Accepted in IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS 2020), 2020
Authors accepted manuscript. The final published version available at https://doi.org/10.1109/MWSCAS48704.2020.9184456

CU Size Decision for Low Complexity HEVC Intra Coding based on Deep Reinforcement Learning

Mohammadreza Jamali and Stéphane Coulombe
Department of Software and IT Engineering
École de technologie supérieure, Université du Québec
Montreal, Canada
mohammadreza.jamali.1@ens.etsmtl.ca, stephane.coulombe@etsmtl.ca

Hamidreza Sadreazami
Bio-engineering Department
McGill University
Montreal, Canada
hamidreza.sadreazami@mail.mcgill.ca

***Abstract*—High efficiency video coding (HEVC) uses a quadtree-based structure for coding unit (CU) splitting to effectively encode various video sequences with different visual characteristics. However, this new structure results in a dramatically increased complexity that makes real-time HEVC encoding very challenging. In this paper, we propose a novel CU size decision method based on deep reinforcement learning and active feature acquisition to reduce HEVC intra coding computational complexity and encoding time. The proposed method carries out early splitting and early splitting termination by considering the encoder and CU as an agent-environment system. More specifically, through early splitting, the proposed method precludes the need for rate-distortion optimization at the current level. In addition, through early splitting termination, it disposes of the lower level computations. The proposed method provides a very fast encoder with a small quality penalty. Experimental results show that it achieves a 51.3% encoding time reduction on average with a small quality loss of 0.041 dB for the BD-PSNR, when we compare our method to the HEVC test model.**

***Index Terms*—Coding unit size decision, deep reinforcement learning, high efficiency video coding (HEVC), H.265, intra video coding, low complexity video coding, video compression**

I. Introduction

High efficiency video coding (HEVC)/H.265 is the most recent video coding standard that achieves a significant improvement in compression efficiency and provides a 50% bit rate reduction compared to the highly successful H.264/advanced video coding (AVC) with the same quality. The improved performance of HEVC is at the expense of much higher computational complexity at the encoder, making it challenging to be deployed in real-time applications. In frame splitting, unlike H.264/AVC which employs 16×16 macroblocks, HEVC introduces coding tree units (CTUs) with a maximum size of 64×64 [1]. The CTU may be split recursively and content-adaptively into coding units (CUs) in a quadtree-based manner, resulting in an efficient coding of background and objects with various sizes and shapes. This frame partitioning along with mode decision result in a significant computational complexity.

To lower this high complexity, previous works have proposed to either avoid processing the entire tree, resulting in a fast CU size decision [2]–[6] or decrease the number of intra modes going through the rate-distortion optimization (RDO) [7]–[12], resulting in a fast mode decision. Some other works combine the two approaches to achieve faster encoders [13], [14]. In [4], a data-driven CU size decision is proposed in which a support vector machine based method is applied at all four CU levels. This method consists of two steps. In the first step, an offline classifier decides on CU split termination or skipping the current CU. In the second step, and for those CUs which are not early skipped or early terminated, an online binary classifier, based on previous frames, is proposed to further refine the CU size decision. In [6], a method is proposed for fast CU size decision using statistical information. This method uses the image complexity along with an adaptive depth prediction process to early determine the size of the CU. The CU splitting is terminated early based on a Bayesian classification and a quadratic discrimination analysis. In [13], a gradient-based approach is proposed for fast CU size decision which uses texture complexity. The gradients are also employed to decrease intra mode candidates.

To lower HEVC encoder high complexity, in this paper, a new framework for intra CU size decision is proposed based on deep reinforcement learning (RL) and active feature acquisition [15]. The proposed method is realized by incorporating a CU early splitting and a CU early splitting termination. The CU early splitting aims at skipping mode decision at the current level and moving directly to the next depth without RDO computations. In addition, CU early splitting termination aims at terminating the splitting process and skipping mode decision for the next levels. As a result, the proposed method eliminates the non-necessary CU levels from the RDO process. We also propose to consider CU size decision as a markov decision process (MDP) and the encoder as an agent, sequentially taking coding decisions. To the best of our knowledge, this is the first work to consider HEVC intra CU size decision as a sequential decision making problem and to solve it by RL. In addition, it is the first work to utilize sequential feature acquisition in HEVC coding. Sequential feature acquisition is proposed to compute the most discriminative features used to make better decisions by the RL agent. It should be noted that any CU size decision method, including ours, provides an independent module which may be combined with fast mode decision methods to achieve even faster intra coding.

This work was supported by Vantrix Corporation and by the Natural Sciences and Engineering Research Council of Canada under the Collaborative Research and Development Program (NSERC-CRD 428942-11).

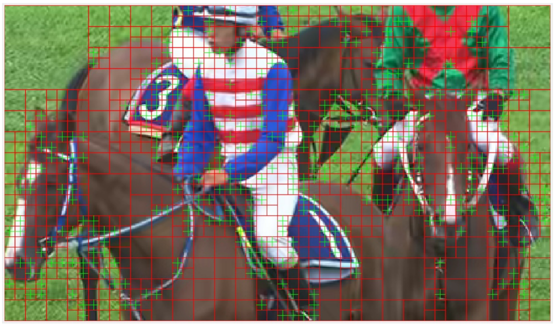
Fig. 1: Example of frame splitting into CUs (red) and PUs (green) based on the HEVC quadtree structure, *RaceHorses*.

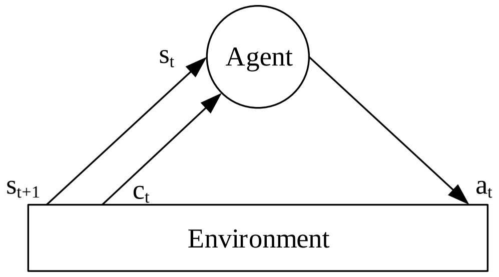

Fig. 2: Reinforcement learning configuration [17].

The paper is organized as follows. Section II discusses HEVC intra coding. Section III introduces requisite RL concepts. In section IV, our method of CU size decision for complexity reduction is proposed. Section V presents the experimental results and finally section VI concludes the paper.

II. Overview of HEVC Intra Coding

HEVC intra coding, compared to previous video codecs, carries some new features such as a new frame splitting approach and an increased number of prediction modes. In frame splitting, it uses a quadtree structure, where the largest CU (with a size of 64×64) is recursively split into four equal-sized CUs. The size of the smallest CU is 8×8. To perform the prediction with intra coding, for each CU a prediction unit (PU) is associated, except for the 8×8 CU, which could be predicted as an 8×8 PU or four 4×4 PUs. Fig. 1 shows an example of dividing a frame into CUs and PUs for one of the sample sequences, namely, the *RaceHorses* sequence [16], using such quadtree structure. From this figure, it can be observed that there are many possibilities to split a picture into multiple blocks and many ways to combine the coding tools. In order to find the best combination, an exhaustive execution of RDO may be performed, which is highly time consuming.

In mode decision, HEVC increases the number of intra modes to 35, compared to 9 modes in H.264. To select the best mode at each CU level, the HEVC test model (HM) operates in two steps. First, it selects the N candidates with the lowest rough mode decision (RMD) costs. Then the RDO process chooses the mode with the lowest RDO cost as the final mode. In the first step, the RMD cost (C_{RMD}) is computed as:

$$C_{RMD} = SATD_l + \lambda_p \times R_p, \tag{1}$$

where λ_p is a Lagrange multiplier, R_p is the mode coding bitrate and $SATD_l$ is the sum of absolute transformed differences (SATD) between the predicted and original blocks.

In the second step, and for the selected candidates, the RDO cost (C_{RDO}) is computed as:

$$C_{RDO} = (SSE_l + \omega_c \times SSE_c) + \lambda_m \times R_m, \tag{2}$$

where λ_m is a Lagrange multiplier and R_m is the PU coding bitrate. SSE_l and SSE_c are the sum of squared errors (SSE) between the reconstructed luma and chroma blocks and the original ones, respectively, and ω_c is the chroma weight determined based on the quantization parameter.

III. Reinforcement Learning

Reinforcement learning is a type of machine learning technique containing a set of powerful algorithms to find optimal solutions for complicated sequential problems. In an RL system, an agent is learning how to take actions in an environment to minimize (maximize) a cost (reward) function. The agent, at time t, observes the environment state as s_t and then takes action a_t which results in an immediate cost (reward) of $c_t(r_t)$ while the environment goes to the next state of s_{t+1}. Fig. 2 shows a configuration of an RL system.

To apply RL to the CU splitting problem, we describe it as a sequential decision making problem, where the CU is an MDP and the encoder is an agent sequentially taking coding decisions. An MDP is defined as a tuple of (S, A, T, C), where S is a set of states, A is a set of actions, C is a stochastic cost function and T is a stochastic function, which determines the transitions between the states. By taking an action a, the agent changes the state of the MDP from s to s'. In such a system, a policy π is defined as a function which outputs an action a for a given state s such as $a = \pi(s)$. The value of an action performed in a certain state is defined by a state-action value function $Q^{\pi}(s,a)$, which indicates the expected return cost, when starting from state s, taking action a and then following policy π. This function is called Q-function and defined as:

$$Q^{\pi}(s,a) = E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^{\,k} c_{t+k} \middle| s_t = s, a_t = a\right], \tag{3}$$

where $0 \leq \gamma < 1$ is a discount factor to make future costs less important than the immediate cost.

If the MDP is large with too many states and actions (or continuous states and actions), the state-action value function is represented with function approximation as follows:

$$\hat{Q}_{\pi}(s,a) \approx Q_{\pi}(s,a). \tag{4}$$

This makes it possible to generalize from observed states and actions to unobserved ones. In our proposed method, and as a function approximation method, we propose the use

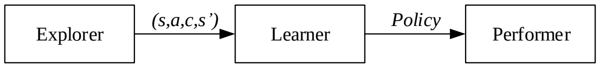

Fig. 3: Three agents of a batch-mode reinforcement learning system.

of neural networks as they are known to approximate any function by using non-linearity via different activations.

Batch-mode Reinforcement Learning

In batch-mode RL, the agent tries to find the best policy using a set (batch) of transition samples. Batch-mode RL algorithms are popular because of their low complexity, data-efficiency and stability. In batch-mode RL, the agent receives a set of n transition samples $\mathcal{F} = \{(s_t, a_t, c_t, s_{t+1}) | t = 1, ..., n\}$ and then finds the best possible policy based on these samples. Unlike online learning, where the performance agent and the learning agent are the same, in batch-mode RL they are clearly separated with an additional agent for exploring purpose. Fig. 3 shows the three agents of a batch-mode RL system.

IV. Fast CU Size Decision

We consider CU size decision as a sequential decision making problem and formulate it by RL. We deploy a batch-mode RL, based on fitted-Q iteration (FQI) [18], to find a policy for coding unit size decision in the context of intra HEVC coding. To this end, we use n transition samples in the form of four-tuples (s, a, c, s') in the set $\mathcal{F} = \{(s, a, c, s')_l | l = 1, ..., n\}$ to estimate the state-action value function (Q-function). The policy is derived based on this estimation, which starts with $\hat{Q}^0(s, a) = \bar{q}^0$ for all states and actions, where $\bar{q}^0$ is the initial value and is set to zero ($\bar{q}^0$ is a constant and, therefore, is not a function of s and a). $\hat{Q}^1(s, a)$ is then estimated using $\hat{Q}^0(s, a)$. It is noted that the exact $Q^1(s, a)$ is the conditional expectation of the immediate cost as given by:

$$Q^1(s, a) = E\left[c_t \middle| s_t = s, a_t = a\right]. \tag{5}$$

It is the expected cost of an action a in a given state s if the agent interacts with the environment just one time. After the first estimation, the following two steps are iterated starting from $i = 0$ until a terminating condition is satisfied:

1- The set

$$\mathcal{T}^{i+1} = \{(s, a, \bar{q}^{i+1}_{s,a}) | \bar{q}^{i+1}_{s,a} = c + \gamma \min_{a' \in A} \hat{Q}^i(s', a')\} \tag{6}$$

is generated based on all transition samples provided by the set $\mathcal{F}$. In the above definition, c and s' are functions of s and a, so $\bar{q}^{i+1}_{s,a}$ is a function of s and a.

2- A supervised learning method is used to find $\hat{Q}^{i+1}$ as an approximation of Q^{i+1} based on the set $\mathcal{T}^{i+1}$.

Having $\hat{Q}^N(s, a)$, and by assuming a discrete action space, the approximated optimal policy is obtained as:

$$\hat{\pi}^*_N(s) = \operatorname*{argmin}_{a \in A} \hat{Q}^N(s, a). \tag{7}$$

Using this policy, the performance agent can take an action in a given state, although the state has never been seen before.

A. Problem Formulation

If the system dynamics and the cost function are available, based on dynamic programming (DP), an optimal policy π^* exists for the system. However, in the proposed method, we aim at finding a policy based on a finite set of observed sample transitions. Since theoretically it is not tractable to find an optimal policy based on this finite set of samples, our objective is to find an approximation of such policy. In the proposed method, we consider S to be multi-dimensional and continuous and A to be a finite set of discrete actions.

The first step is an exploration phase, where the encoder runs and the data is collected. In this step, which is the exploring phase in the batch-mode RL, the actions are chosen randomly to see the impact of good and bad actions. In the next step, we choose neural networks and in particular, multi-layer perceptrons (MLPs) to approximate the Q-function since they are capable of approximating nonlinear functions very well. This step, as the learning agent, is the second module of the algorithm. Finally, the encoder behaves as the performance agent by using the learned Q-function and takes actions based on the policy obtained in the learning phase.

Algorithm 1 shows different steps of the proposed method for CU size decision in the form of a fitted learning algorithm. In order to obtain the approximated optimal policy, we use a modified version of (7) to make the implementation more straightforward. In this approach, the set $\mathcal{F}$ is split based on different actions resulting in multiple $\mathcal{F}_a$ for each action. Then a regression algorithm can be applied to each of these subsets to derive the corresponding $\hat{Q}^N_a(s)$. Then all these Q-functions are computed and the action which its related $\hat{Q}^N_a(s)$ is minimum is selected. In view of this, we apply the proposed supervised learning method based on MLPs separately to each action. After all iterations, a separate $\hat{Q}^N_a$ is achieved for each action. For a given state, the performance agent (encoder in our case) chooses the action with the lowest value of $\hat{Q}^N_a$ as the selected action. In the algorithm, N_A is the total number of actions and n_{a_m} is the number of four-tuples in $\mathcal{F}_{a_m}$. In addition, $A_{s'_l}$ is the set of valid actions in state s'_l.

B. State Representation

To apply RL to high-dimensional environments, the state space should be mapped to a low-dimensional feature space. Thus, one of the key components of any RL system is *state representation*, i.e., mapping states to low-dimensional feature vectors using field knowledge. This includes determining the number of features and also the areas they are targeting. If the state space is n-dimensional, the mapping is:

$$\phi : R^n \to R^m, m \ll n, \tag{8}$$

where m is the dimension of the feature space. As a result, the collected data set is:

$$\mathcal{F}_\phi = \{(\phi(s), a, c, \phi(s')) | (s, a, c, s') \in \mathcal{F}\}, \tag{9}$$

Algorithm 1 Encoder training phase (learning agent)

```
Input: $\mathcal{F} = \{(s, a, c, s')_l | l = 1, ..., n\}, A, N_A$
Output: $\hat{Q}_a^N$ as an approximation of the value function $Q_a^N$
    for all action $a \in A$
1:  $k = 0$
2:  initialization: set $\hat{Q}_{a_m}^0$ to zero everywhere on $s$, $m = 1$ to
    $N_A$
3:  repeat
4:    for $m = 1$ to $N_A$ do
5:      $\mathcal{F}_{a_m} = \{(s, a, c, s') \in \mathcal{F} | a = a_m\}$
6:      $\mathcal{T}_{a_m} = \{(i_l, o_l), l = 1, ..., n_{a_m}\}$ where:
7:      $i_l = s_l, o_l = c_l + \gamma \min_{a' \in A_{s'_l}} \hat{Q}_{a'}^k(s'_l)$
8:      Function approximation based on NN
9:      $\hat{Q}_{a_m}^{k+1} \leftarrow \text{NN}(\mathcal{T}_{a_m})$
10:   end for
11:   $k = k + 1$
12: until $k = N$
13: return $\hat{Q}_{a_m}^N$, $m = 1$ to $N_A$
```

in which $\phi(s) = (f_1(s), f_2(s), ..., f_{N_F}(s))$, where N_F is the number of features describing the state. To find the best mapping ϕ, we experimentally obtain the best state representation based on the most relevant features. To this end, in the proposed method, the selected features for the problem of CU size decision are:

-*featureRMDcostBlock*: the RMD cost of the CU.

-*featureRMDcostBlockComputed*: binary value indicating whether the RMD cost is computed for the CU.

-*featureRDOcostBlock*: the RDO cost of the CU.

-*featureRDOcostBlockComputed*: binary value indicating whether the RDO cost is computed for the CU.

-*featureMGA*: the mean of gradient amplitudes (MGA) of the CU.

-*featureMDGA*: the mean of directional gradient amplitudes (MDGA) of the CU.

-*featureEarlySplitTermination*: binary value indicating whether the CU splitting is early terminated.

-*featureEarlySplitting*: binary value indicating whether the CU is early splitted.

-*featureStandardSplitting*: binary value indicating whether the CU is splitted similar to the standard HEVC implementation.

MGA is an appropriate measure to evaluate block's variation. A CU with a low degree of pixel variation is less likely to be split into four smaller CUs. The MGA as a measure of global gradient is given by:

$$MGA = \frac{1}{n} \sum_i \sum_j (|G_X(i,j)| + |G_Y(i,j)|), \tag{10}$$

where n is the number of pixels in the CU, and G_X and G_Y are the gradient components computed by Sobel operator.

MDGA shows if a CU can be accurately predicted by an angular mode. The MDGA is given by:

$$MDGA = \frac{1}{n} \sum_i \sum_j (|G_X(i,j)| + |G_Y(i,j)|) \times cos(\theta(i,j)), \tag{11}$$

where θ is the angle between the best angular mode and the gradient. Using these features, each state can be represented by a 9-dimensional feature vector. All the states are categorized into two classes: terminal and non-terminal states. Terminal states are the states in which the encoding process is terminated for the CU and the episode ends. There are three terminal states categories as follows:

-*Early terminated (ET) state*: is the state of a CU in which the encoding process is terminated and the encoding of the lower depths is avoided. The *featureEarlySplitTermination* for this state is 1.

-*Early split (ES) state*: is the state of a CU in which the encoding process at the current depth is interrupted and the encoder moves to the lower depth. The *featureEarlySplitting* for this state is 1.

- *Standard split (SS) state*: is the state of a CU for which the encoding process at the current depth is fully performed and the encoder moves to the lower depth to continue the encoding process. This CU is treated exactly as it is treated in the HM (the exhaustive approach). The *featureStandardSplitting* for this state is 1.

Non-terminal states are those in which *featureEarlySplitTermination*, *featureEarlySplitting* and *featureStandardSplitting* are zero and the encoding process of the corresponding CU has not yet terminated.

C. Action Space and Cost Function

The action space $A = \{a_1, a_2, ..., a_{N_A}\}$ contains N_A discrete number of actions. In our problem, there are in total five actions available for the encoder as the performance agent. Not all actions are valid in all states. We define two kinds of actions. First, there are information-gathering actions (IGAs) which need to be taken just one time and gain some knowledge about the state of the system. For the purpose of this work, we consider obtaining the value of a feature as the desired knowledge. Second, there are episode terminating actions (ETAs) which lead to a terminal state and like IGAs are taken only one time in each episode.

The reason to define and apply IGAs is well-understood by considering the problem of active feature acquisition [15] in a prediction scenario. Expensive features, in terms of computation, which are not much useful for an accurate prediction, should not be computed. In view of this, the agent should be aware of whether computing a given feature increases the accuracy of prediction or not. This is very challenging when dealing with systems for which feature computation requires a great deal of computational resources. This problem is not usually considered in a traditional machine learning framework, where the features are assumed to be known a priori and without any cost (feature acquisition and prediction are assumed to be independent). To address this issue, we define IGAs in such a way that the agent associates the cost of computing a feature with the cost of taking an action in the RL, i.e., incorporating the concept of active feature acquisition in the learning problem. Following is the list of actions along with the states they are valid in:

- *actEarlySplitTermination*: this action early terminates the splitting procedure and removes the lower depths to be processed. It is an ETA and is valid when *featureRDOcostBlockComputed=true*.
- *actEarlySplitting*: this action early splits the CU before completing the encoding process for the current depth. It is an ETA and is valid when *featureRDOcostBlockComputed=false*.
- *actStandardSplitting*: this action splits the CU, as standard procedure in HM, into four blocks. It is an ETA and is valid when *featureRDOcostBlockComputed=true*.
- *actComputeRMDcostBlock*: this action computes the RMD cost of the CU. It is an IGA and is valid when *featureRMDcostBlockComputed=false*.
- *actComputeRDOcostBlock*: this action computes the RDO cost of the CU. It is an IGA and is valid when *featureRDOcostBlockComputed=false* and *featureRMDcostBlockComputed=true*.

It should be noted that no action is valid in a terminal state because the episode ends in that state.

The cost function is defined to reflect the main trade-off of an encoding system between computational complexity and visual quality. Since we aim at designing a fast encoder providing high rate-distortion performance, the time required by each action as well as the RD loss resulted from taking that action are considered as the costs for that action. The following definition provides a combination of these two costs:

$$c_a = c_{rd} + \eta c_T, \tag{12}$$

where c_a is the cost of each action. c_{rd} is the rate-distortion cost, c_T is the computational time we pay as a result of taking an action and η is the trade-off weight between c_{rd} and c_T.

V. Experimental Results

In this section, the performance of the proposed RL-based method for CU size decision is investigated and the obtained results are presented. For implementation, we use the HEVC test model HM 15.0 and a PC equipped with an Intel® Core™ i7-4790 CPU @ 3.60 GHz and 32 GB of RAM on a Windows 10 system. The configuration and profile are set to *all-intra* and *Main profile*, respectively. The trade-off weight η of (12) is set to 0.2. The results are based on the common test conditions recommended in [16], and are reported based on time reduction (TR), Bjøntegaard delta rate [19] (BD-Rate) and Bjøntegaard delta peak signal-to-noise ratio [19] (BD-PSNR). The sequences belong to five classes to cover various resolutions and applications. The results are averaged over four quantization parameters (QPs): 22, 27, 32 and 37. The training phase is performed based on Algorithm 1. For Q-function approximation in line 9 of the algorithm, the MLP is used which comprises one input layer, two hidden layers having 18 and 9 neurons, respectively, and one output layer. The number of layers and the number of neurons in each layer are selected through a grid searching optimization technique. The activation functions for the hidden layers are *tanh* and for the output layer is linear. The *tanh* activation function is a zero-centered function which overcomes the non-zero centric problem. This makes learning for the next layer more straightforward. A linear activation function can be used in the output layer to result in an unbounded value or to control the shape and range of output independently. The *RMSProp* is used as the optimizer, with learning rate of 0.001, in the training process and mean-squared error cost function is used to measure the performance of the model. The entire network is trained in a batch mode of size 32 and the updates are performed by back-propagation which iteratively updates the weights and minimizes the cost function.

Table I gives the results for the proposed RL-based method. This method leads to a time reduction of 51.3% with a BD-Rate increase of 0.84%; a trade-off which is considerable. In the exploring phase, the *RaceHorses* sequence is used as the training sequence, and hence, the average results are presented with and without considering this sequence. To compare our work to the other methods, Table II presents results of related works on coding unit size decision. As seen from this table, the proposed method offers an improved complexity-efficiency trade-off compared to other methods. It is about 20% faster than method in [13] at a slight penalty in RD performance. The quality is improved over [6] and both quality and complexity are improved over [4]. Compared to [4] we introduce more relevant textural features. In addition, in our proposed method, features are only computed when they are useful for CU size prediction. Otherwise, resources are not wasted for features computations. Moreover, while [4] uses previous frames for CU size decision, our method is only based on the current frame. This makes our method applicable to HEVC intra coding when it is applied to still image coding. It should be noted that our method is using features based on gradient. As a result, and compared to other works, it is specially useful for sequences where there are dominant directional gradients for most CUs or when the gradient for most CUs in the frame is close to zero. This is the case in sequences such as *SteamLocomotive* where there is one large object in a homogeneous background. In these sequences, CUs in the homogeneous areas become larger in size. As a result, CU size decision becomes more straightforward for our method. Our method also works well for sequences such as *BQTerrace*, where there are long distinct edges in a frame making it easier to find a dominant directional gradient and as a result the proper CU sizes. It should be noted that the proposed RL-based CU size decision method can be combined by any mode decision method such as the method proposed in [7] to achieve even faster HEVC intra coding.

TABLE I: Experimental results while implementing RL-based CU size decision compared to HM

Class	Video sequence	TR (%)	BD-Rate (%)	BD-PSNR (dB)
A (2560×1600)	Traffic	-54.5	1.21	-0.059
	PeopleOnStreet	-54.4	1.25	-0.060
	Nebuta	-52.9	0.21	-0.018
	SteamLocomotive	-52.8	0.22	-0.019
B (1920×1080)	Cactus	-49.6	0.85	-0.029
	Kimono	-59.4	1.68	-0.056
	ParkScene	-50.2	0.76	-0.032
	BasketballDrive	-57.1	1.33	-0.033
	BQTerrace	-46.9	0.42	-0.028
C (832×480)	BQMall	-51.8	0.61	-0.036
	PartyScene	-37.6	0.27	-0.021
	RaceHorsesC	-48.5	0.52	-0.038
	BasketballDrill	-50.3	0.90	-0.039
D (416×240)	RaceHorses	-43.6	0.65	-0.049
	BasketballPass	-50.3	0.65	-0.042
	BlowingBubbles	-42.9	0.27	-0.012
	BQSquare	-42.7	0.19	-0.020
E (1280×720)	FourPeople	-61.1	1.61	-0.078
	Johnny	-58.3	1.73	-0.088
	KristenAndSara	-60.3	1.45	-0.063
Average (with training sequence)		**-51.3**	**0.84**	**-0.041**
Average (without training sequence)		**-51.7**	**0.85**	**-0.041**

TABLE II: CU size decision methods comparison

	TR (%)	BD-Rate (%)
Proposed method	-51.3	0.84
[13]	-31.0	0.70
[6]	-55.5	1.01
[4]	-50.3	1.41

VI. Conclusion

In this work, we have proposed a novel method for coding unit size decision based on deep reinforcement learning. This method takes advantage of two approaches to reduce the intra coding complexity. In the first approach, the CU is early split and the mode decision computations are excluded from the current level. In the second approach, the CU splitting process is early terminated and all computations for the next levels are omitted which results in a considerable time reduction. In the proposed method, we have considered the CU size decision as a sequential decision making problem and formulate it by reinforcement learning. To this end, we have deployed a batch-mode RL to find a coding policy for coding unit size decision in the context of HEVC intra coding. In our problem, we aimed at finding a policy based on a finite set of observed sample transitions. Moreover, we used new features for CU size decision which are more relevant than features proposed in the literature. In summary, the competitiveness of the proposed method can be attributed to the fact that only most discriminative features are computed at any given step, resulting in a significant complexity reduction.

References

[1] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, 2012.

[2] L. Shen, Z. Zhang, and Z. Liu, "Effective CU size decision for HEVC intracoding," *IEEE Trans. on Image Processing*, vol. 23, no. 10, 2014.

[3] M. Jamali and S. Coulombe, "Coding unit splitting early termination for fast HEVC intra coding based on global and directional gradients," in *2016 IEEE Workshop on Multimedia Signal Processing (MMSP)*, 2016.

[4] Y. Zhang, Z. Pan, N. Li, X. Wang, G. Jiang, and S. Kwong, "Effective data driven coding unit size decision approaches for HEVC intra coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 11, pp. 3208–3222, 2018.

[5] A. Mercat, F. Arrestier, M. Pelcat, W. Hamidouche, and D. Menard, "Machine learning based choice of characteristics for the one-shot determination of the HEVC intra coding tree," in *Picture Coding Symposium (PCS)*, pp. 263–267, June 2018.

[6] D. Lee and J. Jeong, "Fast intra coding unit decision for high efficiency video coding based on statistical information," *Signal Processing: Image Communication*, vol. 55, pp. 121–129, July 2017.

[7] M. Jamali and S. Coulombe, "Fast HEVC intra mode decision based on RDO cost prediction," *IEEE Transactions on Broadcasting*, vol. 65, pp. 109–122, March 2019.

[8] M. Jamali, S. Coulombe, and F. Caron, "Method and system for fast mode decision for high efficiency video coding," *US Patent, US20160127725A1*, Nov 2018.

[9] D. Ruiz, G. Fernández-Escribano, J. L. Martínez, and P. Cuenca, "Fast intra mode decision algorithm based on texture orientation detection in HEVC," *Signal Processing: Image Communication*, vol. 44, 2016.

[10] M. Jamali and S. Coulombe, "RDO cost modeling for low-complexity HEVC intra coding," in *2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–5, May 2016.

[11] I. Marzuki, J. Ma, Y.-J. Ahn, and D. Sim, "A context-adaptive fast intra coding algorithm of high-efficiency video coding (HEVC)," *Journal of Real-Time Image Processing*, pp. 1–17, 2016.

[12] M. Jamali, S. Coulombe, and F. Caron, "Fast HEVC intra mode decision based on edge detection and SATD costs classification," in *2015 Data Compression Conference*, pp. 43–52, April 2015.

[13] A. BenHajyoussef, T. Ezzedine, and A. Bouallegue, "Gradient-based pre-processing for intra prediction in high efficiency video coding," *EURASIP Journal on Image and Video Processing*, Dec. 2017.

[14] W. Zhu, Y. Yi, H. Zhang, P. Chen, and H. Zhang, "Fast mode decision algorithm for HEVC intra coding based on texture partition and direction," *Journal of Real-Time Image Processing*, vol. 17, April 2018.

[15] H. Shim, S. J. Hwang, and E. Yang, "Why pay more when you can pay less: A joint learning framework for active feature acquisition and classification," *CoRR*, vol. abs/1709.05964, 2017.

[16] F. Bossen, "Common test conditions and software reference configurations," *Joint Collaborative Team on Video Coding (JCT-VC), JCTVC-L1100*, 2013.

[17] R. Sutton and A. Barto, *Reinforcement learning: An introduction*. Adaptive computation and machine learning, MIT Press, 1998.

[18] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *Journal of Machine Learning Research*, 2005.

[19] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," *VCEG-M33*, 2001.