Dynamic convolutional capsule network for In-loop filtering in HEVC video codec

Su, LiChao; Cao, Mengqing; Yu, Yue; Chen, Jian; Yang, XiuZhi; Wu, Dapeng

Published in:
IET Image Processing

Published: 07/02/2023

Document Version:
Final Published version, also known as Publisher's PDF, Publisher's Final version or Version of Record

License:
CC BY

Publication record in CityU Scholars:
Go to record

Published version (DOI):
10.1049/ipr2.12644

Publication details:

Citing this paper
Please note that where the full-text provided on CityU Scholars is the Post-print version (also known as Accepted Author Manuscript, Peer-reviewed or Author Final version), it may differ from the Final Published version. When citing, ensure that you check and use the publisher's definitive version for pagination and other details.

General rights
Copyright for the publications made accessible via the CityU Scholars portal is retained by the author(s) and/or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights. Users may not further distribute the material or use it for any profit-making activity or commercial gain.

Publisher permission
Permission for previously published items are in accordance with publisher's copyright policies sourced from the SHERPA RoMEO database. Links to full text versions (either Published or Post-print) are only available if corresponding publishers allow open access.

Take down policy
Contact lbscholars@cityu.edu.hk if you believe that this document breaches copyright and provide us with details. We will remove access to the work immediately and investigate your claim.

Download date: 26/03/2024
Dynamic convolutional capsule network for In-loop filtering in HEVC video codec

LiChao Su1 | Mengqing Cao2 | Yue Yu2 | Jian Chen2 | XiuZhi Yang2 | Dapeng Wu3

Abstract
Recently, several in-loop filtering algorithms based on convolutional neural network (CNN) have been proposed to improve the efficiency of HEVC (High Efficiency Video Coding). Conventional CNN-based filters only apply a single model to the whole image, which cannot adapt well to all local features from the image. To solve this problem, an in-loop filtering algorithm based on a dynamic convolutional capsule network (DCC-net) is proposed, which embeds localized dynamic routing and dynamic segmentation algorithms into capsule network, and integrate them into the HEVC hybrid video coding framework as a new in-loop filter. The proposed method brings average 7.9% and 5.9% BD-BR reductions under all intra (AI) and random access (RA) configurations, respectively, as well as, 0.4 dB and 0.2 dB BD-PSNR gains, respectively. In addition, the proposed algorithm has an outstanding performance in terms of time efficiency.

1 | INTRODUCTION

Similar to other lossy video coding standards, HEVC [1] adopts a block-based hybrid coding framework and introduces deblocking filtering (DBF) and sample adaptive offset (SAO) to alleviate the blocking artifacts [2] and ring artifacts [3]. Unfortunately, video coding loss is a very complex nonlinear problem, and existing in-loop filtering schemes have difficulty meeting the demand for high-resolution video in different application scenarios.

On this basis, in recent years, the CNN-based video/image filtering algorithms [4–14] have undergone a rapid development. Dong et al. [4] and Dai et al. [5] successively introduced CNN into JPEG (Joint Photographic Experts Group) and HEVC intra encoding as a post-processing filter, while some scholars used CNN to replace the in-loop filter in HEVC [6, 7]. In [6], Yu et al. put forward a new image-level loop filter based on CNN to replace DBF, and adopted a full variation model for network training to achieve better regularization effects. In [7], Lee et al. replaced SAO with two subnetworks, one of which calculates the optimal offset and the other estimates the type of error and content simultaneously. In addition, some improved CNN algorithms, such as a super resolution convolutional neural network (SRCNN) in [8], an enhanced deep convolutional neural network (EDCNNN) in [9] and the residual-reconstruction-based convolutional neural network (RRNet) in [10], have been proposed to replace the entire loop filter. Considering combining the advantages of CNN-based filters with traditional in-loop filters, the residual highway convolutional neural network (RHCNN) [11] and the multi-level feature residual network (MFRNet) [12] were designed as an auxiliary filter following DBF and SAO to increase coding gains. These methods made full use of the spatial information of the reconstructed images, while some other methods further considered the temporal information to improve the reconstruction effect [13–15]. These methods mentioned above so far have exhibited superior performance improvement. However, the authors in [16] and [17] argue that the CNN-based filtering methods apply only a single model to the whole image, and a single model cannot adapt well to all types of regional features in an image.
In [17], Jia et al. introduced content-aware mechanism into CNN to improve the adaptability of in-loop filtering, but its complexity is rather high. The in-loop filtering algorithms based on CNN mentioned above mainly improve the performance on traditional HEVC filter with two aspects: spatial and temporal. In spatial domain, some algorithms use various CNN models to partially or completely replace the filter modules in HEVC, some cascade them after HEVC’s loop filter to further improve the filter quality, and some improve the adaptive ability of filters from the perspective of attention mechanism. In time domain, reference frame is introduced to improve the filtering performance of prediction frame. However, most of these algorithms are difficult to balance coding gain and complexity.

In [18], a dynamic routing algorithm in CapsNet is proposed, which used the routing-by-agreement mechanism to adaptively weight each input vector instead of using fixed weights. If dynamic routing is localized, the network can let signals pass through different routing paths, which is equivalent to combining different CNN models. In addition, CapsNet is a unified network architecture that simultaneously performs image classification and image reconstruction, and the reconstruction result depends on the classification result. Therefore, we thought that the filtering result could be improved by classifying the local features. In fact, several works [19–21] have attempted to use CapsNet for different types of classification tasks with varying results. However, no research has yet been conducted to apply capsules, a new type of network structure, to video/image filtering. However, direct dynamic routing for the whole high-resolution image requires a huge amount of computation, which influenced the effectiveness of the algorithms seriously. To address these problems, an in-loop filtering algorithm based on a dynamic convolutional capsule network (DCC-Net) is proposed and adapted to the HEVC hybrid video coding framework as a new type of loop filter, which improves the filter’s self- adaptability as well as reduces the algorithm complexity. The main contributions of this study are as follows.

Firstly, we propose a novel localized dynamic routing algorithm, which improves the self-adaptability of the filter to different local features in the image.

Secondly, a dynamic segmentation method based on the idea of reinforcement learning is designed to train the network, realizing the image segmentation adaptively without the classification labels.

Finally, the proposed DCC-net is integrated into the HEVC encoding loop, and has an outstanding performance in terms of time efficiency compared to the previous algorithms.

2 | CAPSNET INTRODUCTION

A capsule [18] is a group of neurons that can learn to detect a specific category of targets or features from a given image area. The capsule outputs a vector whose magnitude indicates the probability of the existence of the target or feature and whose direction indicates other detailed attributes (such as position, size, rotation etc.) of the target or feature. That is, the characteristic attributes are directionally encoded according to specific categories. In this section, we briefly present the principle of the CapsNet.

CapsNet contains two capsule layers, namely, a PrimaryCapsules layer and a DigitCaps layer. The PrimaryCapsules layer is implemented by a conventional convolutional layer, which divides the feature map output from the previous convolution into multiple sub-feature maps by channel, representing different feature categories. The DigitCaps layer takes all the capsules output from the PrimaryCapsules layer as input. Each input capsule $u_j$ with dimension $a$ is mapped to its output space with dimension $ξ_j$ by a corresponding weight matrix $w_j^a$ to obtain a proposal vector (PV) $\hat{a}_{ji}^j$ as in (1), then the total input to a capsule $s_j$ is a weighted sum over all PVS. such as (2):

$$\hat{a}_{ji}^j(\xi_j) = w_j^a(\xi_j)u_j(a),$$

$$s_j = \sum c_j\hat{a}_{ji},$$

where the $c_j$ represents coupling coefficients that are determined by the iterative dynamic routing process.

The dynamic routing algorithm uses a routing-by-agreement mechanism to make inferences, which iterates to adjust the attention weight of each input vector to eliminate abnormal or irrelevant inputs, and outputs a better local feature representation. Because a capsule uses the magnitude to represent the probability of the corresponding category, a squash activation function is applied to normalize the magnitude of the capsule $s_j$ between 0 and 1.

3 | PROPOSED METHOD

The proposed algorithm primarily consists of three stages. The first stage of the algorithm introduces the localized dynamic routing and dynamic segmentation modules, as shown in Figure 1. The second stage explains the dynamic convolutional capsule network (DCC), which is composed of feature extraction, feature inference and feature reconstruction, as is shown in Figure 2. In the last stage, the DCC-net is integrated into HEVC encoder as shown in Figure 3.

3.1 | Localized dynamic routing and dynamic segmentation modules

3.1.1 | Localized dynamic routing

In order to improve the adaptability of the filter to different local features in the image and make the algorithm compatible with different sized input images, we need to localize the input of dynamic routing process in the convolution capsule layer. The pink rounded rectangle in Figure 1 shows a schematic diagram of localized dynamic routing, which inputs $N_s$ sub-feature maps with $N_s$ channels, and outputs $N_f$ sub-feature maps with...
FIGURE 1  Structure of localized dynamic routing and dynamic segmentation

FIGURE 2  Dynamic convolutional capsule network

FIGURE 3  DCC-net based in-loop filtering in HEVC encoder
N_{Z} channels. Each sub-feature map represents one feature category. Among them, the feature vector at each spatial position in each sub-feature map is a capsule node.

As is shown in Figure 1, we take the localization of the capsule at the spatial position \((x, y)\) in the \(i\)-th input sub-feature map as an example. First, we take the input capsule \(u_{i}\) with a neighborhood range of \(N_{\text{KK}} \times N_{\text{KY}}\) centering on the position \((x, y)\) as an input. Second, a \(N_{\text{KK}} \times N_{\text{KY}}\) convolution kernel is used to map \(u_{i}\) to \(\hat{u}_{j|i}\) in the output space linearly, which is called the proposal vector (PV) at \((x, y)\) as in:

\[
\hat{u}_{j|i}(\zeta, x, y) = \sum_{k_{x}, k_{y}} w_{j|i}(\zeta, k_{x}, k_{y}) u_{i}(a, x + k_{x}, y + k_{y}),
\]

\[
\forall x, y, i \in [1, N_{j}], j \in [1, N_{j}], a \in [1, N_{\lambda}], \zeta \in [1, N_{Z}],
\]

\[
k_{x} \in \left[x - \frac{1}{2}(N_{\text{KK}} - 1), x + \frac{1}{2}(N_{\text{KK}} - 1)\right],
\]

\[
k_{y} \in \left[y - \frac{1}{2}(N_{\text{KY}} - 1), y + \frac{1}{2}(N_{\text{KY}} - 1)\right],
\]

where \((x, y)\) refers to the spatial position of the capsule in the sub-feature map, \((k_{x}, k_{y})\) refers to the offset distance between the current convolution point and the convolution kernel center, and \(N_{\lambda}\) and \(N_{Z}\) represent the dimensions of the input capsules and the output capsules, respectively.

Third, all the \(u_{i}\)'s in the \(i\)-th input feature map constitute the proposal vectors map (PVM) \(\mathbf{u}_{j|i}\) of the \(j\)-th output sub-feature map from the \(i\)-th input sub-feature map. Do the same operation for other input sub-feature maps until \(N_{j} \times N_{j}\) PVMs are obtained. Subsequently, extract the capsules whose spatial position are \((x, y)\) from each PVM, treat them as \(N_{j} \times N_{j}\) PVs, and use them as the input of dynamic routing. Finally, the \(j\)-th output of dynamic routing \(v_{j}\) at the spatial position \((x, y)\) can be determined by:

\[
v_{j}(x, y) = \text{squash}\left(\sum_{i} c_{j|i} \hat{u}_{j|i}(x, y)\right).
\]

In this way, the routing is independent at different spatial location. Based on this, the algorithm can dynamically select different routing paths for different image regions. In other words, the algorithm is inherently adaptive to various regional features.

According to original dynamic routing algorithm [18], for each input sub-feature map, all the capsules within the range of \(N_{\text{KK}} \times N_{\text{KY}}\) around \((x, y)\) are first extracted without operating local convolution. Next, these capsules will be linearly mapped to the output space of the output sub-feature maps at the corresponding position according to the formula (1). So, in the end, there will be \(N_{j} \times N_{j} \times N_{\text{KK}} \times N_{\text{KY}}\) PVs as the input for dynamic routing, which is \(N_{\text{KK}} \times N_{\text{KY}}\) times of the input for our localized dynamic routing algorithm. In other words, in the original algorithm, there are too many PVs participating in routing at each spatial location, which results in excessive parallel computing power, and is not conducive to the filtering operation of high-resolution images. Instead, the proposed localized dynamic routing algorithm alleviates the computational complexity by reducing the number of input capsules for dynamic routing. Furthermore, our proposed algorithm can be compatible with the input images of different sizes.

### 3.1.2 Dynamic segmentation

As is described in the section of Localized dynamic routing algorithm, \(N_{j}\) output capsules for each spatial location are obtained. To reduce the storage burden of large numbers of capsules, we try to segment the image by the following way: for each spatial location, a category which is corresponding to the output capsule with the largest module length is assigned to its neighborhood. For each spatial position, we reconstruct only the capsule with the largest magnitude and ignore the capsules corresponding to other categories, that is, only the most significant local feature is reconstructed. The process is called dynamic segmentation (DS) algorithm, which is shown in blue rectangle in Figure 1. In this way, the network can selectively train optimal local maps for different local feature categories, further increasing the adaptability of the filtering algorithm to different local features. The above algorithm is described in detail below.

First, the probability of the capsule of the \(j\)-th category at location \((x, y)\) is calculated as:

\[
p_{j}(j) = \text{softmax}\left(\sqrt{\sum_{\zeta} v_{j}(\zeta, x, y)^{2}}\right),
\]

where \(p_{j}(j)\) is a probability distribution of the \(j\)-th output capsule normalized by the softmax function.

Second, the regional feature category at location \((x, y)\) can be determined by:

\[
\hat{l}_{j} = \text{argmax}_{j} p_{j}(j).
\]

Finally, the most significant feature representation \(\mathbf{V}'\) will be reconstructed selectively from \(\mathbf{V}\), which is the output of localized dynamic routing. Figure 4 shows the schematic diagram of the dynamic segmentation from \(\mathbf{V}\) to \(\mathbf{V}'\), where each smallest cuboid represents an output capsule. It is assumed that there are only \(2 \times 2\) spatial positions and 3 output categories in the \(\zeta\)-th output channel. Each red capsule represents the category with the highest probability in that location. For instance, \(l'_{2}(x, y, \zeta) = r_{2}(x, y)\) is the capsule reconstructed and output after using dynamic segmentation algorithm at the spatial position \((x, y)\). Similarly, we can obtain output capsules at the positions \((x, y + 1), (x + 1, y)\) and \((x + 1, y + 1)\) positions, thus constituting the final output feature \(l'\) in the \(\zeta\)-channel.

Since there are no classification labels and we want the categories at each pixel position to minimize distortion of the output image, we adopt reinforcement learning [22] to train the distribution of \(p_{j}\). Specifically, for each pixel position, we can use the negative output distortion of its neighborhood as a reward to adjust the probability of the current classification. The local
reward at position \((x, y)\) is defined as follows:

\[
R(x, y) = - \sum_{k_x, k_y} \omega(k_x, k_y) [\hat{F}(x + k_x, y + k_y) - F(x + k_x, y + k_y)],
\]

(7)

where \(\hat{F}(x + k_x, y + k_y)\) and \(F(x + k_x, y + k_y)\) refer to the filtered result and the corresponding original pixel at the current spatial position, respectively. \(\omega\) is a set of customized weights for calculating the reward, which represents the correlation between the output distortion of each pixel in each neighborhood and the category \(\hat{I}_{xy}\) in the neighborhood center. Each component of \(\omega\) will be non-negative, thus \(R(x, y)\) is always non-positive.

The goal of training is to minimize the following single sample loss function for DS with respect to network parameters \(\theta\) as follows:

\[
L_{DS}(\theta) = \sum_{x, y} \left( \int_{\theta} - R(x, y)(d(\log p_{xy}(\hat{I}_{xy}|\theta))) \right).
\]

(8)

In fact, DS is a dynamic classification algorithm for local features, whose goal is to obtain a local reconstruction with less distortion. In addition, dynamic segmentation can also be regarded as a kind of regularization, for that segmentation is performed to enhance reconstruction, and reconstruction is used to strengthen segmentation. The training algorithm will converge better only when segmentation and reconstruction achieve synergy.

### 3.2 Dynamic convolutional capsule network structure

Based on the algorithms of localized dynamic routing and dynamic segmentation, we design a new type of network structure named dynamic convolutional capsule (DCC), which is composed of three stages: feature extraction, feature inference and feature reconstruction, as is shown in Figure 2.

In the first stage, the input images are cascaded and fed into a feature extraction network composing of a 1-layer 64-channel input convolutional layer and 4-layer 64-channel convolution residual blocks to obtain the feature maps.

The second stage is the feature inference network, which consists of PrimaryCapsules layer, DigitCaps layer and DecisionCapsules layer. First in the PrimaryCapsules layer, the feature maps obtained from the previous level are mapped into multiple sub-feature maps by a 1-layer 64-channel convolutional layer, namely PrimaryCapsules layer. Next, these sub-feature maps are sent to the DigitCaps layer to perform localized dynamic routing so as to obtain deeper feature representations, that is, DigitCaps, for multiple categories. Then the DigitCaps input to DecisionCapsules layer are dynamically segmented and the DecisionCapsules with highest probability are finally sent to the feature reconstruction network.

In the last stage, the original image is reconstructed through the feature reconstruction network, which is composed of 1-layer 64-channel input convolutional layer, 4-layer 64-channel convolutional residual blocks, and 1-layer 1-channel output convolutional layer. Similar to the deconvolution network, it receives the DecisionCapsules output from the feature inference network, and maps each feature vector into an image patch. By the superposition of these patches, the output image is reconstructed.

In addition, we use the residual learning technique [23] by adding a skip connection between the input and output to increase the speed of training.

The entire inference process of the proposed network can be expressed as an end-to-end mapping:

\[
(\hat{F}, p_{xy}) = \text{model}(F^\text{IN}, \cdot),
\]

(9)

where \(F^\text{IN}\) is the network input, which consists of a cascade of unfiltered image and the reference image; \(\hat{F}\) represents the network output result, \(\theta\) represents the learnable network parameters, and the category of each pixel is represented by its probability distribution \(p_{xy}\).

As we know, losses are inevitable in the reconstruction network, which are called the reconstruction losses of the network. The reconstruction losses of network consist of two parts, as is defined in (10).

\[
L(\theta) = L_{\text{SSE}}(\theta) + \lambda L_{DS}(\theta),
\]

(10)
where $\lambda$ is a hyperparameter called Lagrange multiplier, which represents the weight ratio of the two parts of the loss. $L_{DS} (\cdot)$ is the single subsample loss function after dynamic segmentation as shown in (8), while $L_{SSE} (\cdot)$ is the sum-of-square error (SSE) between the filtered result and the original image $F$, which is formulated as:

$$L_{SSE} (\theta) = \left\| \hat{F} (FIN, \theta) - F \right\|^2. \quad (11)$$

Hence, according to formulas (10) and (8), the gradient function $g(\theta)$ of the loss function $L(\theta)$ with respect to the network parameter $\theta$ can be obtained by (12):

$$g(\theta) = \nabla_{\theta} L_{SSE} - \lambda \sum_{x,y} R(x,y) \nabla_{\theta} \log p_{xy}(\hat{I}_{xy} | \theta), \quad (12)$$

where $\nabla_{\theta}$ represents the partial derivative with respect to $\theta$.

### 3.3 Integration into HEVC

The proposed DCC-net is integrated into the HEVC encoder as a filter to replace the DBF as shown in Figure 3. According to the rate-distortion optimization (RDO) strategy, the encoder compares the image-level distortion of DCC-net and DBF to determine whether the current reconstructed frame uses DCC-net or DBF, and then transmits a 1-bit of filter control flag to indicate the result of selection in Slice-Head. Afterwards, SAO is employed to further reduce the artifacts in the outputs from DCC-net or DBF adaptively.

### 4 EXPERIMENT

In this section, we evaluate the performance of the proposed DCC-net for Y component through the following experiments. The section of Experimental settings introduces the network training process and the test configuration, followed by experimental results and performance analysis.

#### 4.1 Experimental settings

##### 4.1.1 Network training

We adopted the image sets BSDS500 [4] and DIV2K [24] as training data. Under the all intra (AI) configuration with the quantization parameters (QPs) set to 22, 27, 32, 37, we encoded all video frames with the HM-16.7 [25] reference software and extracted unfiltered reconstructed images, predicted images and original images from the encoder. For each input image, we cascaded the unfiltered image and the predicted image as the input $FIN$ and used the original image $F$ as the label to constitute training set $D = \{FIN, F\}$.

We trained with 4 QPs (22/27/32/37) separately under the all intra configuration. The training environment was PyTorch on a computer equipped with an NVIDIA GTX-1080Ti GPU acceleration. The training process used here was in accordance with Procedure 1, where $N_E$ represents the number of training iterations, the lagrange multiplier $\lambda$ is set as 0.1 and the learning rate $\epsilon$ is set as 0.001. In the training process, the calculation of the gradient function $g$ is divided into two stages, one step is the gradient of the $L_{SSE}$ term as shown in line 9, and the other step is adding the gradient of the $L_{DS}$ term as shown in line 12. In particular, if dynamic segmentation is not used in the inference process to select the output capsule with the highest probability at each position, all the output capsules are used for reconstruction, and the operations in lines 11 and 12 can be ignored. In addition, we used steepest descent method (SGD) to update the network parameter $\theta$ in step 15. In fact, we could use the Adam optimizer to update $\theta$.

##### 4.1.2 Test configuration

We integrate the model into the HEVC test module HM-16.7 using the deep learning framework LibTorch [26]. The frames under QP 22-26 /27-31 /32-36 /37-41 share the model trained under QP 22 /27 /32 /37. We measured the Bjontegaard delta bitrate (BD-BR) and the Bjontegaard delta peak signal-to-noise rate (BD-PSNR) [27] of Y component to denote the coding performance under all intra (AI), random access (RA) and low-delay P (LDP) configurations. The test QPs are 22, 27, 32, and 37, and the test sequences are Class A, B, C, D and E [28] (Algorithm 1).
4.2 Results and analysis

4.2.1 Ablation experiments

In order to verify the improvement effect of each component for the proposed network structure in Figure 2, we train and verify the following network structures: CNN-net, which only contains feature extraction and feature reconstruction modules; CC-net, which contains not only feature extraction and reconstruction modules, but also feature inference module including PrimaryCapsule and DigitCaps layers; DCC-net, which is the complete network architecture as shown in Figure 2. The peak signal-to-noise rate (PSNR) gains of the three network structures on the training set and the validation set of DIV2K relative to the original HM-16.7 algorithm are shown in Table 1. It can be seen from Table 1 that after adding the capsule network with dynamic routing and dynamic segmentation, the PSNR gains on the training set and the validation set are higher than those of CNN-net. Although the PSNR gains of CC-net are slightly higher than those of DCC-net, it needs encoding more network parameters.

In order to illustrate the generalization of local dynamic routing and dynamic segmentation algorithms integrated into HEVC, we test our algorithms with the sequences of Class A to E under QPs 22, 27, 32 and 37, by taking standard HM-16.7 as anchor when DBF and SAO are turned on. Experiments show that the model trained under the AI configuration is also effective under the RA and LDP configurations. The BD-BR reductions and the BD-PSNR gains of the CC-net and DCC-net are shown in the Tables 2 and 3, respectively. It is observed from the tables that 7.9%, 5.4% and 4.1% BD-BR saving can be achieved under AI, RA and LDP configurations for the CC-net, and 7.9%, 5.9% and 4.3% BD-BR saving can be achieved under AI, RA and LDP configurations for the DCC-net, respectively. In addition, our proposed DCC-net obtains 0.4 dB, 0.2 dB and 0.2 dB BD-PSNR gains under the above three configurations similar to those of CC-net. To sum up, DCC-net has better generalization than CC-net.

4.2.2 Objective comparison

To better demonstrate the improvement of coding efficiency, the rate-distortion curves (RD-curve) of the anchor and our proposed algorithm for some standard sequences are given in Figure 5, where the horizontal axis is the bit-rate, and the vertical axis is the PSNR. Through the rate distortion curves of DCC-net (red) and HM16.7 (blue), we can intuitively see the improvement of our methods in the rate-distortion (RD) performance.

To further verify the advantage of our algorithm, we compare BD-BR of DCC-net and CC-net to those with the current algorithms such as Dai [5], Lee [7], Yu [6], Wei [10] and Jia [17]. The results are showed in Table 4. As indicated in Table 4, both CC-net and DCC-net obtain 7.9% BD-BR reduction on average for luma channel under AI configuration, which is higher than that of other methods except [10]. Although [10] has better performance in AI configuration, but poor generalization ability makes the performance low in RA configuration. In addition, our proposed DCC-net obtains 5.9% BD-BR reductions on average under RA configuration outperforming other methods except [17], which achieves slightly better objective quality than ours at the cost of high complexity. The test results show that our algorithm is effective for sequences with complex motion foreground, such as Class D and Class E. This may be because the DCC-net is like a multi-modal CNN model, and is therefore suitable for sequences of Classes D and E with rich foreground targets and relatively fixed background.

4.2.3 Subjective assessment

Some regions of compressed video sequences are taken as examples in Figure 6 with QP=32 under the RA configuration. It is observed from Figure 6a that the collar in Kimonol has the most abundant texture folds and line contrast by using our proposed DCC-net algorithm. Similarly, for RaceHorses in Figure 6b, the edges of the human eyes are much clearer by our approach than other methods with varying degree of ringing blur. In a word, these examples show that our approach achieves better visual quality especially for the regions with rich detail features.

4.2.4 Time complexity

Furthermore, the encoding and decoding time complexity of the proposed DCC-net are discussed under RA configuration in this subsection. The time overhead, denoted as T, is calculated as in:

\[ T = \frac{T_{pro} - T_{HM}}{T_{HM}} \tag{13} \]

where \( T_{HM} \) is original enc/dec time of HM-16.7 and \( T_{pro} \) is the enc/dec time of our proposed algorithm. All of the complexity evaluations are conducted with GPU acceleration, and the results with GPU are provided in Table 5. It is obvious from Table 5 that the time overheads of our encoding and decoding complexity (i.e. \( T_{enc} \) and \( T_{dec} \)) are 0.11 and
### Table 2: Test results of CC-net

<table>
<thead>
<tr>
<th>Class</th>
<th>Sequence</th>
<th>AI</th>
<th>RA</th>
<th>LDP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BD-BR (%)</td>
<td>BD-PSNR (dB)</td>
<td>BD-BR (%)</td>
</tr>
<tr>
<td>A</td>
<td>PeopleonStreet</td>
<td>-7.9</td>
<td>0.5</td>
<td>-2.9</td>
</tr>
<tr>
<td></td>
<td>Traffic</td>
<td>-8.5</td>
<td>0.5</td>
<td>-6.3</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>-8.2</td>
<td>0.5</td>
<td>-4.7</td>
</tr>
<tr>
<td></td>
<td>Kimono</td>
<td>-5.8</td>
<td>0.2</td>
<td>-1.4</td>
</tr>
<tr>
<td></td>
<td>ParkScene</td>
<td>-6.5</td>
<td>0.3</td>
<td>-4</td>
</tr>
<tr>
<td>B</td>
<td>Catus</td>
<td>-6.9</td>
<td>0.3</td>
<td>-6.2</td>
</tr>
<tr>
<td></td>
<td>BasketballDrive</td>
<td>-6.0</td>
<td>0.2</td>
<td>-3.1</td>
</tr>
<tr>
<td></td>
<td>BQTerrace</td>
<td>-4.2</td>
<td>0.3</td>
<td>-8.2</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>-5.9</td>
<td>0.3</td>
<td>-4.6</td>
</tr>
<tr>
<td></td>
<td>BasketballDrill</td>
<td>-12.3</td>
<td>0.6</td>
<td>-4.6</td>
</tr>
<tr>
<td></td>
<td>BQMall</td>
<td>-7.9</td>
<td>0.5</td>
<td>-5.1</td>
</tr>
<tr>
<td>C</td>
<td>PartyScene</td>
<td>-4.9</td>
<td>0.4</td>
<td>-2.1</td>
</tr>
<tr>
<td></td>
<td>RaceHorses</td>
<td>-4.8</td>
<td>0.3</td>
<td>-2.7</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>-7.5</td>
<td>0.5</td>
<td>-3.6</td>
</tr>
<tr>
<td></td>
<td>BasketballPass</td>
<td>-8.8</td>
<td>0.5</td>
<td>-4.7</td>
</tr>
<tr>
<td></td>
<td>RaceHorses</td>
<td>-7.9</td>
<td>0.5</td>
<td>-5.1</td>
</tr>
<tr>
<td>D</td>
<td>BQSquare</td>
<td>-4.9</td>
<td>0.4</td>
<td>-2.1</td>
</tr>
<tr>
<td></td>
<td>BlowingBubbles</td>
<td>-4.8</td>
<td>0.3</td>
<td>-2.7</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>-7.5</td>
<td>0.5</td>
<td>-4.2</td>
</tr>
<tr>
<td></td>
<td>FourPeople</td>
<td>-10.9</td>
<td>0.7</td>
<td>-10.6</td>
</tr>
<tr>
<td>E</td>
<td>Johnny</td>
<td>-10.1</td>
<td>0.4</td>
<td>-9.7</td>
</tr>
<tr>
<td></td>
<td>KristenAndSara</td>
<td>-10.1</td>
<td>0.5</td>
<td>-9.7</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>-10.4</td>
<td>0.5</td>
<td>-10</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td>-7.9</td>
<td>0.4</td>
<td>-5.4</td>
</tr>
</tbody>
</table>

**Figure 5** RD curves of DCC-net for BQMAll (Class C) and FourPeople (Class E) under AI, RA and LDP configurations.
6.15 on average, respectively. By comparison, the encoding complexities for Dai [5] and Jia [17] are 1.1 and 1.3, respectively, while the corresponding decoding complexities are 44.59 and 116.56, respectively. To be more specific, the encoding time of our proposed algorithm is 10 times faster than those of Dai [5] and Jia [17], while the decoding time is 7.3 and 19.0 times faster than those of Dai [5] and Jia [17], respectively. In other words, our approach with GPU consumes the least time among all methods. The contrast experiments demonstrated that the proposed DCC-net as a learning-based in-loop filter has an outstanding performance in terms of time efficiency.
5 | CONCLUSION

This paper proposes an in-loop filter algorithm based on a dynamic convolutional capsule network with localized dynamic routing and dynamic segmentation algorithms. Experiment results show that the algorithm effectively improves the coding performance, and reduces the BD-rate by 7.9% under the AI configuration and 5.9% under the RA configuration, respectively. Since video signals have strong temporal correlations in the time domain, we plan to improve the algorithm from the perspective of the time domain in future work. Furthermore, the author will try to embed the algorithm into VVC, AV1 and other new generation video coding standards, as well as introduce new comparison metric, such as MS-SSIM or VMAF, in the future.

ACKNOWLEDGEMENT

The paper is supported by the National Natural Science Foundation of China(62001117 and 61902071).

CONFLICT OF INTEREST

We declare that we have no conflicts of interest to this work. And we declare that we do not have any commercial or associational interest that represents a conflict of interest in connection with the work submitted.

DATA AVAILABILITY STATEMENT

Data not available due to ethical/legal/commercial restrictions.

ORCID

LiChao Su https://orcid.org/0000-0003-2641-5901
Jian Chen https://orcid.org/0000-0001-7556-3121

REFERENCES


https://doi.org/10.1049/ipr2.12644