Video coding that pursues the highest compression efficiency is the art of computing for rate-distortion optimization. The optimization has been approached in different ways, exemplified by two typical frameworks: block-based hybrid video coding and end-to-end learned video coding. The block-based hybrid framework encompasses more and more coding modes that are available at the decoder side; an encoder tries to search for the optimal coding mode for each block to be coded. This is an online, discrete, search-based optimization strategy. The end-to-end learned framework embraces more and more sophisticated neural networks; the network parameters are learned from a collection of videos, typically using gradient descent-based methods. This is an offline, continuous, numerical optimization strategy. Having analyzed these two strategies, both conceptually and with concrete schemes, this paper suggests investigating hybrid-optimization video coding, that is to combine online and offline, discrete and continuous, search-based and numerical optimization. For instance, we propose a hybrid-optimization video coding scheme, where the decoder consists of trained neural networks and supports several coding modes, and the encoder adopts both numerical and search-based algorithms for the online optimization. Our scheme achieves promising compression efficiency on par with H.265/HM for the random-access configuration.

CCS Concepts: • Computing methodologies → Image compression;

Additional Key Words and Phrases: Hybrid optimization, numerical optimization, offline optimization, online optimization, rate-distortion optimization, search-based optimization, video coding

ACM Reference Format:
Shuai Huo, Dong Liu, Haotian Zhang, Li Li, Siwei Ma, Feng Wu, and Wen Gao. 2024. Towards Hybrid-Optimization Video Coding. ACM Comput. Surv. 56, 9, Article 225 (April 2024), 36 pages. https://doi.org/10.1145/3652148

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM 0360-0300/2024/04-ART225
https://doi.org/10.1145/3652148

This work is supported by the Natural Science Foundation of China under Grants 61931014 and 62021001 and by the Fundamental Research Funds for the Central Universities under Grant WK3490000006.

Authors’ addresses: S. Huo, D. Liu (Corresponding author), H. Zhang, L. Li, and F. Wu, University of Science and Technology of China, No. 443 Huangshan Road, Hefei 230027, China; e-mails: huoshuai@mail.ustc.edu.cn, dongeliu@ustc.edu.cn, zhanghaotian@mail.ustc.edu.cn, lil1@ustc.edu.cn, fengwu@ustc.edu.cn; S. Ma, Peking University, No. 5 Yiheyuan Road, Beijing 100871, China; e-mail: swma@pku.edu.cn; W. Gao, Peng Cheng Laboratory, No. 2 Xingkeyi Street, Shenzhen 518055, China, and Peking University, No. 5 Yiheyuan Road, Beijing 100871, China; e-mail: wgao@pku.edu.cn.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
1 INTRODUCTION

Video coding refers to the art of computing to convert digital videos to binary codes (encoding), and convert binary codes back to digital videos (decoding) [108]. A digital video is indeed a sequence of pictures that are captured continuously, where a picture, also known as a frame, is conventionally recorded as a two-dimensional (2D) array of pixels. A video consisting of many pixels is of high volume, e.g., a 1080p (each frame has 1920 × 1080 pixels), 25 frames-per-second (fps), 8-bit RGB (each pixel has three color values: red, green, blue, and each color value is represented in 8 bits) video has a data rate of 155,520,000 bytes per second. Such video data shall be compressed into several orders of magnitude smaller codes, otherwise they cannot be well accommodated by the existing storage or transmission systems. Thus, video coding is also known as video compression. As a special case of video coding, the coding of only one picture is also known as image coding or image compression.

To compress videos, video coding schemes normally use carefully designed technologies to convert pixels to other kinds of symbols (and further convert symbols to binary codes) to reduce the coding rate, and convert the symbols back to reconstruct the pixels. The conversion, however, may lose information and cause certain differences between the reconstructed and the original pixels (termed distortion). Thus, the compression efficiency of video coding schemes shall be evaluated at both aspects: rate and distortion. Ideally, video coding shall pursue the lowest possible rate as well as the lowest possible distortion. This is termed rate-distortion (RD) optimization [96, 113].

Video coding technologies have been developed for more than five decades, when numerous efforts were made to design various methods for converting pixels to different kinds of symbols. For one example, as spatially neighboring pixels have high correlations among them, a small 2D array of pixels (termed a block) may be transformed by, e.g., 2D discrete cosine transform (DCT) to a set of coefficients [4]. These coefficients may be then converted to binary codes. For another example, as consecutive video frames have much duplicate content, a block in one frame may be very similar to another block in another frame, where the two blocks have different locations, indicating that the corresponding content was moving from one location to another location. Thus, a motion vector (MV) may be calculated as the difference between the two locations and then converted to binary codes. These different methods may be integrated together. A video coding scheme may divide a frame into multiple blocks, using the DCT-based method for some blocks and the MV-based method for the other blocks. In such a scheme, different methods are also known as different coding modes. The mode of each block shall be represented by a symbol and be coded, then the corresponding method would be used at the decoder side to convert the symbols back to pixels, e.g., performing 2D inverse DCT to convert the coefficients to pixels, or retrieving the block indicated by the MV to reconstruct the pixels. This multi-mode idea, also known as the block-based hybrid video coding, has been very popular and leads to a series of successful video coding schemes, some of which became standards: H.261 [47], H.262/MPEG-2 [49], H.263 [48], H.264/AVC [129], H.265/HEVC [112], H.266/VVC [11, 12], EVC [19], LCEVC [89], AV1 [33], and AVS3 [145] are all block-based hybrid video coding standards. Among these standards, H.264/AVC and H.265/HEVC are widely deployed, while H.266/VVC and AVS3 represent the highest compression efficiency till now. Historically, more and more modes were designed and integrated into video coding schemes, which was believed essential for improving compression efficiency [95, 128].

While manually designing the pixel-to-symbol conversion methods has achieved great success, another approach trying to obtain the conversion automatically has received increasing attention in recent years. This is attributed to the breakthrough of deep learning, which automates the construction of computing models for symbols (also known as features, representations, or latents in the deep learning literature) through learning from massive data, either supervised or unsupervised [56]. Deep learning-based video coding has been developed in two approaches: deep
tools that are used inside the block-based hybrid video coding framework, and deep schemes that are distinctly different [70]. Deep schemes are also known as end-to-end learned video coding, where the conversion from pixels to latents and the conversion from latents to pixels are jointly learned. Either conversion is realized by a deep neural network, and the network parameters are learned by, e.g., a gradient descent-based algorithm to minimize a predefined loss function on a given training dataset. End-to-end learned video coding has been developed for less than one decade, but it has demonstrated exciting potential in terms of compression efficiency: for image compression, end-to-end learned schemes have clearly surpassed H.266/VVC [34]; for low-delay video compression, which requires to code the frames in the natural temporal order, end-to-end learned schemes outperform H.266/VVC under certain conditions [60]; for random-access video compression, which aims at the highest compression efficiency by reordering the frames to code, end-to-end learned schemes perform on par with H.265/HEVC under certain conditions [15]. Nonetheless, end-to-end learned schemes are far from being as mature as the block-based hybrid schemes. There is no standardized solution for end-to-end learned video coding 1. The end-to-end learned schemes have not been widely deployed, so the practical usefulness and robustness are not fully verified.

Nowadays, video coding technologies have been utilized extensively in many disciplines, where the requirements for video coding are quite diverse. One kernel requirement is still to increase the compression efficiency to further reduce the storage or transmission cost. How can video coding technologies be further developed, especially, how can the compression efficiency be further improved? Since video coding pursues RD optimization, optimization theory, strategies, and algorithms are definitely crucial for the development of video coding. In this paper, we provide a comprehensive survey of video coding technologies from an optimization perspective. We argue that the block-based hybrid video coding and the end-to-end learned video coding are quite different in terms of optimization problems, algorithms, and occasions. We then propose to combine the advantages of both, and to study hybrid-optimization video coding. We design a concrete hybrid-optimization video coding scheme and investigate its compression efficiency. Our experimental results and analyses show that the idea of hybrid-optimization video coding is of great potential.

The remainder of this paper is organized as follows. In Section 2, we review the existing video coding technologies from an optimization perspective. In Section 3, we theoretically analyze the optimization problem for video coding and propose the hybrid-optimization idea. Section 4 presents our designed hybrid-optimization video coding scheme as a concrete implementation of the idea. Section 5 presents the experimental results. Section 6 discusses the potential of hybrid-optimization video coding, and Section 7 concludes this paper.

2 OVERVIEW OF VIDEO CODING: AN OPTIMIZATION PERSPECTIVE

In this section, we review the existing video coding technologies from an optimization perspective. For the existing technologies, block-based hybrid video coding and end-to-end learned video coding represent two dominant frameworks. The two frameworks differ significantly in terms of optimization problems, algorithms, and occasions. Even within one framework, there are some variants from the optimization perspective. Thus, we divide the existing technologies into five categories, which are discussed in the following five subsections, respectively. For ease of reading, Table 1 provides a road map of this section, and Table 2 summarizes the representative schemes/methods we will discuss. Note that we focus on the differences among video coding technologies from the optimization perspective, instead of providing a complete review of all the existing technologies.

---

1End-to-end learned image coding standards are expected to be published in 2024, including JPEG AI (https://jpeg.org/jpegai/) and IEEE 1857.11 (https://sagroups.ieee.org/fvc/)
Table 1. Categories of Video Coding from an Optimization Perspective

| Category                                                                 | Section | Offline Optimization | Online Optimization | End-to-End Optimization | Representative Studies |
|-------------------------------------------------------------------------|---------|----------------------|----------------------|--------------------------|------------------------|
| Block-Based Hybrid Video Coding (1984 - now)                           | 2.1     | N/A                  | Search-based         | No                       | [46, 112]              |
| Block-Based Hybrid Video Coding with Online Numerical Optimization (1996 - now) | 2.2     | N/A                  | Search-based/Numerical | No                       | [79, 62]               |
| Block-Based Hybrid Video Coding with Deep Tools (2015 - now)            | 2.3     | Numerical            | Search-based         | No                       | [23, 99]               |
| End-to-End Learned Video Coding (2015 - now)                           | 2.4     | Numerical            | N/A                  | Yes                      | [118, 58]              |
| End-to-End Learned Video Coding with Online Optimization (2019 - now)   | 2.5     | Numerical            | Search-based/Numerical | Yes                      | [14, 133]              |

Table 2. Representative Schemes/Methods Reviewed in Section 2

| Scheme/Method | Year | Optimization Objective | Key Features |
|---------------|------|------------------------|--------------|
| H.120 [46]    | 1984 | –                      | SKIP mode and INTRA mode |
| H.261 [47]    | 1990 | E                      | Motion-compensated prediction with integer-pixel MV |
| H.263 [48]    | 1995 | E                      | Half-pixel MV, bi-prediction, variable block size |
|             | [96, 113] | RD                  | Joint RD cost as the optimization objective |
| H.265 [112]   | 2012 | RD                    | Quarter-pixel MV, quadtree-based block partition |
| CCLM [146]    | 2018 | E                      | Linear models for cross-component prediction |
| ALF [119]     | 2013 | RD                    | Wiener filters for adaptive in-loop filtering |
| BDOF [5]      | 2010 | RD                    | Optical flow-based bi-prediction refinement |
| ARCNN [23]    | 2015 | D                      | NN-based reconstructed picture filtering |
| MIP [99]      | 2018 | RD                    | Matrix-based intra prediction with multiple modes |
|             | [84] | R                      | NN-based probability distribution estimation for entropy coding |
|             | [65] | RD                    | CNN-based down- and up-sampling for resampling-based coding |
|             | [26] | Accuracy              | CNN-based block partition decision |
| DVC [83]      | 2019 | RD                    | End-to-end optimized video compression |
| DVC [58]      | 2021 | RD                    | Contextual coding instead of residual coding |
| DVC [83]      | 2019 | RD                    | End-to-end optimized video compression |
| DVC [58]      | 2021 | RD                    | Contextual coding instead of residual coding |
| DVC [83]      | 2019 | RD                    | End-to-end optimized video compression |
| DVC [58]      | 2021 | RD                    | Contextual coding instead of residual coding |
| DVC [83]      | 2019 | RD                    | End-to-end optimized video compression |
| DVC [58]      | 2021 | RD                    | Contextual coding instead of residual coding |
| Remarks:      |      | Optimization objectives include: E (residual energy), R (rate), D (distortion), RD (rate-distortion), and Accuracy (accuracy for RD-optimized decision). NN: neural network; CNN: convolutional neural network; RNN: recurrent neural network.
2.1 Block-Based Hybrid Video Coding

A digital video consists of multiple frames. These frames may be compressed individually, each using an image coding scheme. Image coding schemes exploit the intraframe correlations for compression, but cannot exploit the interframe correlations. Thus, using image coding for multiple frames is also known as all-intra video coding. For example, JPEG is a famous image coding standard \[122\]; Motion JPEG just reuses JPEG for all-intra video coding. Since the frames in a natural video have been captured continuously, there is much duplicate content across different frames. If dividing a frame into small blocks, one block in one frame may be highly similar to another block in another frame. This is known as interframe data redundancy. How to remove such redundancy is a kernel problem in video coding.

One simple idea is to distinguish which content is duplicate and which is not. This idea was implemented by a conditional replenishment (CR) method in H.120, the first international video coding standard published by the International Telecommunication Union Telecommunication Standardization Sector (ITU-T) in 1984 \[46\]. The CR method consists of two modes: SKIP mode and INTRA mode. The SKIP mode indicates that the related area is duplicate (i.e., these data can be skipped), while the INTRA mode indicates that the related area is not duplicate and shall be coded with an image coding method. The CR method is indeed a multi-mode video coding solution, which sets up a prototype for the following block-based hybrid video coding schemes.

In natural videos, the duplicate content between different frames may be very similar but not exactly the same in pixel values. For such content, prediction plus residual coding is very efficient. For one block in one frame to be coded, we may find another block in another frame as a reference block, and use the reference block as a prediction; we subtract the prediction from the to-be-coded block and obtain the residual; we then code the residual, maybe using an image coding method such as 2D DCT plus coefficients quantization. Note that the reference block may have a displacement with respect to the to-be-coded block, so we need an MV to record the displacement. The method is named motion-compensated prediction (MCP)\(^2\). Then, one block is represented by one MV and a set of residual coefficients, showing a combination of the DCT-based and MV-based methods mentioned in Section 1.

H.261 published by ITU-T in 1990 \[47\] is the first video coding standard using the MCP method. In H.261, one frame is divided into \(16 \times 16\) non-overlapping blocks, each of which has one MV. The residual of a \(16 \times 16\) block is further divided into four \(8 \times 8\) subblocks, each of which uses 2D DCT to obtain residual coefficients. H.261 is the first successful block-based hybrid video coding standard, whose basic structure is inherited in the following standards till now. Following H.261, several international standards about video coding were published by International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) and ITU-T in the 1990s, including MPEG-1 \[45\], H.262/MPEG-2 \[49\], and H.263 \[48\]. These standards further refined the MCP method. Since the motion may not be aligned to discrete pixels, half-pixel MV is proposed. When using a half-pixel MV for MCP, the prediction shall be calculated by interpolating the imaginary pixels at half-pixel locations. Variable-block-size MCP is also proposed, where a \(16 \times 16\) block may have one MV for the entire block, or be split into four \(8 \times 8\) subblocks each having one MV. Moreover, bi-prediction is proposed, which allows one block to use two reference blocks and record two MVs respectively; to contrast, the previous MCP is termed uni-prediction. Bi-prediction is especially useful when the two reference blocks are before and after (in the

\(^2\)Interestingly, the MCP method is also applicable to image coding if one image is divided into blocks and the blocks are sequentially coded: later coded blocks may use earlier coded blocks as references \[72\]. This method is found to be especially useful for screen content images \[98\].
natural temporal order) the to-be-coded block respectively, which appears in the random-access configuration.

Along with the standard development, several difficult problems emerged. The first is motion estimation (ME), which tries to find out the best MV for a block to use the MCP method. Some heuristic ME algorithms enumerated possible MV candidates and compared them to find the best MV leading to the minimal residual energy (energy defined as sum-of-squares or sum-of-absolutes). The second is mode decision (MD), which tries to select the best mode for a block. Here candidate modes include different block sizes (if enabling variable-block-size MCP), uni-prediction/bi-prediction (if enabling the latter), and so on. Some heuristic MD algorithms tried possible modes and compared them to find the best mode leading to the minimal residual energy. The heuristic algorithms were quickly found problematic in both compression and computation. For compression, the heuristic algorithms limit the compression efficiency. For computation, more and more candidate MVs (e.g., enabling half-pixel MVs) or modes make the exhaustive search-based algorithms unaffordable.

To address the problems of ME/MD, several RD optimization methods were proposed in 1998 [96, 113]. These studies revealed that ME/MD trying to find out the optimal MV/mode is indeed solving RD optimization problems. Thus, ME/MD algorithms shall not rely on the residual energy, but instead consider the actual coding rate and the pixel reconstruction distortion. To optimize both rate and distortion, they suggested using the joint RD cost $D + \lambda R$ as the objective, where $D$ and $R$ are distortion and rate respectively, and $\lambda$ is the Lagrangian multiplier. An ideal RD-optimized ME/MD algorithm shall work as follows: for each candidate MV/mode, it calculates the corresponding rate (uses the MV/mode to perform encoding, then calculates the coding rate including all the symbols such as MV, mode, and residual coefficients) and distortion (uses the coded symbols to perform decoding, then calculates the distortion based on the reconstructed pixels); the best MV/mode is chosen to have the lowest joint RD cost.

The RD optimization methods [96, 113] had a big impact on block-based hybrid video coding schemes. Indeed, all the schemes developed after 2000 have incorporated more or less RD optimization methods. The reference software of H.264/AVC [129], H.265/HEVC [112], and H.266/VVC [12] all used RD optimization methods extensively. However, the ideal RD-optimized algorithm is never implemented in practice. First, practical ME algorithms do not calculate the true rate and distortion, because the calculation incurs unaffordable computational cost; instead, an approximate cost is calculated as $E + \sqrt{\lambda R_{MV}}$, where $E$ is the residual energy (defined as, e.g., sum-of-absolutes), $R_{MV}$ is the MV coding rate (not considering the residual coding rate), and $\sqrt{\lambda}$ is used as the weighting factor per empirical evidence [113]. Second, as there are more and more coding modes in H.264/AVC, H.265/HEVC, and H.266/VVC, enumerating all possible modes is virtually impossible. For example, H.265/HEVC introduced quadtree-based block partitioning, where a $64 \times 64$ block can be divided recursively to as small as $8 \times 8$ blocks. Actually, for a $64 \times 64$ block, there are $83,522$ different partition modes [25]. Moreover, some coding modes can be combined, such as quadtree-based block partition, uni-/bi-prediction, and integer-/half-/quarter-pixel MV. Accordingly, the mode decision is a combinative optimization problem, further increasing the number of candidate modes. Indeed, if we consider ME and MD jointly, this is also a combinative optimization that cannot be solved practically by exhaustive search-based algorithms.

Based on the above analyses, we observe that the block-based hybrid video coding schemes have some common characteristics from the optimization perspective. First, the optimization problem is discrete: the MV for MCP is either integer or of fixed precision (half-pixel, quarter-pixel, etc.); the modes are finite and enumerable; so the optimization problem is defined over a finite set of variables. Second, the optimization algorithm is search-based: the ME/MD algorithms enumerate the candidate MVs/modes to find out the optimal one, where different algorithms differ in how to
define the optimality. Third, the optimization occasion is online, i.e., the optimization procedure is performed within the coding process: all the modes have been designed and integrated into the encoder/decoder; for a given video, specifically for a block to be coded, the optimal MV/mode is decided and the corresponding symbols are written into the codes.

Due to these characteristics, block-based hybrid video coding schemes have several limitations. Since the optimization problem is discrete, the optimal solution must be selected within the pre-defined candidates. Thus, the candidates should not miss the promising solutions. Indeed, designing more and more modes is to enlarge the candidate set in the hope of finding a better solution. However, a larger and larger candidate set incurs a heavier and heavier computational burden for the search-based optimization, especially because the optimization is performed online. In practice, the recent block-based hybrid schemes, especially the encoder that performs online search-based optimization, are more and more difficult to implement.

### 2.2 Block-Based Hybrid Video Coding with Online Numerical Optimization

To address the limitations of search-based optimization, numerical optimization is also considered in block-based hybrid video coding schemes [126]. Numerical optimization refers to effective computational methods for continuous optimization, where the optimization objective is formulated as a continuous function of the variables of interest [94]. For video coding, numerical optimization has been studied in two directions: replacing search-based coding tools and crafting new coding tools.

#### 2.2.1 Direction 1.

As noted in the previous subsection, block-based hybrid schemes rely on ME and MD for efficient MCP-based coding, where both ME and MD were approached via search-based optimization. For MD, since the multiple modes may have fewer correlations, it is quite challenging to use numerical optimization to find out the optimal mode. For ME, the case is quite different, as the MV theoretically has infinite precision. In 1996, a gradient descent-based algorithm was proposed for ME [79], where the optimization objective is casted as a continuous function of MV, and the gradient of the objective function with respect to the MV is used to guide the MV search. Similar algorithms were proposed later in [16, 24, 100]. Compared to search-based ME algorithms, gradient descent-based ME algorithms greatly reduce the number of candidates, leading to lower computational costs. However, gradient descent-based algorithms have their own limitations. First, they may fall into local optimums instead of finding out the global optimum, due to the gradient descent nature. Second, the optimization objective is not the RD cost, as both rate (including MV and residual) and distortion (usually due to residual coefficients quantization) are difficult to write in analytical functions of MV. Therefore, gradient descent-based algorithms did not exhibit prominent superiority over search-based ones for ME. In recent years, MCP-based methods have been greatly improved by taking into account affine motion models [141, 142]. The previous MCP methods used one MV for one block, assuming that the corresponding content had a translational motion. The translational motion model cannot describe more complex motions such as rotations. On the contrary, affine motion models can describe complex motions but require more parameters (usually four or six for one block). When using affine motion models for video coding, search-based ME algorithms incur too much computational complexity to afford. Thus, a gradient descent-based algorithm for affine ME was proposed in [62], which achieved significant compression efficiency improvement with slightly increasing encoder complexity. The affine motion model-based MCP was integrated into the H.266/VVC standard, as a milestone of non-translational motion models in video coding standards [12].

#### 2.2.2 Direction 2.

New coding tools were developed continuously, some of which were based on numerical optimization algorithms. Here we list some representative studies. (1) Adaptive pixel
prediction. A lossless image coding scheme was proposed in [140], where a linear model is built to predict one pixel value from its neighboring pixel values, and the model parameters are obtained by the least-squares approach. Similar studies were reported in [64, 130]. (2) Local illumination compensation. The idea is to refine the MCP signal by considering the local illumination of the to-be-coded block. In one method [73], a linear model is built for the refinement, and the model parameters are estimated by the least-squares approach. (3) Cross-component prediction. One block has different color components such as RGB or YUV (also known as luma and chroma). Prediction of chroma components from luma components is useful to reduce the coding rate of chroma components. In one method known as CCLM [146], a linear model is built between luma and chroma components, and the model parameters are also estimated by the least-squares approach. (4) Adaptive interpolation filter. This is used for sub-pixel (half-pixel, quarter-pixel, etc.) MCP, where the prediction signal shall be interpolated. The Wiener filter has been studied for sub-pixel interpolation in [121, 127]. (5) Adaptive in-loop filter (ALF). It is proposed to filter the reconstructed pixel values to reduce the distortion by Wiener filters, where the filter coefficients may be estimated from the reconstructed and original pixel values and written as symbols [119]. One ALF was adopted into the H.266/VVC standard [52]. (6) Bi-directional optical flow (BDOF). This is proposed to refine the bi-prediction signal by calculating the optical flow gradient [5].

2.2.3 Summary. Numerical optimization methods have been studied in the block-based hybrid video coding framework. In these methods, the optimization problem is continuous: the optimization objective is a continuous function of the variables of interest; the optimization algorithm is numerical, such as least-squares, Wiener filter, and gradient descent; the optimization occasion is still online: the optimization is performed when encoding/decoding a specific video. These methods enrich the coding toolbox for block-based hybrid schemes, while the online optimization may cause computational costs and implementation difficulties.

2.3 Block-Based Hybrid Video Coding with Deep Tools

Deep learning [56] has revolutionized many aspects of video technologies, including video coding. Deep learning-based video coding consists of two approaches: deep tools and deep schemes [70]. Interestingly, both approaches appeared in 2015, in [23] and [118], respectively. This subsection is devoted to deep tools, and the next subsections will address deep schemes.

Deep tools refer to deep neural network-based computing modules that are used together with other coding tools, usually inside block-based hybrid video coding schemes. Some tools that were developed during the H.266/VVC standardization period are introduced in [69]. A more comprehensive review of deep tools may be found in [70]. Here, we divide the existing deep tools into two categories and provide a brief overview: the first category aims to improve compression efficiency, and the second category aims to reduce computational complexity.

2.3.1 Compression Efficiency Improvement Tools. A lot of deep tools have been studied for higher compression efficiency, either replacing their counterparts (non-deep tools) or adding new modules in block-based hybrid schemes.

Inspired by the success of deep learning for image restoration, Dong et al. proposed an image filtering tool to reduce the artifacts (due to lossy compression) in the reconstructed images, namely artifact-reduction convolutional neural network (ARCNN) [23]. ARCNN adopts a deep network of several convolutional layers. The network parameters were trained by a gradient descent-based algorithm with a lot of samples, each sample consisting of a reconstructed image and its corresponding original image. ARCNN does not change the encoding process. Once an image is reconstructed, ARCNN performs an inference using the trained network to filter the reconstructed image. Thus, ARCNN does not change the coding rate but may reduce the distortion,
leading to compression efficiency improvement. Indeed, the network parameters were trained to minimize the distortion, i.e., the difference between network output and original images, for the training samples.

It may be inspiring to compare ARCNN [23] with ALF [52]. Both perform filtering on reconstructed pictures to reduce the distortion. Both adopt numerical optimization algorithms (gradient descent for ARCNN, Wiener filter for ALF). The optimization in ARCNN (training network parameters) is performed on a lot of training samples; the network parameters are determined before encoding/decoding an image. The optimization in ALF (obtaining filter coefficients), on the contrary, is performed on the picture to be coded; the filter coefficients are obtained at the encoder side and shall be written into the codes, otherwise the decoder cannot perform the same filtering process. To contrast, we say that ARCNN performs offline optimization (because the optimization procedure is before the coding process) and ALF performs online optimization.

Based on the same idea, some studies try to replace the online numerical optimization-based tools with offline optimized tools. Corresponding to the tools discussed in Section 2.2 (Direction 2), here we have a series of deep tools: (1) Pixel prediction, using trained networks to predict some pixel values from their neighboring pixel values, e.g., [61]. (2) MCP refinement, using trained networks to refine the MCP signal by considering the neighboring pixel values, e.g., [44]. (3) Cross-component prediction, using trained networks to predict chroma components from luma components, e.g., [66]. (4) CNN-based interpolation filter, used to perform sub-pixel interpolation for MCP, e.g., [134]. (5) CNN-based in-loop filter, following the idea of ARCNN and designing more and more sophisticated networks, e.g., [29, 50, 75]. (6) CNN-based bi-prediction, using trained networks to predict a picture from the two pictures before and after it, e.g., [150].

Some other coding tools in the block-based hybrid schemes also found their deep substitutes. DCT, as a fundamental image coding tool, has been replaced by neural network-based transforms in [71, 135]. Neural networks also enhanced wavelet transforms in [87]. Another series of deep tools address entropy coding, that is to convert symbols to binary codes, where the problem is to accurately estimate the probability distributions of the symbols. In [84, 148], trained neural networks are used for the probability estimation. In resampling-based coding, the down-sampling and up-sampling were traditionally performed with simple linear filters, now are accomplished by trained CNNs in [65].

Moreover, deep learning inspires several new coding tools. In [43], a CNN is trained to generate a picture from multiple preceding pictures, which is known as frame extrapolation. The generated picture serves for MCP, enhancing the capability of interframe prediction. In [78], the MCP signal is used during the inverse quantization process of the residual coefficients. The inverse quantization is performed by trained neural networks.

All these deep tools can be easily integrated into block-based hybrid video coding schemes, since these schemes allow using very different methods as candidate coding modes, as mentioned in Section 2.1. In fact, many deep tools were tested on top of H.265/HEVC or H.266/VVC, where they were added as new candidate modes and tested in the online MD. Considering this nature, some studies designed multiple submodes under a deep tool. In [99], a pixel prediction method known as matrix-based intra prediction (MIP) is proposed, where multiple networks are offline trained and one network is online chosen for each to-be-coded block to perform intraframe prediction3.

2.3.2 Computational Complexity Reduction Tools. The second category of deep tools addresses the computational complexity, especially the encoding complexity, of block-based hybrid schemes. As noted in Section 2.1, more and more coding modes are added into the block-based hybrid schemes, leading to a significant increase in complexity. One solution is to adopt deep learning to replace some of these operations. For example, in H.264, some operations are performed in the frequency domain, which is computationally expensive. If we replace these operations with deep learning, we can significantly reduce the computational complexity.

3In this study, the networks have been simplified to having only one layer, so the network inference is equivalent to matrix multiplication, giving the name MIP.

ACM Comput. Surv., Vol. 56, No. 9, Article 225. Publication date: April 2024.
schemes. Online searching for the optimal mode is computationally expensive. Considering that deep tools are optimized offline, it may be beneficial to design a deep tool for making mode decisions to help or even replace the online search. This idea leads to a series of studies. In [80], trained neural networks are used for fast deciding among the intra prediction modes in H.266/VVC. In [111], trained neural networks are used for fast deciding among the MCP modes in H.265/HEVC. Moreover, a lot of studies have been devoted to block partition deciding in H.265/HEVC and H.266/VVC. As noted before, H.265/HEVC introduces quadtree-based block partition [112], allowing too many partition modes to be exhaustively searched. H.266/VVC further introduces multi-type-tree-based block partition for even larger blocks [12], increasing the number of partition modes by several orders of magnitude. Thus, how to decide block partitions for the highest possible compression efficiency with a reasonable computational cost is very crucial. Recent studies can be found in [26, 53, 81, 132].

2.3.3 Summary. From the above analyses, we observe that deep tools indeed provide a new optimization direction for block-based hybrid video coding schemes, that is to introduce offline optimization. The deep networks are trained offline, before the encoding/decoding process. As a common practice in deep learning, the network training is formulated as a continuous optimization problem, which is usually solved by numerical methods, such as gradient descent. After training, the networks can be used in different coding modes, or used for mode decision, both of which are compatible with the block-based hybrid framework.

One limitation of deep tools is the optimization objective. A deep tool usually performs a part of the computations of the entire encoding/decoding process, while the other computations are performed by the other modules. Then, it is difficult to only train the deep tool to optimize the RD cost. In practice, different deep tools have their own objectives respectively. In-loop filters try to minimize distortion only. Entropy coding tools try to minimize the rate only. Prediction tools try to minimize the residual energy. Mode decision tools try to maximize the accuracy of deciding the optimal mode, with optimality in the RD sense. A lot of studies have tried to design surrogate loss functions that align better with the RD cost for the offline optimization, e.g., [65].

2.4 End-to-End Learned Video Coding

The deep tools reviewed in the previous subsection have demonstrated the potential of using trained deep networks for video coding. Then, it is a natural question to ask whether it is possible to build a video coding scheme using merely deep networks. The answer is yes. Video coding schemes built entirely on deep networks are termed deep schemes in [70]. In this paper, we use another term, end-to-end learned video coding, to highlight its nature in optimization methods rather than in computing models.

End-to-end learned video coding schemes are usually constructed in the following steps. First, an entire deep neural network consisting of both encoder and decoder is designed, with a set of trainable parameters. Second, an optimization objective function is formulated, commonly using the RD cost. Third, the network is trained with a set of samples to minimize the objective function over the samples, commonly using gradient descent-based optimization algorithms. Fourth, an inference of the trained network accomplishes the encoding and decoding of a video.

Compared with deep tools, one obvious advantage of end-to-end learned schemes is that the entire encoder/decoder, including every computing module, is end-to-end optimized. Accordingly, the optimization objective is the true RD cost.

4Due to some technical difficulties, such as dealing with quantization (because it is not differentiable), end-to-end optimization of the true RD is still challenging. Nonetheless, the optimization objective of training the end-to-end learned schemes is much closer to the true RD than that of training the deep tools.
2.4.1 End-to-End Learned Image Coding. Image coding that exploits intraframe correlations is the basis of video coding. Thus, end-to-end learned schemes were first built for image coding. The end-to-end learned schemes all follow the basic idea of autoencoders [37]. In 2015, an end-to-end learned image coding scheme was proposed in [118]. This scheme uses recurrent neural networks as the computational models for encoding/decoding. In the scheme, pixels are converted to latents, and the latents are binary quantized to codes. No entropy coding is applied on the quantized latents, so the coding rate is fully determined by the network structure. Then, the quantized latents are converted to reconstruct pixels. The distortion between original and reconstructed pixels is set as the objective to train the networks.

To optimize not only distortion but also rate, an end-to-end learned image coding scheme that optimizes the joint RD cost was proposed in [6]. In this scheme, the network converting pixels to latents is termed analysis transform, and the network converting latents back to pixels is termed synthesis transform. Indeed, analysis transform plus synthesis transform is an autoencoder. The latents are quantized before being converted to codes. There is another network estimating the probability distributions of the quantized latents, known as the entropy model. Having the probability distributions given by the trained entropy model, the quantized latents can be efficiently coded by, e.g., arithmetic coding. This scheme was followed by a lot of studies, and was improved in different aspects.

- Analysis and synthesis transforms. A CNN with generalized divisive normalization (GDN) operators is used as the analysis transform, and correspondingly, a CNN with inverse GDN operators is used as the synthesis transform, in [6]. Cheng et al. proposed to integrate the attention mechanism into the transforms [18]. Chen et al. introduced a non-local attention module to enlarge the receptive field of the network [17]. Different attention mechanisms are also investigated in [30, 76, 115, 155]. Recent studies adopt transformer-like networks for the transforms [153, 155]. Liu et al. combined transformer and CNN for the transforms [76]. Reversible networks, which incur no information loss, are also considered for the transforms. Inspired by the wavelet transforms, Ma et al. built wavelet-like transforms using CNNs [88]. Helminger et al. used normalizing flows that are also reversible [36]. Xie et al. [131] combined reversible and irreversible networks to enhance the transform ability. Moreover, some studies attempted to build data-dependent transforms [110, 123]. Conditional networks were used to establish variable rate transforms in [20, 21, 57, 110].

- Quantization. For the entire network to be jointly optimized using a numerical algorithm, one technical challenge is to deal with the quantization as it is not differentiable. In [6], adding uniform noise is proposed to replace the hard quantization during training, while the hard quantization (rounding) is still used during encoding/decoding. This causes a mismatch and may worsen the compression efficiency. Several studies have proposed different solutions for this issue, such as stochastic rounding [117], universal quantization [20], soft-then-hard [31], and others [3, 114, 139]. In addition to the uniform scalar quantization (such as rounding), some studies proposed using non-uniform quantization [13, 90, 151] or vector quantization [1, 27, 147, 152].

- Entropy model. Using the entropy model to code the quantized latents was observed effective in reducing the coding rate [6]. The entropy model in [6] is known as the factorized prior model. Later on, the hyperprior model is proposed, which adopts a second transform to convert latents into hyper-latents, codes the hyper-latents, and uses the hyper-latents to estimate the probability distributions of the latents [7]. In this study, the latent probability distributions are assumed to be Gaussian. Gaussian mixtures were then considered in [18]. Multi-layer hyperprior model, which uses the hyperprior idea multiple times, was proposed
in [39]. Minnen et al. further combined the hyperprior model with an autoregressive context model, which is to use the spatially neighboring latents to help estimate the probability distribution of one latent [92]. He et al. proposed a checkerboard context model, which is much more friendly to parallel computing than the autoregressive context model [35]. Minnen and Singh also used the autoregressive idea but applied it for different channels instead of spatially neighboring latents [93]. Advanced networks, such as the transformer, were also used for the entropy model, e.g., [55, 101]. More studies of the entropy model may be found in [17, 30, 34, 63].

A number of end-to-end learned image coding schemes have surpassed H.266/VVC in terms of compression efficiency [34]. Witnessing this evidence, end-to-end learned image coding standards are now developed by ISO/IEC Joint Photographic Experts Group (JPEG) and IEEE Audio Video Coding Working Group (1857WG), respectively.

Furthermore, the idea of end-to-end learned image coding has been applied to the block-based hybrid schemes. Ma et al. proposed to train an artifact reduction network that accepts some coded information (termed side information) as input. The distortion (of the network output image) and the rate (of the side information) are jointly optimized [86]. The trained artifact reduction network is used as a deep tool. Yang et al. proposed to use trained end-to-end networks to allocate the rate between different blocks in a block-based hybrid scheme [136].

### 2.4.2 End-to-End Learned Interframe Coding

An end-to-end learned video coding scheme that exploits not only intraframe but also interframe correlations was proposed in 2019 [83]. The scheme, known as Deep Video Compression (DVC), is an analogy of block-based hybrid schemes but does not split a frame into blocks and does not have different coding modes other than MCP. Indeed, DVC has four parts for interframe coding, all of which are built on deep networks: ME module, MC module, motion information encoding/decoding and entropy model, residual encoding/decoding and entropy model. All the modules are jointly optimized for the joint RD cost, where the rate includes motion information rate and residual rate. Following DVC, Lin et al. proposed multi-frame ME and MC, which uses multiple frames to further reduce the interframe redundancy [68]. Hu et al. adopted deformable convolutions for ME and MC in the feature domain rather than the pixel domain [42].

Another end-to-end learned interframe coding scheme was proposed in 2021 [58]. The scheme, known as Deep Contextual Video Compression (DCVC), differs from DVC in how to use the MCP. In DVC, the MCP signal is subtracted from the to-be-coded frame, resulting in the residual to be coded. In DCVC, the MCP signal (or its features) is used as a context that is injected into the encoding/decoding modules and the entropy model for the to-be-coded frame. In other words, DCVC suggests a paradigm shift from residual coding to contextual coding. Following DCVC, Sheng et al. proposed to learn multi-scale features as the context [106]. Li et al. proposed to improve the contextual entropy model [59]. Li et al. further proposed to use diverse contexts by, e.g., adjusting the quality of different frames [60]. The last scheme [60] is reported to achieve higher compression efficiency than H.266/VVC under certain test conditions.

More end-to-end learned interframe coding schemes have been proposed in recent years than before. Advanced network structures, such as transformers have been used in [91], and normalizing flows have been used in [38]. Bi-prediction has been studied in [15, 137, 143]. Rippel et al. proposed to jointly compress the motion information and residual [103]. Shi et al. introduced conditional intra frames [107]. Rippel et al. proposed a flexible rate scheme that uses one trained network to cover a large range of rates [102]. Other representative schemes can be found in [2, 22, 32, 74, 77, 138].
2.4.3 Summary. Based on the above analyses, the end-to-end learned video coding schemes are quite different from the block-based hybrid video coding schemes from the optimization perspective. First, the optimization problem is continuous: the optimization objective is defined as a continuous (and even differentiable) function of the network parameters to be optimized. Second, the optimization algorithm is numerical: usually, gradient descent or the like algorithms are adopted to train the network parameters. Third, the optimization occasion is offline: the network training happens before the encoding/decoding process.

End-to-end learned schemes have their limitations, too. First, as the optimization objective is assumed to be a continuous function, it is difficult to integrate non-continuous computations, like switching between multiple modes, into the networks to be optimized. Second, numerical algorithms are easily trapped into local optimums. Third, the offline optimization is performed on training samples, so the trained networks are dependent on the training data and may not generalize well to any to-be-coded video.

2.5 End-to-End Learned Video Coding with Online Optimization

To address the limitations of offline optimization, online optimization is also considered for end-to-end learned video coding schemes, including both image coding and interframe coding. As noted before, online optimization may be either search-based or numerical.

**Image coding.** Campos et al. proposed to perform online numerical optimization for learned image coding [14]. Specifically, given an image to compress, the latents are optimized to minimize the RD cost, where the decoding network and the entropy model are kept unchanged. Zhao et al. proposed another latent optimization strategy, which takes the quantization effect into consideration [149]. Yang et al. proposed stochastic annealing to address the quantization in online optimization [139]. Wang et al. also performed online numerical optimization, but not optimizing the latents [124]. Instead, the method finds a substitutional image as the input to the encoding network that leads to the minimal RD cost. Pan et al. proposed to optimize not only the latents but also the decoding network parameters, and thus, to write the updated decoding network parameters as well as the latents into the codes [97]. Zhang et al. investigated a similar method [144]. Some studies investigated online optimization for variable rate coding [28, 51, 110]. Moreover, the online optimization effect is considered in the offline optimization procedure, which is formulated in the meta-learning paradigm [154]. For online search-based optimization, Wang et al. investigated using multiple deep network-based models as an ensemble, and choosing one model for each image to compress [125]. Brand et al. proposed RDONet, which chooses the depth of the encoding network for each image to compress [10].

**Interframe coding.** Lu et al. adopt online numerical optimization upon the offline trained DVC models in [82]. Rozendaal et al. optimized not only the latents but also some decoding network parameters, requiring the coding of latents as well as the video-adaptive network parameters [120]. Lin et al. online optimized the rate for motion information coding [67]. Xu et al. interpreted online optimization as to allocate the rate among multiple frames [133]. Hu et al. proposed an online search-based optimization method, which is to search from several kinds of predefined resolutions for the motion information [40]. They further proposed variable block sizes for the motion information, and used the hyperprior to predict the block size [41].

In summary, online optimization has been studied for end-to-end learned video coding, and achieves certain compression efficiency improvements. As a drawback, online optimization inevitably increases computational complexity. There are some interesting problems to study further, such as the collaboration of offline optimization and online optimization, selecting some variables to be online optimized, and so on.
3 THEORETICAL ANALYSES

In this section, we first revisit the RD optimization problem for video coding, and discuss the difference between offline optimization and online optimization. To solve the optimization problem, we focus on the comparison of search-based optimization and numerical optimization, and then propose the idea of hybrid optimization.

3.1 Optimization Problem Formulation

According to Shannon’s information theory, video coding belongs to lossy source coding [104, 105]. Its compression efficiency has a bound, characterized by the RD function of the source. Here, we start from Shannon’s theory and derive the RD problem formulation with some reasonable assumptions.

Given an information source $X$, its RD function is defined by

$$R(D) = \min_{P(\hat{X}|X)} I(X; \hat{X}), \text{ subject to } \mathbb{E}_{X, \hat{X}}[d(X, \hat{X})] \leq D$$

where $P(\hat{X}|X)$ is a conditional probability distribution, $I(X; \hat{X})$ is the mutual information between $X$ and $\hat{X}$, $\mathbb{E}$ is to take mathematical expectation, $d(X, \hat{X})$ measures the difference between $X$ and $\hat{X}$. The entire RD function can be interpreted as the minimal quantity of mutual information between $X$ and $\hat{X}$, when the distortion (i.e., the expected difference between $X$ and $\hat{X}$) should not exceed $D$. The RD function is the bound of lossy source coding because the coding rate cannot be lower than $R(D_0)$ if the incurred distortion is equal to $D_0$.

We can build a mapping between Equation (1) and a video coding scheme that is abstractly depicted in Figure 1. $X$ corresponds to the original video, and $\hat{X}$ corresponds to the reconstructed video. In Eq. (1), $P(\hat{X}|X)$ is a conditional probability distribution. In a video coding scheme, normally we do not introduce any randomness into the encoder/decoder. Thus, the relation between $X$ and $\hat{X}$ can be formulated as a function $f$. With this in mind,

$$P(\hat{X}|X) = \begin{cases} 1, & \text{if } \hat{X} = f(X) \\ 0, & \text{otherwise} \end{cases}$$

Optimization over functions is usually substituted by optimization over parameters. To this end, we need to parameterize the function:

$$f(X) \triangleq f(X|\Theta), \Theta \in \Omega$$

where $\Theta$ is the parameter, and $\Omega$ is the parameter space consisting of all possible parameters of interest.

---

5 Some studies have introduced randomness into encoding/decoding [85, 116], which are out of the scope of this paper.
As noted in Section 1, a video coding scheme usually converts pixels to symbols for encoding, and converts symbols back to pixels for decoding. This is depicted in Figure 1, where $Z$ refers to the symbols. Here we do not consider the entropy coding module in practical schemes, which converts symbols to binary codes. In the following, $Z$ is termed **semantics**. According to Figure 1,

$$f(X|\Theta) = f_{\text{dec}}(Z|\Theta_{\text{dec}}) = f_{\text{dec}}(f_{\text{enc}}(X|\Theta_{\text{enc}})|\Theta_{\text{dec}})$$

(4)

where

$$\Theta = \{\Theta_{\text{enc}}, \Theta_{\text{dec}}\}$$

(5)

Now, the optimization problem in Equation (1) is recast as

$$\min_{\Theta \in \Omega} I(X; \hat{X}), \text{ subject to } \mathbb{E}_{X, \hat{X}}[d(X, \hat{X})] \leq D$$

(6)

Note that

$$I(X; \hat{X}) = H(\hat{X}) - H(\hat{X}|X)$$

(7)

where $H(\hat{X})$ is the entropy of $\hat{X}$, $H(\hat{X}|X)$ is the conditional entropy of $\hat{X}$ given $X$. The conditional entropy is defined as

$$H(\hat{X}|X) = \sum_{X} \sum_{\hat{X}} -P(X)P(\hat{X}|X) \log P(\hat{X}|X)$$

(8)

Due to Equation (2), $P(\hat{X}|X)$ is 0 or 1, so $P(\hat{X}|X) \log P(\hat{X}|X)$ is always 0, then $H(\hat{X}|X) = 0$, and

$$I(X; \hat{X}) = H(\hat{X})$$

(9)

Note that $\hat{X} = f_{\text{dec}}(Z|\Theta_{\text{dec}})$. According to the information theory, we have

$$H(\hat{X}) \leq H(Z)$$

(10)

In practical coding schemes, we use entropy coding methods to convert $Z$ to binary codes. One entropy coding method defines a set of rules for the conversion. Theoretically, each entropy coding method corresponds to a presumed probability distribution $B(Z)$, and the coding rate (i.e., the expected code length) is

$$R(Z) = \mathbb{E}_Z[-\log B(Z)] = \sum_Z -P(Z) \log B(Z)$$

(11)

Note that

$$H(Z) = \sum_Z -P(Z) \log P(Z) \leq \sum_Z -P(Z) \log B(Z)$$

(12)

Thus, the optimization problem is changed from Equation (6) to

$$\min_{\Theta \in \Omega} R(Z), \text{ subject to } \mathbb{E}_{X, \hat{X}}[d(X, \hat{X})] \leq D$$

(13)

Till now, $X$ is assumed to be an information source whose probability distribution is $P(X)$. However, when we use $X$ to denote videos, the probability distribution of videos is difficult to explicitly characterize. Practically, we have many videos as samples of the implicit distribution. Based on how to define the samples, we have offline optimization and online optimization, respectively.

---

6Here, the word “semantics” does not refer to the meaning of a word as understood by humans. Instead, it refers to the various kinds of symbols that have different meanings, for example, symbols for coding modes, symbols for MVs, symbols for quantized residual coefficients, symbols for quantized latents, and so on.
3.1.1 Offline Optimization. Prior to the encoding/decoding process, we may perform offline optimization using a lot of samples. These samples are assumed to be independently and identically sampled (i.i.d.) from $X$. We use the following denotations. Assuming there are $K$ samples, each sample (video) is denoted by $x_k$ where $k = 1, 2, \ldots, K$. For each sample, the encoder produces the corresponding semantics $z_k$, and the decoder produces the corresponding reconstructed sample $\hat{x}_k$:

$$z_k = f_{\text{enc}}(x_k|\Theta_{\text{enc}}); \hat{x}_k = f_{\text{dec}}(z_k|\Theta_{\text{dec}})$$  \hspace{1cm} (14)

The rate of each sample is $-\log B(z_k)$, and the distortion of each sample is denoted by $\Delta(x_k, \hat{x}_k)$. Then, the optimization problem becomes

$$\min_{\Theta \in \Omega} \frac{1}{K} \sum_{k=1}^{K} -\log B(z_k), \text{ subject to } \frac{1}{K} \sum_{k=1}^{K} \Delta(x_k, \hat{x}_k) \leq D$$  \hspace{1cm} (15)

3.1.2 Online Optimization. During the encoding/decoding process, we may perform online optimization specifically for the sample to be coded. Here, the to-be-coded sample is assumed to be the only sample that we have from $X$. The sample (video) is denoted by $x$. The corresponding semantics is $z$. The reconstructed sample is $\hat{x}$. The rate of this sample is $-\log B(z)$, and the distortion is $\Delta(x, \hat{x})$. Then, the optimization problem becomes

$$\min_{\Theta \in \Omega} -\log B(z), \text{ subject to } \Delta(x, \hat{x}) \leq D$$  \hspace{1cm} (16)

In practice, online optimization is further constrained. The online optimization is specific for the sample to be coded, and the sample is available at the encoder side but not at the decoder side. Thus, practical online optimization can change only the encoding function/parameter $f_{\text{enc}}(\cdot|\Theta_{\text{enc}})$ but not the decoding function/parameter $f_{\text{dec}}(\cdot|\Theta_{\text{dec}})$.

Thus, the practical online optimization problem is

$$\min_{\Theta_{\text{enc}} \in \Omega_{\text{enc}}} -\log B(z), \text{ subject to } \Delta(x, \hat{x}) \leq D$$  \hspace{1cm} (17)

where $\Omega_{\text{enc}}$ is the encoding parameter space.

3.1.3 Summary. We observe that offline optimization and online optimization have two important differences. First, offline optimization minimizes the average RD cost over a set of samples, while online optimization minimizes the RD cost of a specific sample. Thus, online optimization is more adaptive to the sample to be coded. Second, offline optimization can optimize both encoder and decoder, while online optimization can optimize only encoder. Thus, offline optimization has more degrees of freedom. These differences imply that offline optimization and online optimization both have pros and cons. It is then necessary to combine their advantages.

3.2 Optimization Problem Solutions

For either offline optimization (15) or online optimization (17), the optimization problem is a constrained one. Constrained optimization problems are usually recasted as unconstrained ones before

---

7One may be confused here. For end-to-end learned video coding, one online optimization method may optimize not only the latents but also the decoding network parameters, such as in [97]. So why do we claim that online optimization cannot change the decoding parameter? We need to clarify that there is no conflict. Our defined decoding parameter $\Theta_{\text{dec}}$ is not equivalent to the decoding network parameters in [97] and similar methods. Our formulation shown in Figure 1 assumes that the decoding function $f_{\text{dec}}$ processes the entire semantics, and nothing else. In [97] and similar methods, the semantics include not only latents but also decoding network parameters. When interpreting these methods with our formulation, $f_{\text{dec}}$ consists of updating the decoding network parameters and then performing inference with the latents. This $f_{\text{dec}}$ does not change for different videos.
Fig. 2. Conceptual illustration of three optimization solutions. The horizontal axis denotes the parameter space. The vertical axis denotes the optimization objective, which is to be minimized. The red triangle indicates the global optimum. (a) Discrete, search-based optimization, which chooses several parameters (shown as the black/blue dots) and compares their corresponding objective function values. (b) Continuous, numerical optimization, e.g., gradient descent-based optimization, starts from a chosen initial parameter (shown as the black dot) and finds out a local optimum (shown as the blue dot). (c) Hybrid optimization, e.g., also gradient descent-based, starts from several different initial parameters (shown as the black points) and finds out several local optima (respectively (shown as the blue dots and the red triangle)) and compares the local optima. Hybrid optimization combines the concepts of discrete and continuous optimization.

being solved [94]. Specifically, Equation (15) is replaced with

$$
\min_{\Theta \in \Omega} \frac{1}{K} \sum_{k=1}^{K} - \log B(z_k) + \lambda \frac{1}{K} \sum_{k=1}^{K} \Delta(x_k, \hat{x}_k) \tag{18}
$$

Equation (17) is replaced with

$$
\min_{\Theta_{enc} \in \Omega_{enc}} - \log B(z) + \lambda \Delta(x, \hat{x}) \tag{19}
$$

where $\lambda$ is the Lagrangian multiplier. In this subsection, we use $J$ to denote the optimization objective (known as the joint RD cost) in either one of the above two problems. We use $\Theta$ to denote the parameter in either problem. As a conceptual illustration, Figure 2 presents three kinds of optimization solutions.

3.2.1 Search-Based Solution. The idea of search-based optimization is to enumerate possible solutions and compare these solutions to find the optimal one among them. As shown in Figure 2(a), we can choose $N$ different parameters, each of which is denoted by $\Theta_i$, where $i = 1, 2, \ldots, N$. We then calculate $J_i = J(\Theta_i)$, and find out the minimum from the $J_i$’s. The optimal parameter $\Theta^*$ is obtained as

$$
\Theta^* = \arg \min_i J(\Theta_i) \tag{20}
$$

Search-based optimization has no requirement on the objective function except that the function value can be evaluated at each chosen parameter. If the searched parameters are sufficient, the true optimal parameter (that leads to the global optimum of the objective function) has a good chance of being found. Otherwise, if the searched parameters are not sufficient, the global optimum may be missing, as shown in Figure 2(a).

3.2.2 Numerical Solution. There are a lot of numerical optimization methods [94]. We first discuss gradient descent-based methods as they have been widely adopted in video coding [70]. The idea of gradient descent is to iteratively refine a solution using the gradients of the objective function. As shown in Figure 2(b), we can set an initial parameter $\Theta_1$ and calculate $\Theta_i$, $i = 2, 3, \ldots$ in
\[ \Theta_i = \Theta_{i-1} - \eta \frac{\partial f}{\partial \Theta}(\Theta_{i-1}) \]  

(21)

where \( \eta \) is the learning rate. The iteration is terminated if the difference between \( \Theta_i \) and \( \Theta_{i-1} \) is negligible. And the optimal parameter \( \Theta^* \) equals the last \( \Theta_i \).

Gradient-based optimization requires that the objective function is differentiable (implying continuous) at each of the encountered parameters during the iteration. This is a strong assumption. Indeed, due to this assumption, some technical challenges were encountered in practical video coding schemes, such as how to deal with quantization in learned video coding [1, 6, 31]. Moreover, gradient descent usually finds a local optimum that is possibly not the global optimum, as shown in Figure 2(b).

Besides gradient descent, there are other numerical optimization methods. Some methods require the second-order gradients of the objective function, which incurs very high computational complexity. To date, the most popular numerical optimization methods, especially for deep learning, are still gradient descent and its variants.

### 3.2.3 Hybrid Solution

We now compare search-based optimization and numerical optimization. When the objective function is differentiable, we observe that numerical optimization is more efficient for searching in a local area of the parameter space. However, it is easily trapped into local optimums. Therefore, we may combine search-based optimization and numerical optimization to leverage both advantages.

Here we describe a possible solution. As shown in Figure 2(c), we can choose \( N \) different initial parameters, each of which is denoted by \( \Theta_i \), where \( i = 1, 2, \ldots, N \). Around each \( \Theta_i \), we perform gradient descent to obtain a locally optimal parameter \( \Theta^*_i \). At last, the optimal parameter \( \Theta^* \) is obtained as

\[ \Theta^* = \arg \min_i J(\Theta^*_i) \]  

(22)

Compared to gradient descent-based methods, the hybrid optimization method also requires that the objective function is differentiable. As it performs multiple gradient descent processes, its computational complexity is higher (but note that the multiple gradient descents can be calculated in parallel). Meanwhile, it has a better chance to find out the global optimum. And the chance is better if the number of searched initial parameters, \( N \), is larger.

Compared to search-based methods, the hybrid optimization method may require much less searched parameters, since the gradient guidance helps exclude a lot of less promising variables in each local area around \( \Theta_i \). Thus, the computational complexity is probably lower.

In summary, hybrid optimization may strike a better balance between the chance of reaching the global optimum and the computational complexity.

### 3.2.4 Different Optimization Problems and Solutions

Before ending this section, we would like to discuss the relationship between offline/online optimization and search-based/numerical/hybrid optimization.

Conceptually, they are orthogonal: offline/online optimization deals with different optimization problems, and search-based/numerical/hybrid optimization provides different optimization algorithms. Indeed, numerical optimization has been used for both offline optimization (e.g., training deep tools, see Section 2.3) and online optimization (e.g., performing ME, see Section 2.2). Search-based optimization has also been used for online optimization (e.g., mode decision, see Section 2.1). However, search-based optimization is seldom used for offline optimization in video coding, which may be attributed to the very high computational cost. Based on this evidence, we conjecture that hybrid optimization may be more used for online optimization than for offline optimization.
Towards Hybrid-Optimization Video Coding

Fig. 3. Our proposed hybrid-optimization video coding scheme. (a) The scheme includes intraframe coding and interframe coding, where modules are colored to show the adopted optimization strategies. (b) The scheme shows more details of the encoder for interframe coding.

It is more interesting to consider the collaboration between offline optimization and online optimization from the optimization algorithm perspective:

— Suppose that we use a search-based algorithm for online optimization. We know that search-based algorithms need a set of candidate parameters to be compared. The number of candidate parameters directly determines the online computational cost. Thus, we may perform offline optimization to decide these candidate parameters or help to decide. For example, for MD in block-based hybrid schemes, we perform offline optimization and train a neural network. The trained network may decide for each block an adaptive subset of modes to be online searched. In this way, the compression efficiency may be lowered because of missing some better modes, but the computational cost may be decreased significantly. See [26] for a concrete method.

— Then we consider using a gradient descent-based algorithm for online optimization. For gradient descent, the initial parameter is very important. Choosing a good initial parameter not only enhances the quality of the final solution but also accelerates the iterative process. Thus, we may perform offline optimization to decide the initial parameters or help to decide. For example, for online optimization in end-to-end learned schemes, we perform an inference on the offline trained network to obtain a set of latents. These latents are used as the initial value for online gradient descent-based optimization. See [14] for a concrete method.

— Finally, let us consider using the aforementioned hybrid algorithm for online optimization. In the hybrid algorithms, the number of the initial parameters and the location of each initial parameter are all crucial to the performance. We may perform offline optimization to decide or help to decide them. In the next section, we will present a concrete scheme to verify this idea.

Note that offline optimization that helps online optimization may be formulated in the meta-learning paradigm, as mentioned in [154].

4 PROPOSED SCHEME

Inspired by the theoretical analyses, we propose a hybrid-optimization video coding scheme. In this section, we first introduce the scheme design, and then focus on the offline optimization and online optimization procedures in our scheme. Figure 3(a) depicts the scheme including both intra(frame) coding and inter(frame) coding. Figure 3(b) displays more details of the encoder for interframe coding.
4.1 Scheme Design

Our scheme encodes/decodes a video frame by frame, but the frame coding order is allowed to be different from the natural temporal order. When encoding/decoding a frame, the previously coded frames may be (partially) used as reference frames to reduce the interframe data redundancy. We focus on interframe coding as it is much more important than intraframe coding to improve compression efficiency for natural videos. For simplicity, we reuse H.265/HEVC to compress intra frames.

For interframe coding, we follow the most successful paradigm in video coding, i.e., motion-compensated prediction (MCP). MCP is used not only in block-based hybrid schemes [12, 112] but also in end-to-end learned schemes [83]. Accordingly, our scheme has four important modules: motion estimation (ME) or specifically optical flow estimation, motion compensation (MC), residual transform and inverse transform, and entropy coding. An in-loop filter is added to the scheme for distortion reduction. Witnessing the success of deep tools (Section 2.3), the ME, MC, transform and inverse transform, and in-loop filter modules are all based on deep networks; the probability distribution estimation is also based on deep networks for the entropy coding. Thus, the decoder of our scheme (consisting of ME, MC, inverse transform, in-loop filter, and entropy decoding) is purely built on deep networks.

To facilitate hybrid optimization, we introduce multiple coding modes, some of which have parameters to be decided online. Indeed, in the online optimization procedure, the multiple coding modes correspond to the multiple initial parameters in Figure 2(c). If one coding mode has its own parameters, these parameters are online optimized by numerical algorithms. After all the modes are optimized, they are compared in terms of the joint RD cost, and the best mode is selected. We observe that the online optimization indeed follows the hybrid optimization idea mentioned in Section 3.2.

We consider three categories of coding modes, inspired by the practices in block-based hybrid schemes [112]. These modes are depicted as switches in Figure 3(b).

— **Block partition mode.** A block may be coded as one unit, or split into smaller subblocks each being one unit. Following H.265/HEVC, we use quadtree-based block partition, allowing a $64 \times 64$ block to be divided recursively to as small as $8 \times 8$ blocks.

— **Motion mode.** The motion information may be represented in different manners. Currently, we have three motion modes, known as Temporal Merge Mode, Temporal Scale Mode, and Motion Vector Mode. Both Temporal Merge Mode and Temporal Scale Mode require to predict a per-pixel optical flow at the decoder side. The basic idea of optical flow prediction is to use multiple reference frames. For example, if the frame to be coded is the $t$-th frame, and the decoder already obtains the reconstructed $(t-1)$-th and $(t+1)$-th frames. The optical flow from $(t-1)$ frame to $(t+1)$ frame can be estimated by an optical flow network. The estimated optical flow is scaled by 0.5 to be the predicted optical flow from $(t-1)$ frame to $t$ frame, with the uniform-speed motion assumption. In Temporal Merge Mode, the predicted optical flow is directly used for MC at the decoder side; there is no additional bit to signal any motion information except for the mode itself. In Temporal Scale Mode, there are some coefficients to further scale up/down the predicted optical flow, implying non-uniform-speed motion; for each block, there are four scaling coefficients (forward and backward references, horizontal and vertical directions); these scaling coefficients are optimized at the encoder side and written into the codes. Motion Vector Mode does not use the prediction optical flow; instead, it is to assign an MV for a block; the MV is optimized at the encoder side and written into the codes. For both Temporal Scale Mode and Motion Vector Mode, online optimization is addressed by gradient descent-based algorithms.
Towards Hybrid-Optimization Video Coding

— **Residual mode.** A block may have residual, which will be transformed and the resulting coefficients will be coded, or not (named Residual Skip).

One may notice that the number of our designed modes is much less than that in the state-of-the-art block-based hybrid schemes like H.265/HEVC and H.266/VVC. Indeed, H.265/HEVC has much more modes than our scheme. Except for quadtree-based block partition, H.265/HEVC also has **prediction unit (PU)** partitioning for interframe coding. Moreover, H.265/HEVC has many motion- or residual-related modes, such as SKIP, MERGE, AMVP, and so on [112]. H.266/VVC has much more block partition modes as well as more motion- or residual-related modes [12]. As mentioned before, search-based optimization has higher computational complexity if using more modes, so we have intentionally limited the number of modes in our scheme. We hope that the hybrid optimization may relieve the need for too many modes to be searched.

### 4.2 Offline Optimization

For the offline optimization in our scheme, we follow the practices of training deep networks in end-to-end learned schemes (Section 2.4). Since the offline optimization deals with massive training data, search-based algorithms or hybrid algorithms may incur very high computational complexity. Thus, we use only numerical algorithms in the offline optimization.

Our entire scheme has not only deep networks but also multiple coding modes, where the latter is not friendly to numerical algorithms. To solve this problem, we remove all the discrete components (the switches in Figure 3(b)) before training. Specifically, the deep networks being trained constitute a complete end-to-end interframe coding scheme: the ME module produces a per-pixel optical flow; the MC module obtains the MCP signal using the optical flow; one autoencoder equipped with entropy model compresses the residual; another autoencoder equipped with entropy model compresses the optical flow; the in-loop filter takes the reconstructed residual plus the MCP signal as input and obtains the reconstructed frame. All these networks are jointly trained to minimize the RD cost on the entire training dataset.

After training the networks, we augment our scheme with the designed coding modes. As noted in Section 3.2, we also prepare for the online optimization. Our online optimization is hybrid, so we need to define the number of initial parameters as well as the location of each initial parameter (Figure 2(c)). They correspond to the number of modes and the initial parameter (to be online numerically optimized) for each mode, respectively. Currently, the number of modes is predefined in our scheme. Several modes do not have parameters to online numerically optimize. So we consider only the modes that allow online numerical parameter optimization.

Specifically, we consider **Temporal Scale Mode and Motion Vector Mode.** For Temporal Scale Mode, the parameters for online numerical optimization are the scaling coefficients. For Motion Vector Mode, the parameters for online numerical optimization are the MVs. We design appropriate initialization algorithms for these parameters respectively, using the offline trained networks as guidance. For Temporal Scale Mode, the scaling coefficients are initialized by the optical flows (calculated by the offline trained ME module), e.g., the scaling coefficient for \((t - 1)\) frame is set to the ratio of the optical flow from \((t - 1)\) to \(t\) over the optical flow from \((t - 1)\) to \((t + 1)\). For Motion Vector Mode, the MV of one block is initialized as the mean of the optical flow vectors inside the block.

### 4.3 Online Optimization

The online optimization in our scheme follows the hybrid optimization idea (Section 3.2), i.e., online numerical optimization followed by online search-based optimization.

Currently, the online numerical optimization works on only two modes: **Temporal Scale Mode and Motion Vector Mode.** For either mode, there are some parameters that allow numerical
optimization. We need to define the optimization objective for them. In online optimization, a natural objective is the RD cost of the to-be-coded frame. We relate the parameters to per-pixel optical flow: For Temporal Scale Mode, the parameters are used to scale a predicted optical flow (the predicted optical flow itself is regarded as constant). For Motion Vector Mode, the parameters are one MV per block, so the MV is assigned as the optical flow to every pixel in the block. We use the per-pixel optical flow together with the trained networks (MC, residual compression, in-loop filter) to calculate the RD cost. The RD cost with respect to the parameters is then a differentiable function, because the networks are all differentiable. Given the optimization objective, we properly initialize the parameters and update them iteratively by gradient descent-based algorithms.

Next, we perform online search-based optimization. Note that we have three kinds of modes. Their combinative optimization is computationally unaffordable. Thus, we perform a stage-wise search. **Stage 1: motion mode decision.** The frame to be coded is divided into $64 \times 64$ blocks. For each $64 \times 64$ block, the three motion modes are compared to decide the best one. Then similarly, $32 \times 32$ blocks, $16 \times 16$ blocks, and $8 \times 8$ blocks are processed in turn. **Stage 2: block partition decision.** Since quadtree-based block partition allows too many choices [25], we use a heuristic bottom-up algorithm for deciding block partition. The frame to be coded is divided into $8 \times 8$ blocks. For each $8 \times 8$ block, its RD cost is set to the minimum by comparing different motion modes in the first stage. Then the frame to be coded is divided into $16 \times 16$ blocks. For each $16 \times 16$ block, if it is split, $RD_s$ is the sum of the RD costs of the corresponding four $8 \times 8$ blocks, which are obtained in the previous step; if it is not split, $RD_{ns}$ is set to the minimum by comparing different motion modes in the first stage. $RD_s$ and $RD_{ns}$ are compared to decide whether to split or not. After the decision, the RD cost of this $16 \times 16$ block is set to the minimum of $RD_s$ and $RD_{ns}$. Then similarly, $32 \times 32$ blocks and $64 \times 64$ blocks are processed in turn. **Stage 3: residual mode decision.** After the former two stages, the block partition as well as the motion mode of each block have been decided, so the residual is determined, too. The entire-frame residual is redivided into blocks (can be different from the previous block partition to allow flexibility) - renamed residual units to distinguish - and each residual unit may choose its residual mode. After the three stages, the online search-based optimization is finished.

In summary, our proposed scheme performs hybrid optimization, including both search-based and numerical optimization, in the online stage. We adopt offline optimization to assist the online hybrid optimization. Moreover, our scheme is designed for the random-access configuration. These characteristics are rarely found in the existing methods mentioned in Section 2.5.

5 EXPERIMENTS

5.1 Experimental Settings

5.1.1 Coding Configuration. There are three commonly used coding configurations, known as all-intra, low-delay, and random-access. Our experiments concentrate on the random-access configuration due to the following reasons. First, our scheme focuses on interframe coding, so we omit the all-intra configuration. Second, we are studying video coding for compression efficiency, and random-access is believed to have the highest compression efficiency among the three configurations [95]. Third, it seems that end-to-end learned schemes performed less competitively in the random-access configuration than in the low-delay configuration, so we want to improve upon the end-to-end learned schemes.

To test our scheme, we follow the common test conditions defined for the random-access configuration in the H.265/HEVC reference software known as HM (version 16.10)\(^8\). In our scheme,
only the first frame in each intra period is intraframe coded, and the remaining frames are organized into multiple groups of pictures (GOPs). Each GOP consists of 8 consecutive frames. The frames within one GOP (1st to 8th) are divided into 4 temporal layers (L0 to L3), and coded in the following order: L0 (8th), L1 (4th), L2 (2nd, 6th), L3 (1st, 3rd, 5th, 7th). Each frame can use up to two reference frames, but the reference frames must be at the previous layer, or in the previous GOP, or intra frames. For example, the 6th can use the 4th and 8th as references, the 3rd can use the 2nd and 4th as references, and so on. This coding configuration is especially suitable for bi-prediction.

5.1.2 Network Training. The common test conditions of HM specify four quantization parameters (QPs): 22, 27, 32, and 37. Corresponding to each of the four QPs, we train a set of networks. Note that different QPs relate to different $\lambda$ values in the optimization objective (Equation 15 or 17). We use mean-squared error as the distortion metric in the optimization objective. Thus, we can directly use the $\lambda$ values calculated in HM for our optimization. We use the SJTU 4K dataset [109] as well as its downsampled versions to prepare training data. Only the luma component is used for training data. For each QP, we use HM to compress the original videos, then extract the original frames and reconstructed frames according to our defined coding configuration. For example, we may extract the 6th frame from an original video, and the 4th and 8th frames from the corresponding HM reconstructed video, and use them to form a tuple. We further crop the tuples, in the spatial dimensions, into non-overlapping $128 \times 128$, 3-frame clips. In total, we produce about 500,000 clips as the training data.

As for implementation, the network training uses PyTorch and is performed on 4 NVIDIA GTX 1080Ti GPUs. The quantization is approximated using the method described in [117]. The Adam optimizer [54], with hyperparameters 0.9 and 0.999 and no weight decay, is used. The batch size is 16. The learning rate is $10^{-4}$ at the beginning and then reduced to $10^{-5}$.

5.1.3 Other Settings. Our networks are trained with only the luma component. For video sequences normally in the YUV420 format, our scheme encodes Y, U, and V separately and using the same network, without considering cross-component correlations. The online numerical optimization is applied only on L1 frames (i.e., 4th frame of each GOP). The number of online iterations is set to 10. Both settings aim to reduce the online computational cost.

5.1.4 Test Conditions. We try our best to follow the HM test conditions. For example, the intra period is set to 32, which is the default setting in HM (version 16.10). Some changes in test conditions are noted below. First, we use 16 test sequences with different resolutions known as Classes B, C, D, and E [9]. Second, for each sequence, we test the frames of the first intra period. Third, in addition to using the raw sequences in the YUV420 format, we convert the test sequences into the YUV400 format and test again. The results of YUV400 better reflect the real performance of our scheme. These changes are applied to HM as well when we compare our scheme with HM.

5.2 Overall Performance

5.2.1 Comparison with H.265/HM. We experimentally compare our scheme and HM using the same test conditions. Here we use PSNR to measure the quality of the reconstructed frames and calculate BD-rate [8] to quantify the compression efficiency of our scheme, taking HM as the anchor.

Figure 4 presents the average PSNR values of all the tested frames, showing the quality variation within each GOP (lower temporal layer frames have higher quality) as well as between intra frames and inter frames (intra frames have higher quality). It is observed that our scheme is well aligned to H.265/HM in terms of the frame quality levels. Thus, the BD-rate results are reliable.
Fig. 4. Average PSNR values of the frames with respect to different POCs (POC indicates the relative position within a GOP, 0 stands for intra frame) obtained by H.265/HM and our proposed scheme. Our scheme is well aligned with H.265/HM in terms of the frame quality levels.

Table 3. BD-Rate Results of our Proposed Scheme Compared to H.265/HM (YUV400)

| Class | Sequence       | Y (%) |
|-------|----------------|-------|
| Class B |                |       |
|       | Kimono         | −2.4  |
|       | ParkScene      | −4.4  |
|       | Cactus         | −4.1  |
|       | BasketballDrive| 2.5   |
|       | BQTerrace      | 43.4  |
| Class C |                |       |
|       | BasketballDrill| 2.6   |
|       | BQMall         | 4.9   |
|       | PartyScene     | 0.4   |
|       | RaceHorsesC    | 16.6  |
| Class D |                |       |
|       | BasketballPass | −6.4  |
|       | BQSquare       | −14.5 |
|       | BlowingBubbles | −0.8  |
|       | RaceHorses     | 3.9   |
| Class E |                |       |
|       | FourPeople     | −4.6  |
|       | Johnny         | 2.9   |
|       | KristenAndSara | 1.2   |
| Class Summary |    |       |
| Class B |                | 7.0   |
| Class C |                | 6.1   |
| Class D |                | −4.4  |
| Class E |                | −0.2  |
| Overall | Classes B–E   | 2.6   |

Table 3 presents the results on the YUV400 format sequences, which show that our scheme has an average 2.6% BD-rate loss compared to HM. Table 4 presents the results on the normal YUV420 format sequences, which show that our scheme has an average 3.7%, 6.3%, 4.7% BD-rate losses compared to HM for Y, U, V, respectively. The losses on YUV420 are larger than on YUV400 because our scheme did not exploit the cross-component correlations, but H.265/HM had exploited. Based on the YUV400 results, it is evident that our scheme performs on par with H.265/HM.

Taking a per-sequence view, our scheme has a significant gain on BQTerrace, but significant losses on BQTerrace and RaceHorsesC. For the other sequences, the difference between our
Table 4. BD-rate Results of our Proposed Scheme Compared to H.265/HM (YUV420)

| Class   | Sequence     | Y (%) | U (%) | V (%) |
|---------|--------------|-------|-------|-------|
| Class B | Kimono       | 0.9   | 0.4   | 1.0   |
|         | ParkScene    | -3.5  | 1.3   | 2.0   |
|         | Cactus       | -2.9  | -2.5  | -0.7  |
|         | BasketballDrive | 4.3  | 12.0  | 5.1   |
|         | BQTerrace    | 44.9  | 10.5  | 11.2  |
| Class C | BasketballDrill | 4.2  | 12.9  | 9.0   |
|         | BQMall       | 6.2   | 13.5  | 11.1  |
|         | PartyScene   | 0.6   | 3.4   | 3.7   |
|         | RaceHorsesC  | 17.3  | 28.1  | 24.6  |
| Class D | BasketballPass | -4.4 | 3.2   | -4.0  |
|         | BQSquare     | -13.6 | -2.9  | -6.2  |
|         | BlowingBubbles | -0.5 | 3.6   | 5.5   |
|         | RaceHorses   | 6.8   | 21.2  | 14.2  |
| Class E | FourPeople   | -4.4  | -2.8  | -2.9  |
|         | Johnny       | 2.7   | 0.2   | 2.2   |
|         | KristenAndSara | 1.3  | -0.6  | -0.5  |
| Class Summary | Class B | 8.7   | 4.3   | 3.7   |
|         | Class C      | 7.1   | 14.5  | 12.1  |
|         | Class D      | -2.9  | 6.3   | 2.4   |
|         | Class E      | -0.1  | -1.1  | -0.4  |
| Overall | Classes B–E  | 3.7   | 6.3   | 4.7   |

scheme and HM is less than 10%. For BQTerrace and RaceHorsesC, we observed that the poor performance of our scheme was mostly due to inaccurate optical flows. Indeed, BQTerrace contains a large area of thin textures (e.g., wall and railing) and irregular textures (e.g., water ripples). RaceHorsesC contains irregular textures (e.g., grass) and many disocclusions (due to the moving horses). These are difficult cases for optical flow estimation networks. Our scheme based on optical flow estimation thus performs worse. On the other hand, BQSquare has small motions (the camera is moving slowly), which can be accurately captured by the optical flow estimation networks. It is worth noting that optical flow estimation networks are offline numerically optimized, while the ME algorithms in traditional block-based hybrid schemes are online search-based optimized. The inefficiency of optical flow estimation networks indicates some shortcomings of offline numerical optimization, such as lack of robustness and being trapped into local optimums.

5.2.2 Comparison with End-To-End Learned Schemes. Plenty of end-to-end learned schemes have been proposed in recent years. A low-delay coding scheme was reported to outperform H.266/VVC on certain test conditions [60]. Another random-access coding scheme was reported to perform on par with H.265/HEVC [15]. It is worth noting that a lot of comparisons in the previous studies did not obey the same test conditions. For example, end-to-end learned schemes usually take RGB format videos as input and output and exploit the correlations among R, G, and B components implicitly. Our scheme did not address the cross-component correlations at all. Despite these differences, we here provide a set of comparison results to contrast our scheme with some recent end-to-end learned schemes. Note that our scheme and HM process YUV420 format videos, and we convert the original videos and the reconstructed videos into the RGB format and then calculate PSNR in this subsection.
First, we compare with [15], which represents the best compression efficiency of end-to-end schemes for the random-access configuration. On the five test sequences (HEVC Class B), the B-CANF method has a 10.6% BD-rate loss than H.265/HM, where the distortion is measured by RGB PSNR, as reported in the Fig. 5 of [15]. On the same sequences and following the same test condition, our scheme has a 9.3% BD-rate loss than H.265/HM. Thus, our scheme performs on par with B-CANF.

Moreover, we compare with some earlier end-to-end schemes. We use 13 test sequences including Classes B, C, and D. For each sequence, we test the first 100 frames. Some earlier end-to-end schemes used too short intra period (IP), like 10, which is not common practically. To make the comparison informative, we also use short IPs, like 8 and 16 (need to be a multiple of GOP length), for our scheme and HM. Note that our scheme and HM actually support much longer IPs as in the previous results. Figure 5 presents the results of different schemes, including x264 (an open-source implementation of H.264/AVC), x265 (an open-source implementation of H.265/HEVC), DVC [83], HLVC [137], FVC [42], HM, and our scheme. It can be observed that HM performs the best, and our scheme is slightly worse than HM. There is a large gap between our scheme and the others except HM.

5.3 Analyses of Proposed Hybrid Optimization

5.3.1 Benefit of Multiple Modes. To demonstrate the benefit of using multiple coding modes, i.e., using online search-based optimization in our scheme, we perform some ablation studies. In these studies, we use the sequences of Classes B, C, and D in the YUV400 format, and always use HM as the anchor to calculate BD-rate.

During the offline optimization of our scheme, we have built an end-to-end learned interframe coding scheme, as mentioned in Section 4.2. This scheme may be directly used for video coding without any further optimization. It is named ”E2E Flow,” because it requires coding the optical flows. Its performance, as shown in Table 5, is unsatisfactory. One may conjecture that the bi-prediction does not require so much optical flow information.

Our designed Temporal Merge Mode does not signal any motion information. Instead, the decoder side predicts an optical flow for MC, as mentioned in Section 4.1. Based on the offline trained networks, we try to apply only Temporal Merge Mode without any other mode (no block partition.
Table 5. Ablation Study Results Showing the BD-Rate of each Configuration Compared to H.265/HM (YUV400)

| Class | E2E Flow | E2E TMerge | + TScale | + MVM | + VBlock | + ResiSkip |
|-------|----------|------------|----------|-------|----------|------------|
| B     | 130.1    | 29.4       | 19.9     | 13.3  | 8.2      | 8.3        |
| C     | 66.5     | 21.6       | 15.6     | 10.0  | 7.0      | 6.8        |
| D     | 33.5     | 7.6        | 2.8      | −0.4  | −2.9     | −3.5       |
| Average | 80.8     | 20.3       | 13.3     | 8.0   | 4.4      | 4.2        |

(1) E2E Flow: After the end-to-end training, directly compressing the flow without any mode or online optimization. (2) E2E TMerge: After the end-to-end training, only use the proposed Temporal Merge Mode without block partitioning or online optimization. (3) + TScale: adding the proposed Temporal Scale Mode to (2), where block size is fixed to 32 × 32. (4) + MVM: adding the proposed Motion Vector Mode to (3), where block size is still fixed to 32 × 32. (5) + VBlock: enabling variable block sizes (from 8 × 8 to 64 × 64) on top of (4). (6) + ResiSkip: adding the proposed Residual Skip Mode to (5).

It is named “E2E TMerge” in Table 5. Its performance is much better than “E2E Flow,” which supports the conjecture that the bi-prediction can exploit the correlations between multiple reference frames to avoid coding much motion information.

Then, we add Temporal Scale Mode and Motion Vector Mode in turn. These two modes require writing some symbols at the block level, so we intentionally set the block size to 32 × 32. As shown in Table 5, adding Temporal Scale Mode reduces the average BD-rate loss by 7 percent; further adding Motion Vector Mode reduces the average BD-rate loss by another 5.3 percent.

Then, we turn on block partition modes using the proposed bottom-up search algorithm. So the block size is variable from 64 × 64 to 8 × 8. As shown in Table 5, variable block size reduces the average BD-rate loss by 3.6 percent.

Then, we enable residual modes. As shown in Table 5, the average BD-rate loss is further decreased from 4.4% to 4.2%. Although the improvement in BD-rate is small, Residual Skip Mode may also save decoding time in practice.

Till now, we have not enabled the online numerical optimization. These experimental results fully demonstrate the necessity of the online search-based optimization, and also demonstrate the effectiveness of the proposed coding modes, like Temporal Merge and Temporal Scale.

5.3.2 Mode Selection Results. In our scheme, multiple coding modes are being selected in the online optimization. Here, we analyze the mode selection results. Specifically, we calculate the selection ratio of each mode, defined as the ratio of the area of the blocks that select the mode over the entire area. Note that the ratio is calculated by area rather than by count because blocks have different sizes in our scheme.

The selection ratios of different block sizes are visualized in Figure 6. It can be observed that larger blocks are selected more than smaller blocks. Comparing different classes (having different resolutions), Class B (having higher resolution) uses more 64 × 64 blocks than the other two classes. This is attributed to the observation that high-resolution content usually has higher spatial correlations, while low-resolution content contains more details that require finer block partition. As QP increases, more 64 × 64 blocks are selected, while less 32 × 32, 16 × 16, 8 × 8 blocks are selected. Note that higher QP corresponds to lower rate, which prefers larger blocks to reduce the number of coded symbols.

The selection ratios of different motion modes and residual modes are visualized in Figure 7. In this figure, different classes do not exhibit obvious variation, which implies that the motion/residual modes are less dependent on video resolution. Among the three motion modes, Temporal Merge Mode is used the most, followed by Motion Vector Mode and Temporal Scale Mode. It is
already observed in Table 5 that Temporal Merge Mode is highly efficient for bi-prediction, so it is the most frequently selected. As QP increases, more blocks select Temporal Merge Mode, because this mode has no additional symbol to represent motion information, and thus is preferred at low rates. Residual Skip Mode is also frequently selected, especially at higher QP (lower rate), because it saves the rate for writing residual-related symbols.

In summary, the online search-based optimization could select suitable coding modes depending on the video content (e.g., resolution) and the required RD balance (reflected by QP). It exhibits strong adaptive capabilities.

5.3.3 Benefit of Online Numerical Optimization. Our scheme allows for online numerical optimization prior to the online search-based optimization. We have several restrictions on the online numerical optimization, such as using it for only two modes, for only L1 frames (4th frame of each GOP), and the iteration number is at most 10. Even with these restrictions, we observe that the online numerical optimization provides considerable coding gain. Table 6 presents the BD-rate results of turning on the online numerical optimization for the two modes respectively, where we calculate rate and PSNR for only L1 frames and for all frames, respectively. It can be observed that for all frames, the online numerical optimization achieves an average 1.0% BD-rate reduction. For only L1 frames, the average BD-rate reduction is 4.1%, and the highest reduction is 8.8% for Kimono. The two modes exhibit different benefits, e.g., Motion Vector Mode has more gain for Kimono while Temporal Scale Mode performs better for BQTerrace. The benefits of the two modes are largely accumulated when turning on both. These results demonstrate the necessity to perform online numerical optimization in our scheme. Combined with the aforementioned results, they confirm the benefit of the hybrid optimization.
Table 6. BD-Rate Results of Turning on the Proposed Online Numerical Optimization on L1 Frames for Temporal Scale Mode (TScale), Motion Vector Mode (MV), or Both

| Class  | Sequence      | Only L1 frames | All frames |
|--------|---------------|----------------|------------|
|        |               | TScale | MV | Both | Both |
| Class B| Kimono        | −0.7   | −8.6 | −8.8 | −1.9 |
|        | ParkScene     | −0.4   | −0.3 | −0.8 | −0.4 |
|        | Cactus        | −0.9   | −0.7 | −1.1 | −0.4 |
|        | BasketballDrive| −2.1   | −5.1 | −6.5 | −1.8 |
|        | BQTerrace     | −5.8   | −3.7 | −7.7 | −1.4 |
| Class C| BasketballDrill| −0.5   | −0.4 | −0.5 | −0.1 |
|        | BQMall        | −1.5   | −5.4 | −6.9 | −1.5 |
|        | PartyScene    | −1.9   | −1.0 | −2.5 | −0.5 |
|        | RaceHorsesC   | −0.2   | −3.0 | −3.1 | −0.5 |
| Class D| BasketballPass| −2.7   | −2.7 | −4.1 | −1.0 |
|        | BQSquare      | −2.3   | −1.3 | −2.8 | −0.8 |
|        | BlowingBubbles| −2.6   | −2.6 | −3.6 | −0.8 |
|        | RaceHorses    | −0.5   | −5.0 | −5.4 | −1.3 |
| Overall| Classes B–D   | −1.7   | −3.1 | −4.1 | −1.0 |

The anchor is turning off online numerical optimization (YUV400). Rate and distortion are calculated for only L1 frames and for all frames, respectively.

6 DISCUSSIONS

We have observed the performance of a concrete hybrid-optimization video coding scheme. Although the scheme does not set a new state-of-the-art, its compression efficiency is comparable to H.265/HEVC in the random-access configuration. We note that a concurrent study also reported similar compression efficiency, which uses much more sophisticated deep networks [15]. Considering the potential of hybrid-optimization schemes, we note the following two aspects.

To date, the most successful video coding schemes all belong to the block-based hybrid framework. Recently, the end-to-end learned framework has attracted more and more attention and is developing rapidly. Video coding experts have been studying advanced methods in one framework or in the other. However, the two frameworks are almost isolated, making it difficult to migrate the advances in one framework to the other. Moreover, some schemes belonging to one framework have tried to borrow ideas from the other framework, but such idea transfer is random rather than systematic. In other words, the improvements in either framework are difficult to combine, which limits the development of video coding technologies. Our suggested hybrid-optimization video coding may build a bridge between the two frameworks. On the one hand, more subtle coding modes in the block-based hybrid framework, such as advanced block partition, intraframe and interframe modes, and RD-optimized quantization, can be inserted into our scheme. On the other hand, more sophisticated deep networks in the end-to-end learned framework, such as optical flow estimation network, autoencoder, entropy model, in-loop filter, can be adopted in our scheme, too. Thus, our scheme has a better chance to absorb and combine more advanced technologies in both frameworks.

Video coding technologies rely on computations to improve compression efficiency. The computing power restricts the achievable performance. Recently, computing power has been regarded as a kind of infrastructure. Computing power centers have been built to combine and coordinate heterogeneous computing devices, such as CPU, GPU, TPU, and so on. Meanwhile, different
optimization algorithms prefer different kinds of computing power. Search-based optimization is more friendly to parallel computing, while numerical optimization often performs iterative computing. Switching between multiple modes is better supported in CPU, while training neural networks is better supported in TPU. If we stick to one optimization algorithm, we may be unable to take full advantage of a heterogeneous computing infrastructure. On the contrary, hybrid optimization that combines search-based and numerical algorithms may utilize the computing power better. Surely, it requires a lot of studies to distribute and schedule the computations.

7 CONCLUSION

We have reviewed the existing video coding technologies, especially two representative frameworks: block-based hybrid video coding and end-to-end learned video coding, from an optimization perspective. We have analyzed the optimization problems and the optimization solutions, which highlight the difference between offline and online optimization as well as the difference between search-based and numerical algorithms. Our analyses suggest that offline and online optimization should be combined, and it is beneficial to merge search-based algorithms and numerical algorithms into hybrid optimization algorithms. Guided by the theory, we have proposed a concrete hybrid-optimization video coding scheme, which consists of trained deep networks and multiple coding modes, and supports online hybrid optimization. We have experimentally demonstrated the compression efficiency of the proposed scheme, which is comparable to H.265/HEVC in the random-access configuration. Finally, we have analyzed the great potential of hybrid-optimization video coding schemes, on which we anticipate more theoretical and evidential studies.

8 ACKNOWLEDGMENT

The authors would like to thank the following collaborators: Cunhui Dong, Aolin Feng, Changsheng Gao, Jianping Lin, Kang Liu, Changyue Ma, Haichuan Ma, Xihua Sheng, Yefei Wang, Ning Yan, Kun Yang, Runyu Yang, Yan Yi, Ziqiu Zhang, Hengrui Zhao, and Yanchen Zuo.

REFERENCES

[1] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V. Gool. 2017. Soft-to-hard vector quantization for end-to-end learning compressible representations. In NIPS, Vol. 30. 1141–1151.
[2] Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Balle, Sung Jin Hwang, and George Toderici. 2020. Scale-space flow for end-to-end optimized video compression. In CVPR. 8503–8512.
[3] Eirikur Agustsson and Lucas Theis. 2020. Universally quantized neural compression. In NeurIPS, Vol. 33. 12367–12376.
[4] N. Ahmed, T. Natarajan, and K. R. Rao. 1974. Discrete cosine transform. IEEE Trans. Comput. C-23, 1 (1974), 90–93.
[5] Alexander Alshin, Elena Alshina, and Tammy Lee. 2010. Bi-directional optical flow for improving motion compensation. In PCS. IEEE, 422–425.
[6] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. 2016. End-to-end optimized image compression. arXiv preprint arXiv:1611.01704 (2016).
[7] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. 2018. Variational image compression with a scale hyperprior. arXiv preprint arXiv:1802.01436 (2018).
[8] Gisle Bjontegaard. 2001. Calculation of Average PSNR Differences between RD-Curves. Technical Report VCEG-M33.
[9] Frank Bossen. 2011. Common Test Conditions and Software Reference Configurations. Technical Report JCTVC-F900. JCT-VC.
[10] Fabian Brand, Kristian Fischer, and Andre Kaup. 2021. Rate-distortion optimized learning-based image compression using an adaptive hierarchical autoencoder with conditional hyperprior. In CVPR Workshops. 1885–1889.
[11] Benjamin Bross, Jianle Chen, Jens-Rainer Ohm, Gary J. Sullivan, and Ye-Kui Wang. 2021. Developments in international video coding standardization after AVC, with an overview of versatile video coding (VVC). Proc. IEEE 109, 9 (2021), 1463–1493.
Towards Hybrid-Optimization Video Coding

[12] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J. Sullivan, and Jens-Rainer Ohm. 2021. Overview of the versatile video coding (VVC) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 10 (2021), 3736–3764.

[13] Jianrui Cai and Lei Zhang. 2018. Deep image compression with iterative non-uniform quantization. In *ICIP*. IEEE, 451–455.

[14] Joaquim Campos, Meierhans Simon, Abdelaziz Djelouah, and Christopher Schroers. 2019. Content adaptive optimization for neural image compression. In *CVPR Workshops*. 1–5.

[15] Mu-Jung Chen, Yi-Hsin Chen, and Wen-Hsiao Peng. 2023. B-CANF: Adaptive B-frame coding with conditional augmented normalizing flows. *IEEE Transactions on Circuits and Systems for Video Technology* (2023). https://doi.org/10.1109/TCSVT.2023.3301016

[16] O.T.-C. Chen. 2000. Motion estimation using a one-dimensional gradient descent search. *IEEE Transactions on Circuits and Systems for Video Technology* 10, 4 (2000), 608–616.

[17] Tong Chen, Haojie Liu, Zhan Ma, Qiu Shen, Xun Cao, and Yao Wang. 2021. End-to-end learnt image compression via non-local attention optimization and improved context modeling. *IEEE Transactions on Image Processing* 30 (2021), 3179–3191.

[18] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. 2020. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *CVPR*. 7939–7948.

[19] Kiho Choi, Jianle Chen, Dmytro Rusanovskyy, Kwang-Pyo Choi, and Euee S. Jang. 2020. An overview of the MPEG-5 essential video coding standard [standards in a nutshell]. *IEEE Signal Processing Magazine* 37, 3 (2020), 160–167.

[21] Ze Cui, Jing Wang, Shangyin Gao, Tiansheng Guo, Yihui Feng, and Bo Bai. 2021. Asymmetric gained deep image compression with continuous rate adaptation. In *CVPR*. 10532–10541.

[22] Abdelaziz Djelouah, Joaquim Campos, Simone Schaub-Meyer, and Christopher Schroers. 2019. Neural inter-frame compression for video coding. In *ICCV*. 6421–6429.

[23] Dailan He, Ziming Yang, Weikun Peng, Rui Ma, Hongwei Qin, and James Bankoski. 2021. Checkerboard context model for efficient learned image compression. In *CVPR*. 14771–14780.

[24] Amirhossein Habibian, Ties van Rozendaal, Jakub M. Tomczak, and Taco S. Cohen. 2019. Video compression with rate-distortion autoencoders. In *ICML*. 3920–3929.

[25] Dailan He, Ziming Yang, WeiKun Peng, Rui Ma, Hongwei Qin, and Yan Wang. 2022. ELIC: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding. In *CVPR*. 5718–5727.

[26] Leonhard Helminger, Abdelaziz Djelouah, Markus Gross, and Christopher Schroers. 2020. Lossy image compression with normalizing flows. *arXiv preprint arXiv:2008.10486* (2020).

[27] Geoffrey Hinton and Ruslan Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313, 5786 (2006), 504–507.
[38] Yung-Han Ho, Chih-Peng Chang, Peng-Yu Chen, Alessandro Gnutti, and Wen-Hsiao Peng. 2022. CANF-VC: Conditional augmented normalizing flows for video compression. In ECCV. Springer, 207–223.

[39] Yueyu Hu, Wenhan Yang, Zhan Ma, and Jiaying Liu. 2022. Learning end-to-end lossy image compression: A benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 8 (2022), 4194–4211.

[40] Zhihao Hu, Zhenghao Chen, Dong Xu, Guo Lu, Wanli Ouyang, and Shuhang Gu. 2020. Improving deep video compression by resolution-adaptive flow coding. In *ECCV*. Springer, 193–209.

[41] Zhihao Hu, Guo Lu, Jinyang Guo, Shan Liu, Wei Jiang, and Dong Xu. 2022. Coarse-to-fine deep video coding with hyperprior-guided mode prediction. In *CVPR*. 5921–5930.

[42] Shuai Huo, Dong Liu, Bin Li, Siwei Ma, Feng Wu, and Wen Gao. 2021. Deep network-based frame extrapolation with reference frame alignment. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 3 (2021), 1178–1192.

[43] Zhihao Hu, Guo Lu, and Dort Xu. 2021. FVC: A new framework towards deep video compression in feature space. In *CVPR*. 1502–1511.

[44] Shuai Huo, Dong Liu, Bin Li, Siwei Ma, Feng Wu, and Wen Gao. 2021. Deep network-based frame extrapolation with reference frame alignment. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 3 (2021), 1178–1192.

[45] ISO/IEC. 1993. ISO/IEC 11172-2 (MPEG-I): Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1.5 Mbit/s - Part 2: Video.

[46] ITU-T. 1990. *ITU-T Recommendation H.261: Video Codec for Audiovisual Services at p × 64 kbit/s*.

[47] ITU-T. 1994. *ITU-T Recommendation H.262 - ISO/IEC 13818-2 (MPEG-2): Generic Coding of Moving Pictures and Associated Audio Information - Part 2: Video*.

[48] ITU-T. 1995. *ITU-T Recommendation H.263: Video Coding for Low Bitrate Communication*.

[49] ITU-T. 1984. *ITU-T Recommendation H.260: Codec for Videoconferencing Using Primary Digital Group Transmission*.

[50] ITU-T. 1990. *ITU-T Recommendation H.261: Video Codec for Audiovisual Services at p × 64 kbit/s*.

[51] ITU-T. 1995. *ITU-T Recommendation H.263: Video Coding for Low Bitrate Communication*.

[52] Shuai Huo, Dong Liu, Bin Li, Siwei Ma, Feng Wu, and Wen Gao. 2021. Deep network-based motion compensation refinement for video coding. In *ISCAS*. IEEE, 1–4.

[53] Chuanmin Jia, Shiqi Wang, Xinfeng Zhang, Shanshe Wang, Jiaying Liu, Shiliang Pu, and Siwei Ma. 2019. Content-aware convolutional neural network for in-loop filtering in high efficiency video coding. *IEEE Transactions on Image Processing* 28, 7 (2019), 3343–3356.

[54] Wei Jiang, Wei Wang, Songnan Li, and Shan Liu. 2022. Online meta adaptation for variable-rate learned image compression. In *CVPR*. 498–506.

[55] Jooyoung Lee, Seyoon Jeong, and Munchurl Kim. 2022. Selective compression learning of latent representations for variable-rate image compression. In *NeurIPS*. Vol. 35. 13146–13157.

[56] Chuanmin Jia, Shiqi Wang, Xinfeng Zhang, Shanshe Wang, Jiaying Liu, Shiliang Pu, and Siwei Ma. 2019. Content-aware convolutional neural network for in-loop filtering in high efficiency video coding. *IEEE Transactions on Image Processing* 28, 7 (2019), 3343–3356.

[57] Kyungah Kim and Won Woo Ro. 2018. Fast CU depth decision for HEVC using neural networks. *IEEE Transactions on Circuits and Systems for Video Technology* 29, 5 (2018), 1462–1473.

[58] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[59] Wei Jiang, Wei Wang, Songnan Li, and Shan Liu. 2022. Online meta adaptation for variable-rate learned image compression. In *CVPR*. 498–506.

[60] Marta Karczewicz, Nan Hu, Jonathan Taquet, Ching-Yeh Chen, Kiran Misra, Kenneth Andersson, Peng Yin, Taoran Lu, Edouard François, and Jie Chen. 2021. VVC in-loop filters. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 10 (2021), 3907–3925.

[61] Jooyoung Lee, Seyoon Jeong, and Munchurl Kim. 2022. Selective compression learning of latent representations for variable-rate image compression. In *NeurIPS*. Vol. 35. 13146–13157.

[62] Jooyoung Lee, Seyoon Jeong, and Munchurl Kim. 2022. Selective compression learning of latent representations for variable-rate image compression. In *NeurIPS*. Vol. 35. 13146–13157.

[63] Yung-Han Ho, Chih-Peng Chang, Peng-Yu Chen, Alessandro Gnutti, and Wen-Hsiao Peng. 2022. CANF-VC: Conditional augmented normalizing flows for video compression. In *ECCV*. Springer, 447–463.
Towards Hybrid-Optimization Video Coding 225:33

[67] Chih-Hsuan Lin, Yi-Hsin Chen, and Wen-Hsiao Peng. 2022. Content-adaptive motion rate adaption for learned video compression. In PCS. 163–167.

[68] Jianping Lin, Dong Liu, Houqiang Li, and Feng Wu. 2020. M-LVC: Multiple frames prediction for learned video compression. In CVPR. 3546–3554.

[69] Dong Liu, Zhenzhong Chen, Shan Liu, and Feng Wu. 2020. Deep learning-based technology in responses to the joint call for proposals on video compression with capability beyond HEVC. *IEEE Transactions on Circuits and Systems for Video Technology* 30, 5 (2020), 1267–1280.

[70] Dong Liu, Yue Li, Jianping Lin, Houqiang Li, and Feng Wu. 2020. Deep learning-based video coding: A review and a case study. *ACM Computing Surveys (CSUR)* 53, 1 (2020), 1–35.

[71] Dong Liu, Haichuan Ma, Zhiwei Xiong, and Feng Wu. 2018. CNN-based DCT-like transform for image compression. In MMM. Springer, 61–72.

[72] Dong Liu, Xiaoyan Sun, and Feng Wu. 2008. Manipulating image patches for compression. In ICME. 197–200.

[73] H. Liu, Y. Chen, J. Chen, L. Zhang, and M. Karczewicz. 2015. *Local Illumination Compensation*. Technical Report VCEG-AZ06. VCEG.

[74] Haojie Liu, Ming Lu, Zhan Ma, Fan Wang, Zhihuang Xie, Xun Cao, and Yao Wang. 2021. Neural video coding using multiscale motion compensation and spatiotemporal context model. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 8 (2021), 3182–3196.

[75] Jiaying Liu, Dong Liu, Wenhan Yang, SiFeng Xia, Xiaoshuai Zhang, and Yuanying Dai. 2020. A comprehensive benchmark for single image compression artifact reduction. *IEEE Transactions on Image Processing* 29 (2020), 7845–7860.

[76] Jinming Liu, Heming Sun, and Jiro Katto. 2023. Learned image compression with mixed transformer-CNN architectures. In CVPR. 14388–14397.

[77] Jerry Liu, Shenlong Wang, Wei-Chiu Ma, Meet Shah, Rui Hu, Pranaab Dhawan, and Raquel Urtasun. 2020. Conditional entropy coding for efficient video compression. In ECCV. Springer, 453–468.

[78] Kang Liu, Dong Liu, Li Li, and Houqiang Li. 2021. Context-adaptive inverse quantization for inter-frame coding. *IEEE Open Journal of Circuits and Systems* 2 (2021), 660–674.

[79] Lurong-Kuo Liu and Ephraim Feig. 1996. A block-based gradient descent search algorithm for block motion estimation in video coding. *IEEE Transactions on Circuits and Systems for Video Technology* 6, 4 (1996), 419–422.

[80] Zheng Liu, Tianyi Li, Ying Chen, Kajijn Wei, Mai Xu, and Honggang Qi. 2023. Deep multi-task learning based fast intra-mode decision for versatile video coding. *IEEE Transactions on Circuits and Systems for Video Technology* 33, 10 (2023), 6101–6116.

[81] Zhenyu Liu, Xianyu Yu, Yuan Gao, Shaolin Chen, Xiangyang Ji, and Dongsheng Wang. 2016. CU partition mode decision for HEVC hardwired intra encoder using convolution neural network. *IEEE Transactions on Image Processing* 25, 11 (2016), 5088–5103.

[82] Guo Lu, Chunlei Cai, Xiaoyun Zhang, Li Chen, Wanli Ouyang, Dong Xu, and Zhiyong Gao. 2020. Content adaptive and error propagation aware deep video compression. In ECCV. 456–472.

[83] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. 2019. DVC: An end-to-end deep video compression framework. In CVPR. 11006–11015.

[84] Haichuan Ma, Dong Liu, Cunhui Peng, Li Li, and Feng Wu. 2019. iWave: CNN-based wavelet-like transform for image compression. *IEEE Transactions on Multimedia* 22, 7 (2019), 1667–1679.

[85] Haichuan Ma, Dong Liu, Ning Yan, Houqiang Li, and Feng Wu. 2022. End-to-end optimized versatile image compression with wavelet-like transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 3 (2022), 1247–1263.

[86] Guido Meardi, Simone Ferrara, Lorenzo Ciccarelli, Guendalina Cobianchi, Stergios Poularakis, Florian Maurer, Stefano Battista, and Ahmad Byagowi. 2020. MPEG-5 Part 2: Low complexity enhancement video coding (LCEVC): Overview and performance evaluation. In *Applications of Digital Image Processing XLIII*, Vol. 11510. International Society for Optics and Photonics, 115101C.

[87] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. 2018. Conditional probability models for deep image compression. In CVPR. 4394–4402.

[88] Fabian Mentzer, George D. Toderici, David Minnen, Sergi Caelles, Sung Jin Hwang, Mario Lucic, and Eirikur Agustsson. 2022. VCT: A video compression transformer. In NeurIPS. Vol. 35. 13091–13103.
[92] David Minnen, Johannes Ballé, and George Toderici. 2018. Joint autoregressive and hierarchical priors for learned image compression. In NIPS, Vol. 31. 10794–10803.

[93] David Minnen and Saurabh Singh. 2020. Channel-wise autoregressive entropy models for learned image compression. In ICP. IEEE, 3339–3343.

[94] Jorge Nocedal and Stephen Wright. 2006. Numerical Optimization. Springer Science & Business Media.

[95] Jens-Rainer Ohm, Gary J. Sullivan, Heiko Schwarz, Thiow Keng Tan, and Thomas Wiegand. 2012. Comparison of the coding efficiency of video coding standards—including high efficiency video coding (HEVC). IEEE Transactions on Circuits and Systems for Video Technology 22, 12 (2012), 1669–1684.

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

[97] Guanbo Pan, Guo Lu, Zhihao Hu, and Dong Xu. 2022. Content adaptive latents and decoder for neural image compression. In ECCV. Springer, 556–573.

[98] Wen-Hsiao Peng, Frederick G. Walls, Robert A. Cohen, Jizheng Xu, Jörn Ostermann, Alexander Machniss, and Tao Lin. 2016. Overview of screen content video coding: Technologies, standards, and beyond. IEEE Journal on Emerging and Selected Topics in Circuits and Systems 6, 4 (2016), 393–408.

[99] J. Pfaff, P. Helle, D. Maniry, S. Kaltenstadler, W. Samek, H. Schwarz, D. Marpe, and T. Wiegand. 2018. Neural network based intra prediction for video coding. In Applications of Digital Image Processing XLI, Vol. 10752. International Society for Optics and Photonics, 1075213.

[100] Lai-Man Po, Ka-Ho Ng, Kwok-Wai Cheung, Ka-Man Wong, Yusuf Md. Salah Uddin, and Chi-Wang Ting. 2009. Novel directional gradient descent searches for fast block motion estimation. IEEE Transactions on Circuits and Systems for Video Technology 19, 8 (2009), 1189–1195.

[101] Yichen Qian, Ming Lin, Xiuyu Sun, Zhiyu Tan, and Rong Jin. 2022. Entroformer: A transformer-based entropy model for learned image compression. arXiv preprint arXiv:2202.05492 (2022).

[102] Oren Rippel, Alexander G. Anderson, Kedar Tatwawadi, Sanjay Nair, Craig Lytle, and Lubomir Bourdev. 2021. ELF-VC: Efficient learned flexible-rate video coding. In ICCV. 14479–14488.

[103] Oren Rippel, Sanjay Nair, Carissa Lew, Steve Branson, Alexander G. Anderson, and Lubomir Bourdev. 2019. Learned video compression. In ICCV. 3454–3463.

[104] C. E. Shannon. 1948. A mathematical theory of communication. Bell Systems Technical Journal 27, 4 (1948), 623–656.

[105] C. E. Shannon. 1959. Coding theorems for a discrete source with a fidelity criteria. International Convention Record 7 (1959), 325–350.

[106] Xihua Sheng, Jiahao Li, Bin Li, Li Li, Dong Liu, and Yan Lu. 2023. Temporal context mining for learned video compression. IEEE Transactions on Multimedia 25 (2023), 7311–7322.

[107] Yibo Shi, Yunying Ge, Jing Wang, and Jue Mao. 2022. AlphaVC: High-performance and efficient learned video compression. In ECCV. Springer, 616–631.

[108] Thomas Sikora. 2005. Trends and perspectives in image and video coding. Proc. IEEE 93, 1 (2005), 6–17.

[109] Li Song, Xun Tang, Wei Zhang, Xiaokang Yang, and Pingjian Xia. 2013. The SJTU 4K video sequence dataset. In QoMEX. 34–35.

[110] Myungsao Song, Jinyoung Choi, and Bohyung Han. 2021. Variable-rate deep image compression through spatially-adaptive feature transform. In ICCV. 2360–2369.

[111] Iago Storch, Luciano Agostini, Bruno Zatt, Sergio Bampi, and Daniel Palomino. 2022. FastInter360: A fast intermode decision for HEVC 360 video coding. IEEE Transactions on Circuits and Systems for Video Technology 32, 5 (2022), 3235–3249.

[112] Gary J. Sullivan, Jens Ohm, Woo-Jin Han, and Thomas Wiegand. 2012. Overview of the high efficiency video coding (HEVC) standard. IEEE Transactions on Circuits and Systems for Video Technology 22, 12 (2012), 1649–1668.

[113] Gary J. Sullivan and Thomas Wiegand. 1998. Rate-distortion optimization for video compression. IEEE Signal Processing Magazine 15, 6 (1998), 74–90.

[114] Heming Sun, Lu Yu, and Jiro Katto. 2022. Improving latent quantization of learned image compression with gradient scaling. In VCIP. 1–5.

[115] Zhisen Tang, Hanli Wang, Xiaokai Yi, Yun Zhang, Sam Kwong, and C.-C. Jay Kuo. 2022. Joint graph attention and asymmetric convolutional neural network for deep image compression. IEEE Transactions on Circuits and Systems for Video Technology 33, 1 (2022), 421–433.

[116] Lucas Theis and Eirikur Agustsson. 2021. On the advantages of stochastic encoders. arXiv preprint arXiv:2102.09270 (2021).

[117] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. 2017. Lossy image compression with compressive autoencoders. arXiv preprint arXiv:1703.00395 (2017).

[118] George Toderici, Sean M. O’Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar. 2015. Variable rate image compression with recurrent neural networks. arXiv preprint arXiv:1511.06085 (2015).
Towards Hybrid-Optimization Video Coding

[119] Chia-Yang Tsai, Ching-Yeh Chen, Tomoo Yamakage, In Suk Chong, Yu-Wen Huang, Chih-Ming Fu, Takayuki Itoh, Takashi Watanabe, Takeshi Chujoh, Marta Karczewicz, and Shaw-Min Lei. 2013. Adaptive loop filtering for video coding. IEEE Journal of Selected Topics in Signal Processing 7, 6 (2013), 934–945.

[120] Ties van Rozendaal, Johann Brehmer, Yunfan Zhang, Reza Pourreza, and Taco S. Cohen. 2021. Instance-Adaptive Video Compression: Improving Neural Codecs by Training on the Test Set. arXiv preprint arXiv:2111.10302 (2021).

[121] Yuri Vatis and Joern Ostermann. 2008. Adaptive interpolation filter for H.264/AVC. IEEE Transactions on Circuits and Systems for Video Technology 19, 2 (2008), 179–192.

[122] Gregory K. Wallace. 1992. The JPEG still picture compression standard. IEEE Transactions on Consumer Electronics 38, 1 (1992), xviii–xxxiv.

[123] Thomas Wedi. 2006. Adaptive interpolation filters and high-resolution displacements for video coding. IEEE Transactions on Circuits and Systems for Video Technology 16, 4 (2006), 484–491.

[124] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra. 2003. Overview of the H.264/AVC video coding standard. IEEE Transactions on Circuits and Systems for Video Technology 13, 7 (2003), 560–576.

[125] Xiao Wang, Ding Ding, Wei Jiang, Wei Wang, Xiaozhong Xu, Shan Liu, Brian Kulis, and Peter Chin. 2022. Substitutional neural image compression. In CVPR. 17379–17388.

[126] Yao Wang, Jörn Ostermann, and Ya-Qin Zhang. 2002. Video Processing and Communications. Vol. 1. Prentice Hall Upper Saddle River, NJ.

[127] Tongda Xu, Hua Ye, Guang Deng, and John C. Devlin. 1999. Least squares approach for lossless image coding. In International Symposium on Signal Processing and its Applications (ISSPA), Vol. 1. IEEE, 63–66.

[128] Hui Yuan, Jiande Sun, Hechao Liu, and Yujun Li. 2012. Affine model based motion compensation prediction for zoom. In CVPR. 6628–6637.

[129] M. Akın Yılmaz and A. Murat Tekalp. 2022. End-to-end rate-distortion optimized learned hierarchical bi-directional video compression. IEEE Transactions on Image Processing 31 (2022), 974–983.

[130] Honglei Zhang, Francesco Cristi, Hamed Rezazadegan Tavakoli, Maria Santamaria, Yat-Hong Lam, and Miska M. Hannuksela. 2021. Learn to overfit better: Finding the important parameters for learned image compression. In VCIP. IEEE, 1–5.
[145] Jiaqi Zhang, Chuanmin Jia, Meng Lei, Shanshe Wang, Siwei Ma, and Wen Gao. 2019. Recent development of AVS video coding standard: AVS3. In PCS. IEEE, 1–5.

[146] Kai Zhang, Jianle Chen, Li Zhang, Xiang Li, and Marta Karczewicz. 2018. Enhanced cross-component linear model for chroma intra-prediction in video coding. IEEE Transactions on Image Processing 27, 8 (2018), 3983–3997.

[147] Xi Zhang and Xiaolin Wu. 2023. LVQAC: Lattice vector quantization coupled with spatially adaptive companding for efficient learned image compression. In CVPR. 10239–10248.

[148] Ziqiu Zhang, Changyue Ma, Dong Liu, Li Li, and Feng Wu. 2021. Improving VVC intra coding via probability estimation and fusion of multiple prediction modes. In ICIG. Springer, 654–664.

[149] Jing Zhao, Bin Li, Jiahao Li, Ruiqin Xiong, and Yan Lu. 2021. A universal encoder rate distortion optimization framework for learned compression. In CVPR. 1880–1884.

[150] Zhenghui Zhao, Shiqi Wang, Shanshe Wang, Xinfeng Zhang, Siwei Ma, and Jiasheng Yang. 2019. Enhanced bi-prediction with convolutional neural network for high efficiency video coding. IEEE Transactions on Circuits and Systems for Video Technology 29, 11 (2019), 3291–3301.

[151] Zhisheng Zhong, Hiroaki Akutsu, and Kiyoharu Aizawa. 2020. Channel-level variable quantization network for deep image compression. In IJCAI. 467–473.

[152] Xiaosu Zhu, Jingkuan Song, Lianli Gao, Feng Zheng, and Heng Tao Shen. 2022. Unified multivariate Gaussian mixture for efficient neural image compression. In CVPR. 17612–17621.

[153] Yinhao Zhu, Yang Yang, and Taco Cohen. 2022. Transformer-based transform coding. In ICLR. https://openreview.net/forum?id=IDwN6xjHnK8

[154] Nannan Zou, Honglei Zhang, Francesco Cricri, Hamed R. Tavakoli, Jani Lainema, Miska Hannuksela, Emre Aksu, and Esa Rahtu. 2020. L2C – learning to learn to compress. In MMSP. 1–6.

[155] Renjie Zou, Chunfeng Song, and Zhaoxiang Zhang. 2022. The devil is in the details: Window-based attention for image compression. In CVPR. 17492–17501.

Received 29 August 2022; revised 19 February 2024; accepted 7 March 2024