

Received 9 February 2023, accepted 22 February 2023, date of publication 27 February 2023, date of current version 2 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3249284

RESEARCH ARTICLE

Reinforcement Learning for Rate-Distortion Optimized Hierarchical Prediction Structure

JUNG-KYUNG LEE¹, NAYOUNG KIM¹, AND JE-WON KANG^{1,2}, (Member, IEEE)

¹Department of Electronic and Electrical Engineering, Ewha Womans University, Seoul 03760, South Korea

²Graduate Program in Smart Factory, Ewha Womans University, Seoul 03760, South Korea

Corresponding author: Je-Won Kang (jewonk@ewha.ac.kr)


This work was supported in part by NRF through MSIT under Grant NRF-2022R1A2C4002052; in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korean Government, MSIT, "Development of ultra high resolution unstructured plenoptic video storage, compression, and streaming technology for medium to large space," under Grant 2020-0-00920; and in part by MSIT through the Information Technology Research Center (ITRC) Support Program, supervised by IITP, under Grant IITP-2023-2020-0-01460.

ABSTRACT Video coding standards use a prediction structure to arrange video frames and exploit temporal correlations. In this aspect, it is crucial to resolve complicated temporal dependencies among frames to improve coding efficiency because the coding of a preceding frame affects the rate-distortion (R-D) performance of the subsequent frames. Previous algorithms have attempted to address the problem using handcrafted features or analytical models even though natural videos display various temporal characteristics. In this paper, we propose a reinforcement learning (RL)-based decision algorithm to build the optimal hierarchical prediction structure under a random-access configuration (RA-HPS) in Versatile Video Coding (VVC). Our goal is to maximize coding efficiency by selecting a series of optimal group of pictures (GOP) structures for coding. Accordingly, we formulate an adaptive GOP selection algorithm with a binary tree to represent a policy. We generate an optimal binary tree to minimize the sum of the R-D costs among all plausible binary trees. A new RL policy representation is defined, and the optimal policy is obtained by a sequential update. The tree grows with a hierarchical state-action and a reward sequence in each node. For efficient learning, the proposed technique uses a deep Q-network architecture to capture the temporal correlation between frames, which helps learn the policy of the tree-based RL framework effectively. Experimental results demonstrate that the proposed technique achieves a significant Bjontegaard-Delta (BD)-rate reduction compared with state-of-the-art GOP size-selection algorithms.

INDEX TERMS Reinforcement learning, hierarchical B prediction, adaptive GOP, rate-distortion optimization, deep Q-network, VVC.

I. INTRODUCTION

Video content is increasingly delivered over the Internet and stored in data centers. The growth rate of video consumptions is higher than ever before, while wireless spectrum becomes scarce and costly. With the explosive growth of video traffic, high-efficiency video coding technology is crucial to provide a more engaging experience for users. Recently, the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) have developed the

The associate editor coordinating the review of this manuscript and approving it for publication was Diego Oliva .

Versatile Video Coding (VVC) standard [1], aiming at 50% bit-rate reduction compared with previous standards [2]. It is expected for the VVC to support emerging data-intensive video services, such as 4K ultra high definition (UHD), 360-degree video, and virtual reality (VR).

Video coding efficiency can be significantly improved by exploiting temporal correlation among adjacent video frames. There have been several studies using deep learning to reduce temporal redundancy and improve coding efficiency, by producing more accurate motion vectors and reference blocks [3], [4], [5], [6]. In [3] and [4], a virtual reference frame was generated from previously coded frames using

convolutional neural network (CNN). The generated signals were used to accurately predict the current sample by expressing irregular motions [5], [6]. In [7], a prediction block was generated by affine transformation and adaptive spatially-varying filters. The CNNs have been trained to approximate the reference block to the original block.

Although the deep learning-based inter coding methods improved coding performance, they have overlooked temporal dependency between the current frame and the reference frame. A coding order of a preceding frame can significantly affect coding performance of subsequent frames, and, accordingly the coding dependency needs to be carefully addressed. Current video coding standards use a prediction structure to encode video frames in an order and improve rate-distortion (R-D) performance through exploiting the temporal correlation. Hierarchical prediction structure under a random-access configuration (RA-HPS) was actively applied to High Efficiency Video Coding (HEVC), in which quantization parameters (QPs) were adjusted with a temporal layer (TL) of a frame in the prediction [8], [9]. It was further used for the VVC standard to meet various requirements.

The prediction structure is built with sequential and independent groups of intra-frames (I-frames) and inter-frames (P- or B-frames). In the group, intra- or inter-coded frames belonging to the lowest TL (TL-0) are called key frames. The key frames are coded with the highest fidelity. Non-key frames are coded using reference frames from lower layers. Key frames tend to be regularly positioned, and each length of a prediction structure is same. However, different positions of the key frames produce improved coding performance because a temporal segment of a video often exhibits diversified characteristics. Furthermore, a coding order of frames and a selection of reference frames are subsequently changed based upon the decision, which affects coding performance of subsequent frames.

Previous studies have identified the intriguing problem and attempted to present optimized prediction structures. The prediction structure is developed to resolve the temporal dependencies and improve coding efficiency [10], [11], [12], [13], [14]. In [15], an adaptive key-frame selection based on spatial and temporal features was presented. In [12], the coding order of B-frames was optimized recursively. The optimization of the reference picture selection was solved using the Viterbi algorithm [16]. Temporally dependent R-D optimizations (TD-RDO) was proposed to improve coding performance by measuring the distortion propagated from the previously coded frames [11], [17]. Several studies have developed a coding scheme to divide the prediction structures, by using scene change detection [18], [19]. The more accurate the temporal changes of a frame, the easier it is to decide the size of a prediction structure, which enhances R-D performance.

In this paper, we focus on developing a reinforcement learning (RL)-based decision algorithm to recursively divide video intervals into several groups and locate key and non-key frames in an RA-HPS in VVC [1]. The previous studies

have attempted to solve the problem with analytical models and handcrafted features [12], [18], [19] or apply action proposals to perform temporal localization and segmentation using CNNs [20]. However, it was difficult to manage complicated temporal dependencies in high-dimensional videos displaying various temporal characteristics. In fact, the coding order and the display order of video frames differ in an RA-HPS, which causes more complicated temporal dependency among frames, whereas the other prediction structures, such as low-delay (LD), yield relatively simple dependencies to sequentially encode frames [11], [21]. Thus, the coding of a preceding frame significantly affects the R-D performance of the subsequent frames [11], [17].

The contributions of our proposed method are summarized as follows:

- To the best of authors' knowledge, this is the first study to use the RL to build an R-D optimized prediction structure in VVC, extended from our previous work [22]. In the proposed method, we make a partition of a video interval as a sequence of a state-action for an agent to conduct an RL. The optimal solution is obtained by maximizing a reward function of the RL policy representation.
- We formulate an optimization of a prediction structure for coding, by a sequential growing of a tree and a hierarchical state-action and reward sequence. We develop a new learning scheme and a deep Q-network architecture based on the problem formulation.
- We verify the results of the proposed method compared with state-of-the-art studies. Experimental results demonstrate that the proposed method can adapt to a dynamic video sequence and achieve optimal coding performance.

The rest of the paper is organized as follows. In Sec. II and Sec. III, we review the related works and the RL. We explain the proposed technique in Sec. IV. Experimental results and analysis are presented in Sec. V. The conclusion is remarked in Sec. VI.

II. RELATED WORKS

A. TEMPORAL PREDICTION STRUCTURE

Temporal prediction structure has been actively studied in several video compression applications such as R-D optimization [16], [23], perceptual optimizations [18], [24], rate control [25], [26], and video streaming Internet-of-Things (IoT) [27]. It is used to maximize coding efficiency and provide scalable features in video coding [28].

In video coding standards, a video frame is involved to a group-of-picture (GOP) as a unit of a prediction structure to determine a reference frame for a current frame. A GOP structure is built using the current key frame and the frames between two consecutive key frames. The key frames tend to be coded with the highest fidelity among the grouped frames. Non-key frames of the same TLs in a GOP are usually coded with the same coding schemes, while using an incremented

offset to the QP of the key frames. The coding order is chosen in a way that the reference frames of the lower TLs are coded before.

RA-HPS uses a hierarchical B prediction structure. Fig. 1 displays an RA-HPS with four temporal hierarchy layers, in which all frames are hierarchically grouped in TLs. In an RA-HPS, the coding order and the display order of video frames differ due to the hierarchical B-prediction. In Fig. 1(a), a B-prediction is applied to B_4 in the second layer (TL-1) by referencing I_0 and P_8 in TL-0. B_2 in the next layer is subsequently coded using I_0 and B_4 . The frames of the lower TLs are more critical because their reconstruction qualities should affect the coding of the other frames. Several works have indicated that a fixed coding mechanisms in an HPS cannot offer optimal coding performance due to the different degrees of motion dynamics in video sequences [29], [30].

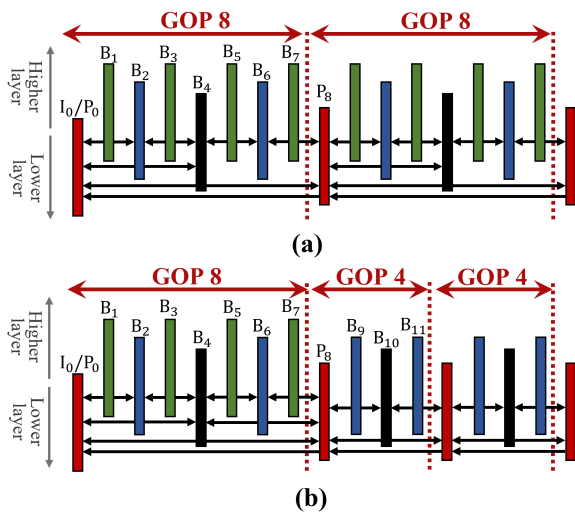


FIGURE 1. RA-HPS using (a) a fixed-size GOP and (b) an adaptive-size GOP. The red frames refer to key frames in TL-0. The higher frame in the figure, the higher the TL.

B. PREVIOUS STUDIES FOR ADAPTIVE PREDICTION STRUCTURE

The interval between two consecutive key frames (or the size of a GOP) is not necessarily uniform for coding efficiency. For instance, in one hand, a large group size is more beneficial to encode static video scenes because there are slight motion changes between two key frames. In the other hand, a small group size is chosen for dynamic motions. The GOP sizes can be changed based upon the temporal characteristics of a video sequence. In Fig. 1(b), the second partition consists of two GOPs with a size of 4, which is more favorable to scene changes occurring within the partition. The fixed GOP structure cannot manage the various temporal properties, appropriately.

Several studies have improved R-D performance by selecting an adequate GOP size and avoiding computational complexity in an encoder [18], [19], [31], [31], [32], [33], [34],

[35], [36]. In [32], when temporal changes between frames are detected, a video sequence is segmented and encoded. The video sequence is partitioned using entropy changes and pixel differences at which each I-frame is inserted to build the adaptive GOP. The GOP size has been determined based on the changes in the motion activity [33], [36]. In [36], Ding et al. presented a method using motion vectors and residues to find abrupt temporal changes and determine the size. In [33], Sakamoto et al. analyzed spatial and motion complexity of video frames and selected an optimal GOP size to improve coding performance. It has been applied to UHD video coding using HEVC. In [19] and [31], Poobalasingam et al. proposed to use various GOP sizes by measuring the temporal steadiness of video contents. In [19], they used encoding parameters as texture features not to rely on the previous estimation of motion vectors in a video sequence. In [31], a homogeneous texture descriptor was selected to determine the size.

A fixed GOP size can result in inaccurate selection of reference key frames and flickering artifacts in the decoded video. Vijayanagar and Kim [34] developed two-phase block classification scheme and a prioritized block classification scheme to improve the efficiency. Coarse-to-fine refinements were conducted to determine the final GOP size [35]. The approach started with a coarse GOP size and then modified it by iterative checking the size based on a frame-level intra mode decision method. Chen et al. [18] developed a perceptual hash algorithm-based adaptive GOP selection method. This approach used a hashing algorithm to measure the difference in every two successive frames and determined the GOP size between 1~3. Ascenso et al. [37] presented a method to decide the size based on hierarchical clustering based on block statistics. These features were used for detecting changes in both global and local motion. The decision task of an adaptive GOP size was formulated as a classification problem. Huong et al. [38] created several spatial and temporal features generated from pixel values and motion vectors, respectively. A decision tree algorithm was used for training samples and classification to decide a GOP size.

III. REINFORCEMENT LEARNING

An RL system uses two key concepts that are an agent and an environment to find optimal sequential decisions. It is defined as a tuple of (s, a, r, t) , where s is a state, a is an action, r is a reward, and t is a transition to the next state s' . The agent learns a policy to take an action from s to s' to maximize a reward from its environment even though it may not have analytical knowledge about the environment. At each time t , the policy $\pi(s_t)$ outputs an action a_t in the current state s_t to maximize the total reward while receiving an immediate reward of r_{t+1} . Accordingly, an episode, which is the full interaction process between an agent and an environment, produces a state-action and reward sequence $\{s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_T\}$ ending at the terminal state s_T . r can be replaced with a cost to minimize the total cost equivalently.

The agent learns how to act in an environment to achieve to maximize the sum of rewards. This is achieved by predicting how many values the state or the action provides. For this purpose, under a given policy π , we define a state-action value function (also known as Q -function) $Q_\pi(s, a)$ [39] as

$$Q_\pi(s, a) = E_\pi \left[\sum_{k=0}^{T-1} \gamma^k r_{t+1+k} | s_t = s, a_t = a \right], \quad (1)$$

where $\gamma \in [0, 1]$ is a discount factor used to penalize the future rewards. Eq. (1) can be expressed as the Bellman equation [40] that decompose the value function into the immediate reward and the future values.

$$Q_\pi(s, a) = E_\pi [r_{t+1} + \gamma E_{a \sim \pi} Q(s_{t+1}, a) | s_t = s, a_t = a], \quad (2)$$

which is computed with the recursive update process.

There exists an optimal policy π^* such that $Q_{\pi^*}(s, a) = \max_\pi Q_\pi(s, a)$ for all plausible pairs of states and actions in the action-replay process [40]. Thus, the optimal action a_t^* is taken by,

$$a_t^* = \arg \max_a Q_{\pi^*}(s_t, a). \quad (3)$$

A. RL IN VIDEO CODING

Several pioneering studies have applied RL algorithms to video coding applications. RL was used to formulate HEVC intra-frame rate control [41], in which each QP value of coding tree units (CTUs) in a frame is sequentially determined to minimize the distortion of a frame within rate constraints. They considered the texture complexity of CTUs and the bit balance as a state, the QP value as an action, and the distortion of the CTU as a reward. They adopted a three-layered fully connected (FC) neural network for Q-learning. In [42], the study was extended to inter-frame rate control. They define an RL agent to allocate bits to each frame to maximize cumulative RL rewards and eventually prevent fluctuations of bit-rates over time. In [43], more comprehensive state vectors including block properties with pixel values and gradients and coding properties with bit balance are defined both at the frame-level and CTU-level. Action has been taken to determine the QP values relative to the frame level.

Some efforts were made to use the RL system to optimize the R-D performance and speed up the mode decisions of HEVC encoders. In [44], Helle et al. presented an RL framework to automatically learn complexity-scalable encoder strategies from a video sequence. It incorporates a continuous trade-off between the RD-cost and the computational complexity. In [45], Chung et al. utilized the RL to facilitate the coding unit (CU) split decision in HEVC. They used the sample values of a CU as the state and the split decision as an action. The reduction in R-D cost relative to maintaining the current CU was counted for the reward. For the Q-learning, CNNs with three convolutional layers and two FC layers are used for different CU sizes. In [46], a CU size

decision based on RL was developed to decrease HEVC intra-coding complexity. They conducted early termination of the quad-tree splitting process, by considering the encoder as an agent, sequentially making coding decisions.

IV. PROPOSED METHOD

In this paper, we present an efficient coding technique using temporal segmentation of video frames and key-frame allocation with a deep RL method. The goal is to maximize coding efficiency by building optimal RA-HPS structures. Because the RA-HPS uses a hierarchical B-prediction structure, a division of a prediction structure is recursively conducted. The RL is chosen to organize the recursive structure and select important video frames to affect coding performance of subsequent frames. We use a GOP as a basic prediction structure, and coding gains of the frames within a GOP are used for rewards in the RL system.

We assume that the period of I-frames is already determined as L , and there can be several GOPs divided in the period. The prediction structure is independently optimized in every period. Furthermore, we assume the size of the GOP is 2^l , $l = 2, 3, \dots, \log_2 L$. The smallest size of the GOP is 8 in our setting, and the size can expand to L . Our objective is to determine the dyadic hierarchical divisions of the given video frames $\mathcal{F} = \{f_0, \dots, f_{L-1}\}$ into several different sizes of GOPs of RA-HPS.

A. PROBLEM FORMULATION WITH A BINARY TREE

We use a binary tree T to divide \mathcal{F} into several GOPs and automatically determine their sizes as depicted in Fig. 2. Each node $n = (k, l) \in T$ represents a coding state to encode $l - 1$ frames between f_k and f_{k+l} when f_k and f_{k+l} have been already coded as the key frames of the current GOP.

Starting from the root node $n^0 = (0, L)$, a tree may bifurcate from n^i in the i -th layer to the left node n_l^{i+1} and the right node n_r^{i+1} . A node becomes a leaf node as the final state when a GOP is determined with its key frames. It is noted that, once the position of the key frame and the size of the GOP are specified, the prediction structure of the in-between frames are all determined using the hierarchical B-prediction. An intermediate node is split into two child nodes that may represent two subdivided GOPs or continue the bifurcation. Therefore, a tree T generates a unique prediction structure of \mathcal{F} . An edge $e \in T$ represents an action to be described later.

Let $J(\tau; T)$ define the R-D cost as

$$J(\tau; T) = D + \lambda R, \quad (4)$$

where R and D stand for the bit-rates and the distortion when f_τ is encoded. λ is the Lagrangian multiplier. It is differently specified with QPs and frame types in the R-D optimization of VVC reference software.

We attempt to generate an optimal tree T^* by minimizing the sum of the R-D costs

$$T^* = \arg \min_{T \in \mathcal{T}} \sum_{\tau} J(\tau; T), \quad (5)$$

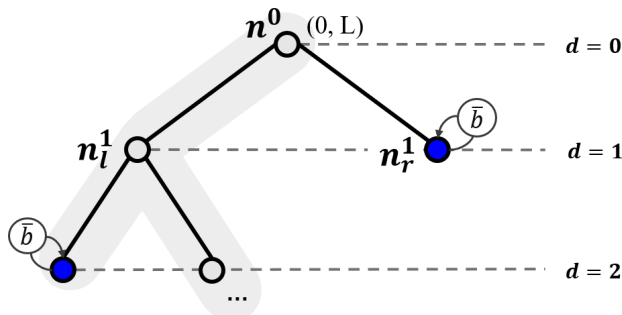


FIGURE 2. A binary tree T to divide a video sequence into several GOPs.

where T presents all plausible binary trees representing different GOP structures.

The computations of the R-D costs depend on the reconstruction states of frames in different prediction structures. However, there are a large number of possible binary trees, approximating the Catalan number [47]. It is computationally expensive to obtain the solution in Eq. (5) by exhaustively searching for all binary trees in T . In our method, we attempt to adopt an RL system to solve the problem more efficiently.

B. POLICY REPRESENTATION

We formulate an RL problem to calculate the minimum costs in Eq. (5). We recall that a node $n = (k, l)$ is a state to encode the frames $\{f_{k+1}, \dots, f_{k+l-1}\}$ when f_k and f_{k+l} are given.

In an intermediate node, an encoder has a choice about whether to split the current GOP. We define such an action $a_n \in \{b, \bar{b}\}$ at the current node n (or a state). b is taken to create the child nodes. In other words, it divides the GOP into two sub-GOPs and sets the middle frame as a key frame for coding. The child nodes can be further split in the next depth. In contrast, \bar{b} is taken to set the node as a leaf node, which determines to encode the frames as in the current GOP. Fig.2 displays an example of the tree in which nodes take the actions.

A policy π now consists of a sequential binary decision at each node in a tree. To apply RL to a high-dimensional environment, the state space was mapped into a low-dimensional feature space in the previous studies [43], [44], [45], [46]. In these studies, the states are represented by feature vectors consisting of encoding parameters. This includes the dimension of features and knowledge on block-level or frame-level encoding parameters.

However, it is challenging to apply the same Markov decision process (MDP) framework to our problem because the state representation has a larger dimensionality. In such cases, the RL cannot offer an adequate near-optimal policy and require numerous training samples. For instance, considering two subdivided GOPs, the key frame of the first sub-GOP in the sibling nodes would affect the coding of the second sub-GOP. Therefore, we use an approximated policy iteration with hierarchical binary-state space decomposition motivated by [48]. This decomposition implies that, by decomposing

the original tree into multiple sub-trees with smaller state spaces, it is possible to derive superior local policies, and a near-optimal global policy can be achieved

Let a Q-function $Q_\pi(k, l; a)$ denote the expected costs to encode the frames from f_{k+1} to f_{k+l-1} when following π in the current node n . Then, by using the hierarchical binary-state space decomposition, the optimal Q-function to achieve Eq. (5) is obtained by

$$\min_a \{Q_\pi(k, \frac{l}{2}; a) + Q_\pi(k + \frac{l}{2}, \frac{l}{2}; a) + J_a(k + \frac{l}{2})\}, \quad (6)$$

where we applied the Bellman equation with $\gamma = 1$ in Eq. (2) while replacing max operation due to the cost function J instead of a reward function.

$J_a(k + \frac{l}{2})$ computes the R-D cost to encode the middle frame in the GOP. In the codec, the Lagrangian multiplier λ in J_a depends on the temporal layer of the frame and QP values as in Eq. (4), given as

$$\lambda = \alpha \times W_k \times 2^{\frac{(QP-12)}{3}}, \quad (7)$$

where α is determined as the number of B-frames used for referencing, and W_k is defined as [1]

$$W_k = \begin{cases} 0.442, & \text{TL} = 0 \\ 0.3536 \times \text{Clip}(2, 4, \frac{(QP - 12)}{6}), & \text{Otherwise,} \end{cases} \quad (8)$$

where the $\text{Clip}(v_l, v_u, v)$ function limits the value v between v_l and v_u . Therefore, the cost function relies on the current action. J_a calculates differently depending on whether the middle frame is determined as a key frame by $a = \bar{b}$.

The node chooses an optimal action a^* to take, by conducting the optimization as

$$a^* = \arg \min_a Q_{\pi^*}(k, l; a), \quad (9)$$

where we obtain the solution through deep Q-learning [49] described in the following subsection.

We obtain the near-optimal global policy by combining the local policies in each sub-tree [48]. Therefore, an optimal prediction structure is given by aggregating each leaf node of the derived tree T_π^* . The interaction process is illustrated in Fig. 3. In Fig. 2, assuming that \mathcal{F} includes 32 frames, the last GOP includes 16 frames from f_{16} to f_{31} because n_r is determined as a leaf node to choose \bar{b} . The first GOP includes eight frames from f_0 to f_7 because it has depth of one higher, whereas the other frames are not determined yet. The optimal set of actions \mathcal{A}^* is obtained by tracking back from the leaf nodes to the root node in T_π^* .

C. LEARNING ON POLICY

We adopt a deep Q-network (DQN) [49] to learn the proposed policy and obtain the approximated solution of Eq. (9). The original DQN uses two deep neural networks: a main network and a target network. The main network and target network estimate the current Q-values and the subsequent values in the next state and action, respectively.

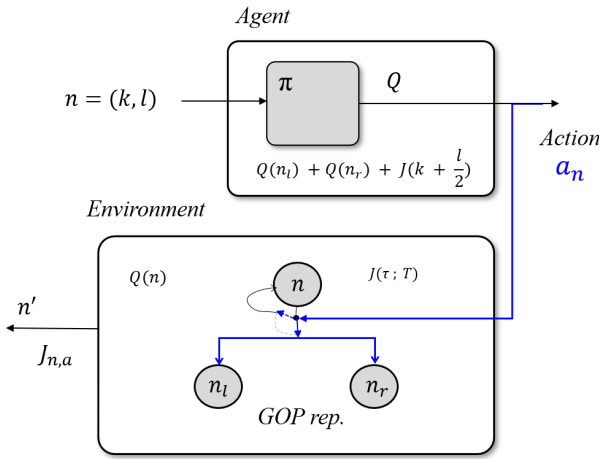


FIGURE 3. RL setting in the proposed technique. A node and an edge refer to a state and an action, respectively. A new state is given by creating the child nodes or determining itself as a leaf node while an agent takes a cost J .

In Q-learning, an agent starts with an initial Q-value and iteratively enhances the estimation by taking an action while observing the cost and the next state. It updates the estimation of the Q-network with weights θ by minimizing the error δ , given as

$$\delta = Q_{\pi_{\theta}}(k, l; a) - \min_{a'} \{Q_{\pi_{\theta'}}(k_l, \frac{l}{2}; a') + Q_{\pi_{\theta'}}(k_r, \frac{l}{2}; a') + J_a(k + \frac{l}{2})\}, \quad (10)$$

where $Q_{\pi_{\theta}}$ and $Q_{\pi_{\theta'}}$ are the Q-values from the main and target networks, respectively. The target and main networks have the same architectures and parameter sets, while the target network operates at one-step ahead of the main network. In the next action, θ goes to θ' . k_r and k_l are the starting frames in the subdivided intervals, respectively.

We recall the cost function J_a depends on several factors such as QPs and TLs. Although most of the previous RL-based video coding studies [41], [42], [43], [44], [45], [46] have used normalized R-D costs in the Q-function, we found that those approaches might not be effective in our framework because they incur unstable learning. Therefore, we define a new constant cost function \tilde{J}_a . For this purpose, we have encoded training video sequences using different GOP structures and examined their R-D costs to determine which action should lead to the optimal solution. Let \mathcal{A}^* denote the trajectory of the ground-truth action (or bifurcation) in a video. Then, \tilde{J}_a is defined as,

$$\tilde{J}_a = \begin{cases} \nu - \varepsilon, & \text{if } a \in \mathcal{A}^*. \\ \nu, & \text{otherwise,} \end{cases} \quad (11)$$

where the function outputs a smaller cost when the current action is a part of the optimal set. ν and ε are both set to 1 in our experiments, respectively. When the current node is the root node, the ν value is set to 3 to prevent the early termination of a tree. We set ν and ε aiming to get the fair

cost function regardless of any possible GOP selection and adjust them heuristically.

We iteratively optimize the Q-network by minimizing the sum of the squared error, *i.e.*, $\sum_B \delta^2$, where B is a sample batch. The parameter set θ of the policy network is updated during the learning using a stochastic gradient descent method.

Algorithm 1 displays the policy update during the learning. For every node n_i at iteration i , the Q-function selects a_i minimizing Eq. (9) and observe \tilde{J}_i . The bifurcation of the tree stops when the nodes are determined as leaf nodes from Eq. (9). It also stops when the tree reaches the maximum depth which is set to 2 in our experiments. When the tree finishes its bifurcation, a new tree is initialized and the current episode continues.

Algorithm 1 Proposed DQN

- 0: Initialize D , Q , and θ
 - 0: **for** episode = 1, ..., M **do**
 - 0: Initialize a tree T
 - 0: Set $w = 0$
 - 0: **for** $i = 1, \dots$, until episode terminates **do**
 - 0: Select $a_i = \arg \min_a Q_{\pi_{\theta_i}}$ using Eq. 9
 - 0: Execute a_i and observe \tilde{J}_i
 - 0: Store transition $(n_i, a_i, \tilde{J}_i, n_{j+1})$ in D
 - 0: Sample batch of transitions $(n_j, a_j, \tilde{J}_j, n_{j+1})$
 - 0: Update θ with the loss δ^2
 - 0: **if** $a \notin \mathcal{A}^*$ **then**
 - 0: $w \leftarrow w + 1$
 - 0: **end if**
 - 0: **if** w is equal to w_{max} **then**
 - 0: Terminate episode
 - 0: **end if**
 - 0: **end for**
 - 0: **end for** = 0
-

For the termination condition of an episode, we define the accumulated costs of a case when $a \notin \mathcal{A}^*$ per episode as w . The costs would increase by ν during iteration, and each episode is terminated when w_{max} has been reached. We set w_{max} to 3. In this manner, even though the tree selects a suboptimal action to lead to a higher cost, the episode does not stop immediately but continues until reaching w_{max} . If the episode continues to run, it results in higher-performing scenarios with smaller accumulated costs.

In our implementation, we adopt a replay memory [50] to facilitate the learning. The replay memory D stores an experience of the agent as a tuple of a transition $(n_i, a_i, \tilde{J}_i, n_{i+1})$ and randomly samples it over episodes. Our method records the last N transition samples in the replay memory and selects a random batch with a uniform distribution from the memory to update a parameter. The uniform sampling provides equal importance to all transition samples in the replay memory to prevent the learning from getting trapped in local minima. It is alleged in [50] that learning directly from

consecutive samples is inefficient because of the high correlations between the samples. The sampling process mitigates this inefficiency problem by decorrelating between video frames and stabilizing the DQN training procedure.

D. MODEL ARCHITECTURE

We build a two-stream network consisting of a leaf-stream (LS) network and a bifurcation-stream (BS) network as shown in Fig. 4. We create the streams to consider various coding states within a current GOP to encode frames using different sub-GOP sizes and key frames. To specify, the two key frames and the middle frame in the current GOP are concatenated for processing by the LS network. It extracts some features about an action in which the current node is set to a leaf node. On the other hand, for the BS network, there are two sub-branches of convolution layers to extract the features of the two subdivided GOPs. The input frames are determined with the current node to which the model operates. For example, in a node $n_0 = (0, 32)$, the LS network takes three concatenated input frames of f_0, f_{16} , and f_{32} as shown in Fig. 4. The BS network takes f_0, f_8 , and f_{16} for the first branch and f_{16}, f_{24} , and f_{32} for the second branch. The features are merged into FC layers to produce an action. The video frames are only the inputs to represent the current state and to compute Q-values and produce an output action.

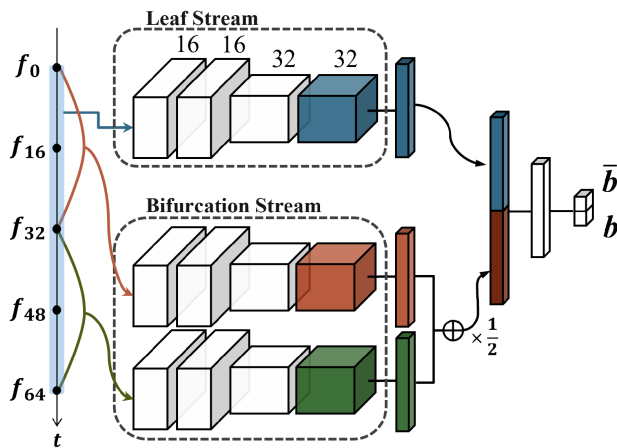


FIGURE 4. The proposed two-stream-network architecture.

We explain more details about the implementation. We use only the luminance (Y) channel of video frames because the Y-channel contains most of the visual information for coding. Let H and W be the height and width of a video frame. Therefore, each stream takes an input of which dimension is $H \times W \times 3$ after the concatenation. All the streams share the same network architecture that share the same set of weights. The stream consists of eight layers, including five convolutional layers followed by a nonlinear function and two FC layers. The first two convolutional layers have 16 filters of size 3×3 , and the other three convolutional layers have 32 filters of size 3×3 . The filtering is conducted with a

stride of 2 pixels and padding with 2 pixels. Two max-pooling layers are carried out over a 2×2 window with stride 2 and a 4×4 window with stride 2 after the first and third convolutional layers, respectively. The output is flattened and passed to an FC layer of 256 hidden units. At the end of BS, the two outputs are added and divided in half. Finally, the outputs from LS and BS are concatenated at the end of the streams, and the two FC layers are applied. The output corresponds to the Q-value for each valid action, b and \bar{b} .

V. EXPERIMENTAL RESULT

A. EXPERIMENTAL SETTINGS

1) TEST CONFIGURATION

We implement the proposed technique based on the recent VVC reference software, i.e., VTM version 16.0 [53]. The tested methods are also implemented with the same software. We conduct the experiments under configuration with the common test conditions (CTC) [54]. The encoder uses an RA configuration with the set of QPs = {22, 27, 32, 37}. The QP differences with difference TIs are set as in the CTC. For the CTC sequences, various resolutions of test video sequences categorized as Class A~Class D are used for performance comparisons. In addition to the CTC test sequences, we use a YouTube test [55] set to examine the performance with various properties of videos. For the YouTube test set, we use 40 test sequences and categorize them into static videos and dynamic videos with respect to the number of scene changes. The static videos have no scene changes. The dynamic videos are divided into two subgroups as shown in Table 1. In both test dataset, 64 frames are used for the experiments.

We set a size of a GOP to 16 through a test video sequence for an anchor. The configuration is referred to as “Fixed GOP 16”. In the experiments, we include various GOP sizes for comparisons, considering different characteristics of test video sequences. “Fixed GOP 8,” “Fixed GOP 32,” and “Fixed GOP 64” represent that the sizes of GOP in VTM software are set to 8, 32, and 64, respectively. The GOP size of less than 8 is not considered, because the size is rarely selected in our empirical results. The experiments are performed with a 3.60 GHz Intel CPU, 8.0 GB RAM, and NVIDIA TITAN X GPU.

2) TRAINING SET

The training videos are collected from YouTube and resized to 416×240 . The training videos are not used in the testing. We attempted to have the training videos display different degrees of motions. For example, the video frames in several training video sequences have been randomly stitched to display more temporal dynamics and scene changes, as shown in Figure 5. During training, we determine the ground-truth GOP sets as the GOPs to provide the highest R-D performance and use them for the supervision in Eq.(11). The YouTube dataset includes approximately 9,000 videos for training and 300 videos for validation.

TABLE 1. BD-rate (in the unit of %) reduction in Y component of the proposed technique compared with [18] and [38] in YouTube test videos. The anchor is a fixed GOP 16 for comparisons. A fixed GOP of 8 and fixed GOP of 32 are tested by changing the size of the GOP in the VTM 16.0 software. Chen et al. [18], Huong et al. [38], Garcia del Molino et al. [51], Li et al. [52], and Lee et al. [22] are the state-of-the-art adaptive GOP algorithm.

Category	# of scene change	Seq.	Fixed GOP			[18]	[38]	[51]	[52]	[22]	Ours	
			8	32	64							
Static video	0	1	16.8%	-5.5%	22.9%	0.0%	22.9%	3.9%	-5.5%	-4.1%	-4.1%	
		2	17.2%	-5.4%	22.0%	0.0%	22%	3.8%	15.8%	3.8%	-4.6%	
		3	17.9%	-9.0%	-6.7%	17.9%	0.0%	-12.0%	-9.0%	0.0%	-6.7%	
		4	-0.7%	0.6%	1.1%	0.0%	0.0%	-0.5%	1.1%	-0.5%	-0.7%	
		5	0.1%	0.2%	2.9%	0.0%	0.0%	0.0%	2.9%	0.0%	-1.4%	
		6	-7.5%	7.0%	42.5%	-7.5%	0.0%	-7.0%	-7.7%	0.0%	-6.8%	
		7	14.1%	-6.8%	-4.4%	-4.4%	-0.4%	-13.9%	-13.9%	-4.4%	-4.4%	
		8	5.9%	0.2%	32%	0.0%	-2.5%	0.0%	5.2%	0.2%	0.2%	
		9	12.5%	-8.5%	12.4%	0.0%	12.5%	3.5%	11.4%	12.4%	-8.5%	
		10	18.1%	-8.8%	-7.8%	-8.8%	18.1%	-8.5%	-8.8%	-7.8%	-7.8%	
		11	15.2%	-5.5%	1.4%	-5.5%	10.8%	7.6%	1.4%	1.4%	-5.5%	
		12	8.7%	-1.1%	6.2%	0.0%	8.7%	2.8%	6.2%	0.0%	-1.1%	
		13	10.6%	-2.6%	18.4%	-2.6%	0.0%	6.2%	0.0%	-2.6%	-2.6%	
		14	13.4%	-4.2%	-0.7%	-4.2%	4.6%	5.7%	-0.7%	-0.7%	-0.7%	
		15	16.3%	-3.4%	4.6%	0.0%	5.3%	5.9%	4.6%	-3.4%	-3.4%	
		16	7.3%	-1.0%	11.8%	-1.0%	5.1%	2.3%	11.8%	2.3%	-1.0%	
		17	9.6%	-0.6%	13.1%	0.0%	9.6%	5.1%	13.1%	0.0%	-0.6%	
		18	16.7%	-5.6%	2.1%	-5.6%	16.7%	-5.6%	2.1%	-5.6%	-5.6%	
		19	11.5%	-1.8%	7.1%	-1.8%	11.5%	-1.8%	7.1%	-1.8%	-1.8%	
		20	10.6%	-3.2%	9.2%	0.0%	10.6%	2.7%	9.2%	0.0%	-1.8%	
	Average BD-rate			10.7%	-3.2%	9.5%	-1.2%	7.8%	0.3%	2.3%	-0.5%	-3.4%
Dynamic video	1	21	13.6%	-7.0%	-8.1%	-8.1%	-8.1%	-14.2%	1.9%	-8.1%	-8.1%	
		22	14.6%	-7.0%	13.1%	13.1%	13.1%	1.2%	3.7%	-7.0%	-7.0%	
		23	9.8%	-4.4%	-5.7%	-4.4%	8.8%	-17.5%	-4.4%	-5.7%	-5.7%	
		24	9.5%	-1.9%	21.9%	-1.9%	7.1%	-7.1%	0.9%	-7.1%	-7.1%	
		25	15.4%	-6.7%	-8.8%	15.4%	9.4%	-12.5%	0.9%	-12.5%	-8.8%	
		26	15.7%	-7.3%	-8.2%	15.7%	-8.2%	-16.3%	4.1%	0.0%	-8.2%	
		27	16.9%	-4.6%	25.2%	0.0%	11.3%	5.0%	10.3%	25.2%	-4.6%	
		28	2.5%	4.0%	29.6%	0.0%	0.4%	-5.6%	4.7%	4.0%	-10.7%	
		29	28.9%	-12.3%	-13.3%	0.0%	-13.3%	-7.7%	5.3%	-7.7%	-13.3%	
		30	13.2%	-6.3%	-5.6%	13.2%	0.0%	-15.5%	-6.3%	-6.3%	-5.6%	
	Av.			14.0%	-5.4%	4.0%	4.3%	2.0%	-9.0%	2.1%	-2.5%	-7.9%
	≥ 2	31	5.9%	-10.4%	0.9%	0%	5.9%	2.1%	2.4%	2.1%	-9.6%	
		32	11.0%	-2.9%	1.9%	0.0%	1.9%	-4.5%	-2.5%	-4.5%	-3.1%	
		33	8.5%	1.1%	26.7%	8.5%	26.7%	-4.0%	8.2%	-4.0%	1.1%	
		34	8.8%	-0.4%	23.6%	23.6%	6.0%	2.4%	2.3%	2.4%	-4.0%	
		35	-3.9%	5.6%	47.9%	-3.9%	-3.9%	-5.6%	7.6%	-3.9%	-3.9%	
		36	5.4%	3.4%	47.0%	0.0%	3.2%	-3.2%	2.6%	-3.2%	-3.2%	
		37	-0.3%	3.8%	29.7%	3.8%	-0.3%	-6.9%	9.2%	-0.3%	-7.9%	
		38	1.0%	5.0%	41.1%	5.0%	41.1%	-3.6%	1.0%	-3.6%	-3.6%	
		39	-5.2%	5.9%	59.0%	0.0%	0.4%	-9.8%	6.9%	0.0%	-9.8%	
		40	-2.6%	10.3%	36.3%	0.0%	9.4%	1.4%	5.3%	1.4%	1.4%	
Av.			2.9%	2.1%	31.4%	8.4%	9.0%	-2.5%	4.3%	-1.4%	-4.2%	
Average BD-rate			8.4%	-1.6%	17.7%	6.3%	5.5%	-5.8%	3.2%	-1.9%	-6.1%	
Total average BD-rate			9.6%	-2.4%	13.6%	2.6%	6.7%	-2.7%	2.8%	-1.2%	-4.8%	
Encoding time			-	-	-	100%	100%	103%	100%	101%	101%	

3) IMPLEMENTATION DETAILS

We implement the proposed network using Tensorflow and train the network with the Adam optimizer [56] to update the network parameters for 1,000 episodes. We conduct an ϵ -greedy policy with $\epsilon = 0.01$ as in [50]. In our implementation, we have imposed the Q-function to consider the sum of the costs of the descendant nodes up to a depth of 2 to reduce complexity.

B. CODING PERFORMANCE AND ANALYSIS

We evaluate the coding performance of the proposed technique in comparison to several state-of-the-art adaptive GOP algorithms. We justify the efficiency of the proposed technique by comparing the coding performance of

“fixed GOP 8”, “fixed GOP 32,” “fixed GOP 64”, Huong et al. [38], Chen et al. [18], Garcia del Molino et al. [51], Li et al. [52], and Lee and Kang [22]. We further explain the details of the tested methods. Huong et al. utilized a binary tree to solve a classification problem when structuring a GOP. Chen et al. took a discriminative approach by thresholding a relation value between frames. Garcia et al. proposed event segmentation paradigm that used LSTM-based generative network to predict their visual context. In the comparison, the segmentation results have been used to determine the closest GOP size. Li et al. utilized a hidden Markov model to frame-level characteristics such as the correlation coefficient within neighboring frames and noise level. Lee et al. [22] uses a fixed tree structure to optimize the prediction structure

TABLE 2. The BD-rate (in the unit of %) reduction of the proposed technique compared with [18] and [38] in the CTC test videos. The anchor is a fixed GOP 16 for comparisons. A fixed GOP of 8 and fixed GOP of 32 are tested by changing the size of the GOP in the VTM 16.0 software. Chen et al. [18], Huang et al. [38], Garcia del Molino et al. [51], and Li et al. [52] are the state-of-the-art adaptive GOP algorithm.

Category	Test Sequence	Fixed GOP			[18]	[38]	[51]	[52]	Ours
		8	32	64					
A1	Campfire	1.2%	0.0%	-3.7%	0.0%	1.2%	0.5%	-3.7%	-3.7%
	FoodMarket4	2.2%	0.0%	1.7%	0.0%	0.0%	0.6%	1.7%	0.0%
	Tango2	2.7%	0.9%	5.1%	0.9%	2.5%	0.8%	0.0%	-0.9%
A2	CatRobot1	5.2%	0.0%	3.0%	0.0%	5.1%	2.6%	3.0%	0.0%
	DaylightRoad2	10.5%	-1.2%	2.8%	2.8%	10.5%	9.4%	2.8%	-1.2%
	ParkRunning3	4.7%	0.0%	3.2%	3.2%	4.7%	1.7%	3.2%	0.0%
B	BasketballDrive	5.7%	0.9%	6.9%	0.9%	0.0%	2.5%	5.7%	0.2%
	BQTerrace	7.9%	-1.9%	-12.1%	-12.1%	6.5%	3.3%	-12.1%	-12.1%
	Cactus	8.7%	-1.6%	0.6%	-1.6%	2.8%	3.9%	0.6%	-2.4%
	MarketPlace	9.7%	-0.3%	2.7%	-0.3%	0.0%	2.9%	2.7%	-1.8%
	RitualDance	4.9%	-0.5%	5.5%	-0.5%	4.9%	1.5%	4.9%	1.5%
C	BasketballDrill	10.9%	-4.2%	-5.4%	0.0%	7.9%	8.0%	-4.8%	-4.8%
	BQMall	7.7%	-1.9%	4.8%	-1.9%	1.6%	5.1%	4.8%	-3.4%
	PartyScene	9.9%	-2.0%	3.0%	3.0%	6.7%	-2.9%	-2.9%	-2.9%
	RaceHorses	6.0%	0.8%	4.5%	0.0%	6.0%	4.3%	-0.1%	-0.1%
D	BasketballPass	2.0%	-0.2%	4.7%	-0.2%	2.0%	1.5%	4.7%	-1.4%
	BlowingBubbles	6.7%	0.0%	13.4%	0.0%	1.7%	4.3%	0.0%	-2.5%
	BQSquare	15.8%	-2.7%	7.1%	0.0%	10.6%	6.0%	-2.8%	-2.8%
	RaceHorses	4.4%	0.3%	8.0%	0.3%	1.4%	-1.0%	0.3%	-1.0%
Total average BD-rate		6.7%	-0.7%	2.9%	-0.3%	4.0%	2.9%	0.4%	-2.1%

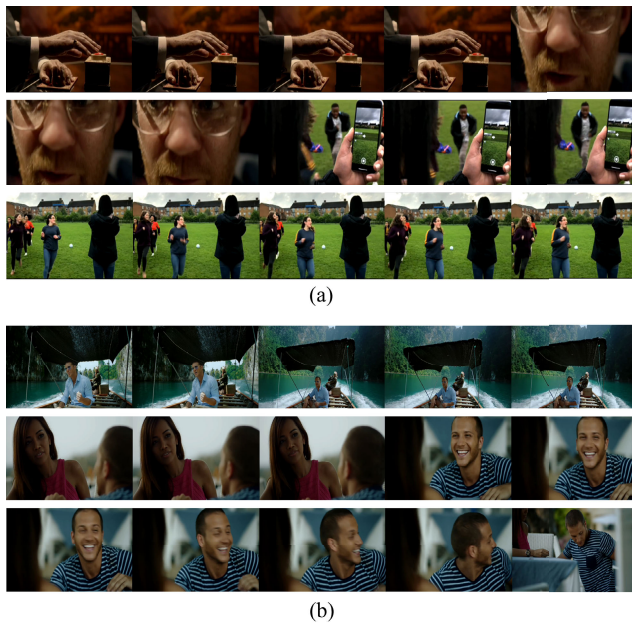


FIGURE 5. Examples of YouTube training videos with dynamic scene changes and different degrees of motions.

using a RL-based decision algorithm. The Bjontegaard-Delta rate (BD-rate) reductions are used for calculating the coding performance. In comparison, a negative BD-rate implies coding performance superior to the anchor (i.e., fixed GOP 16). We also report the encoding time, because the proposed technique conducts encoder-optimization. It is measured under the CPU setting.

Table 1 displays the improved R-D performance of the proposed technique for the luma component in YouTube videos. The proposed technique provides significantly improved

coding gains of more than 4.8% in BD-rate reductions on average. In Table 1, the fixed GOP 32 provided only slightly improved coding gain of approximately 2.4%. The fixed GOP 8 and the fixed GOP 64 even displays a coding loss about 9.6% and 13.6%, respectively. Furthermore, the proposed technique provides coding performance of approximately 7.4% and 11.5% higher than Chen et al. and Huang et al., respectively. The encoding time is only slightly increased by approximately 1.0 % compared with the anchor. The complexity of the RL decision is substantially lower than the video encoding.

The coding efficiency of the proposed technique varies by different types of the test video sequences. We observe that GOP size larger than 32 provides comparable performance to the proposed technique in static videos without scene change because our RL model predicts a GOP size larger than 32 for almost all of the videos in the statics video categories. The proposed technique exhibits -3.4% coding performance outperforming any fixed sized GOP. Chen et al. and Lee et al. [22] provide the improved coding gains of 1.2% and 0.5% respectively, whereas Huang et al., Garcia et al., and Li et al. provide degraded coding performance of approximately 7.8%, 0.3% and 2.3%, respectively.

In dynamic sequences, the proposed technique provides the highest coding performance among the compared methods. In the first group with one scene change, the proposed technique exhibits a significantly improved coding gain of approximately 7.9% on average whereas the fixed GOP 8 and 32 display coding losses about 14.0% and 4.0%, respectively. In the second group with more than one scene change, the proposed technique provides a coding gain of approximately 5.0% on average. Garcia et al. could provide a comparable coding gain of approximately 9.0% in the first group and 2.5% in the second group.

Lee et al. [22] also presents a minor coding gain of approximately 2.5% in the first group and 1.4% in the second group. However, other comparison methods, [18], [38], and [52], degrade the coding performance about 6.3%, 5.5% and 3.2% in dynamic video categories. Especially, both [18] and [38] incur substantial coding loss *e.g.* in the 34-th video of the second group. The proposed technique adapts to the video and prevents such coding loss.

The superior coding performance of the proposed technique is also displayed in Table 2. The proposed technique provides reliable coding gains in different resolutions of test videos in CTC. The average coding performance is 2.1%. In comparisons, Chen et al. yield coding gains of 0.3%. However, the coding performance is degraded in fixed GOP 8 and Huang et al.. There was a significant coding loss in the “DaylightRoad2” sequence. Since the CTC sequences contain only few scene transitions, the fixed GOP 32 also provides comparable coding gains of approximately 0.7%.

We present several video clips for which the proposed technique conducts accurate prediction of the adaptive GOP. Fig. 6 illustrates results in which the proposed technique predicts the same trajectory of the ground-truth, thus leading to the highest coding performance. Fig. 6 (a) presents the result of a static video referred to as “Sequence 15” in Table 1. “Sequence 15” has no scene change, but contains rather huge motion change since it is a video of playing tennis. Our technique predicts a GOP size of 32, providing a coding gain about 3.4%. However, Li et al. predict three sub-divided GOPs with sizes of 16,16 and 32. In Fig. 6 (b) and (c), our technique predicts the smaller sizes of GOPs because the videos contain more than one scene change. Fig. 6 (b) corresponds to “Sequence 28” in Table 1 in which the proposed technique achieves high coding performance. In comparisons, [51] provides degraded coding performance about 4.7% as a result of the wrong prediction to GOPs 16,16, and 32. Fig. 6 (c) shows “Sequence 37”, which contains two scene changes at the moments of f_{36} and f_{61} . It is shown that the proposed technique predicts the GOP to the five sub-divided GOPs with sizes of 32 and 8 and yields 3.2% coding gain over the anchor.

Fig. 7 displays the results in which the prediction is mismatched with the ground-truth. In Fig. 7 (a), sample videos of “Sequence 14” do not contain a scene transition, and the proposed technique selects the size of GOP as 64 for a coding. However, the best performance is achieved when using two 32-sized GOPs, and the proposed technique underperformed about 3.5% compared to Chen et al.. It seems that the coding performance of smaller GOP sizes is higher because Fig. 7 (a) contains irregular motion of the foreground object. Fig. 7 (b) displays a similar case in which our technique incurs some coding loss in $f_{28} \sim f_{64}$. We argue that the coding loss would occur because the proposed technique considers frame-level temporal dynamics such as scene changes to determine the size of the GOP rather than object-level factors such as textures and local motions. This loss may be alleviated by a block-level adaptive GOP decision at the expense of

computational complexity and coding delays, which will be investigated in the future research.

The complexity of the tested methods is presented in Table 1, which presents only slight increase of encoding measurement time. While the proposed network may bring a slight increase in encoding time, the process of selecting GOP is conducted only once (depth=0) to a maximum of 7 times (depth=3), making the increase negligible compared to encoding the entire 64 frames.

C. DISCUSSION

In this subsection, we show several variants of the proposed technique to discuss the performance and conduct ablation studies. The test videos are from YouTube dataset, in which the proposed technique provides 4.1 % BD-rate reductions in Table 1.

1) NETWORK ARCHITECTURE

We present the coding performance when using different network structures of the DQN in Fig. 4. Specifically, Table 3 presents the performance with the different number of streams as compared to the anchor. We recall that we used the two-stream network using three concatenated frames. In this comparison, we use an one-stream network architecture while keeping the same number of layers and filters as described in IV-D. We clearly see that the coding efficiency of the two-stream network is higher than the one-stream network.

TABLE 3. The BD-rate (in the unit of %) reduction of the proposed algorithm with different network structure in YouTube test videos.

Network structure	# of frames	Anchor : Fixed GOP 16
		Y BD-rate (%)
One-stream	3	-1.0%
	8	-1.2%
Two-stream	3	-2.9%
	8 (proposed)	-4.8%

TABLE 4. The BD-rate (in the unit of %) reduction of the proposed algorithm in YouTube test videos as compared with various deep learning methods.

Method		Anchor : Fixed GOP 16
		Y BD-rate (%)
GOP Classification	VGG-16 [57]	3.3%
Temporal segmentation	PySceneDetect [58]	-0.5%
	CES [51]	-2.7%
RL	$\gamma = 0$	3.9%
	$\gamma = 0.5$	-1.7%
	$\gamma = 0.99$	-4.8%

In Table 3, we also test a scenario when the networks use the different number of inputs. The proposed network uses three input frames representing two key frames and the middle frame at each node, which follows a coding order in the current GOP. In the comparisons, we attempted to use eight frames to enable a network to see more frames in the GOP even though it might cause some frame delays

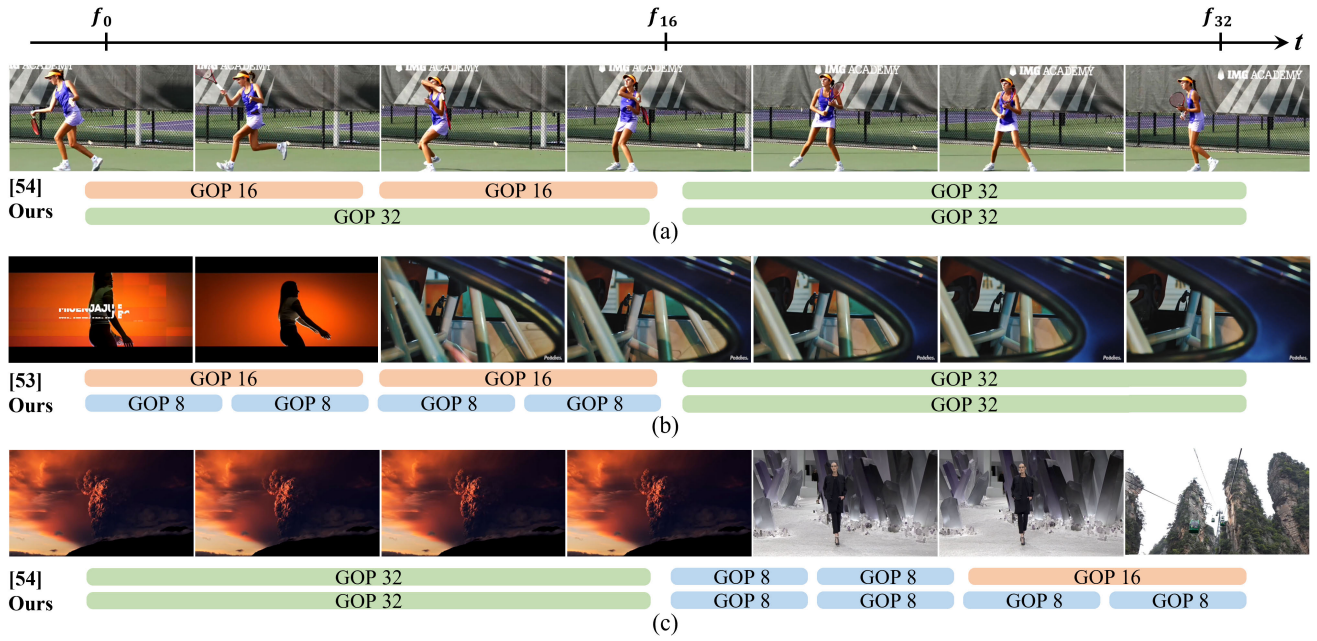


FIGURE 6. Sample videos for which the proposed technique conducts accurate prediction. The proposed technique predicts the same results as the ground-truth. The video in (a) is a static video with large motion. The videos in (b) and (c) are dynamic videos. The videos (a)~(c) correspond to the sequence numbers 15, 28, and 37 in Table 1, respectively.

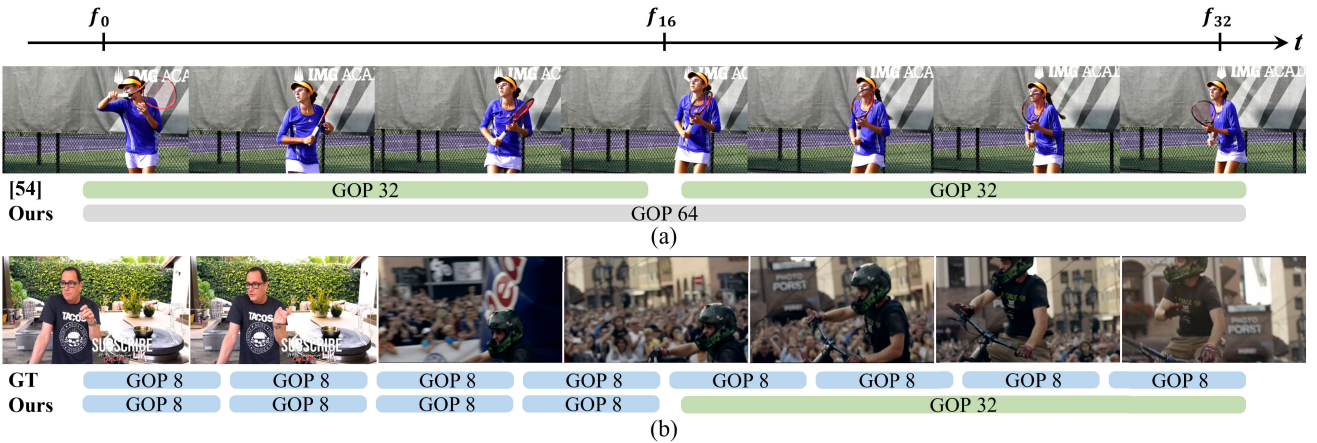


FIGURE 7. Sample videos for which the proposed technique conducts inaccurate prediction. GT indicates the best performing GOP structure. The videos in (a) and (b) correspond to the sequence numbers 14 and 40 in Table 1, respectively.

for coding. It is shown that the number of the inputs would not impact coding performance much. The BD-rate reduction is 1.2% when the one-stream network uses eight input frames, which is slightly higher than using three input frames. The highest performance is achieved with the two-stream network using three inputs.

2) DECISION SCHEMES

We conduct several experiments to compare various decision schemes. The results are presented in Table 4.

First, we evaluate the performance of classification-based decision scheme that could be an alternative to the proposed

RL-based decision scheme. The classification method utilizes an one-hot encoding of labels using VGG-16 architecture [57]. A classifier is trained with the supervision of the ground-truth, and the decision is made to output the probabilities. In our experimental setting, because we set the period of an I-frame to 32 frames, there are five possible adaptive GOP structures. Each GOP structure corresponds to a possible tree. Accordingly, we train a classifier to solve the multi-class classification problem with five labels. As shown in Table 4, the classification method exhibits degraded coding performance of approximately 3.3% whereas the proposed technique provides a superior coding gain of approximately

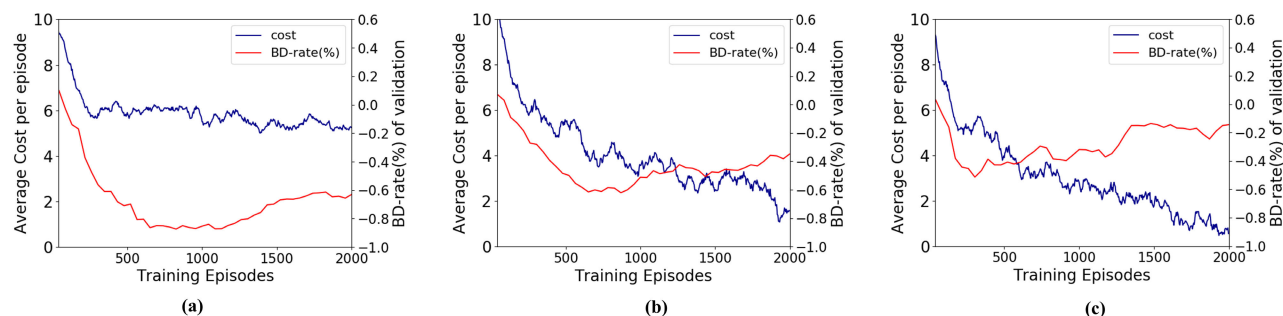


FIGURE 8. The plots show average cost per episode and BD-rate(%) of validation videos during training with (a) $\gamma=0.99$, (b) $\gamma=0.5$ and (c) $\gamma=0$.

4.8%. We observe that the RL-based scheme attempts to approximate the optimal solution at each node to minimize the expected costs even though it does not obtain the optimal tree structure in every scenario. However, the classification method lacks such an optimization scheme, so it often incurs significant coding loss when the prediction fails.

Second, the proposed technique is compared with a preprocessing scheme using content-aware scene detection [58] and contextual event segmentation (CES) [51]. The proposed technique is applied during encoding, in which it uses only the reference frames and the current frames. In comparison, the preprocessing is applied to determine the sizes of GOPs beforehand, and the frames are encoded with the decision. For this test, we first obtain several key frames from the scene detector and then generate groups of temporally homogeneous frames to create GOP structures. As shown in Table 4, the coding performance of [58] is approximately 0.5% and [51] is 2.7%. The preprocessing tends to yield an accurate temporal segmentation but has no considerations in R-D optimization.

We adjust the γ values in Eq. (2) and present the results in Table 4. The coding gain is improved with a higher gamma value. It achieves BD-rate reduction of approximately 4.8% for $\gamma = 0.99$ and 1.7% for $\gamma = 0.5$. For $\gamma = 0$, the degraded performance is shown about 3.9% since each node makes a greedy decision by taking only the present cost into consideration. Furthermore, a smaller γ value is vulnerable to the overfitting problem because it considers only the immediate cost. Fig. 8 displays the average cost per episode and the coding performance during validation. The blue line and red line exhibit the average cost per episode and the BD-rate savings, respectively. As shown in Fig. 8, the costs decrease as the episodes repeat. The BD-rates also decrease to a certain training point. However, the curves increase in Fig. 8 (b) and (c) because of excessive training episodes whereas those in Fig. 8 (a) are reliable.

VI. CONCLUSION

In this paper, we proposed an RL-based GOP decision algorithm with a tree-based framework to improve coding efficiency. The proposed technique utilized a binary tree to

divide a given video into several GOPs and automatically determined their sizes. For this, we defined a new policy representation and obtained near-optimal global policy by combining the local policies in each sub-tree. The proposed method employed a deep Q-network with a modified architecture to capture temporal correlation between frames and iteratively trained the Q-network using effective learning schemes. Owing to the tree-based RL framework, the proposed technique could efficiently manage complicated temporal dependencies in high-dimensional videos. The proposed technique demonstrated significantly improved coding efficiency compared with the state-of-the-art GOP selection algorithms. It was shown in experimental results that the proposed technique exhibited -4.8% and -2.1% coding gains in YouTube test videos and CTC test videos compared with fixed GOP 16. In the future work, we will study how this RL-based framework can be applied to block-level optimization for various video coding applications.

REFERENCES

- [1] B. Bross, J. Chen, J. R. Ohm, G. J. Sullivan, and Y. K. Wang, "Developments in international video coding standardization after AVC, with an overview of versatile video coding (VVC)," *Proc. IEEE*, vol. 109, no. 9, pp. 1463–1493, Jan. 2021.
- [2] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.
- [3] J. K. Lee, N. Kim, S. Cho, and J.-W. Kang, "Convolution neural network based video coding technique using reference video synthesis," in *Proc. Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf.*, Nov. 2018, pp. 12–15.
- [4] L. Zhao, S. Wang, X. Zhang, S. Wang, S. Ma, and W. Gao, "Enhanced motion-compensated video coding with deep virtual reference frame generation," *IEEE Trans. Image Process.*, vol. 28, no. 10, pp. 4832–4844, Oct. 2019.
- [5] J.-K. Lee, N. Kim, S. Cho, and J.-W. Kang, "Deep video prediction network-based inter-frame coding in HEVC," *IEEE Access*, vol. 8, pp. 95906–95917, 2020.
- [6] Z. Zhao, S. Wang, S. Wang, X. Zhang, S. Ma, and J. Yang, "Enhanced bi-prediction with convolutional neural network for high-efficiency video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 11, pp. 3291–3301, Nov. 2019.
- [7] H. Choi and I. V. Bajic, "Affine transformation-based deep frame prediction," *IEEE Trans. Image Process.*, vol. 30, pp. 3321–3334, 2021.
- [8] B. Li, J. Xu, D. Zhang, and H. Li, "QP refinement according to Lagrange multiplier for high efficiency video coding," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2013, pp. 477–480.

- [9] B. Li, D. Zhang, H. Li, and J. Xu, "QP determination by lambda value," Joint Video Experts Team (JVET) ITU-T SG, Tech. Rep. JCTVC-I0426, 2012.
- [10] Y. Gao, C. Zhu, S. Li, and T. Yang, "Source distortion temporal propagation analysis for random-access hierarchical video coding optimization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 2, pp. 546–559, Feb. 2019.
- [11] Y. Gao, C. Zhu, and S. Li, "Hierarchical temporal dependent rate-distortion optimization for low-delay coding," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2016, pp. 570–573.
- [12] J.-W. Kang, Y.-Y. Lee, C.-S. Kim, and S.-U. Lee, "Coding order decision of B frames for rate-distortion performance improvement in single-view video and multiview video coding," *IEEE Trans. Image Process.*, vol. 19, no. 8, pp. 2029–2041, Aug. 2010.
- [13] S. Li, C. Zhu, Y. Gao, Y. Zhou, F. Dufaux, and M.-T. Sun, "Lagrangian multiplier adaptation for rate-distortion optimization with inter-frame dependency," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 117–129, Jan. 2016.
- [14] S. Li, C. Zhu, Y. Gao, Y. Zhou, F. Dufaux, and M.-T. Sun, "Inter-frame dependent rate-distortion optimization using Lagrangian multiplier adaption," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jun. 2015, pp. 1–6.
- [15] X. Zhao, J. Liu, G. Hu, and L. Zhang, "Adaptive key-frame selection based on image features in distributed video coding," in *Proc. Int. Conf. Comput. Problem-Solving (ICCP)*, Oct. 2013, pp. 1–4.
- [16] H. Li, B. Li, and J. Xu, "Rate-distortion optimized reference picture management for high efficiency video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1844–1857, Dec. 2012.
- [17] Y. Gao, C. Zhu, S. Li, and T. W. Yang, "Temporally dependent rate-distortion optimization for low-delay hierarchical video coding," *IEEE Trans. Image Process.*, vol. 26, no. 9, pp. 4457–4470, Sep. 2017.
- [18] C. Chen, F. Ding, and D. Zhang, "Perceptual hash algorithm-based adaptive GOP selection algorithm for distributed compressive video sensing," *IET Image Process.*, vol. 12, no. 2, pp. 210–217, Feb. 2018.
- [19] V. Poobalasingam and E. Izquierdo, "Steadiness analysis for optimal GOP size selection in HEVC," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 799–803.
- [20] S. Zhu and Z. Xu, "Spatiotemporal visual saliency guided perceptual high efficiency video coding with neural network," *Neurocomputing*, vol. 275, pp. 511–522, Jan. 2018.
- [21] Y. Gao, C. Zhu, S. Li, and T. Yang, "Layer-based temporal dependent rate-distortion optimization in random-access hierarchical video coding," in *Proc. IEEE 18th Int. Workshop Multimedia Signal Process. (MMSP)*, Sep. 2016, pp. 1–6.
- [22] N. K. Jung-Kyung Lee and J.-W. Kang, "Rate-distortion optimized temporal segmentation using reinforcement learning for video coding," in *Proc. Asia Pacific Signal Inf. Process. (APSIPA)*, Dec. 2021, pp. 1–4.
- [23] D. Liu, D. Zhao, X. Ji, and W. Gao, "Dual frame motion compensation with optimal long-term reference frame selection and bit allocation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 3, pp. 325–339, Mar. 2010.
- [24] Z. Pan, P. Jin, J. Lei, Y. Zhang, X. Sun, and S. Kwong, "Fast reference frame selection based on content similarity for low complexity HEVC encoder," *J. Vis. Commun. Image Represent.*, vol. 40, pp. 516–524, Oct. 2016.
- [25] S. Wang, S. Ma, S. Wang, D. Zhao, and W. Gao, "Rate-GOP based rate control for high efficiency video coding," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 6, pp. 1101–1111, Dec. 2013.
- [26] L. Li, B. Li, D. Liu, and H. Li, " λ -domain rate control algorithm for HEVC scalable extension," *IEEE Trans. Multimedia*, vol. 18, no. 10, pp. 2023–2039, Jul. 2016.
- [27] K. Panagidi, C. Anagnostopoulos, and S. Hadjiefthymiades, "Optimal grouping-of-pictures in IoT video streams," *Comput. Commun.*, vol. 118, pp. 185–194, Mar. 2018.
- [28] H. Schwarz, D. Marpe, and T. Wiegand, "Analysis of hierarchical B pictures and MCTF," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2006, pp. 1929–1932.
- [29] T. Yang, C. Zhu, X. Fan, and Q. Peng, "Source distortion temporal propagation model for motion compensated video coding optimization," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2012, pp. 85–90.
- [30] E.-H. Yang and X. Yu, "Rate distortion optimization for H.264 interframe coding: A general framework and algorithms," *IEEE Trans. Image Process.*, vol. 16, no. 7, pp. 1774–1784, Jul. 2007.
- [31] V. Poobalasingam, E. Izquierdo, S. G. Blasi, and M. Mrak, "Optimised selection of structure of pictures for video coding," in *Proc. IEEE 18th Int. Workshop Multimedia Signal Process. (MMSP)*, Sep. 2016, pp. 1–4.
- [32] B. Zatt, M. Porto, J. Scharcanski, and S. Bampi, "GOP structure adaptive to the video content for efficient H.264/AVC encoding," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2010, pp. 3053–3056.
- [33] Y. Sakamoto, R. Yokoyama, M. Takeuchi, Y. Matsuo, and J. Katto, "Improvement of H.265/HEVC encoding for 8K UHD TV by GOP size and prediction mode selection," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2019, pp. 1–2.
- [34] K. R. Vijayanagar and J. Kim, "Dynamic GOP size control for low-delay distributed video coding," in *Proc. 18th IEEE Int. Conf. Image Process.*, Sep. 2011, pp. 157–160.
- [35] K. DinhQuoc, X. HoangVan, and B. Jeon, "An iterative algorithm for efficient adaptive GOP size in transform domain Wyner-Ziv video coding," in *Proc. Pacific-Rim Symp. Image Video Technol.* Cham, Switzerland: Springer, 2011, pp. 347–358.
- [36] J.-R. Ding, J.-F. Yang, and J.-K. Lin, "Motion-based adaptive GOP algorithms for efficient H.264/AVC compression," in *Proc. 9th Joint Int. Conf. Inf. Sci.*, 2006, pp. 49–52.
- [37] J. Ascenso, C. Brites, and F. Pereira, "Content adaptive Wyner-ZIV video coding driven by motion activity," in *Proc. Int. Conf. Image Process.*, Oct. 2006, pp. 605–608.
- [38] T. N. T. Huong, H. P. Cong, T. V. Huu, and X. H. Van, "Artificial intelligence based adaptive GOP size selection for effective Wyner-Ziv video coding," in *Proc. Int. Conf. Adv. Technol. Commun. (ATC)*, Oct. 2018, pp. 120–124.
- [39] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Trans. Autom. Control*, vol. 42, no. 5, pp. 674–690, May 1997.
- [40] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.
- [41] J.-H. Hu, W.-H. Peng, and C.-H. Chung, "Reinforcement learning for HEVC/H.265 intra-frame rate control," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.
- [42] L.-C. Chen, J.-H. Hu, and W.-H. Peng, "Reinforcement learning for HEVC/H.265 frame-level bit allocation," in *Proc. IEEE 23rd Int. Conf. Digit. Signal Process. (DSP)*, Nov. 2018, pp. 1–5.
- [43] M. Zhou, X. Wei, S. Kwong, W. Jia, and B. Fang, "Rate control method based on deep reinforcement learning for dynamic video sequences in HEVC," *IEEE Trans. Multimedia*, vol. 23, pp. 1106–1121, 2021.
- [44] P. Helle, H. Schwarz, T. Wiegand, and K.-R. Müller, "Reinforcement learning for video encoder control in HEVC," in *Proc. Int. Conf. Syst., Signals Image Process. (IWSSIP)*, May 2017, pp. 1–5.
- [45] C.-H. Chung, W.-H. Peng, and J.-H. Hu, "HEVC/H.265 coding unit split decision using deep reinforcement learning," in *Proc. Int. Symp. Intell. Signal Process. Commun. Syst. (ISPACS)*, Nov. 2017, pp. 570–575.
- [46] M. Jamali, S. Coulombe, and H. Sadreazami, "CU size decision for low complexity HEVC intra coding based on deep reinforcement learning," in *Proc. IEEE 63rd Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2020, pp. 586–591.
- [47] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2001.
- [48] X. Xu, C. Liu, S. X. Yang, and D. Hu, "Hierarchical approximate policy iteration with binary-tree state space decomposition," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 1863–1877, Dec. 2011.
- [49] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.
- [50] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, pp. 253–279, Jun. 2013.
- [51] A. G. del Molino, J.-H. Lim, and A.-H. Tan, "Predicting visual context for unsupervised event segmentation in continuous photo-streams," in *Proc. 26th ACM Int. Conf. Multimedia*, Oct. 2018, pp. 10–17.
- [52] B. Li, J. Han, and Y. Xu, "Adaptive GOP size decision for multi-pass video coding based on hidden Markov model," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 1575–1579.
- [53] (2020). *VVC Test Model (VTM) Reference Software 16.0*. [Online]. Available: <https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftwareVTM/-tree/VTM-16.0>

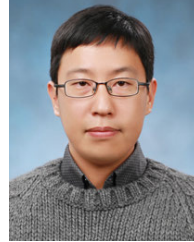
- [54] F. Bossen, J. Boyce, X. Li, V. Seregin, and K. Sühring, *JVET Common Test Conditions and Software Reference Configurations for SDR Video*, document 16, Joint Video Experts Team (JVET) of ITU-T SG, 2019.
- [55] (2023). *YouTube Test Dataset for Adaptive GOP*. [Online]. Available: <https://github.com/icplab48/RL>
- [56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
- [57] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
- [58] PySceneDetect. (2020). *PySceneDetect: Intelligent Scene Cut Detection and Video Splitting Tool*. [Online]. Available: <https://pyscenedetect.readthedocs.io/en/latest/>



JUNG-KYUNG LEE received the B.S. and M.S. degrees in electronics engineering from Ewha Womans University, Seoul, Republic of Korea, in 2018 and 2020, respectively, where she is currently pursuing the Ph.D. degree with the Department of Electronic and Electrical Engineering. Her research interest includes video processing and compression.



NAYOUNG KIM received the B.S. and Ph.D. degrees in electronics engineering from Ewha Womans University, Seoul, Republic of Korea, in 2016 and 2022, respectively. Her research interests include video processing, computer vision, and deep learning.



JE-WON KANG (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 2006 and 2008, respectively, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 2012. He was a Visiting Researcher with Mitsubishi Electric Research Laboratories, Boston, MA, USA, in 2010, and Tampere University, Nokia Research Center, Tampere, Finland, in 2011. He was a Senior Engineer with the Multimedia Research and Development and Standard Team, Qualcomm Technologies, Inc., San Diego, CA, USA, from 2012 to 2014. He is currently an Associate Professor with Ewha Womans University, Seoul, where he is also the Head of the Information Coding and Processing Laboratory, Department of Electronics and Electrical Engineering. He has been an active contributor to the recent international video coding standards in JCT-VC, including high-efficiency video coding (HEVC) standard and the extensions to multiview videos, 3-D videos, and screen content videos. His current research interests include image and video processing and compression, computer vision, and machine learning.

• • •