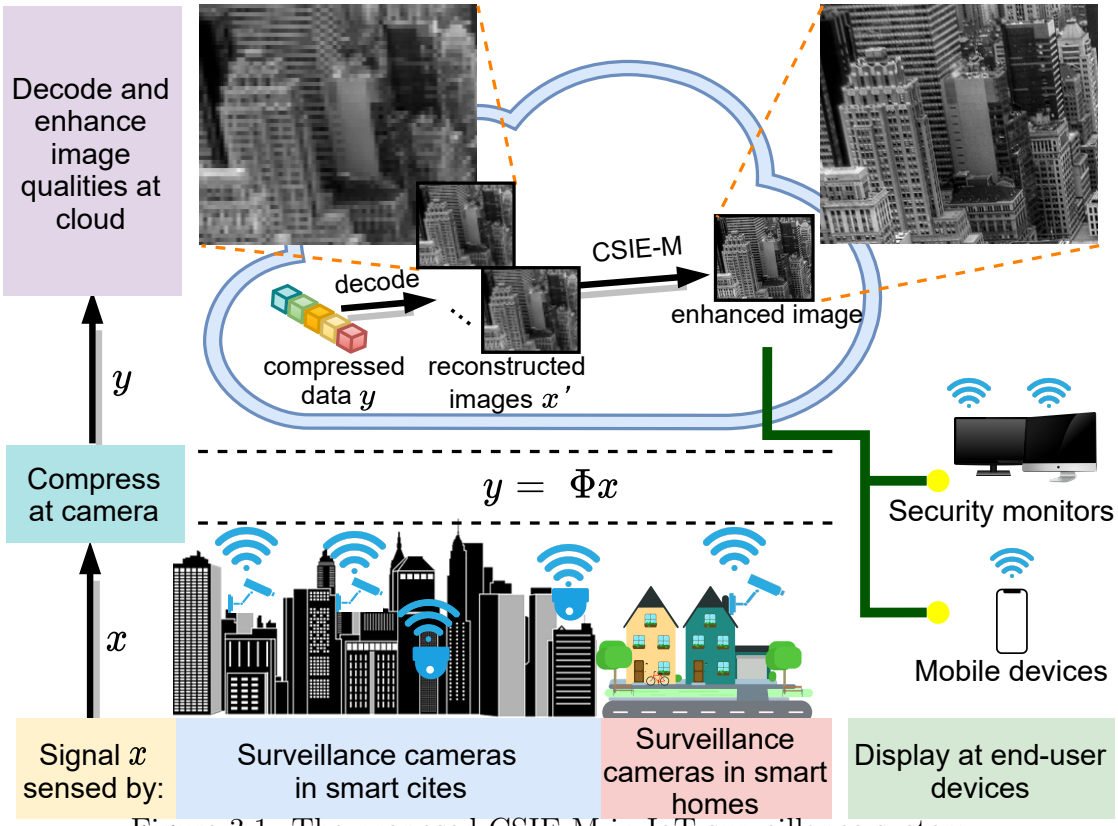


Figure 3.1: The proposed CSIE-M in IoT surveillance system.

Convolutional Neural Network (CNN) is well-known for learning complex functions

where the designed filters are not flexible enough to model. In recent years, CNN has been widely applied and obtained remarkable results in image quality enhancement. In this work, we take the next step towards an AIoT approach for enhancing CS reconstructed images sensed by IoT devices (Figure 3.1). Images are sensed and encoded in IoT cameras before being sent to the IoT cloud. Unlike the existing image enhancement works, where enhancing the image relies on a single degraded image [31, 72–77] or neighboring frames [78, 79], this work takes advantage of different CS reconstructed images with different qualities and introduces a deep learning-based **c**ompressive **s**ensing **i**mage **e**nhancement framework using **m**ultiple reconstructed signals (CSIE-M). At IoT cloud, compressed data are decoded by the commonly used CS reconstruction algorithms: L1 equality constraints via primal-dual algorithm (L1EQPD) [26], Spectral Projected-Gradient for L1 (SPGL1) [27], Orthogonal Matching Pursuit (OMP) [20], and Sparsity Adaptive Matching Pursuit (SAMP) [21]. In learning multiple-to-one mapping, it is necessary to decide which branch an image should be input. Therefore, a No-reference quality assessment module, namely Scorenet, is proposed to score and rank reconstructed images before feeding them to the quality enhancement network by order of quality. In the quality enhancement module, a multiple-input residual recurrent network (MRRN) is proposed for enhancing the reconstructed images by exploiting and enriching useful features via a recurrent mechanism. MRRN takes the best-quality image, which will be added to the enhanced feature for the main branch. The two lower-quality images are input to the supporting branches. Finally, enhanced images at the IoT cloud are displayed in monitoring devices for surveillance solutions or transfer to other image processing tasks such as recognition and detection. The experimental result shows that CSIE-M improves 1.88 to 8.07 dB PSNR compared to the main input image on various sampling rates from 0.125 to 0.75.

This work's main contributions are summarized as follows. First, it is the first time to design a compressive sensing image enhancement framework using multiple

reconstruction signals. In CSIE-M, a No-reference quality assessment module scores and ranks reconstructed images before feeding them to the quality enhancement module. Second, the proposal outperforms the state-of-the-art works in distorted image enhancement. Finally, a study on the number of input images is also conducted to show the effect of using multiple inputs on enhancing the reconstructed image quality.

3.2 Related Knowledge

3.2.1 Compressive sensing

Given sensed signal x that can be represented by a $n \times 1$ sparse vector s in the domain ψ, CS encoder simply calculates the $m \times 1$ measurement y by:

$$y = \Phi x = \Phi \psi s \tag{3.1}$$

where Φ is the $m \times n$ measurement matrix fixed in the encoder and the decoder [19]. The commonly designed measurement matrices include random matrices, binary matrices, and structural matrices. The quotient of m and n defining the system's compression ratio, also known as sampling rate (SR), represents the amount of data sent to the decoder. The process of decoder, on the other hand, is more complicated. In the CS-based image compression, the decoder reconstructs the image x back into the pixel domain by solving an underdetermined matrix equation where $m < n$. Therefore, reconstructing x at the decoder is solving an ill-posed problem. There have been many CS reconstructing algorithms include greedy algorithms, convex optimization, and gradient-based algorithms. The proposal adopts L1 optimization include L1EQPD [26] and SPGL1 [27], and greedy algorithms include OMP [20] and SAMP [21] for CS image reconstruction.

3.2.2 Deep Learning-based distorted image enhancement

Image enhancement is one of the essential components in image processing and image-display applications [80, 81]. Concerning deep learning-based distorted image enhancement, single image enhancement [31, 72–77], and the multi-frame enhancement [78, 79] are mainly focused in removing compression artifacts and denoising. Zhang *et al.* introduce a denoising convolutional neural network (DnCNN) [31] that can deal with different Gaussian noise levels, single image super-resolution, and JPEG image artifacts caused by different quality factors. For denoising real-noisy images, Anwar *et al.* introduce a novel single-stage blind Real image denoising network (RIDNet) [73]. In RIDNet, local skip connections, short skip connections, and long skip connections are utilized to exploit low-frequency information over the feed-forward. Jia *et al.* introduce a content-aware loop filtering scheme based on multiple CNN models (CACNN) [72] for improving the performance of the High Efficiency Video Coding (HEVC) by enhancing the quality of decoded frames. In [74], the authors introduce attention-guided denoising convolutional neural network (ADNet). ADNet combines dilated convolutions, standard convolutions, and an attention mechanism for real-noisy image denoising and blind denoising. In [75], both noise removal and noise generation tasks are trained in a Bayesian network that learns the joint distribution of the pairs of the clean and distorted images. The authors [76] propose a Block artifact removing convolutional neural networks (BARCNN) for JPEG image enhancement. BARCNN can be integrated to the receiver side to enhance image quality without any additional cost on the IoT node ends. Building upon on DnCNN, the authors [77] propose a theoretically-grounded blind and universal deep learning image denoiser, namely Blind Universal Image Fusion Denoiser (BUIFD), for additive Gaussian noise removal. Recent approaches [78, 79] take advantage of the temporal correlation between adjacent frames to enhance the low-quality image by using its neighboring high-quality video frames. Apart from these works, this work proposes a deep network that enriches

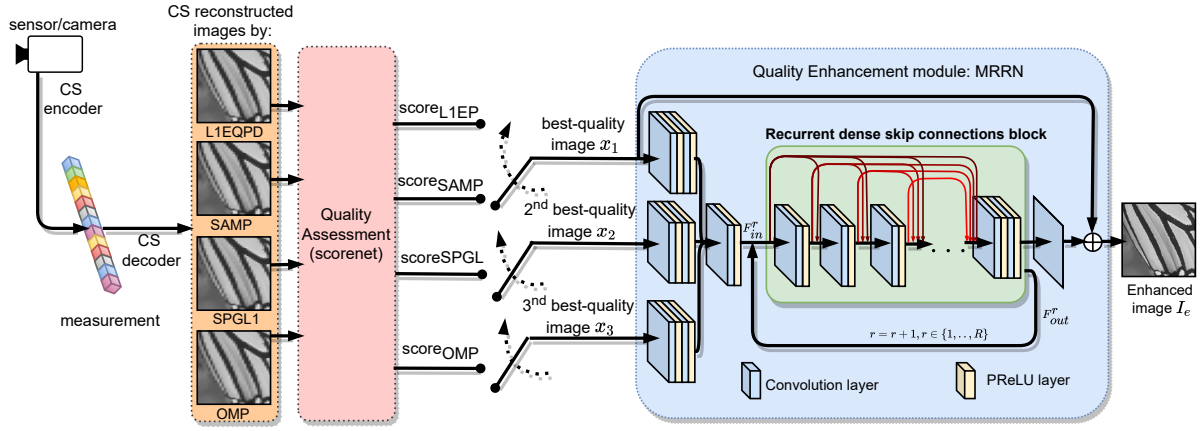


Figure 3.2: The proposed CSIE-M architecture. Sensed signal is compressed by CS in cameras. At the decoder, compressed data (measurement) is recovered by different CS reconstruction algorithms. Reconstructed images are judged by the No-reference quality assessment module Scorenet and fed to the quality enhancement network MRRN by order of quality scores for producing the enhanced image.

and synthesizes useful information via a recurrent mechanism performing on extracted features of CS decoded images.

3.3 The proposed CSIE-M framework

3.3.1 Overview of the CSIE-M framework

As mentioned above, there have been many approaches for solving the nonlinear inverse problem at CS decoder. Different CS reconstruction methods model the solution differently, resulting images recovered with different qualities. Using different CS reconstructed images provides more representations for CNN to exploit and enhance the performance in recovering the original signals. To make full use of this property, we propose a deep learning-based compressive sensing image enhancement framework using multiple reconstructed signals that learns a multiple-to-one mapping from the reconstructed images to the original one (shown in Figure 3.2).

First, a sensed signal is encoded by CS in sensors or cameras. At the decoder, different CS reconstruction algorithms perform on the compressed data to obtain dif-

ferent reconstructed images. Let $X = \{x_{alg}, alg \in \{L1EQPD,\ SPGL,\ OMP,\ SAMP\}\}$ indicate the images reconstructed by the four algorithms L1EQPD, SPGL, OMP, and SAMP. In CSIE-M, the highest-quality reconstructed image takes the highest responsibility in generating the enhanced image. The other inputs are considered additional features generated by designed filters: CS reconstruction algorithms. We propose a No-reference quality ranking module including a Deep Learning-based No-reference image quality assessment (IQA) Scorenet for scoring and ranking CS reconstructed images. Scorenet predicts the quality score z_{alg} of image x_{alg} as:

$$z_{alg} = f_{scorenet}(x_{alg}) \tag{3.2}$$

Images in list X are sorted based on the corresponding quality scores in list Z. The best quality image x_1, the second best-quality image x_2, and the third best-quality image x_3 are fed to MRRN denoted as f_{MRRN}, and the enhanced image I_e can be formulated as:

$$I_e = f_{MRRN}(x_1, x_2, x_3) \tag{3.3}$$

3.3.2 No-reference quality assessment module: Scorenet

In reality, the original image does not always exist. Therefore, full-reference metrics PNSR and SSIM cannot be used for assessing the distorted image quality. In this work, we propose a Deep Learning-based no-reference quality assessment module, called Scorenet, to guide the enhancement module. Our Scorenet simulates full-reference IQA metrics: estimating the difference between the distorted image and the reference image. Scorenet (shown in Figure 3.3) includes two main components: the Reference generative net G for generating the pseudo-original image and the Quality-score prediction network S predicting the quality score of the distorted image.

Given the distorted image x, our objective is to infer the quality score z. In full-

reference IQA tasks, quality scores can be calculated by comparing the distorted image and the reference image. Scorenet simulates the full-reference IQA by estimate the pseudo-original image $y' = f_{\theta_G}(x)$ where $f_{\theta_G}(x)$ indicates the learnt mapping of Reference generative net G . Predicting score z of an image x can be formulated as:

$$z' = f_{scorenet}(x) = f_{\theta_S}(x, f_{\theta_G}(x)) \tag{3.4}$$

where θ_G and θ_S denote the learnt parameters of G and S, respectively. In general, the Reference generative net G has the same purpose of the Quality Enhancement module MRRN: to generate undistorted images. We design the Reference generative net and the Quality Enhancement module to share some architectures and the loss function. The difference between the Reference generative net architecture and MRRN architecture is the number of inputs, where the prior takes one, the latter takes three.

We define the basic convolution layer in our network as $Conv(k, s)$ with k kernels size $s \times s$. For the Reference generative net G, the distorted image is first fed into two convolution groups, each group is defined as $Conv(32, 3) \rightarrow PReLU \rightarrow Conv(32, 3) \rightarrow PReLU$. In Reference generative net G, PReLU activation function follows all the convolution layers except the final one. We set stride and padding to one during convolution. The main part of the Reference generative net G is the Recurrent dense skip connection block (explained in section 3.3.3), also used in MRRN. Feature maps outputted from the Recurrent dense skip connection block are synthesized in the final $Conv(32, 3)$ producing an enhanced feature. This enhanced feature is added onto the distorted image x for a pseudo-original image y'.

In the quality-score prediction net S, $ReLU$ activation function follows all the convolution layer $Conv(32, 3)$. Our network, inspired by the VGG16 model [82], aiming to infer the quality score given the pair of a distorted image and the corresponding pseudo-original image. Image is first split into 224×224 patches with a stride of 40. A pair of 224×224 patches from the distorted image x and pseudo-original image

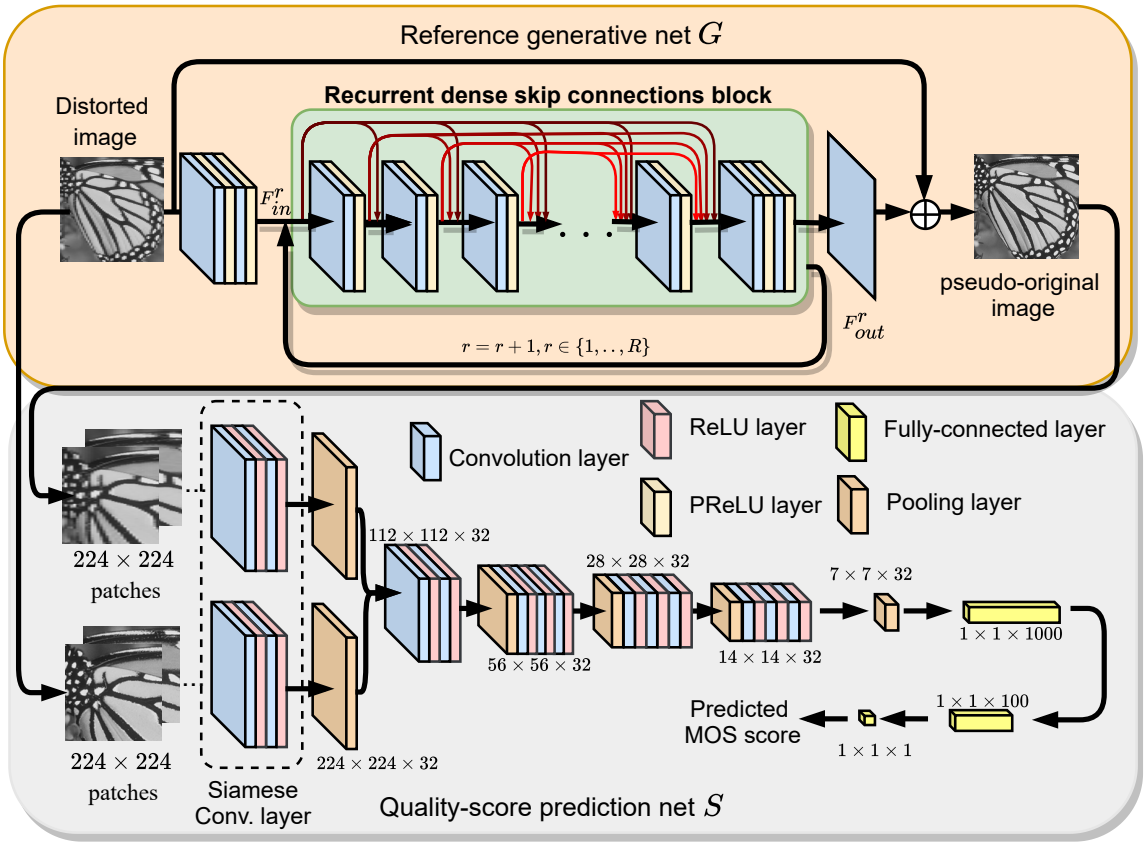


Figure 3.3: The proposed Scorenet architecture. A distorted image is inputted to the Reference generative net G for the corresponding pseudo-original image. Distorted image and the pseudo-original image are then split into 224×224 patches by a stride of 40 before being fed to Quality-score prediction net S. The final predicted score of the input image is the average score of all patches.

y' are fed to a group of Siamese convolution layer includes $Conv(32, 3) \rightarrow ReLU \rightarrow Conv(32, 3) \rightarrow ReLU$. We use a Siamese convolution on both the distorted and pseudo-original images to extract comparable feature maps. The rest of the net then focuses on finding the difference between these feature maps. Feature maps after Siamese convolutional layer will be pooled by a Pooling layer with a window of 2×2 to the size of 112×112. Pooled feature maps from two inputs will be concatenated before feeding into the network. The output from the final fully connected layer is the predicted Mean Opinion Score (MOS) score of a patch. The score of the entire image is the average score of all patches. $L2$ has been chosen as the loss function. Training S on N training

samples becomes minimizing loss function L_S:

$$L_S = \frac{1}{2N}\sum_{i=1}^{N}(z_i - f_{\theta_S}(x_i, y_i'))^2 \tag{3.5}$$

3.3.3 Quality enhancement component: Multiple-input Residual Recurrent Network (MRRN)

Recently, Li *et al.* introduce a recurrent neural network SRFBN [83] which has achieved outstanding results in image super-resolution tasks. Inspired by [83], this work introduces a feedback mechanism for enhancing the quality of input images. Our network architecture MRRN (Figure 3.2) includes the main branch, two supporting branches, a recurrent dense skip connections block, and a global skip connection for residual learning. x_1, x_2, and x_3 respectively denote the best, the second-best, and the third-best-quality images by order of the predicted MOS scores. The original image before CS encoding is denoted as y. The goal of our Quality Enhancement module is to learn the mapping f_{MRRN} between the input images x_1, x_2, x_3 and the target enhanced image y':

$$y' = f_{MRRN}(x_1, x_2, x_3; \Theta) \tag{3.6}$$

Input images are fed into a group of $Conv(n_f, 3) \rightarrow PReLU \rightarrow Conv(n_f, 3) \rightarrow PReLU$ in each branch. In the quality-score prediction net S, the first convolution group extracts the same features from the distorted image and the pseudo-original image. The rest of the network S aims to find the differences between these features. MRRN, on the other hand, aims to keep the diversity of extracted features from different input images. Therefore, convolution layers for each input are preferred over a Siamese convolution layer. We formulate the output feature map f at convolution layer l^{th} at

branch k as:

$$f_l^k = PReLU(f_{l-1}^k * w_l^k + b_l^k) \tag{3.7}$$

where w_l^k and b_l^k is respectively the learnt weight and bias of convolution layer l^{th} in branch k and f_0^k is $\{x_k, k = 1, 2, 3\}$. After the first two convolution layers, n_f feature maps in each branch will be concatenated to a convolution layer f_3:

$$f_3 = PReLU(cat(f_2^1, f_2^2, f_2^3) * w_3 + b_3) \tag{3.8}$$

where $cat(f_1, f_2, ...f_n)$ denotes the concatenate operation for the list of the feature maps $f_1, f_2, ...f_n$ on channel dimension. Later, the output from the 4^{th} convolution layer f_4 is input to the Recurrent dense skip connections block. Stride for every convolution layer is set as one. To keep the size of input image through doing convolution, all the paddings are set as $(s-1)/2$.

Feedback mechanism. We introduce the Recurrent Dense skip connections block (RDBlock) that integrates the two elements: depth and skip connections. In RDBlock, no information is omitted: the extracted low-level features F_{in}^r are reused in each loop and inside the RDBlock, and the high-level features F_{out}^{r-1} are also used to fine-tune the low-level feature input F_{in}^r of RDBlock. RDBlock does recurrence for R times, each $r = \{1, 2, ..., R\}$ corresponds to an output $\hat{y}_r$ of the network. In the loop r^{th}, RDBlock returns an output of F_{out}^r given a pair of F_{in}^r and F_{out}^{r-1}:

$$F_{out}^r = RDBlock(F_{in}^r, F_{out}^{r-1}) \tag{3.9}$$

F_{out}^{r-1} at $r = 1$ is set as F_{in}^r. For $r \geq 2$, the output feature maps F_{out}^r are stored and be concatenated with the F_{in}^r for the next loop until $r = R$. In training recurrent neural network, the feedback mechanism receives the information from previous loop $(r-1)^{th}$ to further improve the input of the recurrent mechanism. We denote g_0 is the

input of the RDBlock:

$$g_0 = cat(F_{in}^r, F_{out}^{r-1}) \tag{3.10}$$

Our RDBlock is built from nine convolution layers and dense skip connections. In each loop r^{th}, the input g_0 and feature maps from the lower-level layers are concatenated and fed to the higher-level ones. Let $g_{j,j\in[1,9]}$ denote output from convolution layer j^{th}. In our RDBlock, all layers j^{th}, excluding the final one, synthesize information from $(j-1)$ previous layers and the input g_0. The final layer g_9 takes only feature maps outputted by convolution layer g_8. The output of each convolution layer at layer $j^{th}, j < 9$ in the RDBlock defined as:

$$g_j = PReLU(cat(g_0, g_1, ..., g_{j-1}) * w_j + b_j) \tag{3.11}$$

Convolution layer j^{th} with $j \in [1, 8]$ in RDBlock takes an input of $(j+1) \times n_f$ concatenated feature maps and outputs n_f feature maps. In this work, we set the value of n_f to 32 for all convolution layer in RDBlock and the number of loop R is set as four. There is no PReLU activation for the final convolution of MRRN. The enhanced feature from each loop will be added onto the best-quality image input x_1 for the output $\hat{y}_r$. There is only one ground-truth label y_0 for R outputs of the network. In testing, only output from the final loop y_R is considered the result of the network. However, all the outputs $\hat{y}_r$ are used to calculate the loss function:

$$L(\Theta) = \frac{1}{M.R} \sum_{i=1}^{M} \sum_{r=1}^{R} I^r \parallel y_0^i - \hat{y}_r^i \parallel_1 \tag{3.12}$$

where M is the number of training samples, and Θ denotes the learned network parameters. L1 loss is used to optimize the network parameters. I^r is the importance of output r in the outputs list. Following [83], we set I^r to 1 for all the outputs.

3.4 Experimental results and comparison

3.4.1 Experiment settings

MRRN settings. For training MRRN, we use images with different sizes from DIV2K [84] for the training set and images BSD500 training set [85] for evaluating during training. Set 5 [86], Set 14 [87], Urban100 [88] and 200 images from the testing set of BSD500 [85] are used for testing. The Inverse Fast Walsh-Hadamard transform and binary Hadamard matrix are applied for transforming and measuring the input images. The other CS reconstruction algorithms and other matrices can replace the reconstruction algorithms and measurement matrix used in CSIE-M. Images in the training and testing sets are fed to the Scorenet to get the ranking scores and then fed to the Quality Enhancement module MRRN by order of quality scores.

Scorenet settings. For training Scorenet, we mostly focus on the dataset TID2013 [89], which is commonly used for learning no-reference IQA tasks. TID2013 contains a total of 3000 images generated by 24 types of distortions. We randomly divide reference images into 80% for training and 20% for testing as the no-reference IQA works have done [90]. Distorted images will go to the set that its reference image belongs to, ensuring no overlap between the training and testing sets. In testing, 20% images in the testing set are used for evaluating our Scorenet. Like the other IQA work [90], we evaluate the efficiency of our Scorenet via Spearman's Rank Order Correlation Coefficient (SROCC) and the Linear Correlation Coefficient (LCC). The higher SROCC and LCC represent the higher correlation between the predicted scores and the human-ranked MOS scores.

Training settings. Our experiments on the CSIE-M framework are conducted on Pytorch 1.0.0 with NVIDIA Tesla V100 GPUs' support. Adam optimization is used in training all the networks in CSIE-M. In training MRRN, the learning rate starts at 0.0001 and is divided by two every 300 epochs. The commonly used Peek-signal-to-

noise (PSNR) and Structural Similarity Index (SSIM) metrics are applied for evaluating quality enhancement results. For these metrics, the higher value indicates a better result. Four models for four sampling rates (SR) of 0.125, 0.25, 0.5, and 0.75 are trained and converged in different training epochs.

3.4.2 Ablation studies

Study on Scorenet. Table 3.1 presents the ten-time average of SROCC and PLCC metrics of our Scorenet compared to the widely used Full-reference IQA metrics PSNR and SSIM. Observe from Table 3.1, our Scorenet performs closer prediction to the human eyes (presented in MOS score) than the Full-reference IQA metrics PSNR and SSIM. In detail, we obtain SROCC relative improvements of 28.02% and 21.02% compared to PSNR and SSIM metrics, respectively. In the PLCC metric, improvements of 30.52% and 21.88% are achieved compared to PSNR and SSIM metrics. These results show an adequate ability of Scorenet to guide MRRN on enhancing the reconstructed image quality.

Table 3.1: SROCC and PLCC comparison on TID2013 dataset of our Scorenet compared to the widely used metric PSNR and SSIM. Note that Blue indicates the best result.

Metric	PSNR	SSIM	Scorenet (Ours)
SROCC	0.571	0.604	0.731
PLCC	0.593	0.635	0.774

To verify the effectiveness of the proposed Scorenet, Scorenet has been replaced by a random-based input ranking. We perform a ten-time random-based input ranking and average these results for comparison. Table 3.2 shows the quality comparison of using random-based ranking and Scorenet-based ranking. As observed from Table 3.2, applying Scorenet can gain up to 1.127 dB and 4.8 $\times 10^{-3}$ on average PSNR and SSIM

improvements compared to randomly setting the input images of MRRN when it comes to three-input MRRN. On the other hand, CSIE-M with multiple-input images and Scorenet can significantly improve the reconstructed image quality compared to using one input image or randomly set the order for multiple input images.

Table 3.2: PSNR and SSIM$\times 10^{-2}$ comparison of the entire proposal with and without Scorenet in N-input MRRN at sampling rate of 0.5.

Number of MRRN inputs	Max quality diff. between inputs	Random-based input ranking	Scorenet-based input ranking
	ΔPSNR / ΔSSIM	PSNR / SSIM	PSNR / SSIM
1	0.23 / 0.21	31.647 / 94.862	32.772 / 95.207
2	0.218 / 0.21	31.747 / 94.882	32.964 / 95.36
3	0.33 / 0.22	31.882 / 94.94	33.000 / 95.387
4	0.33 / 0.21	31.871 / 94.946	32.978 / 95.384

Study on the number of input images. This experiment validates the effect of the number of input images on the performance of enhancing CS reconstructed images. We train our dataset on single-input, two-input, three-input, and four-input MRRN. The N-input MRRN takes N images ranked by the No-reference quality ranking module. To fully evaluate the performance of N-input MRRN, reconstructing time is also considered besides PSNR and SSIM. Table 3.2 shows the PSNR and SSIM of N-input network, $N \in \{1, 2, 3, 4\}$. We choose a sampling rate of 0.5 for validating this experiment. Obtain from Table 3.2, the three-input MRRN obtains the highest performance compared to the other N-input MRRNs. From single-input MRRN to two-input MRRN, average PSNR and SSIM have significantly increased by 0.19 dB and 0.002, respectively. It is well-known in learning deep networks that the more representations are obtained, the better results can be achieved. In learning to enhance CS reconstructed images, other supporting images x_2 and x_3 can be considered the representations obtained from special filters: CS reconstruction algorithms. Moreover, the reconstructing time increases the amount of five milliseconds from single-input MRRN to two-input

Table 3.3: CSIE-M performance in terms of PSNR and SSIM comparison on difference combinations on CS reconstruction algorithms.

SR	CSIE-M		CSIE-M*		CSIE-M**	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
0.125	26.824	0.825	26.815	0.813	26.811	0.786
0.25	27.747	0.864	27.612	0.849	27.691	0.849
0.5	33.000	0.953	32.924	0.949	32.951	0.948
0.75	38.240	0.984	38.193	0.982	38.186	0.982
SR	CSIE-M***		CSIE-M****		CSIE-M*****	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
0.125	26.823	0.814	26.781	0.813	26.823	0.814
0.25	27.665	0.849	27.707	0.853	27.746	0.853
0.5	32.905	0.949	32.982	0.949	32.981	0.949
0.75	38.198	0.982	38.201	0.982	38.164	0.982

Note: Blue indicates the best result and orange indicates the second best result. Reconstruction algorithms for inputs: CSIE-M: L1EQPD, SPGL, OMP, and SAMP. CSIE-M*:TwIST, StOMP, GroupBP, and CoSaMP. CSIE-M**: SAMP, OMP, CoSaMP, and StOMP. CSIE-M***: GroupBP, SPGL, L1EQPD, and TwIST. CSIE-M****: SAMP, OMP, TwIST, and GroupBP. CSIE-M*****: CoSaMP, L1EQPD, SPGL, and StOMP.

MRRN. That has shown the advantage of multiple-input MRRN compared to single-input MRRN: the image quality is significantly improved, and the running time slightly increases. We report the small differences in performance of multiple-input CSIE-M networks, and the best performance, over the test images, belongs to the three-input MRRN.

Performance on other CS reconstruction algorithms. We present the CSIE-M performance on different CS reconstruction algorithms in Table 3.3. In this experiment, four other CS reconstruction algorithms from convex optimization algorithms include Two-step iterative shrinkage/thresholding (TwIST) [91] and Group-sparse basis pursuit (GroupBP) [92], and from greedy algorithms include Compressive Sampling Matching Pursuit (CoSaMP) [22] and Stagewise Orthogonal Matching Pursuit

Table 3.4: Study on numbers of the recurrent iteration R of the network under PSNR and SSIM $\times 10^{-2}$ at sampling rate 0.5.

Dataset	R = 1		R = 2		R = 3	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Set 5	37.485	97.160	37.552	97.173	37.559	97.177
Set14	33.657	94.920	33.673	94.920	33.668	94.940
BSD500	33.278	95.429	33.318	95.452	33.371	95.493
Urban100	31.544	94.893	31.632	94.945	31.773	95.056
Average	32.817	95.266	32.871	95.297	32.949	95.358
Dataset	R = 4		R = 5		R = 6	
	PSNR	SSIM	PSNR	SSIM M	PSNR	SSIM
Set5	37.596	97.19	37.582	97.196	37.601	97.198
Set14	33.740	94.927	33.767	94.948	33.765	94.953
BSD500	33.403	95.509	33.421	95.525	33.424	95.521
Urban100	31.860	95.115	31.923	95.155	31.932	95.145
Average	33.000	95.387	33.032	95.410	33.036	95.400

Note: Average are calculated over the number of images. Blue indicates the best result and orange indicates the second best result.

(StOMP) [93] have been used. Over the testing sets, quality difference between images reconstructed by the above algorithms is up to 1.5 dB, 2.07 dB, 3.22 dB, 2.79 dB PSNR, while the CS reconstruction algorithms used in the original proposal is up to 0.04 dB, 0.07 dB, 0.33 dB, and 0.2 dB PSNR at sampling rates of 0.125, 0.25, 0.5, and 0.75, respectively.

Table 3.3 illustrates the quantitive results (PSNR and SSIM) comparison among the combinations of CS reconstruction algorithms: the proposed CSIE-M, the combination of the other four CS reconstruction algorithms (CSIE-M*), the combination of the greedy algorithms (CSIE-M**), the combination of convex optimization algorithms (CSIE-M***), the other combination of CSIE-M used greedy algorithms and other convex optimization algorithms (CSIE-M****), the combination of CSIE-M used convex

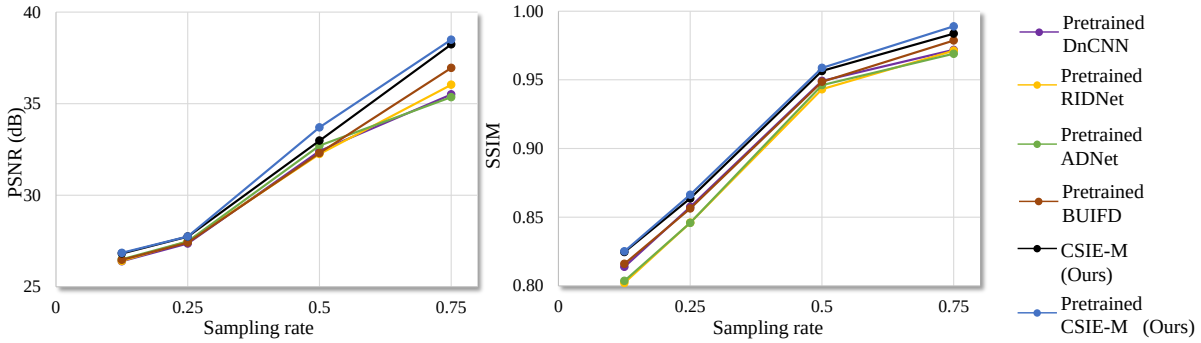


Figure 3.4: PSNR and SSIM comparison to the pretrained state-of-the-art in distorted image enhancement under sampling rates of 0.125-0.75.

optimization algorithms and the other greedy algorithms (CSIE-M*****). It can be seen that CS reconstruction algorithms used in CSIE-M obtain the highest quality in terms of PSNR and SSIM over all the sampling rates. Generally, the other combinations reduce the quality of the enhanced images up to an amount of 0.136 dB PSNR and 3.8×10^{-2} SSIM on average. Also, note that even though two CS reconstruction algorithms are shared between some groups, the results are different since the input images fed to the network are different. For example, OMP, SAMP, and L1EQPD reconstructed images of *foreman* (Set14) have been fed into MRRN as the best-quality, the second-best quality, and the third-best quality image, respectively. Meanwhile, the feeding order in CSIE-M** is respectively StOMP, OMP, and SAMP reconstructed images. It concludes that choosing top-three CS reconstruction algorithms and the order of input images of MRRN take a high responsibility on generating high-quality images.

Study on numbers of iteration R of the Recurrent dense skip connections block. In this experiment, we find the correlation between R and network performance. The investigation is conducted on the three-input network under the sampling rate of 0.5. We separately train five CSIE-M models in which the recurrent iteration R is set as 1, 2, 3, 4, 5, and 6. In Table 3.4, a clear trend is performed: the larger R, the better quality image. At $R = 1$, the network is considered a traditional neural network. It obtains the lowest image quality compared to using the recurrent neural network where

$R \geq 2$. The reconstructing time between different Rs is about four ms if the number of loops is increased by one. This experiment will consider the performance and training time of different Rs. For $R = 1$, the network takes 45 hours to converge. For recurrent network $R \geq 2$, it takes about three days for training the network whose recurrent iteration is 2, 3, and 4. For $R = 5$, it takes more than three days to converge on the DIV2K dataset. Observe from Table 3.4, the network with $R = 5, 6$ shows the best performance; however, on the other hand, the training time increases. The recurrent iteration of four has average performance while the complexity is in the middle. The following experiments take a feedback iteration of four for analysis.

3.4.3 Overall results.

Table 3.5 shows our results in ΔPSNR and ΔSSIM$\times 10^{-2}$ of our proposal CSIE-M compared to the main input x_1 since x_1 takes the highest responsibility in generating the enhanced image. Generally, CSIE-M strongly improves the quality of CS reconstructed images. In the PSNR evaluation metric, the best and the lowest improvements belong to the sampling rate of 0.75 and 0.25, respectively. In words, CSIE-M improves an average PSNR of 2.01 dB, 1.88 dB, 3.89 dB, and 8.07 dB for the sampling rates of 0.125, 0.25, 0.5, and 0.75, respectively. Under the SSIM evaluation, the best and the lowest improvements belong to the sampling rate of 0.125 and 0.25, respectively. Notably, CSIE-M shows outstanding results on dataset Urban100 which contains aliasing edges and complex scene structures. We obtain improvements of 4.71 dB PSNR and 7.04×10^{-2} SSIM average all the sampling rates on Urban100.

Comparison to the State-of-the-art: For a fair comparison, we retrain the related work models DnCNN [31], RIDNet [73], CACNN [72], ADNet [74] and BUIFD [77] on our training set. For the image denoising works [31, 73, 74, 77], we set the CS reconstructed images as the noisy input image, and the original image is set as the clean ground-truth. For enhancing the decoded image by HEVC [72], we assign

Table 3.5: ΔPSNR and ΔSSIM$\times 10^{-2}$ comparison to the state-of-the-arts.

Set	SR	RIDNet ICCV'19 [73]		CACNN TIP'19 [72]		ADNet N.N. '20 [74]		BUIFD TIP'20 [77]		CSIE-M (ours)	
		ΔP	ΔS	ΔP	ΔS	ΔP	ΔS	ΔP	ΔS	ΔP	ΔS
Set5	1/8	3.23	6.72	3.06	5.94	3.37	6.44	3.38	6.31	3.71	6.76
	1/4	3.02	4.96	2.61	4.09	3.01	4.91	3.02	4.59	3.46	5.14
	1/2	4.30	2.85	4.65	2.82	4.79	2.86	4.42	2.74	5.05	3.04
	3/4	5.86	2.86	7.31	2.99	5.80	2.77	7.11	2.90	8.59	3.28
Set14	1/8	2.08	5.72	2.00	5.26	2.14	5.75	2.14	5.50	2.32	5.96
	1/4	1.90	3.86	1.63	3.34	1.83	3.77	1.85	3.62	2.08	4.18
	1/2	3.18	3.66	3.37	3.58	3.45	3.67	3.13	3.49	3.62	3.82
	3/4	5.15	4.09	6.48	4.45	4.88	3.91	6.61	4.55	7.75	4.87
BSD500	1/8	1.45	4.93	1.41	4.64	1.50	4.97	1.51	4.93	1.65	5.3
	1/4	1.24	3.12	1.13	2.80	1.26	3.12	1.24	3.11	1.46	3.57
	1/2	2.81	3.95	3.03	3.80	3.15	4.14	2.91	3.77	3.40	4.15
	3/4	5.37	4.69	6.57	4.80	5.01	4.44	6.40	4.70	7.73	5.19
Urban 100	1/8	2.10	7.37	2.07	6.88	2.24	7.73	2.14	7.18	2.60	8.40
	1/4	2.09	5.71	1.82	4.80	2.09	5.71	1.91	5.19	2.61	6.65
	1/2	3.59	5.37	4.04	5.52	4.32	5.98	3.57	5.15	4.86	6.16
	3/4	5.53	5.81	6.77	6.15	5.36	5.63	7.27	6.26	8.78	6.95
Average	1/8	1.71	5.76	1.67	5.39	1.79	5.89	1.77	5.68	2.01	6.32
	1/4	1.56	4.00	1.39	3.47	1.57	3.99	1.51	3.81	1.88	4.59
	1/2	3.09	4.36	3.38	4.31	3.56	4.67	3.15	4.18	3.89	4.75
	3/4	5.42	4.99	6.64	5.18	5.13	4.76	6.69	5.16	8.07	5.70

Note: Blue indicates the best result and orange indicates the second best result.

the network to take an input and a ground truth of reconstructed images and the corresponding original image, respectively. All the settings are default. No noise or preprocessing is added to the training data of these related works. Since MRRN and related works' inputs are from different CS reconstruction algorithms, it is better to compare how much quality improvements each approach can obtain. In this comparison

experiment, we assume the highest quality image in PSNR for the related works. While the difference in image quality of the related works' input and CSIE-M main input x_1 is 0.02dB PSNR, the proposal significantly improves the image quality in PSNR and SSIM. In other words, the related works increase 1.39 dB to 6.69 dB, where ours is 1.88 dB to 8.07 dB in average PSNR improvement over the testing images. In terms of SSIM, our improvement is 4.59×10^{-2} to 6.32×10^{-2} while the others are from 1.45×10^{-2} to 5.68×10^{-2} on average. Our proposal shows significant improvements compared to the related works in a high sampling rate of 0.75 and complex structures such as edge and aliasing in the test set Urban100. At a sampling rate of 0.75, CSIE-M scores an improvement of 8.07 dB and 5.7 $\times10^{-2}$ in terms of PSNR and SSIM, where the related works are up to 6.69 dB PSNR and 5.16 $\times10^{-2}$ SSIM improvement in average.

We also make a comparison to the pre-trained related works (Figure 3.4). In this experiment, we compare our models using training from scratch and using the pre-trained model with the related works [31, 73, 74, 77]. Noisy images from DIV2K and Flickr2K created by BUIFD [18] at noise levels of 10, 15, and 20 are first used for training MRRN. After convergence, the pre-trained model is used as initialized network weights for training on our dataset. Pertaining to applying transfer learning to MRRN, it has been recorded that image quality improvements have been increased by averages of 0.13dB PSNR and 4×10^{-3} SSIM compared to training from scratch. Compared to the related works, both pre-trained and training-from-scratch CSIE-Ms obtain 0.3 dB - 2.9 dB PSNR and 0.8×10^{-2} to 2.3×10^{-2} SSIM improvements compared to the related works.

We perform the Rate-distortion (RD) curves to visualize the coding performance of CSIE-M compared to the related works (Figure 3.5). In these charts, the horizontal axis and vertical axis indicate the four sampling rates and the corresponding PSNR of the images refines by CNN. We randomly pick images from Set5, BSD500 testing set, and Urban100. It can be seen that at each sampling rate, our proposal CSIE-M exceeds the

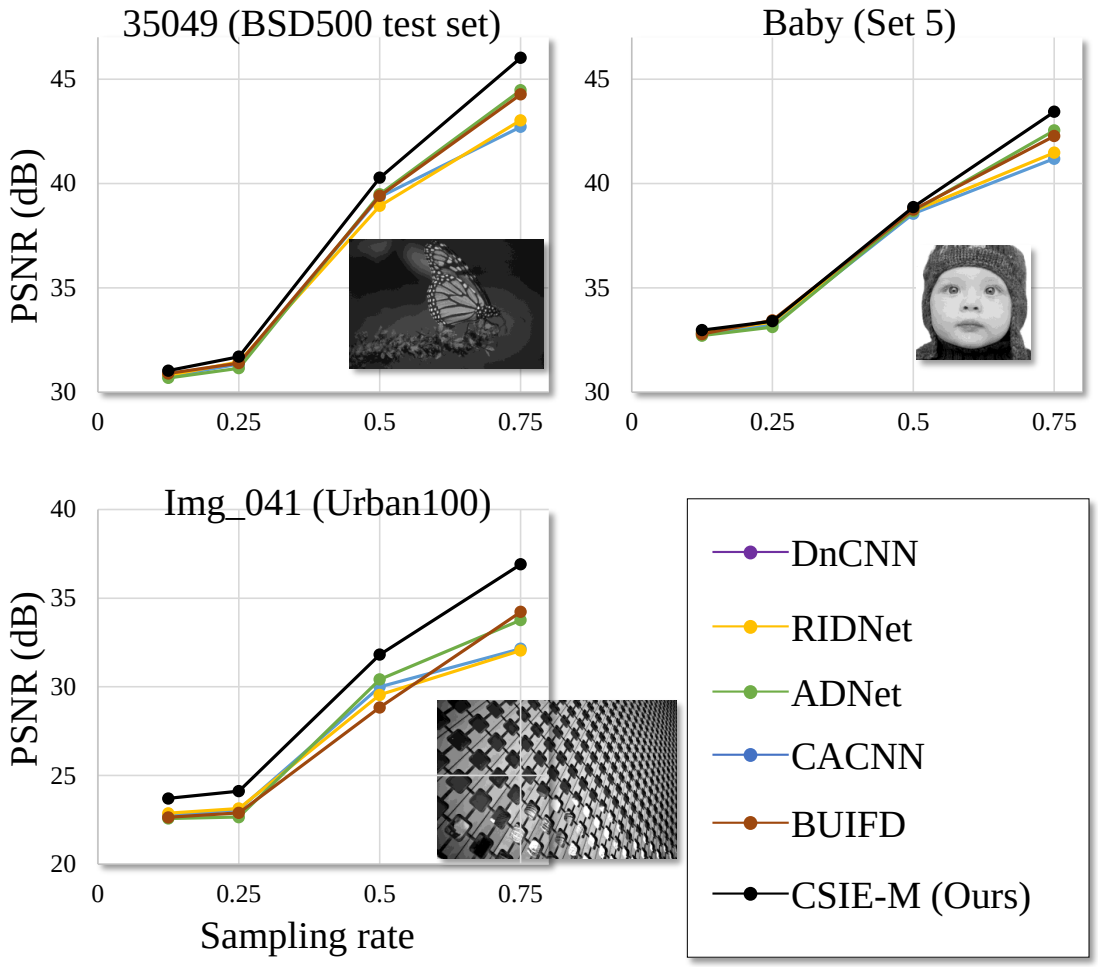


Figure 3.5: RD-curves of CSIE-M and related works on sampling rates 0.125-0.75.

related works in the PSNR metric. Generally, the RD curves of CSIE-M and the related approaches are distinct. In the complex scene of img_004 and img_041 of Urban100, there are significant improvements compared to the related works. Moreover, the improvement of CSIE-M is more stable than related works. In img_041, BUIFD shows third place in terms of PSNR at sampling rates of 0.125 and 0.25, sixth place at sampling rate 0.5, and second place at sampling rate 0.75.

Figure 3.6 visualizes the CS enhanced images by CSIE-M and the related works under sampling rates of 0.125-0.75. From top to down, every two rows are images reconstructed from sampling rates of 0.75 to 0.125. Observe from figure 3.6, CSIE-M returns the enhanced image with sharper edges than the related works. CSIE-M removes compression artifacts such as ringings and aliasing. At the low sampling rates

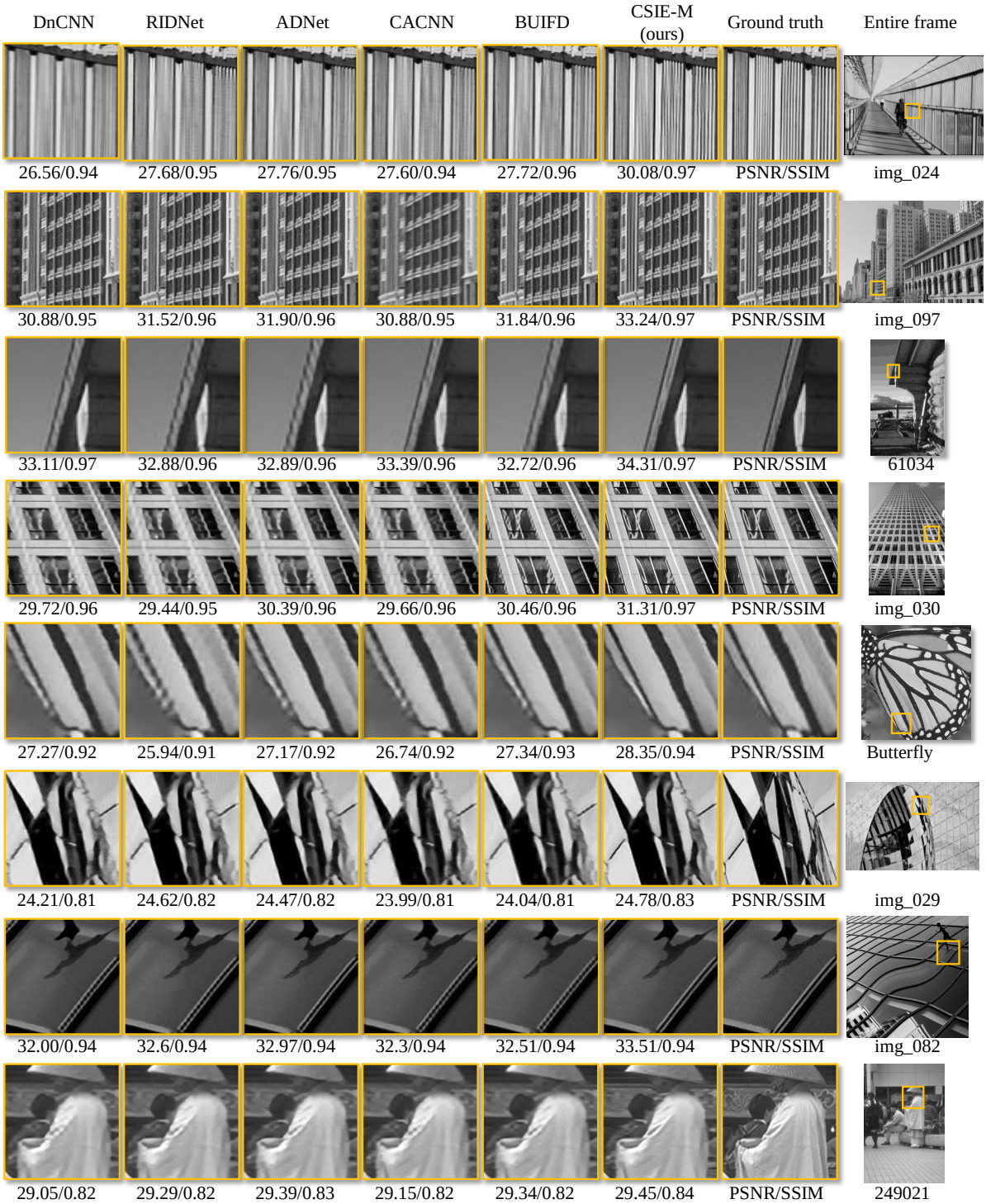


Figure 3.6: Visualizations of reconstructed images enhanced by DnCNN [31], RIDNet [73], CACNN [72], ADNet [74], BUIFD [77], CSIE-M (ours) under the sampling rates of 0.125-0.75, and the ground-truth label.

of 0.125 and 0.25, CSIE-M sharpens the local details such as blurring edges and aliasing. At high sampling rates of 0.5 and 0.75, where more information is sent to the decoder, CSIE-M tends to produce better structures in the enhanced images.

3.5 Chapter conclusion

Different from the other image compression standards, such as JPEG and JPEG2000, CS can reconstruct many images with different qualities. Using this property, this is the first time to propose a deep learning-based compressive sensing image enhancement framework using multiple reconstructed signals. In the decoder, reconstructed images are scored and ranked by a No-reference quality ranking module before feeding to the quality enhancement module. In the quality enhancement module, low-level and high-level features extracted from CS reconstructed images are exploited and enriched by the proposed Recurrent dense skip connections block. As a result, 1.88-8.07dB PSNR improvements under the sampling rates of 0.125-0.75 have been obtained. We further experimented on the effectiveness of CSIE-M with and without the No-reference quality ranking module. The result shows that 1.127 dB PSNR can be improved when using the No-reference Quality enhancement module. Moreover, our framework CSIE-M, which utilizes multiple-input images to enhance the reconstructed image quality, has outperformed the one-to-one learning networks. The proposal can be integrated into IoT imaging systems to enhance the CS reconstructed images, giving better visual quality for end-users and a promising approach for designing AIoT systems.

Chapter 4

In-loop filtering image enhancement for video encoder-decoder

4.1 Introduction

Common video coding standards divide input video into different levels for prediction. Input video is first split into group of pictures, each group could consist of intra frame (I frame), and predictive coded picture (P frame) or bipredictive coded picture (B frame). For more accurate compensation, common video coding standards perform either intra or inter coding on current frame to be encoded. In inter coding, motion compensated prediction (MCP) first searches for the best matching block among the already reconstructed frames in the first reference picture list (List 0) and the second reference picture list (List 1). Fig. 4.1 demonstrates the motion search video coding. MCP aims to predict the current frame from the reference frames which are previously reconstructed and store the residual along with the motion vector between the corresponding blocks, which benefits for reducing the temporal redundancy in inter coding. Since converting signals from the analog domain to the digital domain may omit some data, there is no perfect matching block for compensating the current block. Although multiple coding modes such as fractional interpolation [14] and affine motion estimation [94, 95] have been proposed for better inter block prediction, there is rarely a complete fit for this data.

Beside MCP, lossy compression technique quantization is another main factor causing the discontinuities on reconstructed images. Fig. 4.2 illustrates the process of quantization and dequantization in both encoder and decoder. After transform and

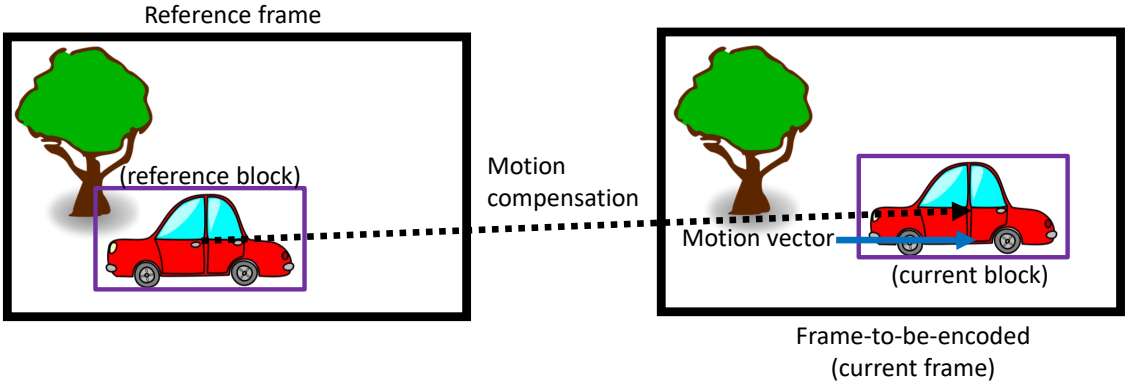


Figure 4.1: Motion compensation in inter coding. Block-based inter coding increases the discontinuities on reconstructed images.

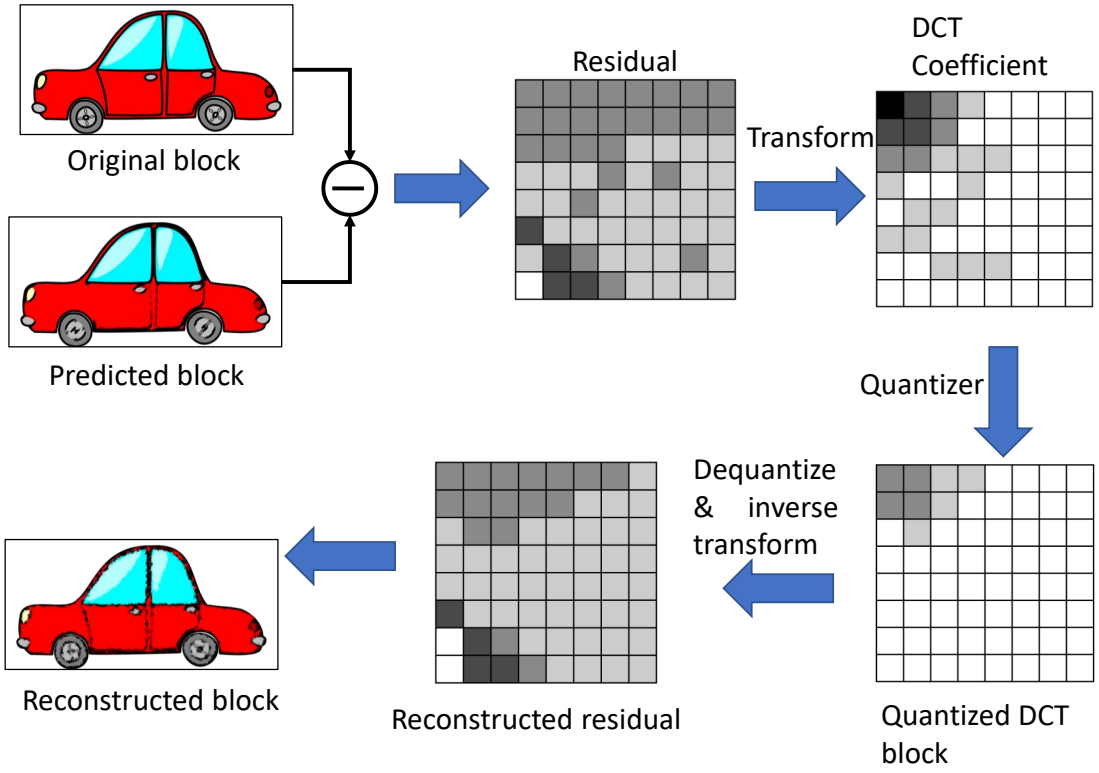


Figure 4.2: Lossy compression technique, quantization, reduces the possibility for fully reconstructing current block.

quantization, signals can not be fully reconstructed, leaving artifacts on reconstructed images due to the loss of high frequency information.

The upcoming video coding standard Versatile Video Coding (VVC) [2] has outperformed its predecessor High Efficiency Video Coding (HEVC or H.265 [12]). Although

gaining 30-50% bitrates saving over HEVC, VVC coding performance is still affected by block-based coding and lossy compression that causes the missing information and coding distortions such as edges, blurring, ringing artifacts on the reconstructed images. The Joint Video Team (JVT) of ITU-T investigates in-loop filtering (ILF) includes deblocking filter (DBF), sample adaptive offset (SAO), and adaptive loop filter (ALF) to reduce these artifacts. Compared to its predecessor, the previous state-of-the-art (SOTA) High Efficiency Video Coding (HEVC or H.265) [12], VVC can perform up to 50% bitrate reduction at the same reconstructed image quality. Although remarkable coding performances have been obtained, these modern video coding standards still suffer from undesirable blocking, noisy, and blurry artifacts on final reconstructed frames. Therefore, in-loop filtering (ILF) has been introduced to remove the coding artifacts of reconstructed images. In Advanced Video Coding (AVC or H.264) [11], the prior video coding standard to HEVC, ILF adopts deblocking filter (DBF) for removing blocking artifacts. In HEVC ILF, sample adaptive offset (SAO) [15] is exploited after DBF. Later on, luma mapping with chroma scaling (LMCS) and adaptive loop filter (ALF) [96] are introduced besides DBF and SAO in the ILF of VVC. These ongoing developments demonstrate that there is always a space to improve further the designed ILF presented in lossy video coding standards.

In recent years, deep learning (DL) and its notable representative convolutional neural network (CNN) have become prominent approaches to reconstructed image enhancement. Dong *et al.* [34] first introduced an end-to-end deep convolution neural network for compression artifact removal, uncovering the ability of CNN in removing compression artifacts and enhancing reconstructed images. Later on, different CNN architectures are used for further improving the coding efficiency of modern video coding standards [60, 97–100]. For more accurate pixel compensation, recent researches [72, 101–104] have introduced DL-based ILF enhancement at the coding tree unit (CTU) level, aiming to refine the reconstructed image to local levels compared to

the entire frame. At each CTU, the rate-distortion (RD) costs for each ILF mode are calculated for the default ILF of VVC and the DL-based filters. The chosen mode, whose RD-cost is the lowest among the candidate modes, is used for filtering the current CTU. For signaling the decoder about the ILF mode of current CTU, bit(s) indicates the ILF mode is compressed under a syntax element defined in Context-adaptive binary arithmetic coding (CABAC) mode. It is apparent that current studies have not made the most of coding information, leaving space for further improvements.

In this work, we take advantage of the encoder and decoder resources to enhance the VVC reconstructed images from the local regions, thus improving the entire image quality. This work first introduces the Spatial-Temporal In-loop filtering (STILF) [1] for VVC: the default ILF of VVC, our Self-enhancement CNN with CU map (SECUM), and our Reference-based enhancement CNN with Optical Flows (REOF). Besides a VVC reconstructed image, SECUM takes the CU partition information in the encoder as one of the inputs. In [104], the authors proposed Reference Frame Selector (RFS) for multi-frame enhancement. In this work, we take a different approach to defining REOF. Reference picture sets including List 0 and List 1 defined by VVC are used as references to avoid the extra computations and keep the references specified by the coding standards. Compared to the related works, we explore and utilize *more potential resources at the encoder and the decoder*. These resources include reference picture sets, List 0 and List 1, reference pictures, and their picture order count (POC) defined by the video coding standards. After filtering, instead of performing RD cost-based ILF mode selection, this work exploits and performs the autonomous mode selection (AMS) using a reinforcement learning approach (STILF-AMS). Our agent is trained to predict the ILF mode of each and every CU in the CU partition. Besides, we adopt split actions on CUs, allowing us to exploit deeper coding unit levels. By these designed filtering and split actions, STILF-AMS can refine the pixel value *to the local levels that are smaller than the CU partition of VVC without storing any bits for indicating ILF mode*, thus

optimizing both rate and distortion. As a result, we outperform VVC and the state-of-the-art in deep learning-based ILF enhancement. Remarkably, up to 18% and an average of 5.9% bitrate savings have been obtained under all configurations. The main contributions of this work are summarized as follows:

- This work utilizes various coding information and presents an adaptive Spatial-Temporal ILF for the SOTA video coding standard VVC. Besides the ILF from VVC, STILF-AMS proposes a set of ILFs: SECUM network exploring the spatial information within the frame, and REOF network exploiting the temporal correlations among frames.

- We propose a reinforcement learning-based autonomous ILF mode selection (AMS) scheme, enabling the ability to adapt to different coding levels. Our agent network is trained to predict the new CU partition and the ILF mode for each CU. By predicting the ILF mode, we cost *zero bit* for signaling the decoder. Exploring ILF at CU levels, STILF-AMS has gone over the CTU-wise ILF selection presented in the existing works.

- STILF-AMS gives a state-of-the-art performance compared to other deep learning-based ILF enhancements for modern video coding standards. Moreover, our STILF-AMS is CU size-independent and requires absolutely no additional design to adapt to different video coding standards.

The rest of this paper is organized as follows. Section 4.2 provides a brief overview of the deep learning-based ILF for current video coding standards. Section 4.3 presents the proposed STILF-AMS in details. Finally, we report the experimental results in Section 4.4 and conclusion in Section 4.5.

4.2 Related Works

4.2.1 Deep learning-based In-Loop filtering for video coding

Besides the built-in ILF, deep learning approaches are widely used in enhancing reconstructed images by video coding standards. Recent research targets designing deeper and more complex networks for better quality compensation. In [99], divergence and second derivative of the input image are used as the attention mechanisms in enhancing VVC reconstructed video frames. The works [97, 98] adopt global skip connections for enhancing VVC reconstructed images. In [60], a residual highway convolutional neural network (RHCNN), structured from residual highway units and convolutional layers, is proposed to improve the performance of HEVC. However, directly replacing or integrating ILF on top of video coding standards might not take full advantage of different in-loop filters. Switch mode-based ILF strengthens the conventional ILF with deep learning-based filters. In [101], Squeeze-and-Excitation Filtering CNN (SEFCNN) consists of Feature EXtracting subnet and Feature ENhancing subnet, objects to enhancing the default ILF in HEVC. To reduce the performance loss of a single CNN, Jia *et al.* propose a content-aware CNN [72] with a switch-mode at CTU level to improve the adaptability of ILF. Huang *et al.* propose an adaptive reinforcement learning-based ILF (ALRF) [103] for improving the ILF of VVC. Although a network selection is performed, a final RD cost-based CTU-wise ILF mode selection following the filters selection eventually increases the bitrate. In the work [104], the authors introduce multi-frame ILF with a switch-mode selection for reducing compression artifacts, thus improving coding efficiency. The coding mode is chosen from the default ILF, the single frame enhancement, or the multi-frame enhancement. For multi-frame enhancement, reconstructed frames are selected by the reference frame selector (RFS) before being used for enhancing the current image. Since the latest video coding standard VVC on its developed, there have been many studies on enhancing the in-loop

filters [97–100, 105, 106]. In [97, 98], global skip connection is used for learning the residual between the image distorted by VVC encoding and the raw video frame. In [100, 106], skip connections have play viral role in designing the dense residual convolutional neural network based in-loop filter (DRNLF) [100] and dense residual convolutional neural network (DRN) [106] for VTM reconstructed image enhancement. The authors of [105] designed an Attention-based Dual-scale CNN (ADCNN) to improve the quality of luminance and chrominance components for VVC decoded images. During the feedforward, encoding information such as Coding Unit (CU) map and quantization parameter (QP) have been used as the attention mechanisms. Dai *et al.* [107] proposed Variable-filter-size Residue-learning CNN (VRCNN) that replace HEVC Deblocking filter and SAO.

Although distortion is well handled at the CTU level, the switch mode-based approach falls to two main drawbacks. First, the existing works can only perform on CTU level, which is fixed to 64×64 in HEVC and 128×128 in VVC while blocking artifacts mainly appear on CU level. Second, to adapt to smaller levels than CTU, such as coding unit (CU), bits for indicating ILF modes are required. Eventually, this approach is not bitrate-friendly. A possible solution for deciding an ILF mode at local levels is to train a network to predict the best ILF mode given reconstructed pixel values. The following Section will summarize some related works on deep learning-based mode decisions in video coding.

4.2.2 Deep learning-based mode decision in video coding

Mode decision is one of the key components that bring high coding efficiency to the video coding standards. It is apparent that the more modes presented, the more possibility that a coding unit is better predicted. However, this coding efficiency is traded by very high computational complexity. CNN-based intra-mode predictions [108, 109] have been presented for replacing the conventional 35 intra-coding modes in HEVC:

33 angular intra prediction modes, the DC mode, and the planar mode. To reduce the complexity of attempting different coding modes, Chen *et al.* propose a low-complex shallow asymmetric-kernel CNN (AK-CNN) [110] for both fast CU partition and fast intra-mode decision. In [111], a modified ResNet18-CNN structure name CtuNet is modified for reducing the computational complexity of HEVC intra-prediction mode. Xu *et al.* introduce deep learning approaches for predicting CU partition, aiming to reduce the HEVC complexity in intra and inter-mode search [112, 113]. Kuanar *et al.* [114] predicts the CU size in HEVC by CNN-based region-wise feature learning and classification. Even though such approaches significantly reduce video coding complexity, they suffer from low coding efficiency. In this work, we combine and solve the problems of low coding efficiency in DL-based mode decision [108–114] and rate increase problem [1, 60, 72, 97–101, 103, 104] when performing ILF mode prediction on deeper CU levels.

4.3 Spatial-Temporal In-loop Filtering with Autonomous mode selection (STILF-AMS): the proposal

4.3.1 Overview the proposed STILF-AMS

Fig. 4.3 illustrates the overview of the proposed STILF-AMS integrated into the VVC video coding framework. Blue arrows and color blocks indicate our implementations integrated into the original VVC presented in black arrows and white blocks. VVC reconstructed frame is first filtered by the default ILF before feedforward to our self-enhancement CNN with CU map (SECUM) and the Reference-based enhancement CNN with optical flows (REOF). Different from other multi-ILF mode selection works, STILF-AMS performs no RD-cost and stores no bit for different filtered images. In-

stead, we present a reinforcement learning-based autonomous mode selection (AMS) for predicting the ILF mode of each and every CU in the reconstructed image. After filtering by three ILF in the network set, images will be fed to the agent network. A proposed agent network decides which CU partition the image should be split and how to filter these CUs, allowing ILF to run on CU level without storing ILF bits. Moving towards this goal, we additionally introduce a set of splitting besides filtering, allowing images to be filtered to the smallest possible CU level.

4.3.2 The proposed network set: STILF.

In this Section, we briefly introduce the two proposed networks in the filter set. A feedback mechanism, namely recurrent dense skip connection block (RDSCB), is proposed for SECUM and REOF to fully exploit the information from reconstructed images. This Section will first introduce the shared RDSCB architecture and then detail the specific designs for SECUM and REOF architectures.

Feedback mechanism: Recurrent dense skip connection block

In this work, a shared architecture named Recurrent dense skip connection is introduced. Recent researches have shown the strength of deep networks in enhancing the quality of the compressed images and videos [60, 72]. However, the deeper network requires a larger number of parameters. To overcome this drawback, [83, 115] introduce recurrent neural networks for single image super-resolution. The studies on recurrent neural networks have shown outstanding results while maintaining the number of network parameters. Inspired by these works, we propose a feedback mechanism where high-level features are used for refining the low-level features. Moreover, dense skip connections are also investigated inside the recurrent mechanism for passing rich spatial information from low to higher-level layers. Our RDSCB (as shown in Fig. 4.4) includes n convolutional layers, layer l^{th} takes $(l+1) \times n_f$ feature maps including out-

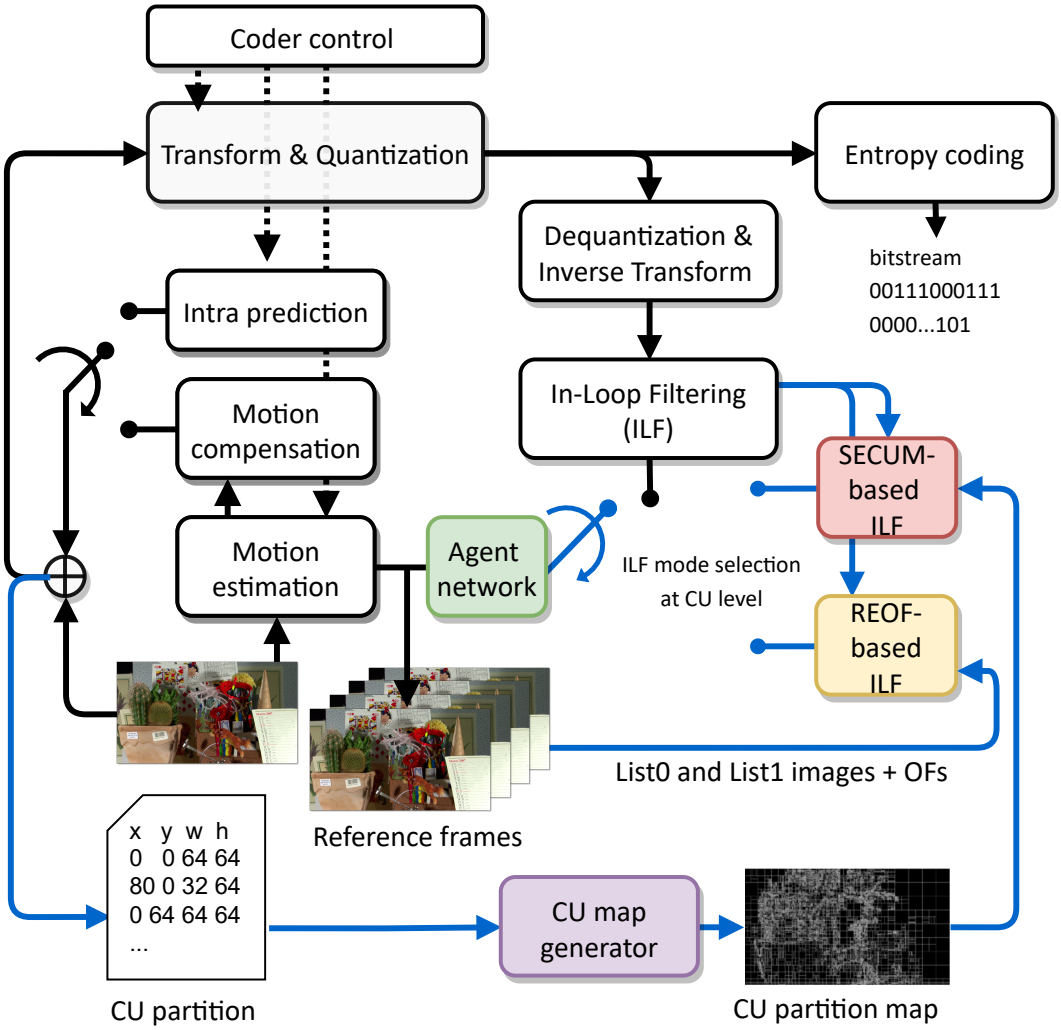
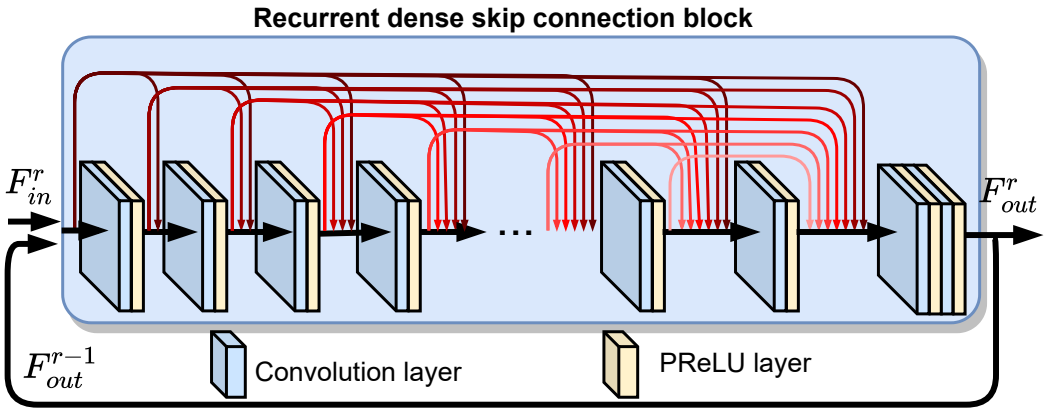


Figure 4.3: Integration of the proposed Spatial-Temporal In-Loop Filtering (STILF-AMS) to video coding VVC. STILF-AMS includes two main components: the filter set and the agent network. First, reconstructed images are sequentially filtered by the proposed ILF set consisting of the default ILF, the SECUM network, and the REOF network [1]. After filtering, the agent network performs and predicts the new CU partition and ILF mode of each CU.

puts from $l-1$ prior convolution layers and $2{\times}n_f$ feature maps from RDSCB input. This holds true for all layers except the final one which takes only n_f feature maps as the input. During feedforward, RDSCB does the feedback for R times. The output F^r_{out} at loop $r^{th}, r \in \{1, ..., R\}$ of RDSCB is concatenated with input F_{in} feature maps to be the input for the next loop $(r+1)^{th}$. The first loop $r = 1$ has no feedback F^r_{out}, we then duplicate input feature maps to be RDSCB block input.



$$r = r + 1, r < R$$

Figure 4.4: Recurrent dense skip connection block architecture.

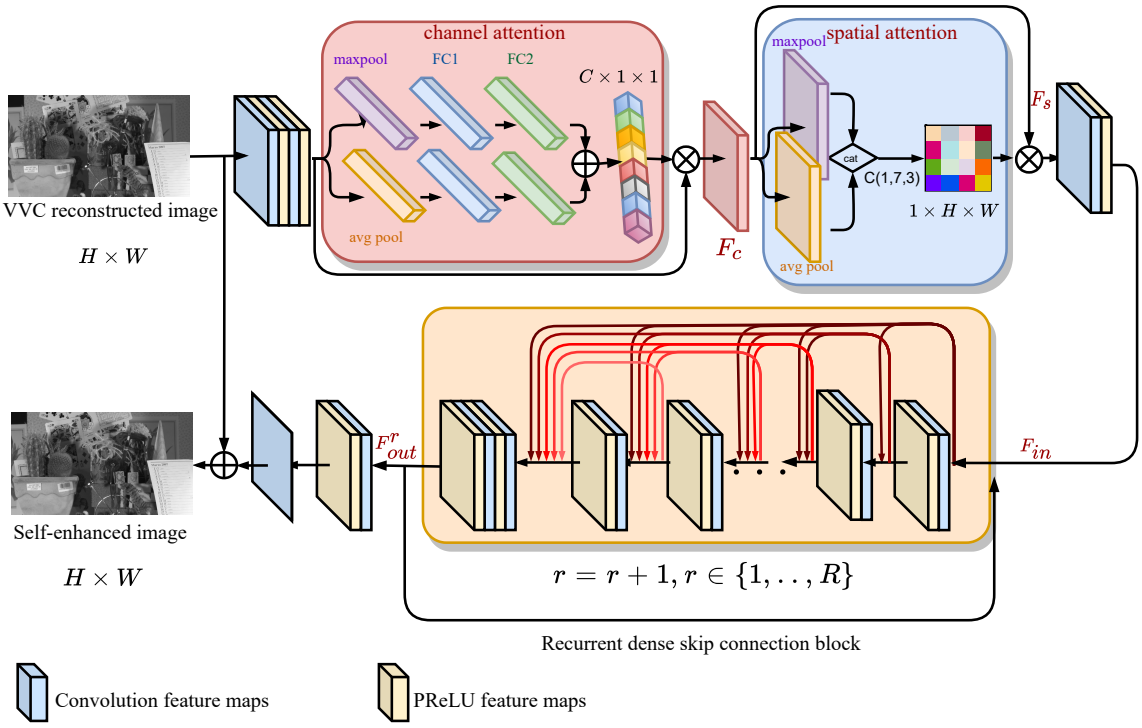


Figure 4.5: Self-enhancement CNN with CU map architecture (SECUM). Our SECUM includes three main components: channel attention, spatial attention, and a recurrent dense skip connection block which contains n convolution layers. Coding unit (CU) map produced from the VVC encoder is also input to the network for signaling the area where blocking artifacts usually appear. $C(k, s, p)$ indicates the convolution layer with k kernels do the convolution with $s \times s$ window and a padding of p. Our RDSCB block does recurrent for R times, output at time $r \in R$ are concatenated with the input F_{in} for the next loop $r + 1$.

Self-enhancement with CU Map (SECUM)

Our SECUM network is visualized in Fig. 4.5. Let $C(k, s, p)$ denotes the convolution layer, which has k kernels size $s \times s$ and padding of p. At first, two convolution layers, each followed by the PReLU layers, sequentially extract the feature maps from the input image. We use three self-attention mechanisms during the feeding forward: channel and spatial mechanisms for emphasizing useful information and a feedback mechanism to take full advantage of the low-level and the high-level features. The channel and spatial attentions [116] then emphasize information from the second convolution layer for more useful representations of the input image. In our channel attention, max-pooling (MP) and average pooling (AP) first extract different information of the input features before being processed by the siamese one-hidden-layer perceptron f. An element-wise summation merges these representations before passing to the sigmoid function σ. The highlighted information of input feature maps F_{in} after channel attention can be calculated by:

$$F_c = \sigma\left(f(\mathrm{AP}(F_{in}) + f(\mathrm{MP}(F_{in}))\right) \otimes F_{in} \tag{4.1}$$

where $\otimes$ denotes the element-wise multiplication operation. On the other hand, the spatial attention layer learns informative areas from the input feature maps. Let $f_{C(1,7,3)}$ denotes the convolution layer has one kernel size 7×7 does convolution with the padding of three, the output feature maps F_s of our spatial attention, given the input feature maps F_c, are computed by:

$$F_s = \sigma(f_{C(1,7,3)}([\mathrm{AP}(F_c); \mathrm{MP}(F_c)]) \otimes F_c \tag{4.2}$$

Since blocking artifacts usually appear on CU borders, we then input CU partition map to SECUM as an attention mechanism. By using the CU partition map, we highlight the parts to which the network should pay attention to to learn better artifact removal. The CU partition map can be visualized as a binary matrix where elements

at the borders between CUs are one, and the other areas are zeros. We introduce an additional branch whose output of convolution layer is concatenated with spatial attention feature map F_s. All the information are synthesize in a convolution layer $C(128, 3, 1)$ before input to RDSCB. Final convolution layers after RDSCB block are $C(64, 3, 1) \rightarrow C(1, 3, 1)$. Other the convolution layers are $C(64, 3, 1)$, and the network does recurrent R of four times. During training, we minimize the $L1$ loss between the ground-truth y and each network output $\hat{y}_r$:

$$L(\Theta) = \frac{1}{N.R} \sum_{i=1}^{N} \sum_{r=1}^{R} I^r \parallel y_0^i - \hat{y}_r^i \parallel_1 \tag{4.3}$$

where N is the number of training samples, and Θ denotes for the learned network parameters. I^r is the weight of the output $\hat{y}_r$, with $r^{th}, (r \leq R)$ indicates the loop order of RDSCB. Follow [83], we set I^r to one for all the outputs.

Reference-based enhancement CNN with Optical Flows (REOF)

The existing CNN-based in-loop filterings for VVC [97–100, 105, 106] have mainly focused on enhancing the quality of the reconstructed image given itself. In our reference-based enhancement CNN (REOF), neighboring frames are used for improving the quality of the current video frame to be encoded. Fig. 4.6 shows the unfolding Reference-based enhancement CNN architecture. Our hypothesis is that if there is an optical flow between the current reconstructed image and the reconstructed reference image, there is also the optical flow between the enhanced current image and the enhanced reference image. Given the optical flow OF_i between the reference image and the current reconstructed image, the enhanced reference image ER_r, which are in the reference picture sets List 0 and List 1 defined by VVC, are remaped with the optical flow OF_r and then, fed to the network at a loop $r^{th}, r \in \{1, .., R-1\}$. Warped images $ER_r + OF_r, r \in \{1, .., R-1\}$ are input to the network by the order of picture order

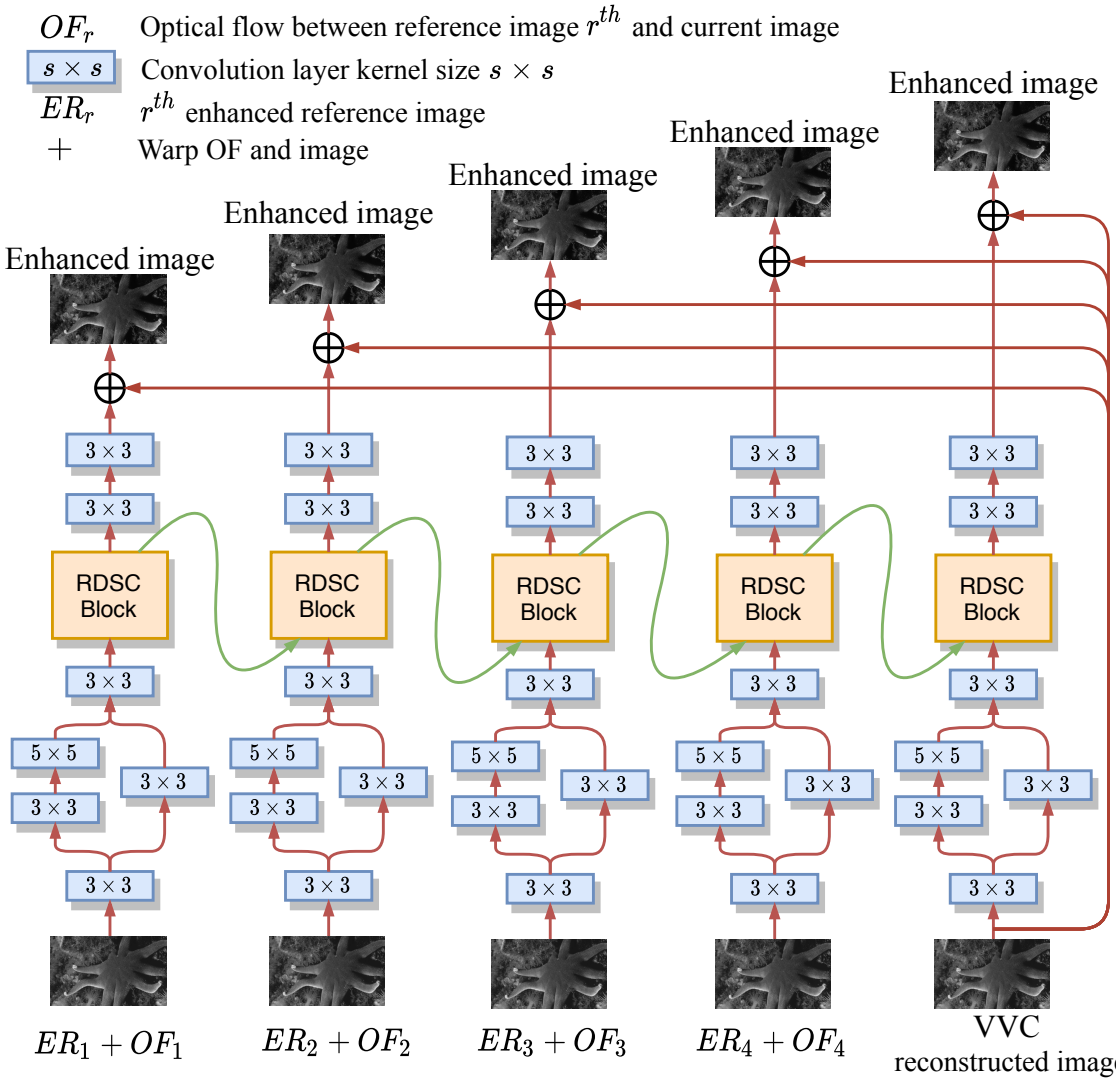


Figure 4.6: Reference-based enhancement CNN with optical flow (REOF) architecture after unfolding. $ER_r + OF_r$ is the warped image of the enhanced reference r^{th} and the optical flow calculated from the current and the reference image. The order r of reference image input to the REOF is the from the long-term to short-term, POC after to before the current image in the reference picture sets List 0 and List 1.

count (POC) long to short term, and POC after to before the current picture. The enhanced current image which is produced by REOF is input to the final loop $r = R$. If any reference images are not activated, the enhanced image of REOF will take that input position instead.

In our reference-based enhancement CNN (REOF), neighboring frames are used for improving the quality of the current video frame to be encoded. Our hypothesis is that

if there is an optical flow between the current reconstructed image and the reconstructed reference image, there is also the optical flow between the enhanced current image and the enhanced reference image. Given the optical flow OF_i between the reference image and the current reconstructed image, the enhanced reference image ER_r, which are in the reference picture sets List 0 and List 1 defined by VVC, are remaped with the optical flow OF_r and then, fed to the network at a loop $r^{th}, r \in \{1, .., R-1\}$. Warped images $\mathrm{ER}_r + \mathrm{OF}_r, r \in \{1, .., R-1\}$ are input to the network by the order of picture order count (POC) long to short term, and POC after to before the current picture. The enhanced current image which is produced by REOF is input to the final loop $r = R$. In several coding configurations, some reference images are inactivated. For these cases, we replace these input ER_r with SECUM enhanced image.

For SECUM, information within the VVC reconstructed image and its CU partition map are used as the input. For REOF, CU partition map, and the self-attention layers such as channel attention and spatial attention are eliminated. Instead, two CNN branches are added. The first branch is described as $C(64, 3, 1) \rightarrow C(64, 5, 2)$ and the second branch can be described as $C(64, 3, 1)$. These output feature maps are concatenated and input to a convolution $C(64, 3, 1)$ before feeding forward to the RDSCB block. We set the number of convolution layers in RDSCB block $n = 9$ for REOF. Including the current picture to be enhanced, there are four reference pictures in the reference picture sets List 0 and List 1. Therefore, R is set as 5 for reference-based enhancement CNN. The loss function can also be written as in equation 4.3.

4.3.3 The proposed reinforcement learning-based autonomous mode selection (AMS)

As mentioned before, current deep learning-based ILF improvement works [1, 60, 72, 101, 103, 104] with the selection controlled by RD-cost suffers from bits required for signaling the decoder when exploiting to deeper levels than CTU level. On the other

hand, designing a syntax element for these signaling bits is tactful work. Insufficient design of syntax elements would cause worse coding results in future frames. This work approaches a reinforcement learning method for autonomous ILF mode selection at the CU level to overcome these drawbacks. By predicting the new ILF mode required for each CU, we can omit the storing bits in the final bitstream while maintaining the effectiveness of multi-mode coding.

Let I_{ilf}, I_{secum}, I_{reof}, and L_{CU} denote VVC reconstructed image, SECUM reconstructed image, REOF reconstructed image, and CU partition of VVC, respectively. Our target is to train an agent to predict a new CU partition $\hat{L}_{CU}$ and an appropriate in-loop filter $F(CU_i)$ for each CU_i in the new partition where $F(CU_i) \in$ {ILF, SECUM, REOF}. Predicted ILF mode is then performed on the corresponding CU_i, resulting in an enhanced image I_e:

$$I_e^{CU_i} = \begin{cases} I_{secum}^{CU_i} & \text{if } F(CU_i) = \text{SECUM}, \\ I_{reof}^{CU_i} & \text{if } F(CU_i) = \text{REOF}, \\ I_{ilf}^{CU_i} & \text{otherwise.} \end{cases} \tag{4.4}$$

Moreover, an action set is introduced, enabling the ability to deal with local areas over the CU level, addressed by none existing works. After filtering, only output image I_e is used. CU partition $\hat{L}_{CU}$ output by the STILF-AMS framework will only be freed without changing the CU partition in the VVC encoder and decoder.

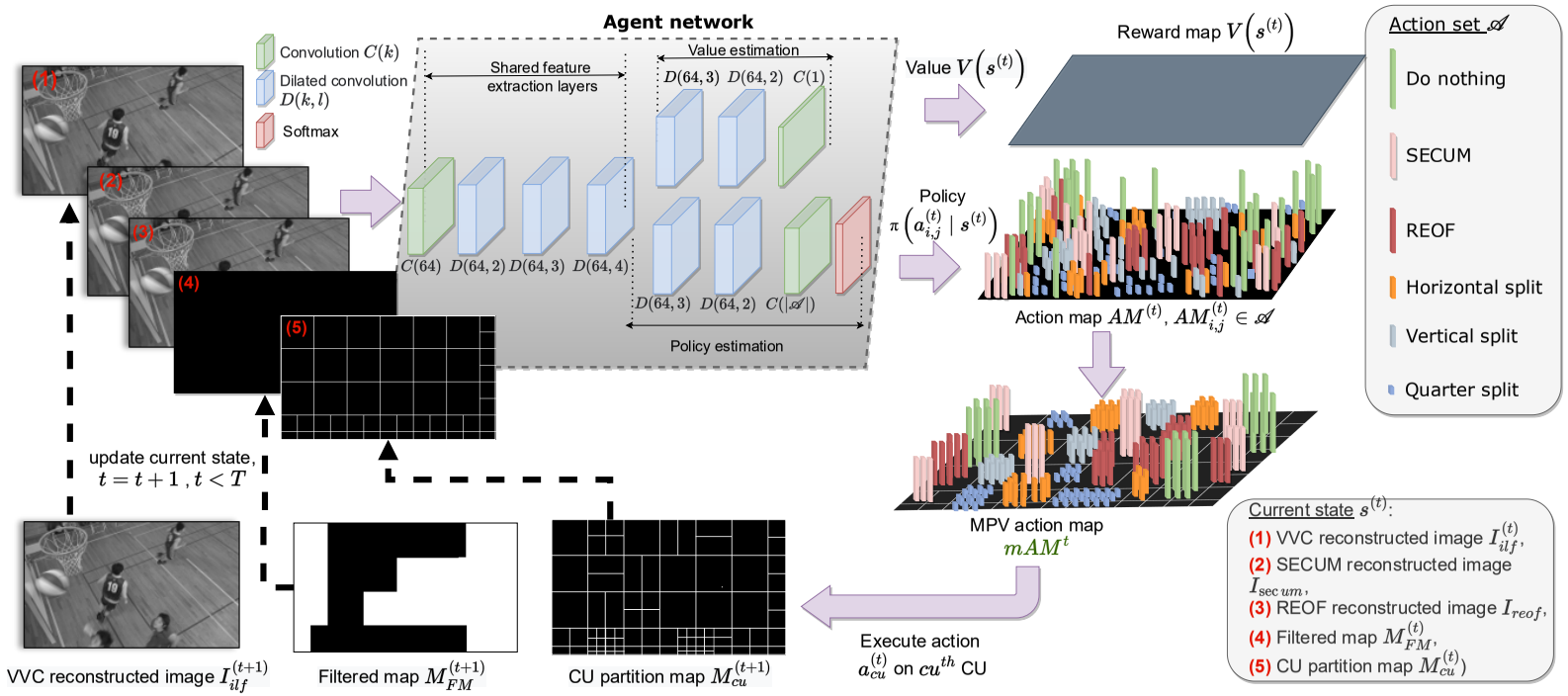


Figure 4.7: Simulation of the reinforcement learning-based ILF-mode selection for VVC video coding at the CU level (STILF-AMS). The agent network iteratively receives and processes a state $s^{(t)}$ of VVC reconstructed image $I_{ilf}^{(t)}$, SECUM reconstructed image I_{secum}, REOF reconstructed image I_{reof}, filtered map $M_{FM}^{(t)}$, and CU partition map $M_{cu}^{(t)}$. Output action map $AM^{(t)}$ present action to be performed on each pixel p_i. In each CU, action with majority votes is then set as the action of the CU, resulting in the most probable value action map $mAM^{(t)}$. VVC reconstructed image, filtered map, and the CU partition map are then updated according to the chosen actions. Agent network continues the process until the current time step t reaches the preset max time step T.

We model the task of selecting ILF for each CU as a sequential decision-making process. At each time step t, our agent observes a state $s^{(t)}$ from the environment and performs an action $a^{(t)}$ predefined in the action set $\mathcal{A}$ following its policy π. After taking action $a^{(t)}$, the agent generates a new observation $s^{(t+1)}$ and receives a feedback reward $r^{(t)}$. The agent iterates this process until either two conditions are satisfied: time step t reaches the T_{max} steps, or the estimated action values are negative. In this section, we present the reinforcement learning-based predicting ILF mode for VVC reconstructed images following asynchronous advantage actor-critic (A3C) approach [117]. The following sections introduce three main components of the proposed AMS: state, agent, and reward.

4.3.4 State definition

In existing RL-based image enhancement works [118–120], a distorted image and its reinforced images at time step t are considered state $s^{(t)}$. In designing RL-based ILF mode selection, we take advantage of coding property for state definition. Each state $s^{(t)}$ is composed of filtered images, filtered map, and CU partition map (as shown in Fig. 4.8). The designation details are as follows.

Filtered images. Similar to other deep learning-based switchable ILF works [1, 60, 72, 101, 103, 104], this work brings up a bank of filters consisting of the default ILF by VVC and the learning filters introduced in 4.3.2. In [1], the default ILF of VVC, our SECUM and REOF are selected at CTU level given their rate-distortion costs. Let $I_{ilf}^{(t)}$ be the ILF image at state t and also the output I_e when $t = T_{max}$. Each and every CU_i in $I_{ilf}^{(t)}$ will be changed according to the predicted action $a^{(t)}$ at time step t. At the end of the episode, CUs that do not choose a filter action in any step $t < T$ will keep the original pixel values from VVC reconstructed image.

Filtered map. During feedforward, it is essential to detect which CU has chosen a filtered rather than the default filter. We then introduce an attention mechanism

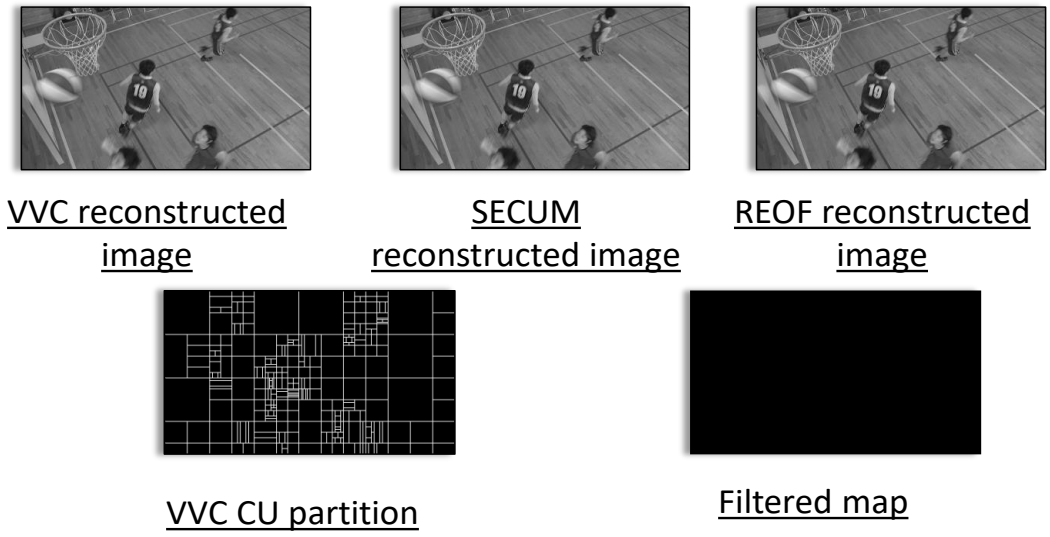


Figure 4.8: Illustration of a state in STILF-AMS framework. The state consists of reconstructed images filtered by the default ILF of VVC, SECUM, and REOF [1], and attention mechanisms including CU partition map and filtered map.

named filtered map to guide the agent. At state $t = 0$, the filtered map $M_{\text{FM}}^{(t)}$ is initialized as a zero matrix whose dimensions are equivalent to the spatial dimensions of filtered images. During feeding forward, if an i^{th} CU chooses a filter action, all of its elements in the filtered map will be set to one. The action types will be clarified in section 4.3.6.

CU partition map. One of the keys leading to the success of modern video coding standards is the ability to adapt to different video content presented in different sizes. The latest video coding standard VVC supports different CU sizes from 128×128 to 4×4. Going over the CU level in ILF tasks, we take into account of lower level than CU size, allow CU to be split by our agent, and support up to the size of 2×2. Given CU partition list output by VVC encoder, a CU partition map can be visualized by a matrix whose elements at CU borders are ones, and the rest are zeros. During the ILF-mode prediction process, if a CU chooses split action, it will be split into two or four equal CUs.

4.3.5 Reward

By receiving incentive rewards from the environment, the agent evaluates the quality of the taken action $a^{(t)}$. Originally, distortion and rate should be considered in multi-mode selection. For our approach, since both the ILF mode decision and the reconstructed image compensation are autonomously performed by the agent, no additional bits are needed for signaling the decoder. Therefore, we can safely remove the rate out of the reward design. In STILF-AMS, each CU is considered an agent and performs an action predicted by the agent network. At time step t, the reward $r^{(t)}$ is calculated after every CU $cu \in M_{cu}^{(t)}$ take action $a_{cu}^{(t)}$, showing how good the action $a_{cu}^{(t)}$ is:

$$r^{(t)} = \|I_{raw} - I_{ilf}^{(t)}\|_2 - \|I_{raw} - I_{ilf}^{(t+1)}\|_2 \tag{4.5}$$

where I_{raw} is the raw video frame input to the video coding standard. As the agent performs action $a^{(t)}$ affecting the future states and actions, cumulative future rewards should be considered:

$$R^{(t)} = r^{(t)} + \gamma V(s^{(t+1)}) \tag{4.6}$$

where the discount factor $\gamma \in (0; 1]$ indicates the impact of future reward on the current reward. Note that reward $r^{(t)}$ is a 2D matrix containing rewards at each pixel of state t. Although STILF-AMS performs on the CU level, convolution is operated on the pixel level. The chosen action on each pixel affects not only itself but neighboring pixels, which also affects the results of the neighboring CUs. Inspired by the proposed reward map convolution in [119], we reformulate our discounted sum of rewards $R^{(t)}$ in equation 4.6 as follows:

$$R^{(t)} = r^{(t)} + \gamma \sum_{i,j=1,1}^{m,m} V(s_{i,j}^{(t+1)}) \tag{4.7}$$

where $m \times m$ is the size of receptive field whose center is the current pixel.

4.3.6 Agent

There have been many works in deep learning-based coding mode prediction [108], or CU partition decision [109, 113] in video coding standards. These CU size-dependent approaches have a drawback: specific designs are required for different video coding standards with various CU sizes. Recent researches in denoising [118, 119] have studied pixel-wise classification, in which pixels are improved by a preset filter. Intuitively, each pixel has the best ILF mode for improving itself. Considering the fact that neighboring pixels in the same CU are commonly coded in similar coding modes, our agent predicts an ILF mode in the set of filters {ILF, SECUM, REOF} for every pixel in the same CU, maintaining the high correlations between them. Inspired by [118, 119], we reformulate the classification on pixel-level to CU-level to adapt with ILF mode selection task.

Action set $\mathcal{A}$. Our action set includes three types of actions: do nothing, filter actions, and split actions as shown in Table 4.1. All pixels in the same CU would perform the action with the most probable value predicted by the agent network. Suppose there is more than one action that has the same majority vote. In that case, priority is applied, biasing for the action with lower complexity. If a CU chooses *do nothing*, regardless of which action has been performed at time $t-1$, all of its pixel values are kept, and the corresponding elements in the filtered map are set as one. Let f denotes the filter actions $f \in \{SECUM,\ REOF\}$ for enhancing the image inside the boundary of a CU: *SECUM* and *REOF*, respectively. For CU chooses filter actions, pixels within its boundary in the current restored image $I_{\text{ilf}}^{(t)}$ would be changed by I_{SECUM} or I_{reof} values.

Practically, *do nothing* and *filter* actions are similar to Rate-distortion (RD)-based coding mode decision. When it comes to reconstructed image quality, simply training the agent to choose which filter enhances a particular CU is not efficient enough. We then design split actions, enabling the ability to exploit deeper local areas smaller than the CU level. If horizontal (or vertical) split actions are selected, the corresponding

vertical (or horizontal) edges are divided. Both vertical and horizontal splits are performed for the four-quadrant split action, resulting in four equal CUs. To avoid the invalid CU size and high complexity, we limit the minimum CU edge size to be split as four. Consequently, the smallest supportive ILF unit is a 2×2 CU, allowing more accurate pixel compensation.

Table 4.1: List of actions, changes on current state $s^{(t)}$, and priorities of actions in the case number of votes in the same CU are equivalent.

Type	Actions at t	Changes in state $s^{(t+1)}$	Priority
Do nothing	Keep pixel values	Filtered map $M_{\text{FM}}^{(t)}$	1^{st}
Filter	SECUM filtering	Current image $I_{\text{ilf}}^{(t)}$ Filtered map $M_{\text{FM}}^{(t)}$	2^{nd}
	REOF filtering	Current image $I_{\text{ilf}}^{(t)}$ Filtered map $M_{\text{FM}}^{(t)}$	3^{rd}
Split	Horizontal split	CU partition $M_{CU}^{(t)}$	4^{th}
	Vertical split	CU partition $M_{CU}^{(t)}$	5^{th}
	Four-quadrant split	CU partition $M_{CU}^{(t)}$	6^{th}

Agent network architecture. Our agent network iteratively takes a state $s^{(t)} = (I_{ilf}^{(t)}, I_{secum}, I_{reof}, M_{CU}^{(t)}, M_{FM}^{(t)})$ as an input, and outputs both action probabilities $AM^{(t)}$ and the expected reward $V(s^{(t)})$ on each pixels. After prediction, pixels in a same CU vote for the most probable value as the action to perform on that CU, resulting in a action map $mAM^{(t)}$ at CU level. CU is then filtered or split according to the predicted actions in $mAM^{(t)}$. After T loops, reconstructed image $I_{ilf}^{(T)}$ is used as compensated result of ILF mode selection. Inspired by recent advantages on deep reinforcement learning-based image denoising works [118, 119], our agent network combines policy and value network in a single architecture by sharing the first four feature extraction layers

(as shown in Fig. 4.7). These sharing layers allow agent network to be trained more efficient and tested with lower complexity. Our agent network f_θ learns to estimate the policy $\pi(a^{(t)}|s^{(t)},\theta)$ and approximate the value function $V(s^{(t)},\theta)$:

$$\pi(a^{(t)}|s^{(t)}), V(s^{(t)}) = f_\theta(s^{(t)}) \tag{4.8}$$

where θ Let $C(k)$ and $D(k,l)$ indicate convolution layer with k kernels and dilated convolution layer with k kernels and a dilated factor of l, respectively. Stride is set to one for all layers. Padding is set to one for convolution layers and l for dilated convolution layers during feedforward. We set the number of kernels k to 64 for all layers except the final convolution layer in each branch. In the policy branch, k is set as the number of actions $|\mathcal{A}|$ while it is one in the value branch. Policy branch ends with a Softmax policy, outputting the probabilities that pixel p_i chooses action $a_{p_i}^{(t)}$, $a_{p_i}^{(t)} \in \mathcal{A}$. Meanwhile, the value branch outputs approximately the expected reward judging the quality of input state $s^{(t)}$. By forming the policy and value function into action map $AM^{(t)}$ and reward map $V(s_{(p_i)}^{(t)})$ to the pixel level, our work is independent of CU sizes, enabling an ability to deal with any modern video coding standards. During training, the agent network aims to find the optimal policy π^*:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_\pi \left(\sum_{t=0}^{\infty} \gamma \bar{r}^{(t)} \right) \tag{4.9}$$

where $\bar{r}^{(t)}$ indicates the mean of reward map $V(s^{(t)})$. Gradients for backward policy branch $d\theta_\pi$ and value branch $d\theta_v$ are defined as follows:

$$d\theta_\pi = -\nabla_{\theta_\pi} \frac{1}{h \times w} \sum_{i,j=1,1}^{h,w} \log \pi(a_{i,j}^{(t)}|s^{(t)}) A(a_{i,j}^{(t)}, s^{(t)}) \tag{4.10}$$

$$d\theta_v = \nabla_{\theta_v} \frac{1}{h \times w} \sum_{i,j=1,1}^{h,w} \left(A(a_{i,j}^{(t)}, s^{(t)}) \right)^2 \tag{4.11}$$

where h, w are respectively height and width of the input image and $A(a_{i,j}^{(t)}, s^{(t)})$ is the advantage function calculated by:

$$A\left(a_{i,j}^{(t)}, s^{(t)}\right) = R^{(t)} - V(s^{(t)}) \tag{4.12}$$

4.4 Experiments

4.4.1 Parameter settings

Training ILF networks. We randomly choose 30 videos from Derf's Test Media Collection [121] for training. It is noteworthy mentioning that chosen sequences are not in the VVC common test sequences. We encode these sequences under the Low Delay P configuration and subsampled frames by a factor of three. We take the first 50 reconstructed frames and theirs CU partitions for training. We randomly chose 30 frames from the rest of the encoded frames for validating. Images are split into 64×64 patches, and a batch size of 64 is used for training. For reference-based enhancement CNN, optical flows between the reference and the current frames are acquired by Lucas–Kanade method. The network parameters of SECUM and REOF are 1.7 $\times 10^6$ and 7×10^5, respectively. The networks are trained on NVIDIA Tesla V100 using PyTorch [122].

Training STILF-AMS agent network. In training, images are cropped into 128×128 patches whose size is the coding tree unit (CTU) size in VVC. Note that the exact CTU is adopted to keep the full CU partition input to the agent network. We set a batch size of 32 and the discount factor γ to 0.95. We set the learning rate starting at 10^{-3} and a step size of one. In each step size, learning rate is cumulatively multiplied by $\left(1 - \frac{\text{current episode}}{\text{max episode}}\right)^{0.9}$. We adopt a transfer learning scheme for transferring knowledge from the pre-trained image denoising model pixelRL [119] to ILF mode selection at CU level. We randomly select 20 videos from [121], which are not overlapped with

datasets in training ILF networks and testing sequences. We acquired the training and validating set by a similar method to generating datasets of ILF networks, except we subsampled chosen frames by a factor of 5. SECUM and REOF first processed training and validating sets to be the input of the agent network. Agent network was trained in 6000 episodes in which a length of six is set for each. ADAM optimizer [123] is used for training. STILF-AMS is implemented using public frameworks Chainer [124] and ChainerRL [125] on NVIDIA Tesla V100.

Video coding testing and evaluation metric. For our testing experiments, we use Bjøtegaard Delta-Rate (BD-rate) [126] with an anchor of the VVC test model (VTM) 9.3 for comparison. For BD-rate, the lower negative number indicates the better result. Four quantization parameters are used: 22, 27, 32, and 37 for calculating BD-rate results. VVC standard test sequences in the common test conditions (CTC) [127] are encoded under using All Intra (AI), Low Delay P (LDP), Low Delay B (LDB), and Random Access (RA) configurations. All the coding parameters are set to default, in which the built-in ILF is enabled.

4.4.2 Study on network set

Results on deep learning-based ILF improvement without autonomous mode selection. In this experiment, we demonstrate the efficiency of the proposed networks SECUM and REOF (STILF). Figure 4.9 shows the proposed STILF (blue arrows) integrated to VVC with RDO-guided mode selection at CTU level. The reconstructed video frame is first filtered by VVC ILF before feeding to SECUM and REOF. A mode selection is performed to select the best in-loop filter at CTU level, which should be decided base on the distortion of the reconstructed image in comparison to the original image. However, adding the new bits for signaling the decoder also affect the total bits, we then check the Rate-Distortion (RD) cost of the enhanced image and additional bits for each coding in-loop filter. In our selection,

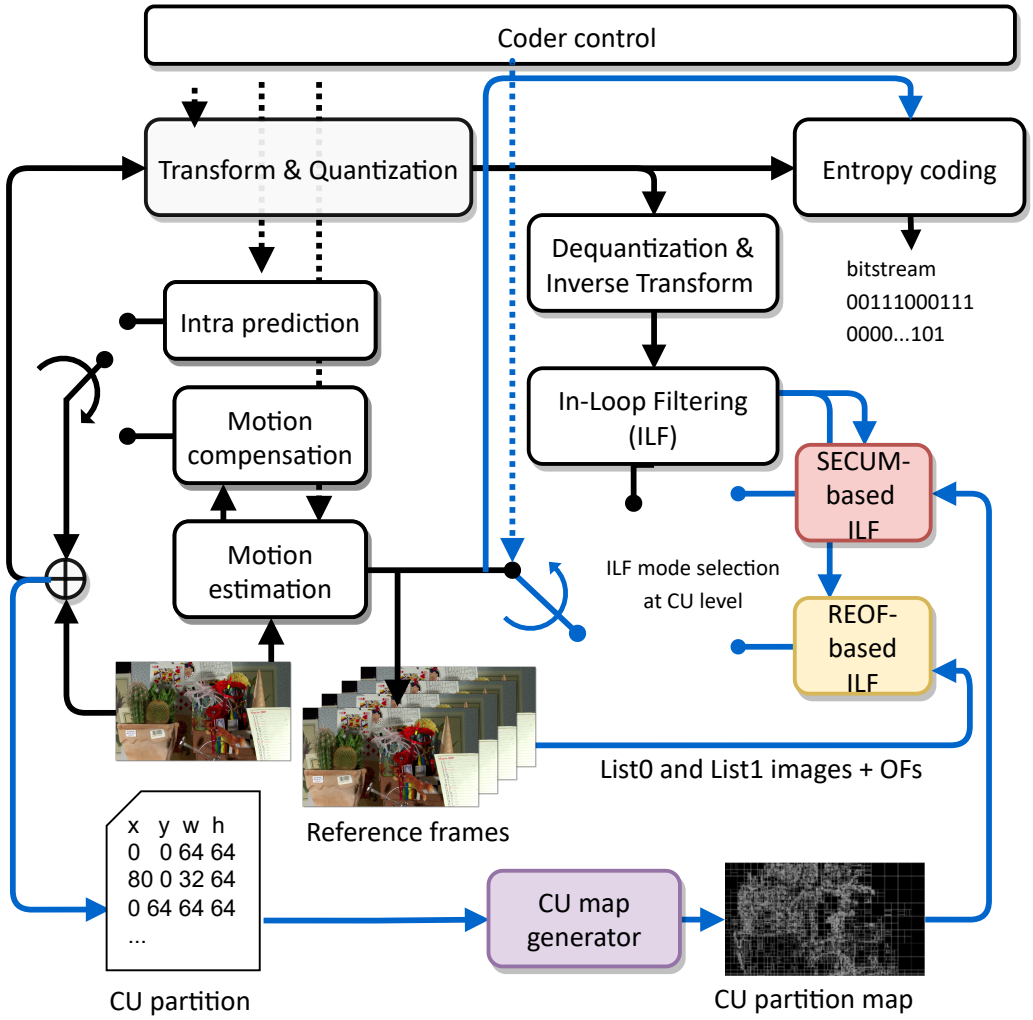


Figure 4.9: Illustration of the proposed STILF (blue arrows) integrated to VVC.

CTU is filtered by three different options, for each option, two bits indicates the filter index will also be simulatively added to the bitstream, and a CABAC estimator calculates how many bits need for compressing this CTU. The RD-cost then bases on the distortion of the image enhanced by the filter and the bits to be encoded of that CTU.

Table 4.2 and 4.3 show the overall performance of the proposed STILF compared to VVC test model VTM 9.3. The work mainly focuses on enhancing the Y component, so the BD-rate of the U and V components slightly increases for some sequences. Table 4.7

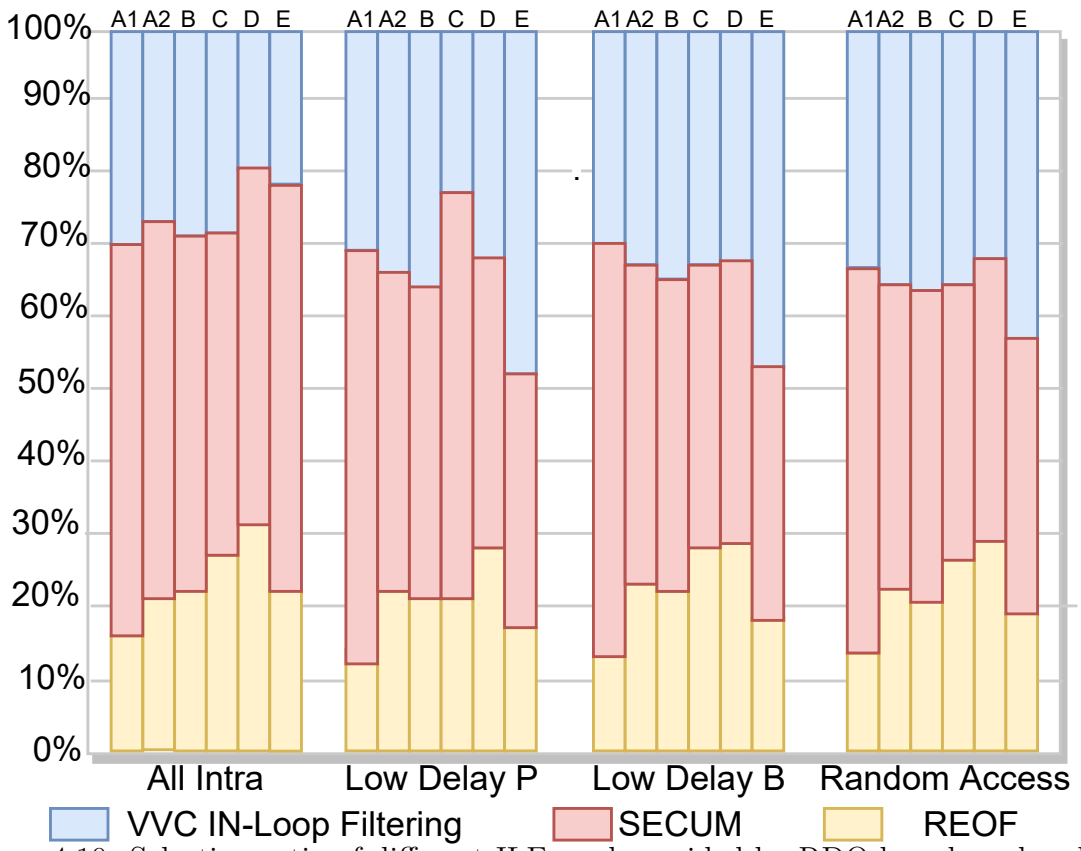


Figure 4.10: Selection ratio of different ILF modes guided by RDO-based mode selection at CTU level.

shows the overall performance of the proposed STILF on the common test conditions recommended by JVET. The work mainly focuses on enhancing the Y component, so the BD-rate of the U and V components slightly increases for some sequences. In AI configuration, since there is no reference assigned by VVC, the warped image is replaced by the VVC reconstructed image. We obtain the Y BD-rate reduction of 3.78%, 6.34%, 6%, and 4.64% on the AI, LDP, LDB, and RA configurations. Besides, we recorded an encoding time of 1.31-1.84 and a decoding time of 8.69-73.03 on average of CTC. The ratio of choosing VVC ILF, our self-enhancement CNN (SECUM), and our reference-based enhancement CNN (REOF) are illustrated in Fig. 4.10. Blue, pink, and yellow bars indicate respectively for the selected ratio of VVC In-loop filtering, SECUM, and REOF. On average, the ratio of choosing VVC In-loop filtering, REOF, and REOF are 33.82%, 44.44%, and 21.74%, respectively.

Study on self-attention mechanism. We perform an ablation study on attention

mechanisms, including self-attention (SA) and CU map (Table 4.4). The results show coding performances have been reduced when cutting these mechanisms.

Figure 4.11 illustrates the visual quality comparison of STILF without AMS and the anchor VTM 9.3. Our STILF better improves the details such as the sweater (yellow outlined area) and forehead (red outlined area) than VTM 9.3 in KristenAndSara. For BQTerrace, the road maker on the bridge reconstructed by ours is perceived superior to VTM's.

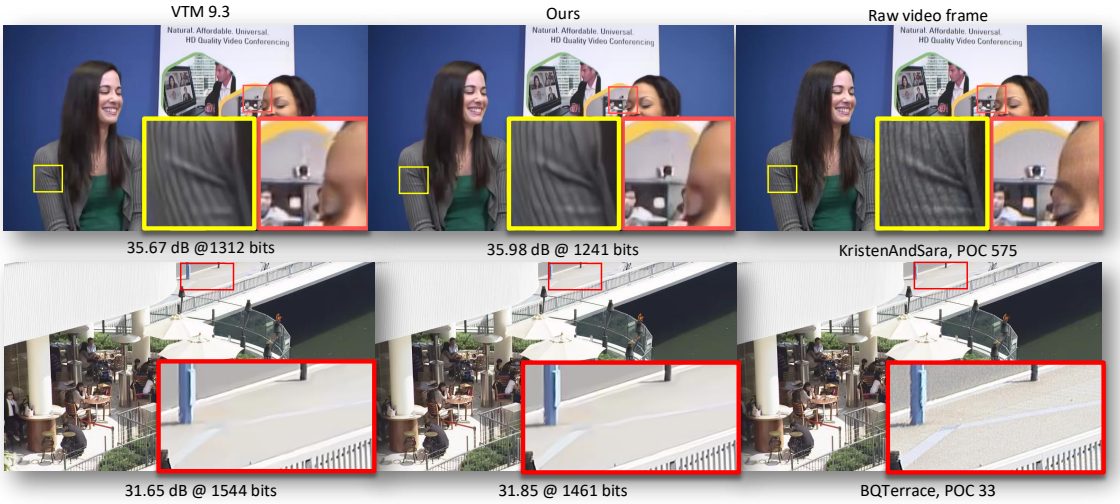


Figure 4.11: Visual comparison of STILF without AMS to VVC (anchor VTM 9.3). From top-down, KristenAndSara POC 575 and BQTerrace POC 33 encoded under Low Delay P with QP 37 are chosen for illustration.

4.4.3 Ablation Study

Action analysis. Fig. 4.12 illustrates the chosen actions on pixels level at each time step t. It is apparent from this figure that the agent network tends to choose split actions at the first several steps and filter actions at the latter steps. At the end of the episode, *REOF* and *do nothing* are the most chosen actions. We come to a conclusion that agents have learned a strategy of first splitting and then filtering the reconstructed images.

Study on input state $s^{(t)}$. To select ILF modes for all CUs in the CU partition,

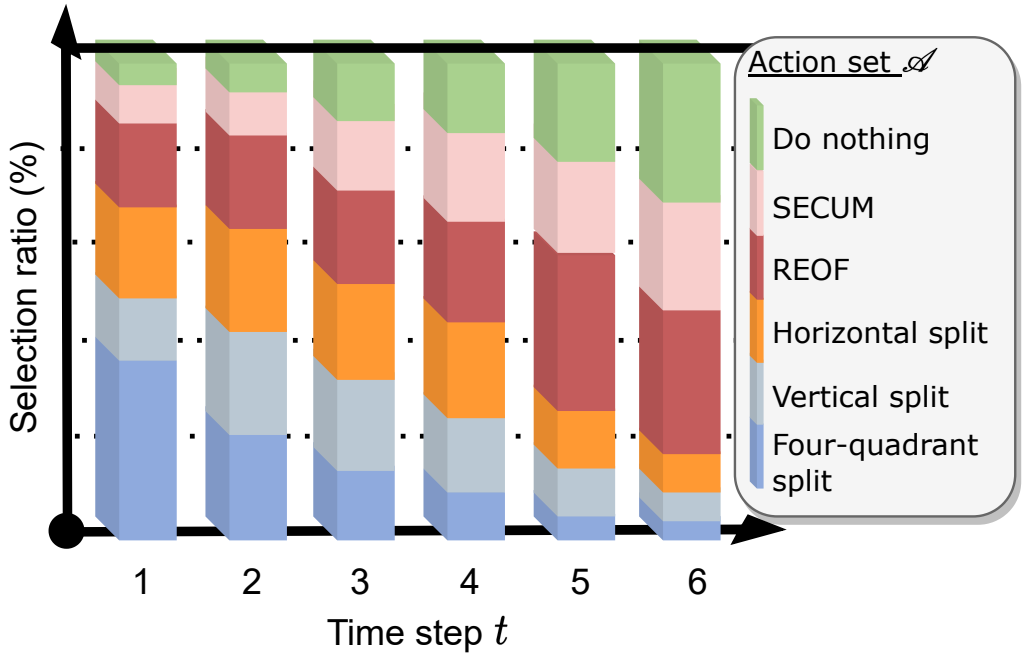


Figure 4.12: Selection ratios of class D's sequences over time step t.

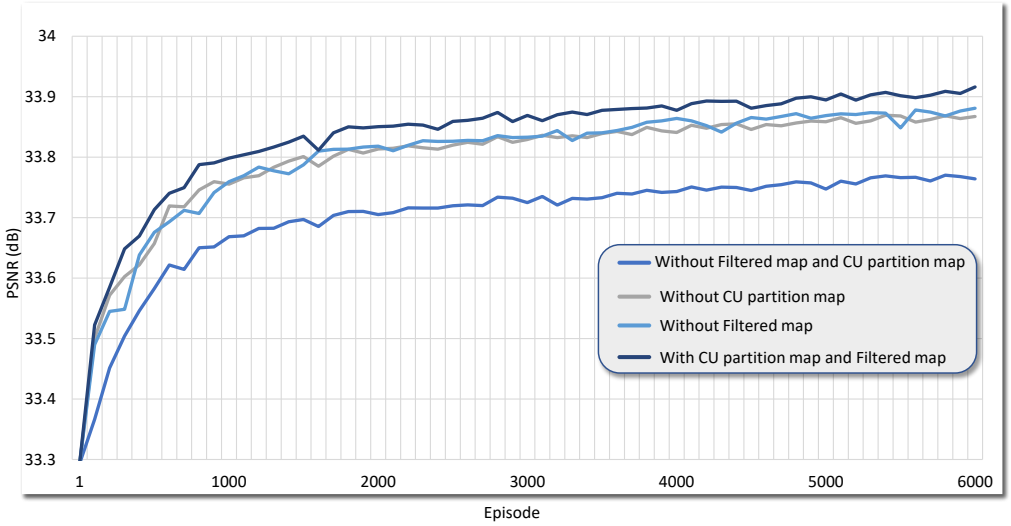


Figure 4.13: Average PSNR on the validation set with different mechanisms disabled for state $s^{(t)}$.

it is essential and meaningful to have ILF, SECUM, and REOF reconstructed images in the input state $s^{(t)}$. In this experiment, we evaluate the effectiveness of two other additional components of input state: CU partition map $M_{cu}^{(t)}$ and filtered map $M_{fm}^{(t)}$. We first experiment on training the agent network with and without these maps (as shown in Fig. 4.13). For visualization, we pick 60 from 6000 training episodes by a factor of 100. The PSNR lines suggest that the network trained with CU partition map

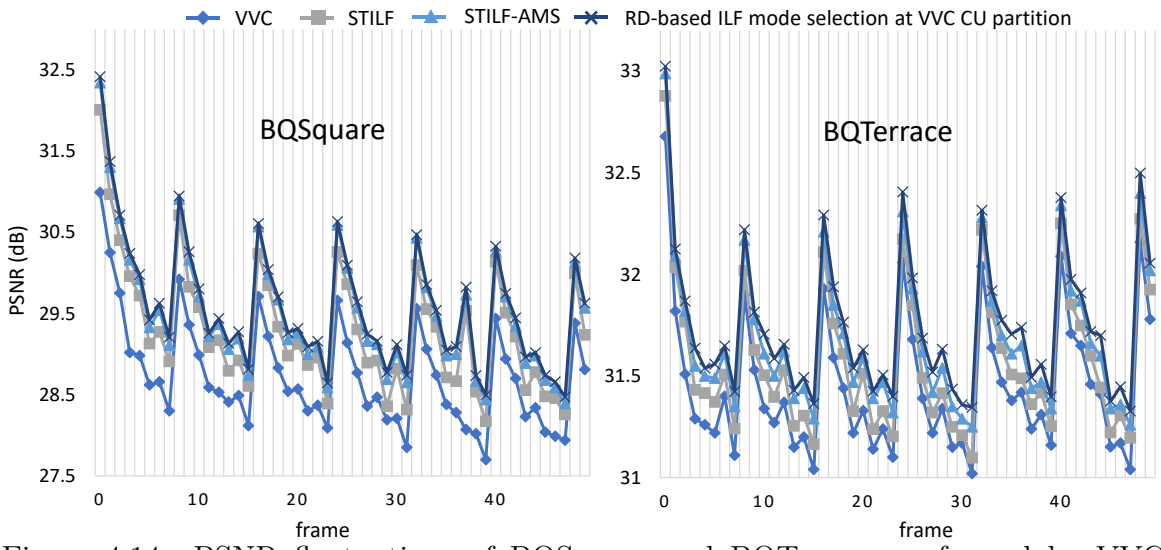


Figure 4.14: PSNR fluctuations of BQSquare and BQTerrace performed by VVC, STILF [1], and STILF-AMS (ours), and the ground-truth RD-based mode selection on VVC CU partition $M_{cu}^{(t=T)}$.

and Filtered map is more stable and obtains the highest PSNR on the validation set. For further verifying the effectiveness of these maps, we perform BD-rates of STILF-AMS with and without these additional components. Results in Table 4.5 have shown that higher bitrate savings can be obtained using both two additional components. Without these components, a 0.2% BD-rate reduction is reduced on average of class C and class D sequences.

PSNR results of different ILF approaches. Fig. 4.14 shows the PSNR fluctuations on different ILF mode: VVC default ILF, our STILF [1], the proposed ILF mode prediction at predicted CU partition (STILF-AMS), and the ground-truth using RD cost-based ILF mode selection at VVC CU partition. It can be seen that STILF-AMS can compensate the reconstructed image closer to the PSNR performed by RD cost-based mode selection. While RD cost-based ILF mode selection on CU level is not practical due to additional bits, STILF-AMS requires no additional bits and offers comparable image quality.

CU partition comparison. For verifying STILF-AMS prediction, we also visualize the CU partition output by VVC and ours in Fig. 4.15. In this visual comparison, the 9^{th} frame of RaceHorseC coded under QP of 37, and RA configuration is chosen for

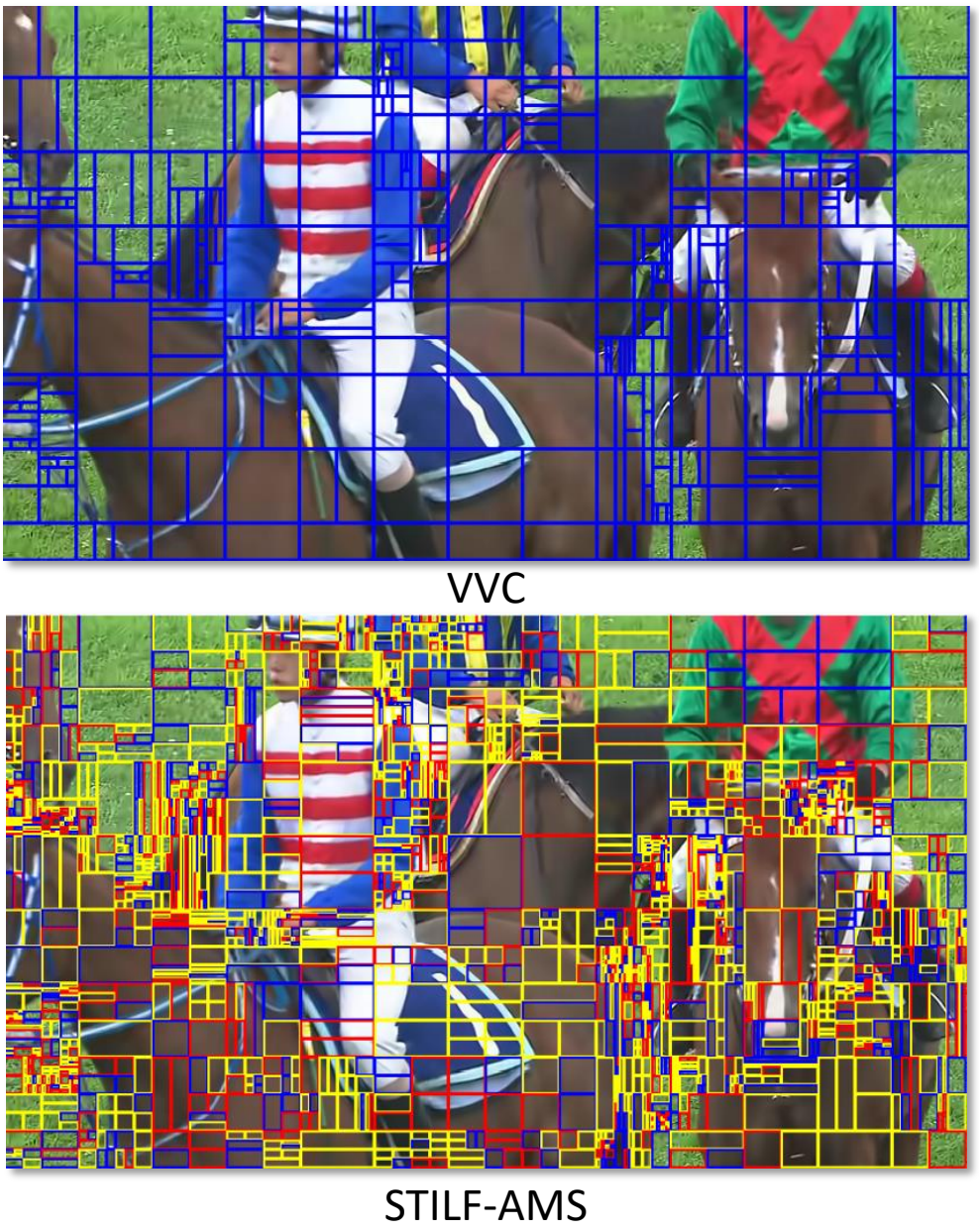


Figure 4.15: Visualizations of CU partitions output by VVC and our STILF-AMS. Blue, red, and yellow rectangles indicate CU that chooses VVC default ILF, SECUM, and REOF, respectively.

visualizing. It can be seen that STILF-AMS tends to split and filter CUs with more details by SECUM and REOF. On the contrary, CUs with less information are simply kept pixels output by VVC default ILF.

Selection ratios. During the coding process, ILF mode selection is performed by the proposed STILF-AMS. To further verify the result of mode selection, we measure the ratio of choosing different ILF modes of the STILF-AMS. Since the mode selection

is performed on CU levels, the ratio should be calculated on the chosen areas than hitting ratios. Given M CUs that choose ILF mode m, $m \in$ {ILF, SECUM, REOF} over the total CUs N, selection ratio for mode m is calculated as follows:

$$\text{Ratio}^{(m)} = \frac{\sum_{i=0}^{M} \text{A}_i^{(m)}}{\sum_{c=0}^{\text{N}} \text{A}_c} \tag{4.13}$$

where $A_i^{(m)}$ and A_c denote the area of CUs that choose ILF mode m and the area of all CUs, respectively. Mode selection ratios performed on sequences of class C and class D are shown in Table 4.6. Specifically, we observe that REOF, in which reference frames are used for enhancing the current frame, is the most chosen mode among the ILF modes introduced in STILF-AMS.

4.4.4 Coding results

Overall performance. Our results compared to VVC with different coding configurations are summarized in Table 4.7. The results show that our proposal can significantly improve the coding efficiency compared to the state-of-the-art video coding standard VVC on CTC sequences. Since STILF-AMS is performing on the Y component, chrominance results for AI configuration do not change, resulting in U and V BD rates being all zeros. We can obtain coding improvements on all components for other configurations where inter-coding is performed. We report a BD-rate reduction result up to 18.65% at the same image quality compared to VVC at sequence *BQSquare*. On average, STILF-AMS obtains bit-rate savings of 4.13%, 7.1%, 6.93%, and 5.5% compared to VVC under AI, LDP, LDB, and RA configurations, respectively.

Qualitative comparisons. Fig. 4.16 presents the reconstructed frames by VVC [2], STILF [1], and the proposed STILF-AMS. Cactus POC 14 and BasketballDrive POC 45 are used for visualization for this qualitative comparison. Although STILF [1]

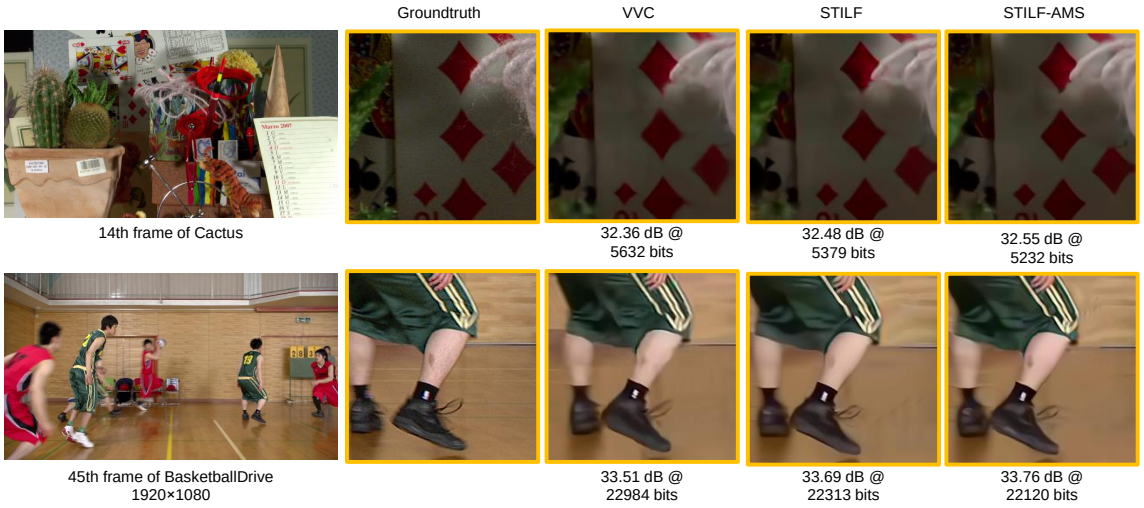


Figure 4.16: Qualitative comparison of VVC [2], STILF [1], and STILF-AMS (ours) at QP of 37. Cactus POC 14 and BasketballDrive POC 45 coded under LDP configuration.

can remove the blurry from VVC, there are areas such as the diffuse diamond in Cactus or unclear birth-mark in BasketballDrive could be improved further. In conclusion, STILF-AMS performs a better PSNR with crisper details at lower bits than VVC, and STILF [1], especially in inter-coding configurations.

Comparison with state-of-the-art (SOTA) deep learning-based ILF for VVC. In order to verify the effectiveness of STILF-AMS, a coding comparison to related works on deep learning-based ILF enhancement for VVC has been conducted. For a fair comparison, we re-implement STILF-AMS and STILF [1] on VTM 6.0. As shown in Table 4.9 and Table 4.10, the proposed STILF-AMS outperforms related researches on learning-based ILF enhancement on all the configurations. Specifically, up to 2.1%, 4.55%, 4.5%, and 3.36% BD-rate reductions are obtained compared to related works [1, 102, 103] under AI, LDP, LDB, and RA configurations, respectively.

Coding complexity analysis. Most of the CNN-based video coding technologies suffer from high computational complexity problems. Table 4.8 provides the computational complexity of the proposed method on GPU compared to VVC. When it comes to encoding complexity, STILF-AMS requires averages of 107% and 133% on AI and inter-coding configurations, respectively. With high-quality compensation, STILF-AMS can

boost the encoder's performance, decreasing the encoding complexity of class D. Note that our STILF-AMS framework is an internal library to VTM, in which each time performs ILF mode selection requires a constructor to load the SECUM, REOF, and STILF-AMS for prediction. Encoding time and decoding time could be significantly reduced by directly integrating the deep learning toolbox into the VVC framework.

4.5 Chapter conclusion

This chapter presents Spatial-Temporal in-loop filtering with autonomous mode selection (STILF-AMS) for the state-of-the-art video coding standard VVC. In this work, we first propose a group of Spatial-Temporal ILFs (STILF) on CU level, including VVC default ILF, the self-enhancement CNN with CU map, and the reference-based enhancement CNN with optical flows for improving the reconstructed image quality. To further improve the coding efficiency, this work is the first to propose a reinforcement learning-based autonomous mode selection (AMS) approach. Our agent is trained to predict the trend of splitting and filtering mode in each CU. By predicting ILF mode and allowing CU to be split more, STILF-AMS requires zero extra bit while ensuring the quality of reconstructed images. As a result, we outperform VVC and the state-of-the-art deep learning-based ILF enhancement. Remarkably, up to 18% and an average of 5.9% bitrate savings have been obtained under all configurations.

Table 4.2: BD-rate(%) of STILF without AMS compared to VVC under AI and RA configurations. (Anchor: VTM 9.3)

Class	Sequence	All Intra			Random Access		
		Y	U	V	Y	U	V
A1	Tango2	-2.39	0.13	0.13	-2.82	0.55	0.84
	FoodMarket4	-4.04	0.11	0.11	-2.86	0.26	0.53
	Campfire	-1.89	0.06	0.08	-2.54	-0.04	0
A2	CatRobot1	-2.89	0.05	0.05	-4.03	-0.69	0.18
	DaylightRoad2	-2.26	0.03	0.03	-4.73	0.29	-0.2
	ParkRunning3	-1.47	0.02	0.02	-1.76	0	-0.14
B	MarketPlace	-2.23	0.12	0.12	-2.83	1.04	0.75
	RitualDance	-4.05	0.28	0.27	-3.09	0.21	0.27
	Cactus	-2.93	0.08	0.07	-3.85	0.5	0.57
	BasketballDrive	-3.39	0.24	0.24	-4.21	-0.09	-0.09
	BQTerrace	-2.89	0.03	0.03	-5.8	0.56	0.34
C	BasketballDrill	-6.98	0.51	0.52	-6.01	-0.19	0.37
	BQMall	-4.61	0.31	0.31	-6.04	-0.19	-0.15
	PartyScene	-3.59	0.12	0.12	-4.81	0.09	0.36
	RaceHorses	-1.91	0.56	0.57	-3.07	-1	0.75
D	BasketballPass	-5.48	0.93	0.94	-5.76	-0.62	0.86
	BQSquare	-7.08	0.53	0.53	-10.9	-0.63	-0.43
	BlowingBubbles	-3.92	1.07	1.06	-5.05	-0.02	-0.3
	RaceHorses	-3.4	1.55	1.54	-3.99	-0.61	0.06
E	FourPeople	-5.14	0.2	0.2	-5.75	0.38	0.34
	Johnny	-5.68	0.25	0.27	-7.46	-0.03	-0.23
	KristenAndSara	-4.92	0.49	0.5	-4.64	0.02	0.4
Average A1		-2.78	0.1	0.11	-2.74	0.26	0.46
Average A2		-2.2	0.03	0.03	-3.51	-0.13	-0.05
Average B		-3.1	0.15	0.15	-3.95	0.45	0.37
Average C		-4.27	0.38	0.38	-4.98	-0.32	0.33
Average D		-4.97	1.02	1.02	-6.42	-0.47	0.05
Average E		-5.25	0.31	0.32	-5.95	0.13	0.17
Average all sequences		-3.78	0.35	0.35	-4.64	-0.01	0.23

Table 4.3: BD-rate(%) of STILF without AMS compared to VVC under LDP and LDB configurations. (Anchor: VTM 9.3)

Class	Sequence	Low Delay P			Low Delay B		
		Y	U	V	Y	U	V
A1	Tango2	-3.93	-0.54	0.24	-3.23	-0.45	-0.32
	FoodMarket4	-3.67	-0.19	-0.34	-3.24	-0.11	0.01
	Campfire	-4.84	-0.16	-0.09	-2.39	-0.09	-0.31
A2	CatRobot1	-4.05	0.63	-0.12	-4.18	-0.14	-0.12
	DaylightRoad2	-4.31	0.25	0.26	-4	0.59	0.65
	ParkRunning3	-1.78	-0.2	-0.21	-1.76	-0.07	-0.12
B	MarketPlace	-3.37	-0.46	1.17	-3.26	0.59	-0.64
	RitualDance	-3.57	-0.09	0.22	-3.59	0.42	-0.34
	Cactus	-5.38	-0.64	-0.79	-7.33	-2.84	-2.72
	BasketballDrive	-5.34	-0.3	-0.2	-5.26	-0.01	0.35
	BQTerrace	-8.41	-0.98	-0.95	-6.9	-0.12	-0.51
C	BasketballDrill	-8.03	-0.32	-0.6	-7.67	0.05	0.92
	BQMall	-8.46	-2	-1.05	-8.25	-0.96	-1.61
	PartyScene	-7.11	-0.55	-0.65	-6.81	-0.92	-1.51
	RaceHorses	-3.58	-0.66	0.42	-3.41	-0.32	0.07
D	BasketballPass	-9	-0.15	-1.04	-8.69	0.27	-1.2
	BQSquare	-18.24	-2.55	-1.33	-17.66	-3.32	-2.24
	BlowingBubbles	-6.65	-0.03	0.42	-6.09	-0.47	-0.62
	RaceHorses	-5.36	-0.42	-0.18	-5.3	-0.68	-0.2
E	FourPeople	-7.46	-0.69	-0.44	-7.07	-0.22	0.35
	Johnny	-9.75	1.72	-1.57	-9.15	-2.24	-1.53
	KristenAndSara	-7.23	0.81	-0.4	-6.77	0.85	1.07
Average A1		-4.15	-0.29	-0.06	-2.95	-0.22	-0.21
Average A2		-3.38	0.23	-0.02	-3.31	0.12	0.13
Average B		-5.21	-0.5	-0.11	-5.27	-0.39	-0.77
Average C		-6.79	-0.88	-0.47	-6.53	-0.54	-0.53
Average D		-9.81	-0.79	-0.53	-9.44	-1.05	-1.07
Average E		-8.14	0.61	-0.8	-7.66	-0.54	-0.04
Average all sequences		-6.34	-0.34	-0.33	-6	-0.46	-0.48

Table 4.4: BD-rate (%) results of the proposed STILF without self attention (SA) mechanisms and CU map under LDP configurations. (Anchor: VTM 9.3)

Class	STILF w/o SA	STILF w/o CU map	STILF w/o SA & CU map	STILF
B	-4.95	-5.15	-4.71	-5.21
C	-6.68	-6.75	-6.12	-6.79
D	-9.75	-9.79	-8.77	-9.81
E	-7.81	-7.58	-6.92	-8.14

Table 4.5: BD-rate (%) results on different mechanisms of the input state (anchor VVC). Results are calculated on the proposal with and without CU partition map $M_{\text{cu}}^{(t)}$ and filtered map $M_{\text{fm}}^{(t)}$ in the input state $s^{(t)}$.

Sequence	w/o $M_{fm}^{(t)}$ and $M_{cu}^{(t)}$	w/o $M_{cu}^{(t)}$	w/o $M_{fm}^{(t)}$	with $M_{fm}^{(t)}$ and $M_{cu}^{(t)}$
BasketballDrill	-8.69	-8.73	-8.77	-8.9
BQMall	-8.9	-8.99	-9.06	-9.15
PartyScene	-7.39	-7.46	-7.51	-7.59
RaceHorses	-3.62	-3.71	-3.76	-3.85
BasketballPass	-9.66	-9.69	-9.72	-9.83
BQSquare	-18.51	-18.55	-18.59	-18.65
BlowingBubbles	-7.23	-7.32	-7.34	-7.44
RaceHorses	-5.7	-5.75	-5.82	-5.9
Average	-8.71	-8.78	-8.82	-8.91

Table 4.6: Selection ratios of three ILF modes on average class C and class D.

QP	VVC ILF	SECUM	REOF
22	30.12	13.74	56.14
27	22.14	33.69	44.17
32	17.42	42.41	40.17
37	22.24	52.49	25.27
Average	22.98	35.58	41.44

Table 4.7: BD-rate (%) measurement of the proposed STILF-AMS compared to the anchor VVC under AI, LDP, LDB, and RA configurations.

Class-Sequence	AI	LDP			LDB			RA		
	Y	Y	U	V	Y	U	V	Y	U	V
A1-Tango2	-2.92	-4.98	0.67	0.4	-4.87	-0.61	-0.47	-4.67	-0.81	-0.72
A1-FoodMarket4	-4.51	-4.64	-0.61	-0.64	-4.66	-0.21	-0.68	-4.25	-0.29	-0.26
A1-Campfire	-2.81	-3.63	-0.33	-0.18	-3.15	-0.1	-0.17	-3.47	-0.01	-0.17
A2-CatRobot1	-3.04	-4.91	-0.02	-0.35	-4.89	-0.32	-0.17	-5.12	-0.67	-0.37
A2-DaylightRoad2	-2.2	-5.17	-0.09	-0.45	-5.06	-0.21	0.27	-5.34	0.16	-0.28
A2-ParkRunning3	-1.65	-2.21	-0.21	-0.4	-2.03	-0.2	-0.22	-2	-0.09	-0.3
B-MarketPlace	-3.06	-6.86	-0.95	-0.83	-6.79	-0.55	-1.73	-5.5	0.57	-0.06
B-RitualDance	-4.43	-4.36	-0.22	0.16	-4.29	0.43	-0.68	-3.89	-0.39	-0.07
B-Cactus	-3.27	-5.85	-0.58	-0.65	-8.24	-3.4	-2.48	-4.93	0.01	0.01
B-BasketballDrive	-3.69	-5.65	-1.34	-0.79	-5.82	-0.41	-0.23	-5.08	-0.48	-0.46
B-BQTerrace	-2.98	-9.01	-1.08	-1.58	-7.41	-1.22	-1	-6.12	0.23	0.36
C-BasketballDrill	-7.25	-8.9	-1.53	-1.22	-8.33	0.18	0.62	-6.55	-0.39	0.87
C-BQMall	-4.92	-9.15	-2.35	-0.89	-9	-0.57	-1.92	-6.8	-0.02	-0.27
C-PartyScene	-3.77	-7.59	-0.63	-0.79	-7.21	-0.91	-1.33	-5.15	0.43	0.38
C-RaceHorses	-2.06	-3.85	0.01	0.17	-3.7	-0.73	0.31	-3.28	-0.73	0.11
D-BasketballPass	-5.81	-9.83	-1.02	-0.58	-9.29	-0.55	-0.88	-6.47	-1.19	0.62
D-BQSquare	-7.34	-18.65	-3.26	-2.81	-18.62	-3.92	-3.04	-11.66	-0.93	-1.03
D-BlowingBubbles	-4.25	-7.44	-0.8	0.27	-6.88	-0.6	-0.34	-5.64	0.36	-0.07
D-RaceHorses	-3.73	-5.9	-0.63	-0.61	-5.58	0.38	-0.37	-4.46	-0.24	0.54
E-FourPeople	-5.63	-8.57	-1.43	-1.41	-8.55	-0.57	-0.25	-7.04	-0.01	-0.05
E-Johnny	-6.04	-10.15	-1.09	-2.31	-10.07	-2.88	-2.82	-8.05	-0.29	-0.75
E-KristenAndSara	-5.4	-8.92	1.08	0.29	-8.12	-0.32	0.73	-5.62	-0.18	0.46
Average A1	-3.41	-4.42	-0.09	-0.14	-2.95	-0.22	-0.21	-2.74	0.26	0.46
Average A2	-2.3	-4.1	-0.11	-0.4	-3.31	0.12	0.13	-3.51	-0.13	-0.05
Average B	-3.49	-6.35	-0.83	-0.74	-5.27	-0.39	-0.77	-3.95	0.45	0.37
Average C	-4.5	-7.37	-1.13	-0.68	-6.53	-0.54	-0.53	-4.98	-0.32	0.33
Average D	-5.28	-10.45	-1.43	-0.93	-9.44	-1.05	-1.07	-6.42	-0.47	0.05
Average E	-5.69	-9.21	-0.48	-1.14	-7.66	-0.54	-0.04	-5.95	0.13	0.17
Average sequences	-4.13	-7.1	-0.75	-0.72	-6.93	-0.79	-0.77	-5.5	-0.23	-0.07

Table 4.8: Encoding time (ET) and decoding time (DT) compared to VVC under AI and average of inter coding configurations.

Class	AI		LDP, LDB, and RA	
	ET (%)	DT (%)	ET (%)	DT (%)
A1	111	14132	128	19238
A2	110	12808	134	17434
B	109	11290	111	3153
C	102	3323	96	1271
D	104	31746	173	12957
E	106	19144	155	7834
Avg.	107	15407	133	10315

Table 4.9: Y BD-rate (%) comparison between STILF-AMS and SOTA deep learning-based Inloop filtering works (anchor VTM 6.0) on AI and RA configurations.

Sequence	All Intra				Random Access			
	VCNN [102]	ALRF [103]	STILF [1]	STILF -AMS	VCNN [102]	ALRF [103]	STILF [1]	STILF -AMS
A1-Tango2	-1.74	-0.58	-2.7	-3.02	-2.45	-1.15	-3.83	-4.13
A1-FoodMarket4	-1.54	-0.46	-3.97	-4.21	-3.95	-2.67	-3.49	-4.65
A1-Campfire	-1.62	-0.45	-2.11	-2.61	-2.31	-1.51	-2.43	-3.67
A2-CatRobot1	-3.48	-1.98	-3	-2.84	-4.09	-2.09	-4.98	-5.22
A2-DaylightRoad2	-2.22	-0.32	-2.23	-2.1	-5.77	-2.4	-5.38	-5.84
A2-ParkRunning3	-2.56	-1.17	-1.22	-1.75	-1.77	-0.91	-1.86	-2.2
B-MarketPlace	-5.58	-3.48	-2.42	-3.26	-3.97	-3.42	-3.32	-6
B-RitualDance	-2.63	-1.13	-4.38	-4.53	-2.68	-1.46	-3.48	-4.19
B-Cactus	-1.39	-1.23	-3.29	-3.37	-1.85	-2.54	-5.02	-5.03
B-BasketballDrive	-1.17	-0.03	-3.48	-3.69	-3.71	-1.26	-5.01	-5.08
B-BQTerrace	-2.26	-0.6	-3.38	-3.08	-3.52	-2.24	-7	-6.32
C-BasketballDrill	-6.99	-3.77	-7.17	-7.25	-4.22	-2.42	-7.22	-6.85
C-BQMall	-4.97	-2.85	-4.81	-4.52	-3.68	-3.08	-7.13	-7.3
C-PartyScene	-3.38	-2.18	-3.7	-3.67	-2.77	-2.61	-5.09	-5.35
C-RaceHorses	-2.26	-0.78	-3.11	-1.76	-2.4	-1.25	-3.27	-3.78
D-BasketballPass	-6.3	-3.88	-5.3	-5.81	-2.55	-2.11	-6.42	-6.57
D-BQSquare	-4.52	-3.01	-7.3	-7.64	-3.49	-2.19	-11.55	-12.26
D-BlowingBubbles	-6.18	-3.45	-3.96	-4.25	-6.83	-4.29	-5.85	-5.64
D-RaceHorses	-5.14	-3.89	-3.87	-3.73	-4.16	-2.51	-4.6	-4.86
E-FourPeople	-5.6	-3.89	-5.26	-5.73	-4.93	-4.12	-7.12	-7.44
E-Johnny	-4.14	-2.45	-5.73	-5.84	-3.77	-3.6	-9.5	-8.55
E-KristenAndSara	-4.3	-2.42	-5.33	-5.6	-3.56	-2.83	-5.75	-5.62
Average A1	-1.63	-2	-2.93	-3.28	-2.9	-1.95	-3.25	-4.15
Average A2	-2.75	-1.16	-2.15	-2.23	-3.88	-1.8	-4.07	-4.42
Average B	-2.61	-1.29	-3.39	-3.59	-3.15	-2.18	-4.77	-5.32
Average C	-4.4	-2.4	-4.7	-4.3	-3.27	-2.34	-5.68	-5.82
Average D	-5.53	-3.56	-5.11	-5.36	-4.26	-2.78	-7.1	-7.33
Average E	-4.68	-2.92	-5.44	-5.72	-4.09	-3.52	-7.46	-7.2
Average All	-3.63	-2	-3.99	-4.1	-3.57	-2.39	-5.42	-5.75

Table 4.10: Y BD-rate (%) comparison between STILF-AMS and SOTA deep learning-based Inloop filtering works (anchor VTM 6.0) on Low Delay configurations.

Sequence	Low Delay P			Low Delay B		
	ALRF [103]	STILF [1]	STILF-AMS	ALRF [103]	STILF [1]	STILF-AMS
A1-Tango2	-0.73	-4.02	-5.08	-0.76	-3.62	-4.87
A1-FoodMarket4	-2.15	-3.29	-4.64	-2.17	-3.39	-4.46
A1-Campfire	-1.68	-2.26	-3.43	-1.51	-2.46	-3.15
A2-CatRobot1	-2.02	-4.81	-4.91	-2.09	-5.31	-5.09
A2-DaylightRoad2	-2.53	-4.89	-5.27	-2.4	-5.09	-4.96
A2-ParkRunning3	-0.9	-1.78	-2.21	-0.91	-1.98	-2.33
B-MarketPlace	-3.5	-3.61	-6.66	-3.42	-3.91	-6.69
B-RitualDance	-1.37	-4.44	-3.96	-1.46	-4.44	-4.49
B-Cactus	-2.21	-5.42	-6.15	-2.14	-5.72	-8.44
B-BasketballDrive	-0.61	-5.04	-5.95	-0.63	-5.34	-5.72
B-BQTerrace	-3.07	-7.76	-8.81	-2.48	-7.46	-7.41
C-BasketballDrill	-3.05	-7.97	-8.6	-2.91	-8.07	-8.23
C-BQMall	-3.26	-8.96	-9.15	-3.16	-8.86	-9.1
C-PartyScene	-2.85	-6.84	-7.79	-2.8	-7.14	-7.11
C-RaceHorses	-0.91	-3.92	-3.75	-0.91	-4.12	-3.8
D-BasketballPass	-2.73	-8.17	-9.83	-2.25	-8.77	-9.39
D-BQSquare	-2.43	-16.42	-18.65	-2.49	-16.32	-18.42
D-BlowingBubbles	-4.61	-6.64	-7.34	-4.38	-6.44	-6.88
D-RaceHorses	-2.48	-6.21	-5.9	-2.42	-6.21	-5.58
E-FourPeople	-4.66	-8.72	-8.57	-4.43	-8.52	-8.15
E-Johnny	-4.28	-10.87	-10.35	-3.67	-11.17	-9.97
E-KristenAndSara	-3.57	-7.75	-8.72	-3.99	-8.35	-8.12
Average A1	-2.53	-3.19	-4.38	-1.84	-3.16	-4.16
Average A2	-1.82	-3.83	-4.13	-1.8	-4.13	-4.12
Average B	-2.15	-5.25	-6.31	-2.03	-5.37	-6.55
Average C	-2.52	-6.92	-7.32	-2.45	-7.05	-7.06
Average D	-3.06	-9.36	-10.43	-2.89	-9.44	-10.07
Average E	-4.17	-9.11	-9.21	-4.03	-9.35	-8.74
Average All	-2.53	-6.35	-7.08	-2.43	-6.49	-6.92

Chapter 5

Conclusion

In this work, deep learning approaches for enhancing reconstructed frames in various codecs have been introduced. This dissertation includes three main proposals, each contributes in either encoder or decoder, adapting bandwidth and resolution requirements as well as minimizing the transmitted rates for modern visual-enabled applications.

In improving reference frames of video encoder, we propose a deep learning-based method for fractional interpolation. We also solve two main problems of applying CNN-based techniques for fractional interpolation in video coding: CNN may change integer pixels after convolution and the lack of training set for fractional interpolation in video coding since fractional samples do not really exist. To solve the first problem, we generate fractional pixels from integer pixel rather than an interpolated image that contains integer and fractional samples. To deal with training set problem, we assume and extract integer and fractional samples in an image and learn a mapping between interpolated integer pixels and fractional samples to avoid the motion shift problem. To further improve coding performance, an RDO-based fractional interpolation selection is implemented at CU level. As a result, we obtain an average BD-rate reduction of 1.2%-2.9% under low various coding configurations.

In improving CS decoders, a deep learning-based image enhancement approach is proposed using multiple reconstructed signals. We take advantage of CS underdetermined problem: be able to recover multiple Cs reconstructed images. CS reconstructed images are then scored and ranked by a No-reference quality ranking module before feeding to the quality enhancement module. In the quality enhancement module, low-level and high-level features extracted from CS reconstructed images are exploited and enriched by the proposed Recurrent dense skip connections block. As a result,

1.88-8.07dB PSNR improvements under the sampling rates of 0.125-0.75 have been obtained. We further experimented on the effectiveness of CSIE-M with and without the No-reference quality ranking module. The result shows that 1.127 dB PSNR can be improved when using the No-reference Quality enhancement module. The proposal can be integrated into IoT imaging systems to enhance the CS reconstructed images, giving better visual quality for end-users and a promising approach for designing AIoT systems.

In improving video codecs, we take into account more coding resources at the encoder and the decoder for improving the in-loop filter by VVC. After being decoded by VVC, reconstructed frames area enhanced by the self-enhancement Convolutional neural network (CNN) with CU map (SECUM) and the reference-based enhancement CNN with the optical flow (REOF). The proposed SECUM and REOF are developed using VVC encoder and decoder coding information, which are well studied in the series of the longest establishing standards. Using this information ensures that spatial and temporal correlations are precisely exploited. To further improve the proposed deep learning-based ILFs for VVC standard, we propose a reinforcement learning-based autonomous mode selection. Note that the ILF mode selection increases the bitrate for each CTU, we define an agent that splits and filters images automatically. By formulating the acts of splitting and filtering as a decision-making problem, this work is the first to propose a reinforcement learning-based autonomous mode selection (AMS) approach. Predicting ILF mode and allowing CU to be split more, STILF-AMS requires *zero* extra bit while ensuring the quality of reconstructed images. As a result, we outperform VVC and the state-of-the-art deep learning-based ILF enhancement. Remarkably, up to 18% and an average of 5.9% bitrate savings have been obtained under all configurations.

Bibliography

[1] C. D. K. Pham, C. Fu, and J. Zhou, "Deep Learning based Spatial-Temporal In-loop filtering for Versatile Video Coding," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021, pp. 1861–1865.

[2] B. Bross, J. Chen, S. Liu, and Y.-K. Wang, "Versatile Video Coding (Draft 10)," *document Rep. JVET-S2001, Teleconference*, Apr. 2020.

[3] C. Evans, I. Julian, and F. Simon, "The sustainable future of video entertainment from creation to consumption," *Futuresource Consulting Ltd.: Hertfordshire, UK*, pp. 1–34, 2020.

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

[5] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG2000 still image compression standard," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36–58, 2001.

[6] P. Tudor, "MPEG-2 video compression," *Electronics & communication engineering journal*, vol. 7, no. 6, pp. 257–264, 1995.

[7] W. Gao, C. Reader, F. Wu, Y. He, L. Yu, H. Lu, S. Yang, T. Huang, and X. Pan, "Avs-the chinese next-generation video coding standard," *National association of broadcasters, Las Vegas*, 2004.

[8] J. Bankoski, J. Koleszar, L. Quillio, J. Salonen, P. Wilkins, and Y. Xu, "VP8 data format and decoding guide," in *RFC 6386*, 2011.

[9] A. Grange, P. De Rivaz, and J. Hunt, "VP9 bitstream & decoding process specification," *Version 0.6, March*, 2016.

[10] P. de Rivaz and J. Haughton, "AV1 bitstream & decoding process specification," *The Alliance for Open Media*, p. 182, 2018.

[11] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[12] G. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[13] J. Lainema, F. Bossen, W.-J. Han, J. Min, and K. Ugur, "Intra coding of the HEVC standard," *IEEE transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1792–1801, 2012.

[14] H. Lakshman, B. Bross, H. Schwarz, and T. Wiegand, "Fractional-sample motion compensation using generalized interpolation," in *28th Picture Coding Symposium*. IEEE, 2010, pp. 530–533.

[15] C.-M. Fu, C.-Y. Chen, Y.-W. Huang, and S. Lei, "Sample adaptive offset for hevc," in *2011 IEEE 13th International Workshop on Multimedia Signal Processing*, 2011, pp. 1–5.

[16] A. Norkin, G. Bjontegaard, A. Fuldseth, M. Narroschke, M. Ikeda, K. Andersson, M. Zhou, and G. Van der Auwera, "Hevc deblocking filter," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1746–1754, 2012.

[17] J. Pfaff, A. Filippov, S. Liu, X. Zhao, J. Chen, S. De-Luxán-Hernández, T. Wiegand, V. Rufitskiy, A. K. Ramasubramonian, and G. Van der Auwera, "Intra prediction and mode coding in VVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3834–3847, 2021.

[18] M. Karczewicz, N. Hu, J. Taquet, C.-Y. Chen, K. Misra, K. Andersson, P. Yin, T. Lu, E. François, and J. Chen, "Vvc in-loop filters," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3907–3925, 2021.

[19] D. L. Donoho *et al.*, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[20] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on information theory*, vol. 53, no. 12, pp. 4655–4666, 2007.

[21] T. T. Do, L. Gan, N. Nguyen, and T. D. Tran, "Sparsity adaptive matching pursuit algorithm for practical compressed sensing," in *2008 42nd Asilomar Conference on Signals, Systems and Computers*. IEEE, 2008, pp. 581–587.

[22] D. Needell and J. A. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Applied and computational harmonic analysis*, vol. 26, no. 3, pp. 301–321, 2009.

[23] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM review*, vol. 43, no. 1, pp. 129–159, 2001.

[24] W. Lu and N. Vaswani, "Modified basis pursuit denoising (modified-bpdn) for noisy compressive sensing with partially known support," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2010, pp. 3926–3929.

[25] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[26] E. Candes and J. Romberg, "l1-magic: Recovery of sparse signals via convex programming," *URL: www. acm. caltech. edu/l1magic/downloads/l1magic. pdf*, vol. 4, p. 14, 2005.

[27] E. Van Den Berg and M. P. Friedlander, "Probing the pareto frontier for basis pursuit solutions," *SIAM Journal on Scientific Computing*, vol. 31, no. 2, pp. 890–912, 2008.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[29] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 2147–2154.

[30] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[31] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[32] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European conference on computer vision*. Springer, 2014, pp. 184–199.

[33] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1646–1654.

[34] C. Dong, Y. Deng, C. C. Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 576–584.

[35] Z. Wang, D. Liu, S. Chang, Q. Ling, Y. Yang, and T. S. Huang, "D3: Deep dual-domain based fast restoration of JPEG-compressed images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2764–2772.

[36] J. Guo and H. Chao, "One-to-many network for visually pleasing compression artifacts reduction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3038–3047.

[37] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, "Dvc: An end-to-end deep video compression framework," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 006–11 015.

[38] G. Lu, X. Zhang, W. Ouyang, L. Chen, Z. Gao, and D. Xu, "An end-to-end learning framework for video compression," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[39] J. Li, B. Li, J. Xu, R. Xiong, and W. Gao, "Fully connected network-based intra prediction for image coding," *IEEE Transactions on Image Processing*, vol. 27, no. 7, pp. 3236–3247, 2018.

[40] W. Cui, T. Zhang, S. Zhang, F. Jiang, W. Zuo, Z. Wan, and D. Zhao, "Convolutional neural networks based intra prediction for hevc," in *2017 Data Compression Conference (DCC)*, 2017, pp. 436–436.

[41] Y. Wang, X. Fan, C. Jia, D. Zhao, and W. Gao, "Neural network based inter prediction for hevc," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*, 2018, pp. 1–6.

[42] Y. Wang, X. Fan, R. Xiong, D. Zhao, and W. Gao, "Neural network-based enhancement to inter prediction for video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 2, pp. 826–838, 2022.

[43] N. Yan, D. Liu, H. Li, and F. Wu, "A convolutional neural network approach for half-pel interpolation in video coding," in *2017 IEEE international symposium on circuits and systems (ISCAS)*. IEEE, 2017, pp. 1–4.

[44] H. Zhang, L. Song, Z. Luo, and X. Yang, "Learning a convolutional neural network for fractional interpolation in hevc inter coding," in *2017 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, 2017, pp. 1–4.

[45] S. Xia, W. Yang, Y. Hu, S. Ma, and J. Liu, "A group variational transformation neural network for fractional interpolation of video coding," in *2018 Data Compression Conference*. IEEE, 2018, pp. 127–136.

[46] C. Pham and J. Zhou, "icnn: A convolutional neural network for fractional interpolation in video coding," in *International Symposium on Artificial Intelligence and Robotics*, 2019.

[47] N. Yan, D. Liu, H. Li, B. Li, L. Li, and F. Wu, "Convolutional neural network-based fractional-pixel motion compensation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 3, pp. 840–853, 2019.

[48] J. Liu, S. Xia, W. Yang, M. Li, and D. Liu, "One-for-all: Grouped variation network-based fractional interpolation in video coding," *IEEE Transactions on Image Processing*, vol. 28, no. 5, pp. 2140–2151, 2018.

[49] S. Xia, W. Yang, Y. Hu, W.-H. Cheng, and J. Liu, "Switch mode based deep fractional interpolation in video coding," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2019, pp. 1–5.

[50] R. Yang, M. Xu, T. Liu, Z. Wang, and Z. Guan, "Enhancing quality for hevc compressed videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 7, pp. 2039–2054, 2018.

[51] Z. Jin, P. An, C. Yang, and L. Shen, "Quality enhancement for intra frame coding via cnns: An adversarial approach," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 1368–1372.

[52] T. Wang, W. Xiao, M. Chen, and H. Chao, "The multi-scale deep decoder for the standard hevc bitstreams," in *2018 Data Compression Conference*, 2018, pp. 197–206.

[53] X. He, Q. Hu, X. Zhang, C. Zhang, W. Lin, and X. Han, "Enhancing hevc compressed videos with a partition-masked convolutional neural network," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 216–220.

[54] L. Ma, Y. Tian, and T. Huang, "Residual-based video restoration for hevc intra coding," in *2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM)*. IEEE, 2018, pp. 1–7.

[55] D. E. Knuth, "Dynamic huffman coding," *Journal of algorithms*, vol. 6, no. 2, pp. 163–180, 1985.

[56] X. Meng, C. Chen, S. Zhu, and B. Zeng, "A new hevc in-loop filter based on multi-channel long-short-term dependency residual networks," in *2018 Data Compression Conference*, 2018, pp. 187–196.

[57] J. Kang, S. Kim, and K. M. Lee, "Multi-modal/multi-scale convolutional neural network based in-loop filter design for next generation video codec," in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 26–30.

[58] X. Song, J. Yao, L. Zhou, L. Wang, X. Wu, D. Xie, and S. Pu, "A practical convolutional neural network as loop filter for intra frame," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 1133–1137.

[59] W.-S. Park and M. Kim, "Cnn-based in-loop filtering for coding efficiency improvement," in *2016 IEEE 12th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*. IEEE, 2016, pp. 1–5.

[60] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2472–2481.

[61] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[62] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h. 264/avc video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.

[63] J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards—including high efficiency video coding (hevc)." *IEEE Transactions on circuits and systems for video technology*, pp. 1669–1684., 22.12 (2012).

[64] Y. Ye, G. Motta, and M. Karczewicz, "Enhanced adaptive interpolation filters for video coding," in *2010 Data Compression Conference*. IEEE, 2010, pp. 435–444.

[65] S. Wittmann and T. Wedi, "Separable adaptive interpolation filter for video coding," in *2008 15th IEEE International Conference on Image Processing*. IEEE, 2008, pp. 2500–2503.

[66] Z. Guo, D. Zhou, and S. Goto, "An optimized mc interpolation architecture for hevc," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 1117–1120.

[67] F. Bossen, "Common test conditions and software reference configurations," *Joint Collaborative Team on Video Coding (JCT-VC) of ITUT SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Doc. JCTVC-J1100, Stockholm, Sweden*, July 2012.

[68] G. Bjontegaard, "Calculation of average psnr differences between rd-curves," *VCEG-M33*, 2001.

[69] Y. Zhang, P. Wang, H. Huang, Y. Zhu, D. Xiao, and Y. Xiang, "Privacy-assured fogcs: Chaotic compressive sensing for secure industrial big image data processing in fog computing," *IEEE Transactions on Industrial Informatics*, 2020.

[70] Y. Zhang, Q. He, G. Chen, X. Zhang, and Y. Xiang, "A low-overhead, confidentiality-assured, and authenticated data acquisition framework for iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 12, pp. 7566–7578, 2020.

[71] Y. Zhang, P. Wang, L. Fang, X. He, H. Han, and B. Chen, "Secure transmission of compressed sampling data using edge clouds," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6641–6651, 2020.

[72] C. Jia, S. Wang, X. Zhang, S. Wang, J. Liu, S. Pu, and S. Ma, "Content-aware convolutional neural network for in-loop filtering in high efficiency video coding," *IEEE Transactions on Image Processing*, vol. 28, no. 7, pp. 3343–3356, 2019.

[73] S. Anwar and N. Barnes, "Real image denoising with feature attention," *IEEE International Conference on Computer Vision (ICCV-Oral)*, 2019.

[74] C. Tian, Y. Xu, Z. Li, W. Zuo, L. Fei, and H. Liu, "Attention-guided cnn for image denoising," *Neural Networks*, vol. 124, pp. 117–129, 2020.

[75] Z. Yue, Q. Zhao, L. Zhang, and D. Meng, "Dual adversarial network: Toward real-world noise removal and noise generation," in *European Conference on Computer Vision*. Springer, 2020, pp. 41–58.

[76] H. Qiu, Q. Zheng, G. Memmi, J. Lu, M. Qiu, and B. Thuraisingham, "Deep residual learning-based enhanced jpeg compression in the internet of things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 2124–2133, 2020.

[77] M. El Helou and S. Süsstrunk, "Blind universal Bayesian image denoising with Gaussian noise level learning," *IEEE Transactions on Image Processing*, vol. 29, pp. 4885–4897, 2020.

[78] Z. Guan, Q. Xing, M. Xu, R. Yang, T. Liu, and Z. Wang, "Mfqe 2.0: A new approach for multi-frame quality enhancement on compressed video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019.

[79] R. Yang, M. Xu, Z. Wang, and T. Li, "Multi-frame quality enhancement for compressed video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6664–6673.

[80] D. Vijayalakshmi, M. K. Nath, and O. P. Acharya, "A comprehensive survey on image contrast enhancement techniques in spatial domain," *Sensing and Imaging*, vol. 21, no. 1, pp. 1–40, 2020.

[81] D. Połap, "An adaptive genetic algorithm as a supporting mechanism for microscopy image analysis in a cascade of convolution neural networks," *Applied Soft Computing*, vol. 97, p. 106824, 2020.

[82] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[83] Z. Li, J. Yang, Z. Liu, X. Yang, G. Jeon, and W. Wu, "Feedback network for image super-resolution," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[84] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[85] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics,"

in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, 2001, pp. 416–423 vol.2.

[86] M. Bevilacqua, A. Roumy, C. Guillemot, and M. line Alberi Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2012, pp. 135.1–135.10.

[87] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *International conference on curves and surfaces*. Springer, 2010, pp. 711–730.

[88] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5197–5206.

[89] N. Ponomarenko, L. Jin, O. Ieremeiev, V. Lukin, K. Egiazarian, J. Astola, B. Vozel, K. Chehdi, M. Carli, F. Battisti *et al.*, "Image database tid2013: Peculiarities, results and perspectives," *Signal Processing: Image Communication*, vol. 30, pp. 57–77, 2015.

[90] X. Liu, J. van de Weijer, and A. D. Bagdanov, "Rankiqa: Learning from rankings for no-reference image quality assessment," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1040–1049.

[91] J. M. Bioucas-Dias and M. A. Figueiredo, "A new twist: Two-step iterative shrinkage/thresholding algorithms for image restoration," *IEEE Transactions on Image processing*, vol. 16, no. 12, pp. 2992–3004, 2007.

[92] W. Deng, W. Yin, and Y. Zhang, "Group sparse optimization by alternating direction method," in *Wavelets and Sparsity XV*, vol. 8858. International Society for Optics and Photonics, 2013, p. 88580R.

[93] D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck, "Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit," *IEEE transactions on Information Theory*, vol. 58, no. 2, pp. 1094–1121, 2012.

[94] L. Li, H. Li, D. Liu, Z. Li, H. Yang, S. Lin, H. Chen, and F. Wu, "An efficient four-parameter affine motion model for video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 8, pp. 1934–1948, 2018.

[95] K. Zhang, Y.-W. Chen, L. Zhang, W.-J. Chien, and M. Karczewicz, "An improved framework of affine motion compensation in video coding," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1456–1469, 2019.

[96] C.-Y. Tsai, C.-Y. Chen, T. Yamakage, I. S. Chong, Y.-W. Huang, C.-M. Fu, T. Itoh, T. Watanabe, T. Chujoh, M. Karczewicz, and S.-M. Lei, "Adaptive loop filtering for video coding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 6, pp. 934–945, 2013.

[97] Y. Dai, D. Liu, Y. Li, and F. Wu, "AHG9: CNN-based in-loop filter proposed by USTC," in *document JVET-M0510, 13th JVET meeting*, 2019.

[98] S. N. K. Kawamura, "A result of convolutional neural network filter," *document Rep. JVET-M0872, Marrakech, MA, USA*, Jan. 2019.

[99] Z. Huang, Y. Li, and J. Sun, "Multi-Gradient Convolutional Neural Network Based In-Loop Filter For VVC," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2020, pp. 1–6.

[100] Y. Li, L. Zhao, S. Liu, Y. Wang, Z. Chen, and X. Li, "Test results of dense residual convolutional neural network based in-loop filter," *document Rep. JVET-M0508, Marrakech, MA, USA*, Jan. 2019.

[101] D. Ding, L. Kong, G. Chen, Z. Liu, and Y. Fang, "A switchable deep learning approach for in-loop filtering in video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 7, pp. 1871–1887, 2019.

[102] Z. Huang, J. Sun, X. Guo, and M. Shang, "One-for-all: An efficient variable convolution neural network for in-loop filter of vvc," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2021.

[103] ——, "Adaptive deep reinforcement learning based in-loop filter for vvc," *IEEE Transactions on Image Processing*, vol. 30, pp. 5439–5451, 2021.

[104] T. Li, M. Xu, C. Zhu, R. Yang, Z. Wang, and Z. Guan, "A deep learning approach for multi-frame in-loop filter of hevc," *IEEE Transactions on Image Processing*, vol. 28, no. 11, pp. 5663–5678, 2019.

[105] M. Wang, S. Wan, H. Gong, and M. Ma, "Attention-based dual-scale CNN in-loop filter for Versatile Video Coding," *IEEE Access*, vol. 7, pp. 145 214–145 226, 2019.

[106] S. Chen, Z. Chen, Y. Wang, and S. Liu, "In-loop filter with dense residual convolutional neural network for VVC," in *2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 2020, pp. 149–152.

[107] Y. Dai, D. Liu, and F. Wu, "A convolutional neural network approach for post-processing in HEVC intra coding," in *International Conference on Multimedia Modeling*. Springer, 2017, pp. 28–39.

[108] T. Laude and J. Ostermann, "Deep learning-based intra prediction mode decision for hevc," in *2016 Picture Coding Symposium (PCS)*. IEEE, 2016, pp. 1–5.

[109] Z. Liu, X. Yu, Y. Gao, S. Chen, X. Ji, and D. Wang, "Cu Partition Mode Decision for HEVC Hardwired Intra Encoder using Convolution Neural Network," *IEEE Transactions on Image Processing*, vol. 25, no. 11, pp. 5088–5103, 2016.

[110] Z. Chen, J. Shi, and W. Li, "Learned fast hevc intra coding," *IEEE Transactions on Image Processing*, vol. 29, pp. 5431–5446, 2020.

[111] F. Zaki, A. E. Mohamed, and S. G. Sayed, "Ctunet: A deep learning-based framework for fast ctu partitioning of h265/hevc intra-coding," *Ain Shams Engineering Journal*, vol. 12, no. 2, pp. 1859–1866, 2021.

[112] M. Xu, T. Li, Z. Wang, X. Deng, R. Yang, and Z. Guan, "Reducing Complexity of HEVC: A Deep Learning Approach," *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 5044–5059, 2018.

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

[114] S. Kuanar, K. Rao, M. Bilas, and J. Bredow, "Adaptive cu mode selection in hevc intra prediction: A deep learning approach," *Circuits, Systems, and Signal Processing*, vol. 38, no. 11, pp. 5081–5102, 2019.

[115] W. Han, S. Chang, D. Liu, M. Yu, M. Witbrock, and T. Huang, "Image super-resolution via dual-state recurrent networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1654–1663.

[116] S. Woo, J. Park, J. Lee, and I. So Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[117] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[118] R. Furuta, N. Inoue, and T. Yamasaki, "Fully convolutional network with multi-step reinforcement learning for image processing," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[119] ——, "Pixelrl: Fully convolutional network with reinforcement learning for image processing," *IEEE Transactions on Multimedia (TMM)*, vol. 22, no. 7, pp. 1704–1719, 2020.

[120] K. Yu, C. Dong, L. Lin, and C. C. Loy, "Crafting a toolchain for image restoration by deep reinforcement learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2443–2452.

[121] C. Montgomery *et al.*, "Xiph. org Video Test Media (Derf's collection), the xiph open source community, 1994," *Online, https://media.xiph.org/video/derf*.

[122] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[123] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[124] S. Tokui, R. Okuta, T. Akiba, Y. Niitani, T. Ogawa, S. Saito, S. Suzuki, K. Uenishi, B. Vogel, and H. Yamazaki Vincent, "Chainer: A deep learning framework for accelerating the research cycle," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 2002–2011.

[125] Y. Fujita, P. Nagarajan, T. Kataoka, and T. Ishikawa, "ChainerRL: A Deep Reinforcement Learning Library," *Journal of Machine Learning Research*, vol. 22, no. 77, pp. 1–14, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-376.html

[126] G. Bjontegaard, "Calculation of Average PSNR Differences Between RD-Curves," *document ITU-T Q. 6/SG16 VCEG, 15th Meeting, Austin, TX, USA*, 2001.

[127] J. Boyce, K. Suehring, X. Li, and V. Seregin, "JVET common test conditions and software reference configurations," *document Rep. JVET-J1010, San Diego, USA*, 2018.

List of Abbreviations

A3C Asynchronous advantage actor-critic

AI All Intra

AIoT Artificial Intelligence of Things

ALF Adaptive loop filter

AP Average pooling

AVC Advanced Video Coding

BD-rate Bjontegaard Delta-Rate

CABAC Context-adaptive binary arithmetic coding

CNN Convolutional Neural Network

CPU Central processing unit

CS Compressive sensing

CTU Coding Tree Unit

CU Coding Unit

DBF Deblocking filter

DL Deep Learning

FPS Frame Per Second

GPU Graphics processing unit

HD High Definition

HEVC High Efficiency Video Coding

ILF In-loop filtering

IoT Internet of Things

JPEG Joint Photographic Experts Group

JVET Joint Video Exploration Team

LCC Linear Correlation Coefficient

LDB Low Delay B

LDP Low Delay P

MOS Mean opinion score

MP Max pooling

MPEG Moving Picture Experts Group

POC Picture order count

PReLU Parametric Rectified Linear Unit

PSNR Peak Signal-to-Noise Ratio

QP Quantization Parameter

RA Random Access

RD Rate-Distortion

ReLU Rectified Linear Unit

RL Reinforcement learning

SAD Sum of absolute difference

SAO Sample adaptive offset

SD Standard Definition

SOTA State-of-the-art

SR Sampling rate

SROCC Spearman's Rank Order Correlation Coefficient

SSIM Structural Similarity Index

VTM VVC Test Model

VVC Versatile Video Coding