Received April 14, 2021, accepted May 31, 2021, date of publication June 4, 2021, date of current version June 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3086566

RL-SPIHT: Reinforcement Learning-Based Adaptive Selection of Compression Ratios for 1-D SPIHT Algorithm

JIN SHIN AND HYUN KIM, (Member, IEEE)

Department of Electrical and Information Engineering and the Research Center for Electrical and Information Technology, Seoul National University of Science and Technology, Seoul 01811, South Korea

Corresponding author: Hyun Kim (hyunkim@seoultech.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korea Government (MSIT) under Grant 2020-0-01304, in part by the Development of Self-Learnable Mobile Recursive Neural Network Processor Technology, and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) through the Ministry of Education under Grant NRF-2019R1A6A1A03032119.

ABSTRACT In recent years, deep learning technologies have been actively used in various applications. In particular, networks trained using reinforcement learning (RL) are widely exploited for auxiliary tasks in various multimedia frameworks, including image restoration, image compression, and computer vision. Discrete wavelet transform (DWT) and set partitioning in hierarchical trees (SPIHT) are the representative lightweight compression methods that are most widely used for the purposes of frame memory compression and LCD overdrive. In precedent research, in order to improve the compression efficiency of DWT-SPIHT algorithms, the relative complexity of DWT coefficients is quantified, and when compressing DWT coefficients with the SPIHT algorithm, the compression ratio (CR) is adaptively allocated to the compression block according to the numerically expressed complexity. However, the SPIHT algorithm has the characteristic of resource limitation, resulting in the occurrence of remaining blocks, which cannot take advantage of allocating the adaptive CR. Moreover, since the equation expressing the block complexity that determines the CR of each block is obtained through machine learning-based linear regression, it lacks the capability to deal with a wide range of real-world images. To compensate for these drawbacks, this paper optimizes the compression efficiency of the 1-D DWT-SPIHT algorithm using the RL-based episodic auxiliary task. In detail, the proposed method optimally adjusts the proportion of CRs, which are adaptively selected for each block according to the DWT coefficient, through the episodic model trained with the RL algorithm. Consequently, the proposed method achieves an average improvement in peak signal to noise ratio (PSNR) of 2.18dB compared to the baseline 1-D DWT-SPIHT with the fixed compression ratio and 0.68dB compared to the precedent research.

INDEX TERMS Reinforcement learning, Deep Q-learning, image compression, discrete wavelet transform, set partitioning in hierarchical trees, compression efficiency, episodic auxiliary task.

I. INTRODUCTION

The use of various mobile multimedia devices supporting displays has increased along with the commercialization of media transmission services, such as web streaming [1]–[3]. As the display resolution has increased to enhance the quality of media services, the complexity and memory bandwidth for image processing have also increased, and accordingly, the power consumption has increased proportionately [4], [5]. In particular, the power consumption issue is more critical for battery-powered mobile devices [6]. To solve these problems in mobile devices, the use of frame memory compression (FMC) based on the embedded compression (EC) schemes has gained attention [7]–[11]. EC-based FMC, which has low latency and low internal power consumption, is a very effective technique for solving external memory bandwidth issues and reducing power consumption by being installed between the processing unit and external memory.

The EC schemes are mainly classified into transform-based and non-transform-based methods [12]–[14]. The non-transform-based schemes such as differential pulse code modulation-variable length coding and block truncation

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

coding are implemented with a very simple structure, but have relatively low compression efficiency and variable bit-stream length. On the other hand, the transform-based method can achieve a relatively high compression rate by lossy-compressing the coefficients of image transform such as discrete cosine transform (DCT) or discrete wavelet transform (DWT) through various quantization techniques [15], [16]. Among various transform-based EC schemes, the combination of DWT and set partitioning in hierarchical trees (SPIHT) has outstanding performance in terms of the trade-off of computational complexity and compression ratio (CR). In addition, it has the advantage of being able to generate bit streams using various fixed target CRs [17]–[20]. In particular, the combination of 1-D DWT and SPIHT [19] supports raster-scan processing, which is suitable for hardware-based FMC, and FMC based on the 1-D DWT-SPIHT can minimize both the operation delay and power consumption issues on mobile devices with only a few hardware resources. Although the 1-D DWT SPIHT structure shows a superior performance in terms of the trade-off between the compression efficiency and the computational complexity, the utilization of spatial correlation in the 1-D algorithm is difficult compared to that in the 2-D algorithm, resulting in significant compression loss. This drawback hinders the 1-D algorithm from utilizing an aggressive CR [14].

To address these problems, the preceding study of this paper [21] proposed a technique for adaptively determining the CR of the SPIHT algorithm according to the DWT coefficients, rather than applying the same CR to all coding blocks in the 1-D DWT-SPIHT. The DWT coefficients are well integrated into the low-pass band for relatively simple DWT blocks (i.e., blocks with low complexity). On the other hand, several coefficients still exist in the high-pass band for relatively complex DWT blocks. Motivated by this fact, the complexity of the DWT blocks is relatively quantified, and the CR appropriate for the complexity of each block is adaptively allocated. Consequently, this method can enhance the compression efficiency of 1-D DWT-SPIHT through relatively simple calculations. However, the preceding study [21] has a problem that the optimal performance cannot be derived because the approach in [21] formulates the correlation between the DWT coefficients and the loss due to compression by linear regression based on machine learning (ML) rather than deep learning (DL). Furthermore, SPIHT has a characteristic of resource limitation to satisfy the total target bit length (TBL), resulting in the occurrence of remaining blocks, which cannot take advantage of the adaptive CR allocation method. This indicates that the preceding study [21] has room for further improvement in terms of compression efficiency.

This study proposes a technique that significantly improves the compression efficiency of the 1-D DWT-SPIHT by adding an episodic auxiliary task that has been trained using reinforcement learning (RL) based on the distribution of the CRs adaptively selected in the previous study [21]. In the proposed method, the complexity distribution of all the blocks is obtained through the DWT process in the first stage. In the second stage, an auxiliary task that biases the CR distribution is performed as a single episode to determine the optimal CR for each block, and SPIHT is adaptively processed with the optimized CRs. Consequently, in this study, the distribution of CRs determined in the first stage is adjusted as close to an optimum as possible through the episodic model trained using RL while retaining the advantages of the preceding study [21] that adaptive selection is possible according to the complexity of the coding blocks. The experimental results show that the proposed method can improve the peak signal-to-noise ratio (PSNR) by 2.18 dB and 0.68 dB on average in comparison with the fixed CR method [20], which applies the same CR to all the blocks, and the previous study [21], respectively.

The remainder of this paper is organized as follows. Section 2 explains the DWT-SPIHT-based adaptive selective CR (ASCR) algorithm [21], which is the motivation of this study, and the background of RL. Section 3 describes the process of applying RL to the proposed RL-SPIHT, and Section 4 presents the experimental environment and results. Finally, Section 5 presents the conclusion.

II. BACKGROUND

This section explains the 1-D DWT-SPIHT compression method, which is the baseline of the RL-SPIHT, and the ML-based ASCR [21]. This section also explains the concept of RL and deep Q-networks (DQN) utilized in this study to optimize the CR distribution.

A. 1-D DWT-SPIHT

For the DWT-SPIHT combination [20], which is the typical transform-based EC technique, the SPIHT algorithm takes DWT coefficients as input and performs compression. DWT which is widely used in various ECs including the representative lossy image compression, JPEG2000, demonstrates a better performance than DCT, which is used in the conventional JPEG [15], [16]. As in the previous study [21], the method of synthesizing DWT in three levels by using a 1-D image pixel block (1×64) as the integer Le Gall 5/3 filter [18] is used in this study. If three-level DWT is applied to 1-D image blocks, the result can be shown in a binary tree form, as shown in Fig. 1. The coefficients are classified as L3, H3, H2, and H1 that constitute the binary tree structure. Each coefficient is represented as a bit plane containing a total of 64 bits. The SPIHT algorithm compresses bit planes transformed in this manner using a specified target CR. Although there are diverse methods for compressing DWT coefficients using SPIHT, the 1-D block-based pass-parallel SPIHT (BPS), which has a hardware-friendly structure and can achieve a high throughput, is used in this study [20]. In general, SPIHT uses three data structures to perform operations on the bit planes for the DWT coefficients. The three data structures are the list of insignificant sets, the list of insignificant pixels, and the list of significant pixels. As the BPS reconstructs these data structures as the

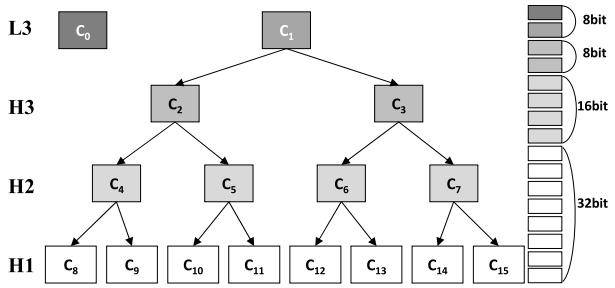


FIGURE 1. Number of bits containing correlation between the DWT coefficients. This figure is modified from Fig. 1 of [21], and the decomposition of DWT is three.

insignificant set pass (ISP), the insignificant pixel pass (IPP), and the refinement pass (RP), both the encoder and decoder can achieve a high processing speed [20]. This type of 1-D DWT-SPIHT shows an excellent performance in terms of the trade-off between the compression efficiency and the computational complexity. However, it has the drawback of an inferior compression performance compared with that of 2-D DWT-SPIHT because it cannot utilize redundancy in the vertical direction.

B. ADAPTIVE SELECTION OF COMPRESSION RATIOS

ASCR was proposed to remedy a relatively lower compression efficiency of 1-D DWT-SPIHT compared with 2-D DWT-SPIHT [21]. The existing 1-D DWT-SPIHT [20] encodes the DWT coefficients in all image blocks with the same fixed CR. Therefore, if the image block has a complex configuration, a considerable amount of information remains in the high-pass band of DWT coefficients, and important data may be lost during the SPIHT process. On the other hand, if the image block is simple, most of the information exists in the low-pass band of DWT coefficient; hence the amount of data loss is relatively small even if the same CR is applied. To take advantage of the difference in complexity of each block, a previous study [21] proposed a technique that adaptively determines the CR of SPIHT according to the DWT coefficients of each block. In other words, complex blocks are compressed slightly, and simple blocks are compressed aggressively while maintaining the total TBL. Using the image of size 256×256 as a reference, the DWT coefficient blocks are divided into 1,024 1×64 pixel blocks. The relative cost of the complexity for all the blocks must be expressed numerically to determine an appropriate CR for each block. Therefore, the complexity of all the blocks is expressed as a cost for the pixels included in the high-pass band (i.e., from H3 to H1) using the following equation:

$$Cost_{DWT} = \sum_{p=H1}^{H3} TRUNC(\log_2 p), \tag{1}$$

where *TRUNC()* function is an operation that discards the decimal point. Based on the fact that the mean squared error (MSE), which is the difference between the pixels of the restored image and the pixels of the original image, is dependent on the complexity of the 1-D DWT block, ASCR proved the correlation between the MSE and $\log(1 + CostDWT)$ with linear regression, and the correlation between these two variables is formulated as follows:

$$log_2\,(1 + MSE(CR_i)) \cong a\ log_2(1 + Cost_i) - bCR_i + c \tag{2}$$

where a, b, and c are constants obtained from the linear regression. Additionally, by applying the optimization technique, the selection of an adaptive CR suitable for the corresponding block can be obtained through the following equation:

$$CR = CR_{Fixed} + \frac{a}{b}(log_2\,(1 + Cost_{DWT}) - S), \tag{3}$$

where S, the average of $\log_2(1+Cost_{DWT})$, is calculated as follows:

$$S = \frac{1}{N}\sum_{i=0}^{N-1} log_2\,(1 + Cost_{DWT})\,. \tag{4}$$

The adaptive CR determined by (3) is sequentially applied to each block before SPIHT is performed.

As the current frame is almost similar to the previous frame in video sequences [21], ASCR uses the average of $\log_2(1+Cost_i)$ of the previous frame to predict the complexity of the corresponding blocks relative to the entire frame. However, since slight differences exist between the consecutive frames, in order to match the TBL exactly, there is a limitation that some blocks within the frame need to be compressed at a certain CR without considering the complexity. In other words, if the remaining blocks are compressed using the most aggressive CR due to the lack of bit length resources, a significant PSNR loss may occur. Furthermore, since ASCR expresses the correlation between the MSE and cost using a simple ML-based linear regression presented in (2) instead of a DL-based approach, optimization was not achieved. Therefore, the proposed study aims to solve this problem by creating an episode model based on the linear regression relationship proven in ASCR and applying the RL model.

C. REINFORCEMENT LEARNING

RL is an area of DL in which the optimal control theory has been systematized for the Markov decision process (MDP), which simultaneously considers the theory of dynamic system including observations, actions, and purposes. RL is a dual-structured model composed of the environment and the agent that controls the environment [22]–[30]. The purpose of RL is to enable the policy of the agent to select an optimal action for a given state in the environment. Since RL is fundamentally based on the dynamic environment, the concept of time exists in RL. In addition, RL can be divided into episodic tasks, which have a finite state space, and continuous tasks, which have an infinite state space, according to the characteristics of the episodes in the environment. Considering the characteristics of the application to which RL is applied, this paper only deals with episodic tasks.

Similar to a single episode, an episodic task can be expressed as a discrete timeline. In general, episodic tasks

include a start state ($t = 0$) and a terminal state ($t = \mathrm{T}$). All the states included in the state space, except for the terminal state, choose an action. Here, selecting the optimal action indicates that this action will result in the greatest reward from the environment in the action space. It should be noted that the concept of reward in RL considers not only the current state but also the cumulative rewards that will be received in the future. The reward G_t to be received at time t is expressed using the following equation:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} \cdots + R_T. \tag{5}$$

The current reward cannot be exactly determined using this equation until the episode is completed. Therefore, a value function that directly estimates the agent value for the state at time t is introduced, which can be expressed using the following equation:

$$v(s) = E_\pi \left[\sum_{t=0}^{T} \gamma^t R_{t+1} \mid S_t = s \right], \tag{6}$$

where $v(s)$ denotes the value of the action for the state at time t in the state space. γ, which is added to (6) on the left side of the condition, generally represents the discount rate for the reward in the next state in RL. γ has a value greater than 0 and less than 1. This factor represents the magnitude of the correlation between the states at times t and t + 1. The greater the influence of the action selected for the state at time t on the state at time t + 1, the closer the value of γ becomes to 1. $v_\pi(s)$ denotes the value function, which takes policy π into consideration, which is responsible for estimating the action for the agent. $v_\pi(s)$ is represented by the Bellman expectation equation indicating the relationship between the value functions of the state at each time t in RL, and can be expressed as follows:

$$v_\pi(s) = E_\pi [R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]. \tag{7}$$

This equation denotes the expectation for the total rewards that will be received if the policy at time t is maintained, and consequently, it can be the criterion for the agent to determine which policy is better.

This algorithm updates the neural network based on the Q-function, which is based on the state-value function described in the previous paragraphs. The Q-function is also called the action-value function, and it represents the maximum aggregate reward that allows each of the possible actions that can be taken in a certain state to have a value function. Consequently, it is possible to determine which action should be selected without having to examine the value function of the next state. The Q-function can be expressed as follows:

$$q_\pi(s, a) = E_\pi [R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]. \tag{8}$$

Although the policy and the methodology for updating this policy are important, RL heavily depends on the concept of states. Therefore, it is very important to determine the states when designing an environment model. In general, a state semantically conveys information to the learner about the state of the environment at a specific time, and the state becomes the output of preprocessing, which is a basic component of the environment. When designing an MDP, it should be considered that S_{t+1}, the state at time t + 1, is dependent on S_t, the state at time t. In other words, S_{t+1} must be obtained using the values altered by the action that the agent selected in S_t. Training is impossible if this dependency is not established. Therefore, the key point of this study is to determine two main factors, the preprocess system for determining the optimal CR of SPIHT according to the complexity of the DWT blocks and the states, which are the components of MDP. The RL algorithm used to solve the proposed episodic environment is described further in Section II-D.

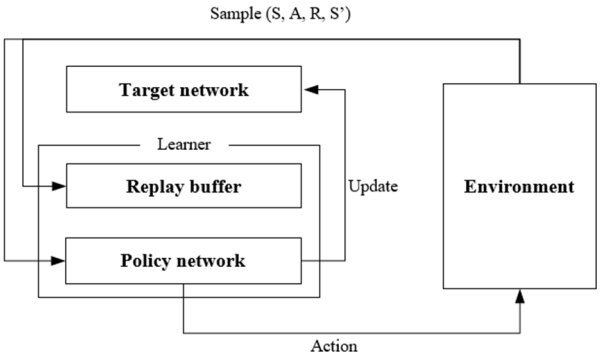


FIGURE 2. Workflow diagram of DQN. (S, A, R, S') of the sample indicate state, action, reward, and next state, respectively. Learner represents essential factors required for training.

D. DQN: DEEP Q-NETWORKS

DQN [22] is an algorithm that was used in AlphaGo [23], a notable example of RL. Studies on DQN have been actively conducted [24]–[30]. DQN can be trained in a model-free method, which does not require a detailed explanation of components, such as the state transition probability in MDP. Moreover, the training process can be simply illustrated using a workflow diagram, as shown in Fig. 2. The replay buffer and target network in the diagram are used for training, and the environment and policy network are used during the actual inference process. The policy network is a Q-function (Q-network) having a Q-value as an output. The environment, which is commonly used by the two processes, has the role of preprocessing the state and reward. For example, in [22], four consecutive frames are first stacked and gray-scaled, and the images are cropped to the size (84 × 84) suitable for the GPU to observe and identify the movements in the frames of the Atari game. Then, the states are preprocessed through this operation. The replay buffer stores the samples received from the environment during training in the form of tuples, and it recycles them using the uniform random sampling when updating. By doing so, the replay buffer solves the sparse reward and high sample dependency issues, which are the problems of the episode-based RL. This data

usage sequentially inserts images of different characteristics in the episodes to generalize the network, thereby facilitating effective training. The target network has the same shape as the policy network and has a rule for copying the weight parameters of the policy network at uniform intervals. This prevents the tendency of errors to diverge or vibrate continuously due to the continuous change in the purpose as the policy network is trained at each step. The q-value of the Q-network and the expectation of the q-value are required to update the Q-network by calculating the loss. The q-value can be expressed using (8), and the expectation of the q-value can be expressed using the following Bellman equation:

$$Q_\pi\,(s, a) = RWD\,(s, a) + \gamma \max_a Q_\pi(s', a). \tag{9}$$

Both s and a of $RWD\,(s, a)$ follow the conditional expression of $S_t = s$ and $A_t = a$, and they represent the reward function for action a performed in the state s. The last term in (9) represents the q-value of the next state s', and the target Q-network is used to calculate this value.

Algorithm 1 Episodic Environment

Input: Direction
Constant: θ, N, Max_{step}, $\text{signal}_{\text{neg}}$, $\text{signal}_{\text{pos}}$
Variable: CR_n, β, step, $\text{count}_{\text{pos}}$, $\text{count}_{\text{neg}}$, done_{epi}, $\text{done}_{\text{step}}$
Output: β

```
Get CR_n from ASCR
Initialize β = 0, step = 0
count_pos, = 0, count_neg = 0, done_epi = False
while done_epi == False do
    step++
    done_step == False
    while done_step == False do
        if (direction == signal_neg) count_neg + +
        if (direction == signal_pos) count_pos + +
        β = ((count_neg * − 1) + count_pos) * θ
        for i = 1 to N
                CR'_i = CR_i + β
                if ([CR'_i] == [CR_i]) done_step = True
                end for
    if (step > Max_step) done_epi = True
    else if (Inference Condition) done_epi = True
    end while
end while
return β
```

Algorithm 2 Deep Q-Learning With Experience Replay

```
Initialize replay memory D to Capacity C
Initialize action-value function Q with random weights
Initialize Env_n with N single frames
for episode = 1, M do
    Initialize sequence s_0 = {D_0, R_0, β_0} from the first stage
    for t = 1, T do
    With probability ϵ, select a random action
    Otherwise, select a_t = max_a Q*(Ø(s_t), a; θ)
    Execute action a_t in two-stage framework
    Observe reward r_t and s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in D
end for
Sample random mini-batch or transitions
(s_t, a_t, r_t, s_{t+1}) from D
Set y_i = { r_j
          { r_j + ϒ max_{a'} Q(Ø_{j+1}, a'; θ)
Perform a gradient descent step on (y − Q(Ø_j, a_j; θ))^2
Change environment sequentially
Save the Q-network according to the result of inference
end for
```

III. PROPOSED METHODS

This section describes the principle of the proposed RL-SPIHT including episodes with the built-in preprocessing procedure in which ASCR [21] is applied in order to optimize the CRs of SPIHT.

A. OVERVIEW OF THE PROPOSED RL-SPIHT

This subsection explains the overall operation of RL-SPIHT. Fig. 3 shows the overall workflow of the proposed RL-SPIHT. RL-SPIHT is primarily divided into two stages. In the first stage, to complete the preprocessing procedure for the second stage, DWT is performed to collect the costs for the complexity of each block, and CRs are allocated to each block according to the ASCR method [21]. In the second stage, through the auxiliary task that has been trained using the proposed RL, the CR value is biased according to the complexity of each DWT block, and SPIHT is performed with the optimal CRs.

In Stage-1, the 1-D block-level DWT procedure is performed on the input frames received in the raster scan order, and the transformed blocks are stored in Buffer 1. Using the DWT blocks stored in Buffer 1, the costs of the pixels included in the high-pass band of each block are calculated using (1). These values are then transformed into the $log(1 + cost)$ format for the next procedure, and the average cost of all the blocks in single frame is calculated using (4). Subsequently, an appropriate CR (i.e., 3/16, 4/16, 5/16, 6/16, 7/17, 8/16, 9/16 in this study) for each block is determined using (3) and stored in Buffer 2.

Stage-2 determines the optimal CR through an auxiliary task that has been trained using RL. In Stage-2, the procedure including "Done Check," "Feature Extractor," "Policy," and "Calculate Bias" is repeatedly performed. The "Done Check" plays the role of determining whether the task has ended. If the task needs to be continued, the CRs in Buffer 2 are sent to the "Feature Extractor". The "Feature Extractor" expresses the complexity of the current image and extracts three types of features that can be analyzed. The features extracted by the "Feature Extractor" are sent to the "Policy". The purpose of the "Policy" is to change all the CRs to the optimal value, and the "Policy" denotes the

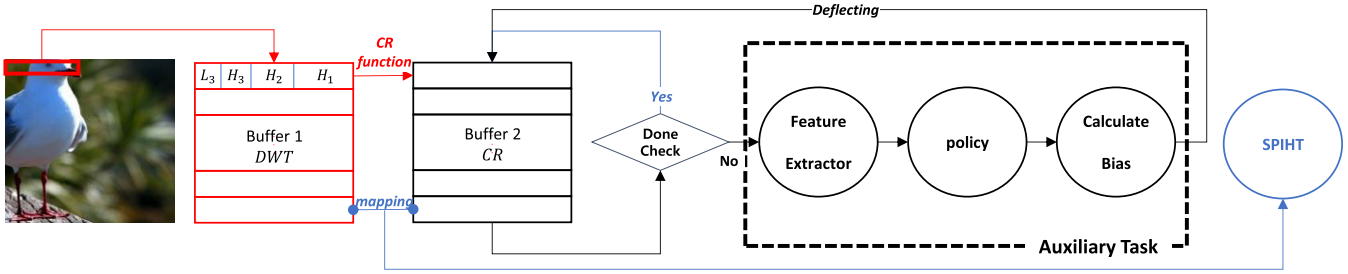


FIGURE 3. Workflow of the RL-SPIHT. Stage-1 with the red lines illustrates the flow of ASCR, and Stage-2 with the black lines illustrates the auxiliary task. The auxiliary task performs the function of biasing the overall CR values resulting from the ASCR, and the bias values for the CRs are determined by the policy, which we intend to train using RL. The blue line shows the process from after the bias values have been determined and until the SPIHT is performed.

neural network trained by RL. In the "Policy" block, the state determines the direction of the bias. The "Calculate Bias" step calculates the degree of the bias in the direction of the bias. After this procedure is completed (i.e., once all the CRs have been biased), the updated CRs are stored in Buffer 2. The CR biasing is repeatedly performing using this loop. When the correction of the CRs has been completed (i.e., all the CRs have optimal values), the "Done Check" step determines that this loop no longer needs to be repeated. In the final step of the procedure, the DWT coefficients stored in Buffer 1 are mapped to the optimized CRs of each block stored in Buffer 2, and the adaptive SPIHT is performed.

B. EPISODIC ENVIRONMENT AND AGENT FOR BIASING

This subsection focuses on the "Feature Extractor" and "Policy" blocks shown in Stage-2 of Fig. 3. The interaction between the environment and agent, which are necessary for RL covered in Section II-C, is the motivation for designing Stage-2. From the perspective of the environment, the CRs stored in Buffer 2 are raw data. The "Feature Extractor" changes the raw data to low-dimension features to decrease the network size and increase the learning speed. The policy is a component of the agent that determines the actions using the features received from the environment. The actions determined by the policy are defined as the directional value that biases all the CRs. This directional value has two values: positive and negative. The process of biasing the CRs is expressed using the following equation:

$$CR' = [CR + \beta\left(D, R, \beta'\right)]. \tag{10}$$

where $[\cdot]$ indicates rounding off, and $\beta(\cdot)$ denotes the discrete bias values selected by the parameter. The parameters, D, R, and β', denote the cumulative relative frequency distribution, information on the remaining blocks, and the β that was applied to the previous distribution, respectively, for the seven classes (i.e., 3/16, 4/16, 5/16, 6/16, 7/17, 8/16, 9/16) in the CR distribution prior to the adjustments (i.e., before correcting the CRs obtained through RL).

Algorithm 1 explains the loop in Stage-2 of Fig. 3 as the episodic environment, which is a component of the RL model. This environment has a unique environment known as compression; hence, it is designed as a finite environment. We assume that an additional increase in PSNR can be achieved by biasing the existing distribution of the CRs obtained from Stage-1 (i.e., CRs obtained using ASCR [21]). However, real-world images have diverse patterns; hence, it is very challenging to train the network to determine the optimal bias on the first attempt. Therefore, we must bias the CR distribution with a low precision and continuously observe the changed states based on the episodic environment.

Accurately understanding this algorithm requires an explanation of the policy of the agent, which is another component of the RL model. The episode of the episodic environment uses the step as the basic unit ($0 \leq step \leq Max_{step}$) and moves continuously from the starting point to the endpoint. The purpose of the episode is to determine the optimal bias, β, that biases N raw data, CR_n, which are stored in Buffer 2 of Fig. 3. The step receives the direction ($signal_{neg}$ or $signal_{pos}$), which is the output of the policy, as an input. The step continually accumulates precision (θ) in β in the direction of the signal ($signal_{neg}$ or $signal_{pos}$). During this repetitive process, if the frequency distribution of CR_n changes even slightly, the step ends. The reason the step ends only when a change occurs is described in Section III-C, along with the explanation of the sparse reward issue. θ is the unit of precision for the amount of change in β, which biases the frequency distribution of CR. θ is always the same in all the episodes and steps. If θ is set too small, the time duration per step increases, but the change in PSNR can be observed accurately and the current frequency distribution and the state of other features can be determined sensitively. This, in turn, allows the policy to be trained better. On the other hand, if θ is set too large, the time duration per step decreases, and consequently, we cannot determine in which step the episode should end. Therefore, we assume that θ is appropriate when the number of θ applied to β in one step is two on average because the variance of the raw data CR_n can vary according to images. Furthermore, we set 0.001 as the unit. The two variables, $count_{neg}$ and $count_{pos}$, represent the count accumulated in the step according to the direction variable, which simplify the calculation. $Done_{epi}$ and $Done_{step}$ are Boolean variables indicating that the episode and step have ended, respectively.

The line-by-line operation of Algorithm 1 is analyzed as follows. The episodic environment first initializes all the variables (Lines 1–3). In the proposed method, one frame becomes one environment episode, and the episode performs the while loop continuously until the termination condition for the episode is satisfied (Lines 4–18). Local variables are always initialized at the beginning of the loop (Lines 5–6). The step processing procedure of the episode is performed continually through the while loop until the termination condition for the step is satisfied (Lines 7–17). The step checks the direction it received as an input and increments the cumulative coefficient variable corresponding to the type of the direction until the end of the episode (Lines 8–9). Subsequently, the bias β is updated through the cumulative coefficients and precision constants accumulated until this step (Line 10). The termination condition of the step is satisfied when the CRs that have been updated and the rounded-off values of the CRs from the previous step are different in at least one of the blocks (Lines 11–14). After the step has been completed, two conditions are checked to determine whether to end the episode. The first condition checks the maximum number of steps (Line 15), and the second condition checks the difference between training and the inference process (Line 16). During training, it is possible to proceed through the last step to store as much diverse data as possible in the replay memory. However, we attempt to end the episode as quickly as possible when performing inference. Hence, the episode does not proceed to the last step. Instead, when an input value different from the input value of the first step is received, that step is performed and then terminated. The logic has been designed in this manner because each step is unique in an episode where the same parameters and images have been applied. Therefore, receiving a different input indicates that the step segment has gone into an infinite loop. Once the episode ends (line 18), the bias value is returned as the final output (line 19).

The episodic environment, which helps solve the challenge of determining the bias value, is important. However, simultaneously, the agent, which receives features from the environment and responds with an action, should be designed well. We have redefined (6) for the agent as follows:

$$\beta\left(D, R, \beta'\right) = Episode(\pi(D, R, \beta')), \tag{11}$$

where π is the policy in Fig. 3, and D, R, β' are inputs to the policy. In terms of the RL algorithm, they can be substituted as the policy of the agent and the three states of the environment, respectively. Therefore, π can be expressed as $q_\pi(s, a)$ of (8). Moreover, the action of the Q-function can be expressed using the following equation:

$$Direction = \underset{a}{\operatorname{argmax}}\, Q(s, a). \tag{12}$$

The output of the Q-function has two arguments, and the direction selects the index of the max value between these arguments as the action. This action is a decision value that decides in which direction the precision should be accumulated in the step procedure of Algorithm 1. Consequently, the optimal policy is trained using the state features and actions, which represent the Q-function, through the DQN algorithm described in Section II-D.

C. REINFORCEMENT LEARNING FOR ASCR

It is important to design both the states and rewards well to train the policy of the agent using RL. It should be noted that rewards have a significant impact on training speed and performance. When the reward is applied to the updated equation, which uses the Bellman equation described in Section II-D, sparse reward and delayed reward should be considered carefully. The sparse reward issue hinders the training process because only a few steps receive a reward ($R \neq 0$) among all the steps in the episode. From the perspective of the episodic environment, when a certain value as bias is added to the CRs received from ASCR [21], the raw data change, but the frequency distribution of CR may not change in the end. If this phenomenon occurs, PSNR does not increase, and consequently, the reward is 0 for the corresponding step. Owing to this phenomenon, in the proposed scheme, the step is not terminated until the frequency distribution of CR changes at least slightly with the precision as described in Section III-B. This scheme prevents samples with the reward of 0 from accumulating in the experienced replay memory, which in turn ensures that the training process is not hindered.

On the other hand, the delayed reward issue occurs when the model for the environment is not accurately known, and it is difficult to predict the changes in the observed state. In such cases, we do not know when the reward should be given. The RL model has a basic tendency for not giving action in the direction of receiving a negative reward. Hence, even if it receives a negative reward due to a slight loss, such as noise, in comparison with the reference value, it determines that it could miss the reward that may occur in the future. Consequently, the effect of training is reduced. Moreover, some other episodes are terminated upon receiving a negative reward. Hence, these episodes tend to miss the opportunity to obtain more positive rewards. If this phenomenon occurs consistently while training, it is difficult to store valuable experience samples in the memory, even for a long exploration, and consequently the network cannot be trained properly. From the perspective of the episodic environment, we can consider a case where PSNR decreases in S_n and S_{n+1}, but the final PSNR increases in S_{n+2} (i.e., a swing scenario) as an example. To solve these issues, the proposed episodic environment should not be terminated even if it receives a negative reward while training. Moreover, to minimize these swings in the reward signal, we delay the time at which the reward signal is received until the PSNR arrives at a randomly accumulated value. By delaying the reward in this manner, it is possible to train more clearly the tendency of the features extracted from the ''Feature Extractor'', which receives CRs from ASCR [21] as an input. The proposed reward function

TABLE 1. Range and meaning of state and action space for MDP.

MDP	Name	Min	Max
state	Distribution	0	1
	R.count	0	1
	R.sum	0	9/4
	R.CR	3/16 or 9/16	
	Bias	-0.5	0.5
action	0	bias toward the negative	
	1	bias toward the positive	

for solving this delayed reward issue is as follows:

$$RWD(\sigma) = \begin{cases} 1, & \sigma < \delta \\ -1, & \sigma \geq -\delta, \end{cases} \tag{13}$$

where σ is the difference between the PSNR of the reference point and the PSNR after performing the current step. The initial value at the reference point is the PSNR when the CRs determined by ASCR [21] have been applied. When $|\sigma|$ arrives at $|\delta|$, the reference point is changed, and σ is initialized with 0. δ postpones the time at which the reward is received; hence, unnecessary steps, such as noise, can be skipped. If δ is too large, the environment becomes a sparse reward environment, and consequently, it is difficult to train the network using RL. On the other hand, if δ is too small, the environment becomes a delayed reward environment because noise may be accepted as it is, and consequently, it is difficult to train the network properly. As a result, when determining δ, PSNR should be obtained by performing the step in one direction for a sufficiently large number of images in the episodic environment. In other words, the variation in PSNR for swinging steps should be checked, and a value greater than the average of the variations should be allocated.

In the previous paragraphs, the core components of MDP for updating the Q-network (i.e., state, action, and reward) are explained. Table 1 lists the range and meaning of the state and action values for the designed MDP. To express all the features regardless of the resolution, the features have been normalized using the total number of DWT blocks so that their values are always relative to all the blocks. The distribution expresses the CRs (raw data) received from ASCR [21] in the RL-SPIHT workflow as relative frequency distribution values based on the frequency distribution with seven classes. The proposed method uses the cumulative relative frequency distribution to reduce the network size further and express the frequency distribution universally. The use of the cumulative relative frequency distribution also provides the advantage that the same weight values of the RL network can be used for all the target CRs. The reason why it is possible to construct an ''integrated'' network that shares the same weight is explained in detail as part of the experimental results in Section IV-B. The cumulative frequency distribution has a value between 0 and 1, and can be expressed as the cumulative relative frequency / total block count. In Table 1, the three features that represent the remaining blocks are indicated as abbreviations: R.count, R.sum, and R.CR. R.count denotes the number of remaining blocks, and R.sum refers to the sum of the CRs allocated to all the blocks. R.CR represents the CR allocated to the remaining block to satisfy TBL, and only the minimum and maximum values of the available CRs can be used for R.CR. Finally, although the unique complexity of the images can be expressed using the values described above, the proposed method has added a bias to express the features more uniquely. Here, the added 'Bias' refers to the previous bias.

The operation of the proposed DQN is described in Algorithm 2 using the components of MDP, which was explained in the baseline DQN [22] algorithm. First, the network weights and replay memory are initialized (Lines 1–2). The baseline DQN initializes the episode using the random parameters of one of the Atari games [22]. However, we instantiate and serialize several episodic environments using several single frames that are independent of each other (Line 3). The episode described in Algorithm 1 is repeated M times using these episodic environments (Lines 4–19). In this procedure, the state is initialized each time the episode is changed (Line 5). As the episode proceeds in the same way as the baseline DQN, the MDP samples necessary for training are continually accumulated in the replay memory until the capacity of the replay memory is reached (Lines 6–13). After one iteration of the episode has been completed, data in the replay memory are randomly sampled up to the size of the mini-batch. Then, the expected value is calculated using the Bellman optimality equation, and the network is updated using the gradient descent method (Lines 13–16). The inference is performed each time the last object of the serialized environment ends the episode, and the result is obtained by calculating the average PSNR for the single frame used in the inference. Finally, if the result is larger than the result of the previous inference, the weights of the network are stored (Line 18).

IV. EXPERIMENTAL RESULTS

In this section, we prove that the features used in the auxiliary task trained with the proposed RL (Q-network) have been properly configured. Moreover, we verify the performance of the proposed 1-D DWT-SPIHT, which utilizes the CRs optimized with the proposed RL model.

A. EXPERIMENTAL ENVIRONMENT

Three 256 × 256 images in the Linneaus-5 JPEG Image Dataset [32] are used to train the Q-network, as shown in Fig. 4(a). In addition, as shown in Fig. 4(b), nine images in the same dataset are used for inference to verify the 1-D DWT-SPIHT algorithm optimized with the DQN. For a fair comparison with previous studies, [20] and [21], that target the video environment, images used for training and inference are converted into the YUV420 format. Moreover, among the components of YUV, the main component Y, which represents the luminance, is used to measure the costs used in the ASCR algorithm [21]. The bias precision and delayed

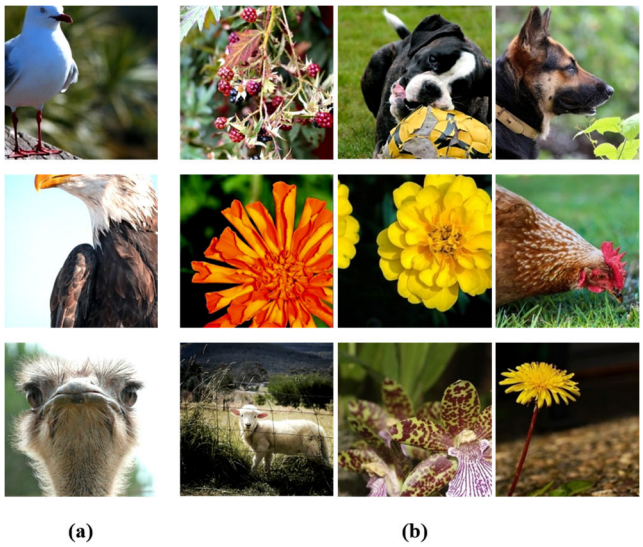


FIGURE 4. Dataset. (a) Training image dataset. (b) Test image dataset.

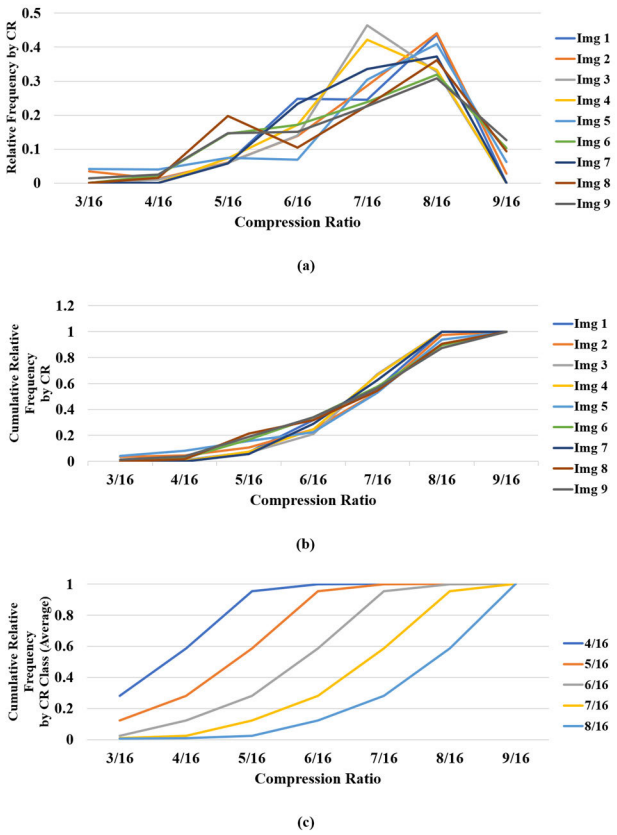


FIGURE 5. Graphs (a), (b), and (c) representing the relative frequency distribution, cumulative relative frequency distribution, and average cumulative relative frequency distribution by target CR, respectively, for the compression ratios of the nine images used in the inference process when the fixed target CR is 7/16.

reward unit are the hyperparameters of the environment, and they are set to 0.001 and 0.05, respectively. The learning rate, discount factor, replay memory size, and mini-batch size are the hyperparameters required for RL training, and they are set to 0.002, 0.98, 50000, and 100, respectively. The leaky

TABLE 2. Results of PSNR (dB) per target CR for the 1-D, ASCR, separated network, and integrated network.

Compression Ratio		PSNR (dB)					
		[20]	[21]	Separated	Integrated	Diff. Int-[20]	Diff. Int-[21]
4/16	Training	33.05	34.30	35.24	35.26	2.21	0.96
	Test	31.47	32.30	32.85	32.85	1.38	0.55
	All	31.86	32.80	33.45	33.45	1.59	0.65
5/16	Training	36.45	38.48	39.89	39.77	3.32	1.29
	Test	34.73	36.61	37.30	37.30	2.57	0.69
	All	35.16	37.08	37.95	37.91	2.75	0.83
6/16	Training	39.90	42.11	43.34	43.31	3.41	1.2
	Test	38.01	40.14	40.65	40.65	2.64	0.51
	All	38.48	40.63	41.32	41.31	2.83	0.68
7/16	Training	43.26	45.41	45.97	45.97	2.71	0.56
	Test	41.17	42.69	43.26	43.30	2.13	0.61
	All	41.69	43.37	43.94	43.97	2.28	0.6
8/16	Training	46.03	47.00	47.47	47.74	1.71	0.74
	Test	43.90	44.67	45.24	45.27	1.37	0.6
	All	44.43	45.25	45.80	45.89	1.46	0.64
Average	Training	39.74	41.46	42.38	**42.41**	**2.67**	**0.95**
	Test	37.86	39.28	39.86	**39.88**	**2.02**	**0.60**
	All	38.33	39.83	40.49	**40.51**	**2.18**	**0.68**

ReLU [33] is used as the activation function for all the fully connected layers. Moreover, the study by Kim *et al.* [20], which proposed the 1-D DWT-SPIHT with fixed CR, and ASCR [21] are used for a comparison with the PSNR of the DWT-SPIHT.

B. REASONABLENESS OF THE FEATURES FOR EPISODES

The RL model can better train the policy of the agent when it has state features that can describe the environment model. If the cumulative frequency distribution is used as a feature, the trained network can respond universally to the image resolution and the diversity of raw data, and simultaneously, the weights can be reduced as much as possible. In this subsection, we prove the appropriateness of the feature selection through the data sampled in the experiment.

Inputting CRs that change sensitively to resolution size into the policy network as image features uses a lot of hardware resources. Therefore, the frequency distribution of the CRs needs to be used as the feature, instead of the CRs, to reduce the feature size and use the feature regularly. However, it may be challenging to perform accurate inference for data having a different resolution than the training dataset because the maximum frequency values are different. Although there is a way to train the network with a lot of datasets having various resolutions to solve this problem, the network may be enlarged accordingly and the efficiency in terms of time may decrease. Therefore, CRs have to be normalized regardless of the resolution by expressing them as a relative frequency dis-

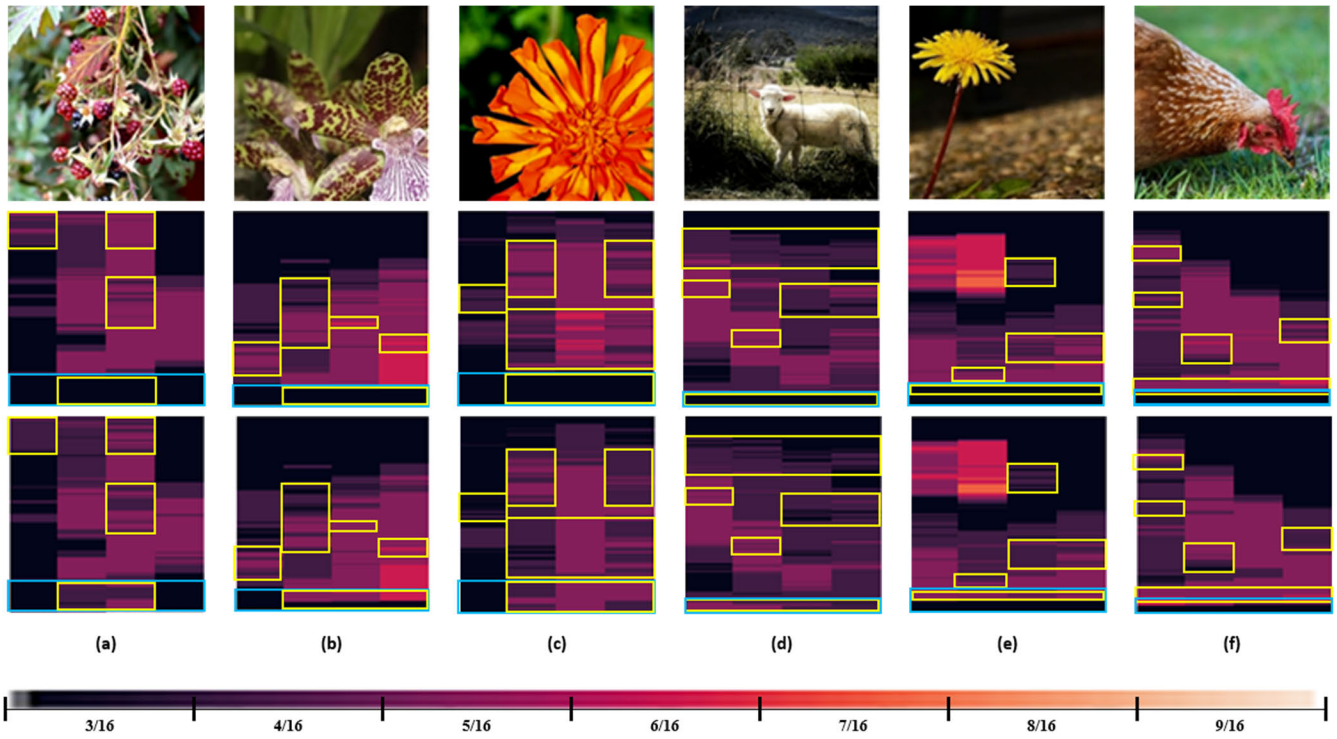


FIGURE 6. Heatmap of CR distribution within the frame according to each compression method when the target CR is 4/16. Top: Original images, Middle: Heatmap by ASCR [21], and Bottom: Heatmap by the proposed method. A block painted with a light tone represents a low CR. The blue box indicates the part where the ASCR method has allocated an unsuitable CR to the block to satisfy the total target bit length. The yellow box indicates the part where there is an apparent difference between the ASCR method and the proposed method on the heatmap. In the proposed method, the blue part can also allocate the optimal CR by securing the resource in the yellow part compared with ASCR.

tribution. However, as shown in Fig. 5(a), even if the normalized data are used, it is difficult to identify the tendency of the network because the difference in frequency between images in a single class is large. Hence, the policy network becomes a ''separated'' network, which uses different weights each time the fixed target CR changes. However, although the CR distribution changes at each step, the values do not change significantly from the initial value. Therefore, if the tendency of the images can be identified when the fixed target CR is the same, it is possible not to change the weights according to the fixed target CR, and this can be solved by converting a relative frequency distribution into a cumulative relative frequency distribution. In Fig. 5(b), although the same images as in Fig. 5(a) are used, the cumulative relative frequency distribution can objectively express the tendency of the images. Therefore, as shown in Fig. 5(c), the average cumulative relative frequencies for the initial state can be expressed by using each fixed target CR for training. Eventually, we can cope with all fixed target CRs as an ''integrated'' network without changing the weights according to the fixed target CRs.

C. PERFORMANCE EVALUATION OF THE PROPOSED SYSTEM

In this subsection, the performance of the proposed model is verified and compared with previous studies [20] and [21]. Table 2 presents the comparison of the PSNR results for each CR with those from the previous studies. For the proposed method, the results of ''separated'' for which RL is applied separately to each target CR, and the results of ''integrated'' for which RL is applied together for all the target CRs, are presented. For each CR, the average PSNRs of the 3 images used for training, 9 images used for inference, and 12 images combined (3 for training + 9 for inference) are presented in order. Experimental results show that the proposed ''separated'' method has higher average PSNRs for all the CRs compared with the previous studies. In addition, the average PSNR of the ''integrated'' method is 0.02 dB higher than that of the ''separated'' method, even though the ''integrated'' method has high compatibility that it can be applied to all target CRs with a single weight. As a result, the ''integrated'' method can achieve an average performance improvement of 2.67 / 2.02 / 2.18 dB and 0.95 / 0.6 / 0.68 dB (training set / test set / total set) compared with the existing fixed [20] and ASCR [21] methods, respectively. This means that the proposed method can reduce the PSNR gap between the 1-D DWT-SPIHT and the 2-D DWT-SPIHT by optimizing the ASCR through sufficient training based on the RL model.

Fig. 6 shows the heatmaps of CR distribution within the frame according to each compression method when the target CR is 4/16. The first row presents six original images and the second and third rows show the CR distribution per block when these six images are compressed using the existing

TABLE 3. Comparison of the number of adaptive blocks selected per target CR. These results are calculated using nine images for the inference process.

Target CR	Method	Adaptive CR						
		3/16	4/16	5/16	6/16	7/16	8/16	9/16
4/16	ASCR	37.6%	28.1%	31.2%	2.9%	0.2%	0%	0%
	Proposed	34.1%	35.4%	28.0%	2.0%	0.1%	0%	0.3%
5/16	ASCR	14.6%	15.4%	30.0%	35.5%	4.3%	0.2%	0%
	Proposed	13.0%	16.2%	32.7%	34.2%	3.6%	0.2%	0%
6/16	ASCR	3.8%	9.4%	15.6%	30.3%	36.3%	4.3%	0.2%
	Proposed	2.9%	9.9%	16.0%	31.8%	35.2%	3.7%	0.5%
7/16	ASCR	1.6%	1.5%	9.6%	15.7%	30.4%	36.3%	4.8%
	Proposed	1.0%	1.6%	9.8%	16.1%	32.2%	34.2%	5.1%
8/16	ASCR	0.8%	0.5%	1.5%	9.7%	15.5%	27.2%	44.9%
	Proposed	0.5%	0.5%	1.3%	9.0%	16.1%	30.2%	42.4%

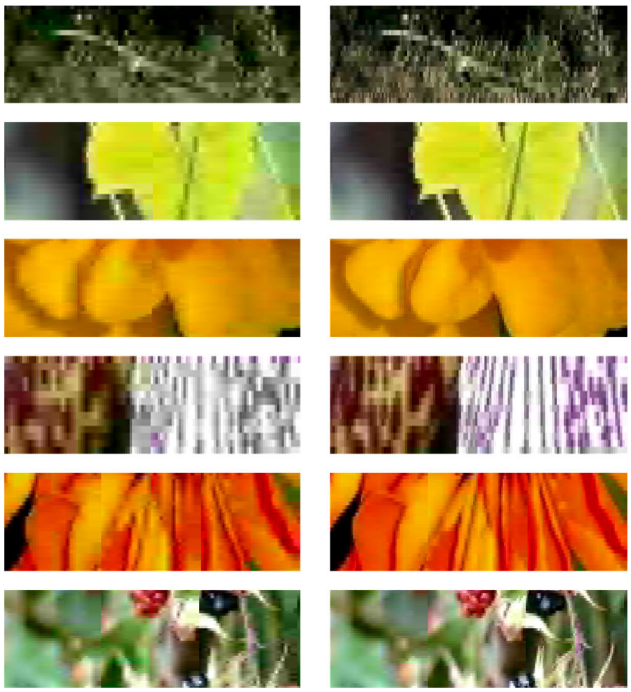


FIGURE 7. Enlarged still images of the reconstructed frames when the target CR is 4/16. (a) Fixed compression ratio [20], (b) Proposed method.

ASCR [21] method and the RL-SPIHT method, respectively. When using RL-SPIHT, the CRs are evenly distributed for most of the images by increasing the bias of the distribution in the backward direction. Hence, the areas assigned with unnecessarily high CRs (yellow box) in the ASCR method are allocated with lower optimal CRs, and it is possible that saved bit lengths are utilized in the lowermost part (blue box) of the heatmaps, which has been forcibly determined as a specific CR to satisfy the TBL in the ASCR method. Such an optimal CR arrangement eventually results in increases in PSNR, as presented in Table 2. Table 3 presents the ratio of the number of adaptive CRs in the ASCR and proposed methods for each target CR. It should be noted that in the "fixed" method, all blocks are compressed equally with the target CR. Experimental results show that the ASCR method selects the CR of 3/16 more often than the "integrated" method in order to match the TBL, which leads to the severe performance degradation. The proposed method can achieve the improvement of the PSNR shown in Table 2 by minimizing such unnecessary selection of the CR of 3/16 through RL-based optimal biasing.

Fig. 7 shows still images compressed and decompressed by the ASCR [21] method (left) and the proposed method (right) using the target CR of 4/16. In Fig. 7, a specific part in each i mage is enlarged to show the difference clearly. It can be seen that vividly visible boundaries between the blocks in the ASCR method have been considerably smoothed in the proposed method. These effects are more visible in the foreground region with a high complexity. Furthermore, the blurred area in the ASCR method has been processed more clearly by the proposed method. These results verify that the proposed method can produce effective deblurring and deblocking effects.

V. CONCLUSION

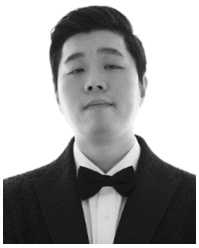
In this study, we optimize the compression efficiency of the 1-D DWT-SPIHT algorithm, a representative EC scheme, by using an RL-based episodic auxiliary task. Although the previous studies uncovered the correlation between the DWT coefficients and the CR of the SPIHT, it could not be optimized sufficiently. Thus, we use an episodic model trained using RL to optimally adjust the adaptively selected CRs for each block according to the DWT coefficients, and apply the optimized CRs to SPIHT. Consequently, compared with the fixed technique that applies the same CR to all the blocks and the preceding research, PSNRs are improved by 2.18 dB and 0.68 dB on average, respectively. In addition, the deblocking and deblurring effects that must be addressed in the block-based compression methods have been achieved. In conclusion, the proposed algorithm can contribute significantly to the performance improvement of the EC-based FMC and can be extended to various studies related to video compression.

REFERENCES

[1] S. Wang and S. Dey, "Adaptive mobile cloud computing to enable rich mobile multimedia applications," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 870–883, Jun. 2013.
[2] H. Knoche, J. D. McCarthy, and M. A. Sasse, "Can small be beautiful?: Assessing image resolution requirements for mobile TV," in *Proc. 13th Annu. ACM Int. Conf. Multimedia (MULTIMEDIA)*, 2005, pp. 829–838.
[3] H. Kim, C. E. Rhee, and H.-J. Lee, "A low-power video recording system with multiple operation modes for H.264 and light-weight compression," *IEEE Trans. Multimedia*, vol. 18, no. 4, pp. 603–613, Apr. 2016.
[4] R. Trestian, A.-N. Moldovan, C. H. Muntean, O. Ormond, and G.-M. Muntean, "Quality utility modelling for multimedia applications for Android mobile devices," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast.*, Jun. 2012, pp. 1–6.
[5] S. He, Y. Liu, and H. Zhou, "Optimizing smartphone power consumption through dynamic resolution scaling," in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw.*, Sep. 2015, pp. 27–38.
[6] D. Sostaric, D. Vinko, and S. Rimac-Drlje, "Power consumption of video decoding on mobile devices," in *Proc. ELMAR*, Sep. 2010, pp. 81–84.
[7] T. B. Yng, B.-G. Lee, and H. Yoo, "A low complexity and lossless frame memory compression for display devices," *IEEE Trans. Consum. Electron.*, vol. 54, no. 3, pp. 1453–1458, Aug. 2008.

[8] A. D. Gupte, B. Amrutur, M. M. Mehendale, A. V. Rao, and M. Budagavi, "Memory bandwidth and power reduction using lossy reference frame compression in video encoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 2, pp. 225–230, Feb. 2011.
[9] T. Wang, M. Chen, and H. Chao, "A novel deep learning-based method of improving coding efficiency from the decoder-end for HEVC," in *Proc. Data Compress. Conf. (DCC)*, Apr. 2017, pp. 410–419.
[10] X. Lian, Z. Liu, W. Zhou, and Z. Duan, "Lossless frame memory compression using pixel-grain prediction and dynamic order entropy coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 223–235, Jan. 2016.
[11] Y. Jin, Y. Lee, and H.-J. Lee, "A new frame memory compression algorithm with DPCM and VLC in a 4×4 block," *EURASIP J. Adv. Signal Process.*, vol. 2009, no. 1, Dec. 2009, Art. no. 629285.
[12] Y. Jin and H.-J. Lee, "A block-based pass-parallel SPIHT algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 7, pp. 1064–1075, Jul. 2012.
[13] C. D. Creusere, "A new method of robust image compression based on the embedded zerotree wavelet algorithm," *IEEE Trans. Image. Process.*, vol. 6, no. 10, pp. 1436–1442, Oct. 1997.
[14] C.-L. Lin, "A low latency coding scheme for compressing reference frame in video codec," in *Proc. Int. Conf. Appl. Syst. Innov. (ICASI)*, May 2017, pp. 1957–1960.
[15] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, pp. 18–34, Feb. 1992.
[16] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding system: An overview," *IEEE Trans. Consum. Electron.*, vol. 46, no. 4, pp. 1103–1127, Nov. 2000.
[17] J. Jyotheswar and S. Mahapatra, "Efficient FPGA implementation of DWT and modified SPIHT for lossless image compression," *J. Syst. Archit.*, vol. 53, no. 7, pp. 369–378, Jul. 2007.
[18] P.-Y. Chen, "VLSI implementation for one-dimensional multilevel lifting-based wavelet transform," *IEEE Trans. Comput.*, vol. 53, no. 4, pp. 386–398, Apr. 2004.
[19] X. T. Nguyen, H.-J. Lee, and H. Kim, "A low-cost hardware design of a 1-D SPIHT algorithm for video display systems," *IEEE Trans. Consum. Electron.*, vol. 64, no. 1, pp. 44–52, Feb. 2018.
[20] S. Kim, D. Lee, J.-S. Kim, and H.-J. Lee, "A high-throughput hardware design of a one-dimensional SPIHT algorithm," *IEEE Trans. Multimedia*, vol. 18, no. 3, pp. 392–404, Mar. 2016.
[21] H. Kim, A. No, and H.-J. Lee, "SPIHT algorithm with adaptive selection of compression ratio depending on DWT coefficients," *IEEE Trans. Multimedia*, vol. 20, no. 12, pp. 3200–3211, Dec. 2018.
[22] 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*. [Online]. Available: http://arxiv.org/abs/1312.5602
[23] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.
[24] Z. Wang, N. D. Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," in *Proc. Int. Conf. Int. Conf. Mach. Learn.*, vol. 48, Jun. 2016, pp. 1995–2003.
[25] H. V. Hasselt, "Double Q-learning," in *Proc. 24th Adv. Neural Inf. Process. Syst. (NIPS)*, 2010, pp. 2613–2621. [Online]. Available: http://papers.nips.cc/paper/ 3964-double-q-learning.pdf
[26] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Aug. 2017, pp. 449–458.
[27] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.
[28] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, *arXiv:1511.05952*. [Online]. Available: http://arxiv.org/abs/1511.05952
[29] M. Fortunato, M. Gheshlaghi Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg, "Noisy networks for exploration," 2017, *arXiv:1706.10295*. [Online]. Available: http://arxiv.org/abs/1706.10295
[30] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Proc. AAAI*, Feb. 2017, pp. 1–14.
[31] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, D. Guo, and C. Blundell, "Agent57: Outperforming the Atari human benchmark," 2020, *arXiv:2003.13350*. [Online]. Available: http://arxiv.org/abs/2003.13350
[32] G. Chaladze and L. Kalatozishvili. (2017). *Linneaus-5 Dataset for Machine Learning*. [Online]. Available: http://chaladze.com/l5
[33] X. Zhang, Y. Zou, and W. Shi, "Dilated convolution neural network with LeakyReLU for environmental sound classification," in *Proc. 22nd Int. Conf. Digit. Signal Process. (DSP)*, Aug. 2017, pp. 1–5.

JIN SHIN received the A.S. degree in computer system and engineering from the Inha Technical College, Incheon, South Korea, in 2017, and the B.S. degree in computer science and engineering from the National Institute for Lifelong Education, Seoul, South Korea, in 2019. His research interests include compression and restoration schemes for multimedia applications.



HYUN KIM (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 2009, 2011, and 2015, respectively. From 2015 to 2018, he was a BK Assistant Professor with the BK21 Creative Research Engineer Development for IT, Seoul National University. In 2018, he joined the Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul, where he is currently working as an Assistant Professor. His research interests include algorithm, computer architecture, memory, and SoC design for low-complexity multimedia applications and deep neural networks.