IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received September 12, 2018, accepted October 9, 2018, date of publication October 22, 2018, date of current version November 14, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2876864

Reduction of Video Compression Artifacts Based on Deep Temporal Networks

JAE WOONG SOH[1], JAEWOO PARK[1], YOONSIK KIM[1], BYEONGYONG AHN[2], HYUN-SEUNG LEE[1,2], YOUNG-SU MOON[2], AND NAM IK CHO[1], (Senior Member, IEEE)
[1]Department of Electrical and Computer Engineering, INMC, Seoul National University, Seoul 08826, South Korea
[2]Visual Display Division, Samsung Electronics Co. Ltd., Suwon 16677, South Korea

Corresponding author: Nam Ik Cho (nicho@snu.ac.kr)

This work was supported in part by Samsung Electronics Co., Ltd., and in part by Projects for Research and Development of Police Science and Technology through the Center for Research and Development of Police Science and Technology and the Korean National Police Agency under Grant PA-C000001.

ABSTRACT It has been shown that deep convolutional neural networks (CNNs) reduce JPEG compression artifacts better than the previous approaches. However, the latest video compression standards have more complex artifacts than the JPEG, including the flickering which is not well reduced by the CNN-based methods developed for still images. Moreover, recent video compression algorithms include in-loop filters which reduce the blocking artifacts, and thus post-processing barely improves the performance. In this paper, we propose a temporal-CNN architecture to reduce the artifacts in video compression standards as well as in JPEG. Specifically, we exploit a simple CNN structure and introduce a new training strategy that captures the temporal correlation of the consecutive frames in videos. The similar patches are aggregated from the neighboring frames by a simple motion search method, and they are fed to the CNN, which further reduces the artifacts. Experiments show that our approach shows improvements over the conventional CNN-based methods with similar complexities for image and video compression standards, such as MPEG-2, AVC, and HEVC, with average PSNR gain of 1.27, 0.47, and 0.23 dB, respectively.

INDEX TERMS Advanced video coding (AVC), compression artifacts, convolutional neural networks (CNN), high efficiency video coding (HEVC), video compression.

I. INTRODUCTION

The standards for image and video compression such as JPEG [1], H.262/MPEG-2 [2], H.264/AVC [3], and H.265/HEVC [4] are widely used to save transmission bandwidth and storage space. They include lossless and lossy compression modes, where the lossy compressed image and video suffer from various compression artifacts, namely blocking, ringing, blurring, mosquito, contour, flickering, etc. Hence there have been a large number of methods for the restoration of the original image from the compressed ones, which is an ill-posed, non-invertible problem due to the quantization process. Many works were mostly focused on the JPEG artifacts because it is the most widely used method for the still image compression [5], [6].

Recently, CNNs have shown great success in high-level vision tasks such as image classification [7]–[9], object detection [10], [11], and semantic segmentation [12], [13]. Additionally, inspired by the great success in high-level vision tasks, CNNs were also adopted for low-level vision tasks, which is to find a mapping from the degraded image to the desired one, such as super-resolution [14]–[16] and denoising [17]. Also, some CNN architectures were developed for the JPEG artifact reduction [6], [18], [19].

Since the dominant artifact with the JPEG is the blocking artifacts, it seems that the CNN-based methods easily learn the filters for the blocking artifacts reduction and improve the PSNR more than 1 dB in a wide range of compression quality factor. In contrast, the state-of-the-art compression algorithm, HEVC yields more complex artifact patterns because it has many compression modes and different sizes of coding units (CUs). For example, FIGURE 1a shows the artifacts in an HEVC intra-coded image, where the directional patterns caused by the intra-prediction are visible. Additionally, in the case of video compression, there are fluctuations or discontinuities in the quality and artifact patterns of consecutive frames, which appear to be flickering artifacts when playing the compressed videos. Eventually, dealing with the artifacts in videos is much more complicated than the case of still images.

In block-based coding methods, one of the most noticeable artifacts may be the blocking artifacts as shown in FIGURE 1b and FIGURE 1c. To alleviate such artifacts, recent compression standards such as AVC and HEVC introduced in-loop deblocking filter [20]. In addition, the HEVC introduced the sample adaptive offset (SAO) [21] to relieve general compression artifacts. These methods not only enhance the quality of each frame but also contribute to improving the compression performance. When the deblocking filter and SAO are applied, it becomes more difficult to obtain the gains by the post-processing because the deblocking and SAO can sufficiently reduce many kinds of artifacts. Hence, as will be shown in the experiments, applying the existing artifact removal CNNs to each frame does not much increase the PSNR of HEVC/AVC-decoded videos, and also does not alleviate the flickering as the frames are independently processed. In this respect, few works tried to improve the HEVC coded images using the CNN. Specifically, Dai *et al.* introduced a CNN for the artifacts removal of HEVC decoded frames [22], but it was limited to the restoration of intra-compressed images without using the deblocking filter and SAO. There are also some methods to replace the deblocking filter and SAO by CNN, which is shown to improve the PSNR compared to the original HEVC [23]. Recently, Yang *et al.* proposed a CNN for enhancement of HEVC compressed frames regardless of the encoding modes [24]. However, they did not exploit the temporal correlation between adjacent frames.

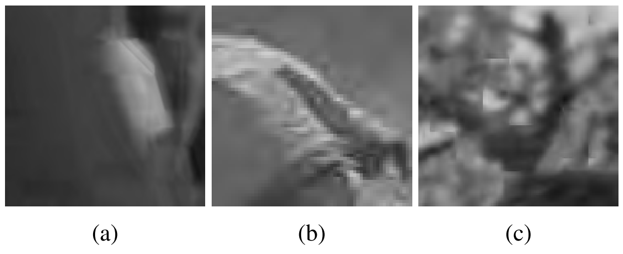

FIGURE 1. Examples of compression artifacts. (a) HEVC directional pattern artifacts (b) JPEG blocking artifacts (c) HEVC blocking artifacts.

In summary, there have been some post-processing CNNs for the reduction of artifacts in the decoded videos, but they have some limitations. Specifically, the post-processor proposed in [22] is applied to HEVC, but it was limited only to the intra-coding mode with the deblocking filter turned off. There is another post-processor that considered various coding modes [24], but it does not consider the temporal correlation between the adjacent frames and requires two networks to handle intra and inter frame separately. In this respect, we propose a new approach that can alleviate the above stated limitations. To be precise, we develop a CNN-based post-processor, and apply it for various kinds of image/video codecs working on various coding modes. Also, in addition to frame by frame information, our model exploits temporal information by applying the input with motion estimated patches.

II. RELATED WORK

A. ARTIFACTS REMOVAL NETWORKS FOR JPEG

The first work that used the CNN for the JPEG artifacts reduction is the Artifacts Reduction Convolutional Neural Network (AR-CNN) [6] which stacked four convolution layers with mean squared error (MSE) loss function. Svoboda *et al.* [18] used a deeper network and residual learning with additional loss function for the artifacts removal. In [19], they developed dual domain convolutional network (DDCN) which adopted the DCT domain prior as an additional input. Cavigelli *et al.* introduced the CAS-CNN [25] which adopts a deep network with multi-scale MSE losses for JPEG artifact reduction. Also, Kim *et al.* [26] modified the Inception module [8] for some low-level vision tasks, and showed that it is also effective and efficient for the JPEG artifacts reduction.

B. SPATIAL-TEMPORAL NETWORKS FOR VIDEO APPLICATIONS

There have been some deep networks that take multiple frames of a video sequence as the input for the CNN. By using the sequential frames as a tensor input, the CNN can capture the temporal as well as the spatial information from a video. Specifically, the deep temporal linear encoding network [28] and the temporal segment network [29] exploit the temporal information for enhancing the action recognition performance. Also, multiple frames are used for video denoising, prediction and video super-resolution by using the recurrent networks [30]–[33]. Also, the spatial-temporal network is proposed for replacing the in-loop filter in the video encoder for improving the compression performance [34]. A CNN structure for video super-resolution is proposed where motion compensated consecutive frames are fed to obtain super-resolved frames for videos [35].

C. CNNs FOR HEVC

In addition to the spatial-temporal networks [34] referenced above, there are also several methods that applied the CNN to the HEVC artifacts reduction. Specifically, Park and Kim replaced the in-loop filter and SAO in the encoder by a CNN [23] and obtained significant gains over the encoder with the in-loop filter. In the case of [22], they used a CNN as a post-processor to the decoded image, but it was limited to the intra-coded images and the test was done without turning on the in-loop filter in the encoder. Also, Yang *et al.* proposed a CNN to handle different properties of I and B/P compressed frames [24], but it does not exploit temporal correlation and requires separate CNNs to handle intra and inter frames.

III. DEEP TEMPORAL NETWORK

For the deep network that maps an input $\mathbf{X}$ to the output $\mathbf{F}(\mathbf{X}; \theta)$, its training is to find the set of parameters θ of the network that makes $\mathbf{F}(\mathbf{X}; \theta)$ as close to as the desired signal $\mathbf{Y}$.

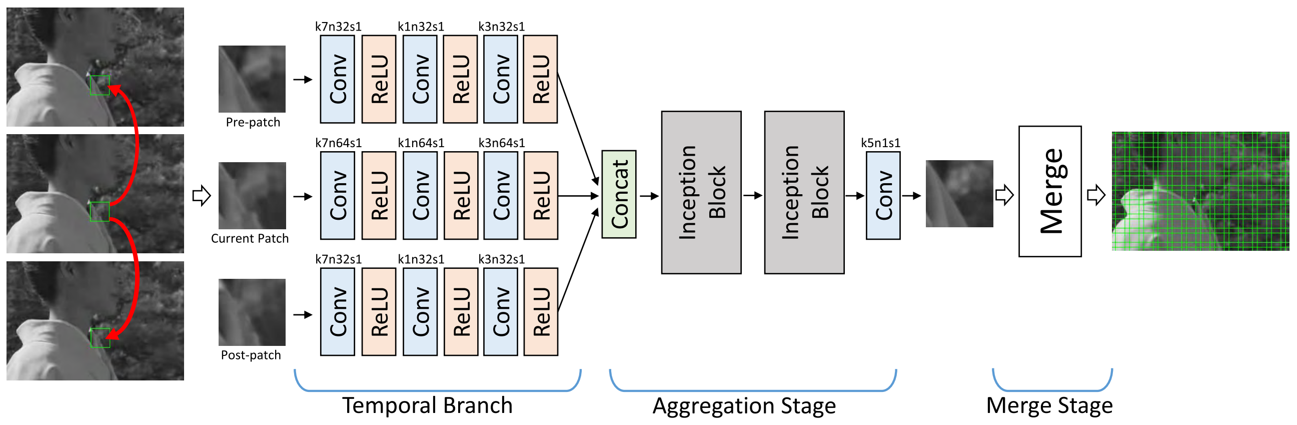

FIGURE 2. **The overall framework of the proposed network, where *k*, *n*, *s* above the convolution layers denote the kernel size, the number of output feature maps, and the convolution strides, respectively.**

In our case, the input is the set of three consecutive frames $\mathbf{X}^{(t-1)}$, $\mathbf{X}^{(t)}$, and $\mathbf{X}^{(t+1)}$, and the desired output denoted as $\mathbf{Y}^{(t)}$ is the original (uncompressed) frame. That is, we exploit the temporal dependencies of consecutive frames to find the current frame, which reduces the artifacts in the frame and also suppresses the temporal artifacts such as flickering. The overall architecture is shown in Fig. 2 which extracts the features from the consecutive frames by deploying three branches of convolution layers which are fed with the related patches from $\mathbf{X}^{(t-1)}$, $\mathbf{X}^{(t)}$, and $\mathbf{X}^{(t+1)}$ respectively. These are named as "temporal branches," whose outputs are then concatenated and fed to the following "aggregation branch." In the rest of this paper, we refer to our network as Artifact Reduction Temporal Network (ARTN).

A. TEMPORAL BRANCH

Each temporal branch in Fig. 2 consists of three convolution layers, each of which is followed by a rectified linear units (ReLU) [36]. The role of a temporal branch is to extract features from the corresponding frame and so it is constructed by simple convolution layers. As shown in the figure, the number of feature maps for the current frame is twice larger than that of the previous or next frame (64 vs. 32) for stressing the current frame and also for some possible mismatch of previous/next frames from the current one.

The artifacts reduction is processed patch by patch where the patch size is determined as 64×64 with the stride of 48 (which is 32+16 so that 1/4 of block length/width overlap with the neighboring ones), which is determined from the fact that the largest size of the coding unit in HEVC is 64×64. For the given patch from the current frame, the patches from the previous and next frames are chosen to be the close ones from the current one. Full search motion estimation (ME) can be used for finding the closet matching patches, but we just use the simple three step search (TSS) algorithm [37] for saving the computations.[1] Also, to cope with the case of abrupt scene change and the failure of ME, the previous (or next) frames' patches are discarded and replaced by the current frame's patch when the mean absolute difference between the patches exceeds a certain threshold (In the experiments, the threshold is set to 25,500 in every experiment).

[1]From the experiments, it is found that using the TSS gives almost the same performance as using the full search method.

B. AGGREGATION STAGE

The role of aggregation stage is to merge the extracted features from the temporal branches and enhance the features. We use the Inception-based network proposed in [26] which is shown to provide comparable denoising performance to the state-of-the-art methods with less number of parameters. This architecture is the modification of Inception module from GoogLeNet [8] which is shown in Fig. 3 for comparison. According to [8] the network-in-network structure in the Inception module helps to extract rich features by using various size kernels while requiring less number of parameters. The modified network-in-network in [26] is to remove the max-pooling of original architecture and, instead, add a larger 7×7 kernel filter. This helps to keep the features that are needed for the JPEG artifacts removal or skin detection as shown in [26] and [27].

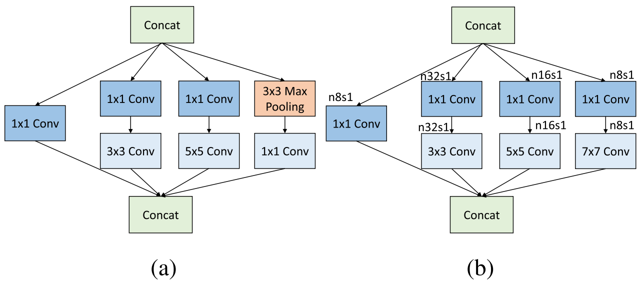

FIGURE 3. **Comparison of network-in-network structure from the (a) original Inception and (b) the modification for artifacts removal. (a) Original. (b) Modified.**

In this paper, we use just two Inception modules followed by one convolution layer as the aggregation stage. Stacking more modules in this stage (and also stacking more

TABLE 1. Parameters of ARTN.

Temporal Branch					
Layer	Pre-section	Current section	Post-section		Total
Conv1	$7 \times 7 \times 1 \times 32$	$7 \times 7 \times 1 \times 64$	$7 \times 7 \times 1 \times 32$		6,272
Conv2	$1 \times 1 \times 32 \times 32$	$1 \times 1 \times 64 \times 64$	$1 \times 1 \times 32 \times 32$		6,144
Conv3	$3 \times 3 \times 32 \times 32$	$3 \times 3 \times 64 \times 64$	$3 \times 3 \times 32 \times 32$		55,296
Aggregation Branch					
Inception1	$1 \times 1 \times 128 \times 8$	$1 \times 1 \times 128 \times 32$	$1 \times 1 \times 128 \times 16$	$1 \times 1 \times 128 \times 8$	8,192
		$3 \times 3 \times 32 \times 32$	$5 \times 5 \times 16 \times 16$	$7 \times 7 \times 8 \times 8$	18,752
Inception2	$1 \times 1 \times 64 \times 8$	$1 \times 1 \times 64 \times 32$	$1 \times 1 \times 64 \times 16$	$1 \times 1 \times 64 \times 8$	4,096
		$3 \times 3 \times 32 \times 32$	$5 \times 5 \times 16 \times 16$	$7 \times 7 \times 8 \times 8$	18,752
Conv8	$5 \times 5 \times 64 \times 1$				1,600
Total number of parameters					122,688

convolution layers in the temporal branches) may increase the performance of the system, but we choose to use the numbers of Inception modules and convolution layers as in Fig. 2 to keep the overall number of parameters close to that of baseline AR-CNN [6] that we compare. Specifically, Table 1 summarizes the number of parameters needed for our architecture, which is shown to be about 123K that is comparable to 106K of baseline AR-CNN. Compared to other recent artifacts removers such as L8 network [18] (220K) and CAS-CNN [25] (5,144K), the proposed method needs fewer parameters while providing better results as will be shown in the experiments.

C. MERGE STAGE

The final output frame is merged as the weighted sum of output patches. As stated previously, 1/4 of horizontal/vertical part of a block overlap with the neighboring ones. Hence there are some parts in the image which are the overlap of two blocks (horizontal or vertical), and also there are some parts which are the overlap of neighboring four blocks (horizontal, vertical and diagonal). For these overlapping parts, we define the Gaussian weights from the block center, multiply them to the pixel values, add the overlapping pixels and then divide by the sum of weights for normalization. To be precise, the weight for a patch is defined as

$$W(i,j) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{d(i,j)^2}{2\sigma^2}\right) \tag{1}$$

where $d(i,j)$ is the Euclidean distance from the center of patch defined as (0,0) to the coordinate (i,j). For each patch, the pixels that contribute the overlapping region are multiplied by the weights. Then all the weighted pixels in the overlapping regions are added and divided by the sum of weights.

D. NETWORK TRAINING

1) DATASET PREPARATION

In the next section, we will show the experimental results for all the widely used video codecs, i.e., MPEG-2, AVC, and HEVC. But we explain the training method only for HEVC due to the limited space and also because the training methods for other codecs are almost the same.

To train the network for the HEVC compression artifacts reduction, we prepare a dataset from HEVC common test sequences of JCT-VC and also from some additional sequences. The sequences are encoded by HM 16.9 software with the *random access main profile* with GOP of 8. The quality factors of QP = 34, 37, 42, and 47 are used for the compression, i.e., we train four models for the corresponding QPs. As addressed in the existing work [18], a single model that covers all the QPs (for the blind case that we do not know the QP of the encoder) performs worse than the models that are specifically trained for the given QP (non-blind case). We assume that we can have QP information from the decoder or we can estimate it by using the CNN-based QP estimator reported in [38].

Our main target resolution is FHD (1920 × 1080), and hence we prepare high-resolution training sets. Specifically, among the videos in the HEVC common test set, the low-resolution ones are excluded, leaving 20 as the training sequences. The network is trained with Y channel only. For constructing the training set, we conduct two steps:

- Base Step: Basic patch extraction for training.
- Refining Step: Refining extracted patches for enriched dataset.

TABLE 2. Sequences for the training set.

Sequence	Resolution (width × height)	Number of base patches	Number of refined patches
BasketballDrill	832 × 480	2420	1720
BasketballDrillText	832 × 480	2420	1825
BQMall	832 × 480	2860	2521
Cactus	1920 × 1080	12220	9230
ChinaSpeed	1024 × 768	4500	3397
Flowervase	832 × 480	1540	875
FourPeople	1280 × 720	6324	4843
Johnny	1280 × 720	6324	3047
Keiba	832 × 480	1540	1067
KristenAndSara	1280 × 720	6324	3591
Mobisode2	832 × 480	1540	640
ParkScene	1920 × 1080	6110	4395
PartyScene	832 × 480	2420	2370
RaceHorses	832 × 480	1320	1067
SlideEditing	1280 × 720	3162	3047
SlideShow	1280 × 720	5270	2871
Tennis	1920 × 1080	6110	3651
vidyo1	1280 × 720	6324	3993
vidyo2	1280 × 720	6324	4011
vidyo3	1280 × 720	6324	4400

TABLE 3. Sequences for the test set.

Sequence category	Sequence	Frames	Characteristics
FHD	BasketballDrive	501	Fast object motion
	BQTerrace	601	Aligned textures
	GTAV	500	Graphics
	Kimono	240	Irregular textures
	Pedestrian	375	Fast near camera motion

TABLE 4. Average PSNR and SSIM results for the test set.

HEVC		QP 34	QP 37	QP 42	QP 47
HEVC	PSNR	34.7265	33.4298	31.1635	28.8384
	SSIM	0.8821	0.8608	0.8202	0.7739
AR-CNN	PSNR	34.7929	33.5159	31.2827	28.9797
	SSIM	0.8828	0.8624	0.8242	0.7792
VRCNN	PSNR	34.7267	33.4296	31.1641	28.8390
	SSIM	0.8821	0.8608	0.8202	0.7739
VSRNet	PSNR	34.9092	33.6354	31.3942	29.0510
	SSIM	0.8846	0.8641	0.8259	0.7801
Baseline	PSNR	34.8638	33.5927	31.3770	29.0260
	SSIM	0.8846	0.8636	0.8254	0.7803
ARTN	PSNR	**34.9442**	**33.6639**	**31.4133**	**29.0622**
	SSIM	**0.8851**	**0.8650**	**0.8261**	**0.7805**

AVC		QP 34	QP 37	QP 42	QP 47
AVC	PSNR	35.3253	33.9823	31.6606	29.2120
	SSIM	0.8913	0.8695	0.8271	0.7777
AR-CNN	PSNR	35.3514	34.0071	31.7993	29.4555
	SSIM	0.8916	0.8699	0.8305	0.7861
Baseline	PSNR	35.6707	34.2924	32.0342	29.6045
	SSIM	0.8962	0.8751	0.8365	0.7903
ARTN	PSNR	**35.7868**	**34.4611**	**32.1313**	**29.6632**
	SSIM	**0.8981**	**0.8776**	**0.8377**	**0.7922**

MPEG-2		QP 20	QP 30
MPEG-2	PSNR	32.9493	31.3361
	SSIM	0.8435	0.8086
AR-CNN	PSNR	33.8265	32.2970
	SSIM	0.8658	0.8375
VSRNet	PSNR	34.0842	32.3552
	SSIM	0.8694	0.8388
Baseline	PSNR	34.0655	32.4580
	SSIM	0.8705	0.8409
ARTN	PSNR	**34.2318**	**32.5964**
	SSIM	**0.8719**	**0.8434**

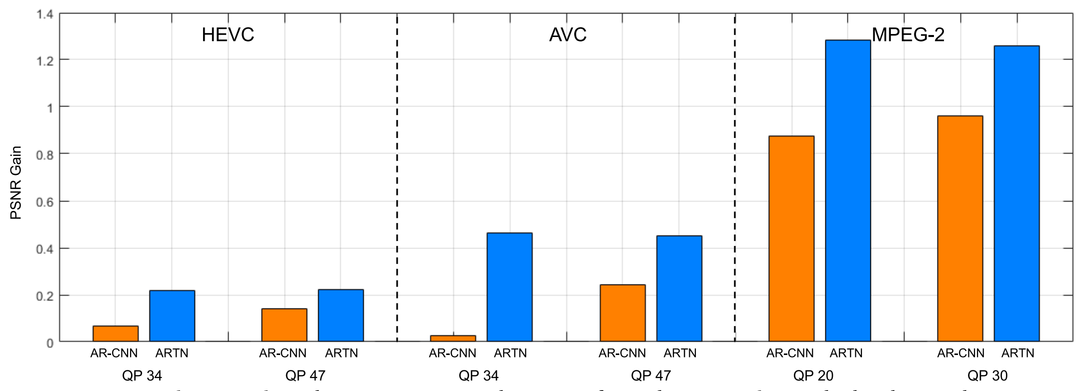

FIGURE 4. PSNR gain comparisons between AR-CNN and our ARTN for each compression method, at low- and high- compression rates.

The 64 × 64 patches are extracted with the stride of 40, from every 50 frames of the training sequences. The TSS parameter is set as $p = 15$. At the base step, the sequences used for the training, their resolution, and the total number of extracted patches are listed in TABLE 2.

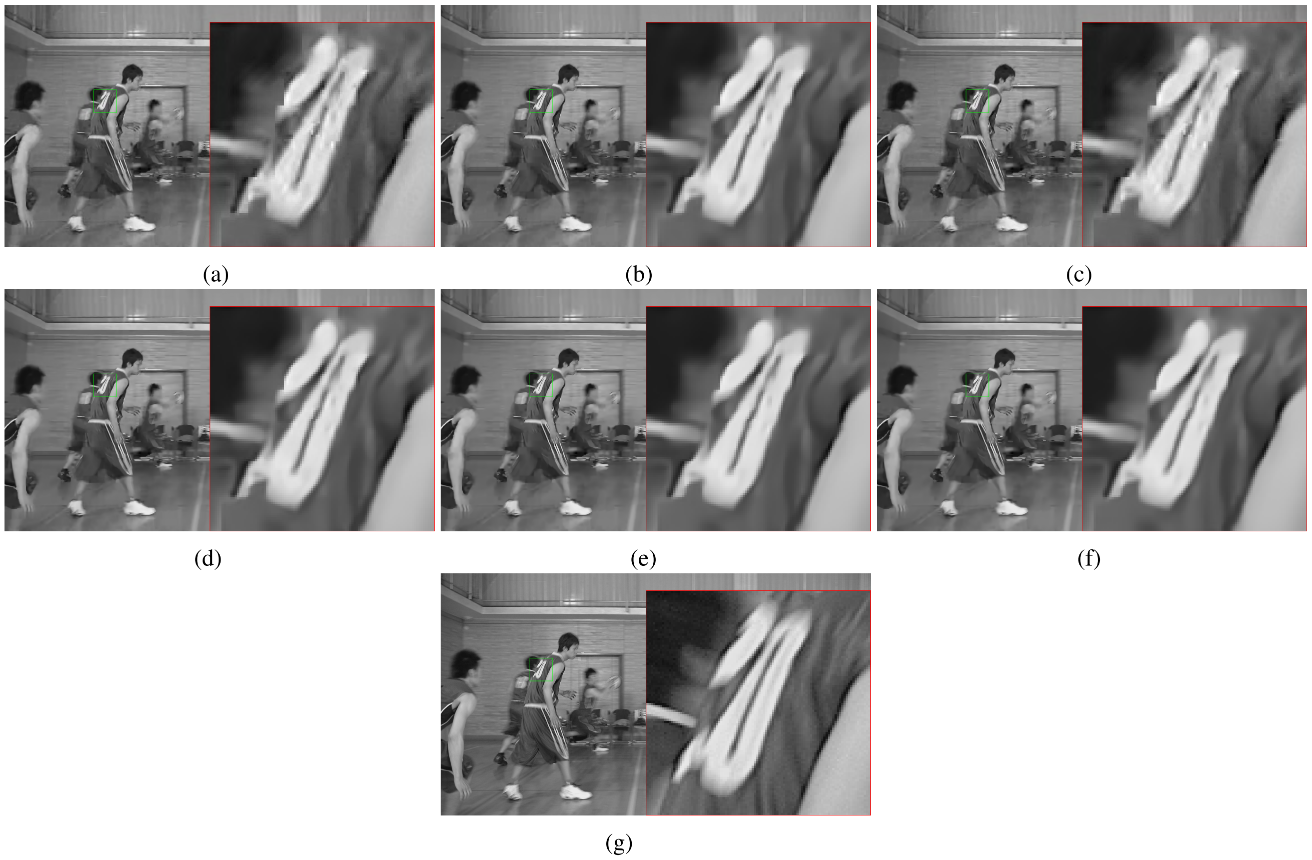

FIGURE 5. Visualization results on the sequence "BasketballDrive" of QP 42 (HEVC). (a) HEVC. (b) AR-CNN. (c) VRCNN. (d) VSRNet. (e) Baseline. (f) ARTN. (g) Ground Truth.

According to our extensive experiments, it is noted that the randomly extracted patches are not good for the training because there are some low-rank patches (*e.g.* flat regions) which do not contain the examples of compression artifacts. On the other hand, the patches with high variance contain many artifacts so that they can be good training samples. Hence, we refine the extracted patches by removing the unnecessary ones that have low variance. Specifically, the patches with variance less than 0.002 (when the pixel values are normalized into [0,1]) are removed from the dataset. TABLE 2 also shows the number of refined patches of each sequence. Since we remove some patches, we augment the data by rotation and flip ($\times 8$).

2) TRAINING DETAILS

We train our networks using L1 loss function instead of L2, based on the experiments in [39] that the L1 loss provides better results than the L2 in terms of PSNR. We implement the ARTN with Caffe framework [40] and use ADAM [41] optimizer. The batch size is 128, and the learning rate is given as 10^{-4} for the entire training. We train four models for HEVC: QP = 34, 37, 42, and 47; four models for AVC: QP = 34, 37, 42 and 47; and two models for MPEG-2: QP = 20 and 30.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

A. EXPERIMENT SETUP

We test our model with five sequences with FHD (1920×1080) resolution listed in TABLE 3. To accelerate the running time of our network, we put as many patches as possible to GPU memory, as a batch for the test environments. The peak signal-to-noise ratio (PSNR) and structure similarity (SSIM) [42] are measured for each of the frames reconstructed from the CNN, and the averaged values of all the frames are compared. For the evaluation of flickering artifacts, we define a flickering score (FS) based on the sum of squared difference (SSD).

To evaluate the effectiveness of temporal branch that we propose, we also implement a network without the temporal branch as an ablation study. Specifically, the branches that receive the pre- and post-patches in Fig. 2 are removed, which is called the "baseline," for the comparison. We also compare the proposed ARTN with the AR-CNN [6] by defining our network's complexity to be similar to AR-CNN. Also, in the case of HEVC, we compare our network with the VRCNN [22] which is proposed for the artifacts reduction in HEVC intra-coded image with the deblocking filter turned off. Additionally, for comparing our method with the existing network that uses temporal information, we modify the Video Super-Resolution Network (VSRNet) [35] which was

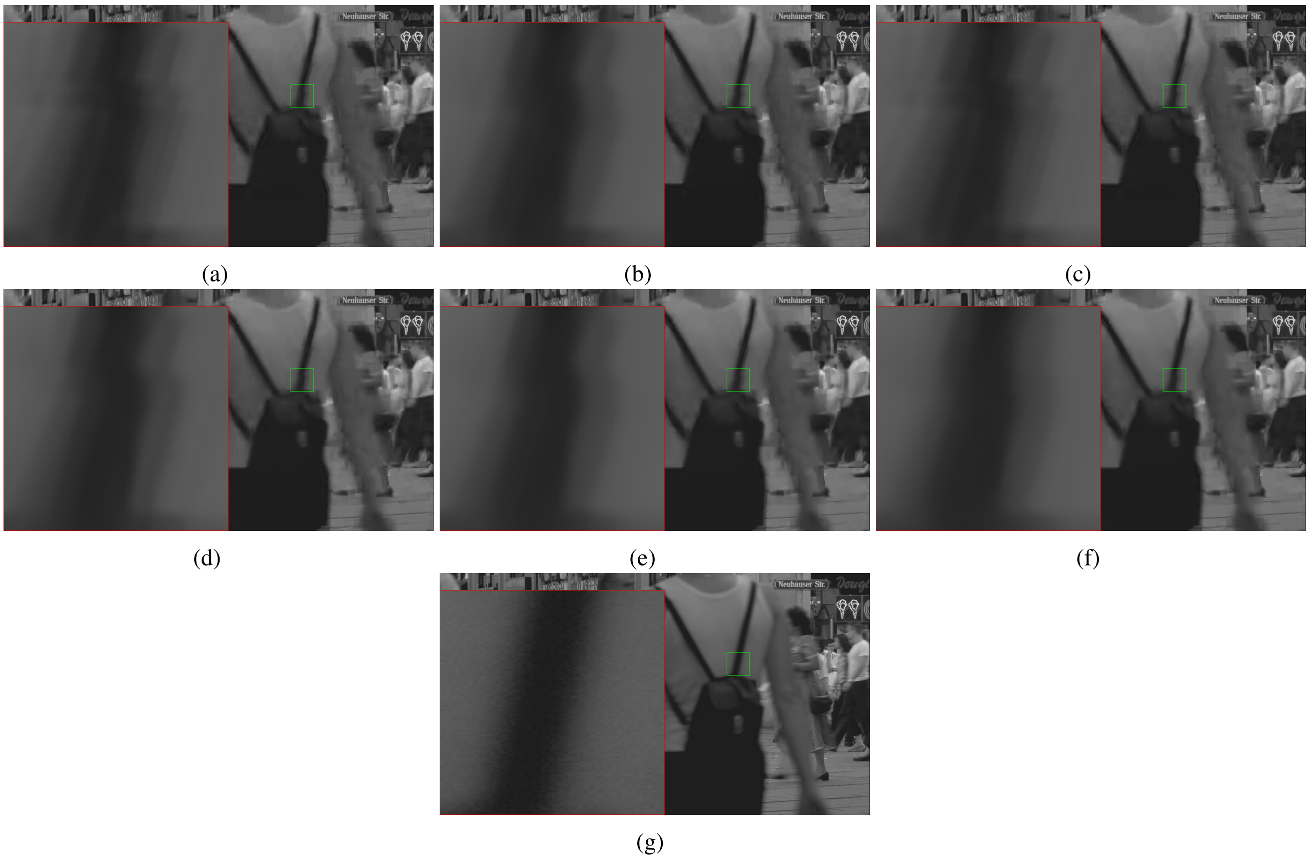

FIGURE 6. Visualization results on the sequence "Pedestrian" of QP 47 (HEVC). (a) HEVC. (b) AR-CNN. (c) VRCNN. (d) VSRNet. (e) Baseline. (f) ARTN. (g) Ground Truth.

originally developed for video super-resolution. We set the number of parameters of this network similar to ours, and also train and test it on the same condition as ours. Also, we re-train the AR-CNN with the same datasets as ours, because the original AR-CNN was trained only for the JPEG compression artifacts reduction. We also train the VRCNN with our dataset (with the deblocking filter turned on), because the original VRCNN is trained only for the intra-coded frames without turning-on the deblocking filter.

B. OBJECTIVE AND SUBJECTIVE COMPARISONS

The average PSNR and SSIM results on the test sequences are shown in TABLE 4. The best PSNR and SSIM results for each QPs are highlighted in bold. As shown, the ARTN achieves the best performance in both PSNR and SSIM compared to the others. We can see that although the VRCNN is effective for reducing the artifacts in intra-coded images according to [22], it is not in the case of turning on the deblocking filter and also in the case of inter-coded frames so that the overall performance is not improved. It is also noticeable that our baseline is better than the AR-CNN, which may demonstrate the effectiveness of our architecture while aggregating the temporal information gives additional gains. It is also seen that the importance of exploiting temporal dependencies between adjacent frames. Our ARTN shows better PSNR and SSIM gain than the VSRNet, while the VSRNet also achieves considerable performance gains by exploiting the temporal information between frames.

Fig. 4 is a graph that shows the PSNR gains of AR-CNN and ARTN for each of encoding methods, at low- and high-compression rates. It can be seen both methods show larger gain in the order of HEVC, AVC and MPEG-2, which also reveals the effectiveness of HEVC compared to MPEG-2. That is, the HEVC is effective enough that further post-processing does not much help whereas there are much to be improved in the case of MPEG-2. It is also observed that the AR-CNN gets a larger gain at a higher QP, whereas the proposed ARTN shows the consistent gain.

Finally, we visualize the results for HEVC images in FIGURE 5 and FIGURE 6, AVC in FIGURE 9 and FIGURE 10, and MPEG-2 in FIGURE 11 and FIGURE 12. By magnifying the images, we can see that the ARTN reduces the artifacts very well as compared to the others.

C. COMPARISONS WITH RECENT CNN-BASED HEVC POST-PROCESSOR

We additionally compare our ARTN with DS-CNN [24] which is a recent CNN for HEVC quality enhancement. Since

TABLE 5. Δ **PSNR (dB) comparisons between DS-CNN and ARTN. The best results are highlighted in bold face.**

Δ PSNR Results		
BasketballDrive		
QP 42	Intra frames	Inter frames
AR-CNN	0.1737	0.1603
DS-CNN	0.2281	0.2039
ARTN	**0.3015**	**0.2541**
BQTerrace		
QP 42	Intra frames	Inter frames
AR-CNN	0.3180	0.2103
DS-CNN	0.3789	0.2987
ARTN	**0.3877**	**0.3085**

the author code is not available, we compare the methods in TABLE 5 only for the common test sequences that appear in the paper, specifically for the BasketballDrive and BQTerrace. That is, the numbers for the DS-CNN and AR-CNN are from the original paper of DS-CNN [24]. As shown in the table, our ARTN achieves better Δ PSNR than the others.

D. MEASUREMENTS OF FLICKERING

Since we are dealing with video, we also measure the temporal artifacts, *i.e.*, the degree of flickering which is caused by the fluctuation of image quality and artifacts patterns between the consecutive frames. There have been some studies to objectively measure the flickering for developing a post-filter or better encoding modes [43], [44] based on the sum of squared difference (SSD). However, since both works targeted on the AVC compression algorithm, the metrics in these studies are not adaptable to our other targets (MPEG-2 and HEVC). Hence, based on these works, we also define a similar measure which will be called the flickering score (FS).

According to [43] and [44], and our observations, the flickering artifacts are

- mainly caused by the changes in intra-prediction modes between the successive frames
- perceived mostly in low motion areas, and between the intra-coded and inter-coded frames
- occurred by the different loss of information between the successive frames due to coarse quantization.

Hence we need to find the candidate areas first (low-motion areas) and then calculate the frame differences in these areas.

1) CANDIDATE BLOCK SELECTION

We select the low-motion blocks by using mean absolute difference (MAD) between the consecutive frames. Specifically, when the i-th block in the current (t-th) frame, denoted as $I_i(t)$, is not much different from the corresponding block (at the same position) in the previous frame (i.e., $I_i(t-1)$) then the block is considered a candidate. Base on this, the set of candidate blocks is defined as

$$\mathbf{I}(t) = \{I_i(t) | \frac{1}{S} ||I_i(t) - I_i(t-1)|| < \tau\}, \qquad (2)$$

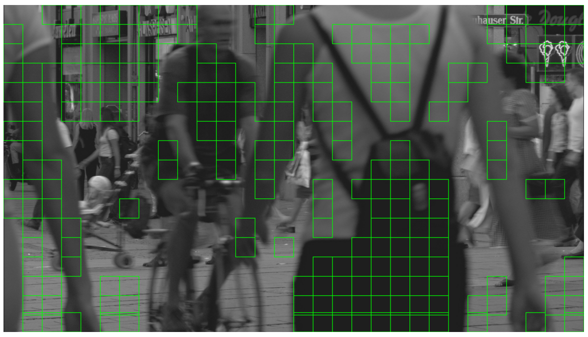

FIGURE 7. **Visualization of candidate blocks.**

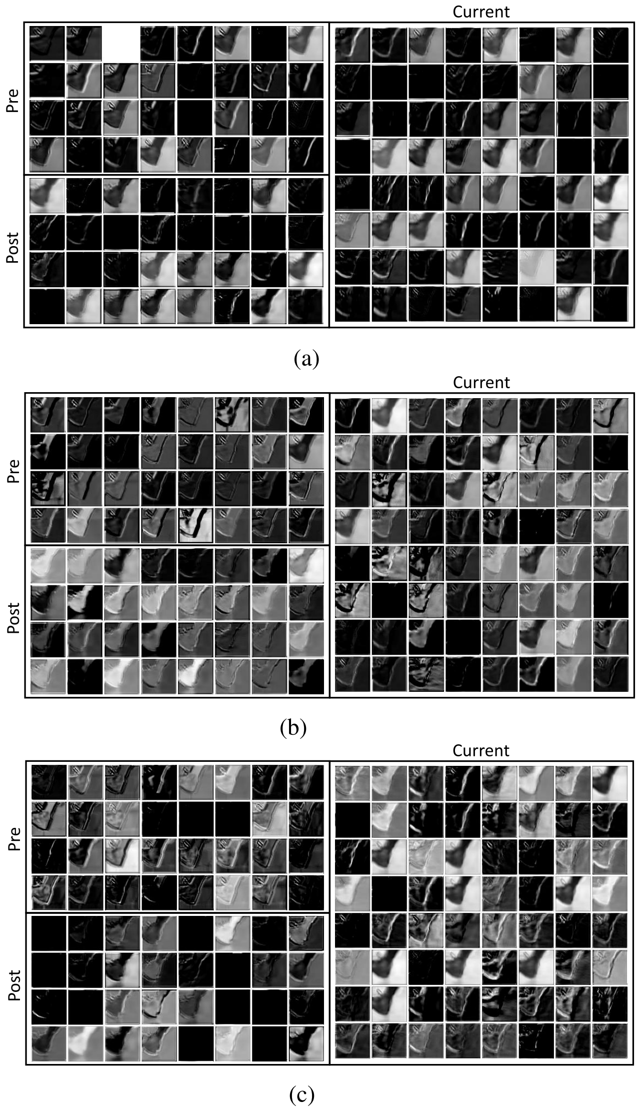

FIGURE 8. **Feature maps of temporal branch. (a) First convolution layer. (b) Second convolution layer. (c) Third convolution layer.**

where S is the number of pixels in the block $I_i(t)$ and τ is the threshold. We use the block size of 64×64 which is the largest CU size in HEVC. In all the experiments we set

$\tau = 7.5$, where FIGURE 7 shows an example of candidate blocks selected with $\tau = 7.5$.

2) METRIC FOR FLICKERING MEASUREMENTS

After selecting the candidate blocks, we measure the FS by calculating the SSD between the target and the reference. The basic idea is that if the temporal difference between the blocks of original frames (denoted as $e_i(t) = |I_i(t) - I_i(t-1)|$) is much different from the temporal difference of encoded frames ($\hat{e}_i(t) = |\hat{I}_i(t) - \hat{I}_i(t-1)|$) then it appears as the flickering, based on the assumption that the original video has no flickering.[2] In summary, the difference between the $\hat{e}_i(t)$ and $e_i(t)$ appears as the flickering, and hence we define the FS at the t-th frame as the sum of these differences in candidate blocks:

$$FS(t) = \frac{1}{N}\sum_i (e_i(t) - e_i(t-1))^2 \tag{3}$$

where N is the number of candidate blocks. Then, the overall FS is calculated by averaging the $FS(t)$ over the time:

$$\mathbf{FS} = \frac{1}{L}\sum_{t=1}^{L} FS(t) \tag{4}$$

where L is the number of frames in the sequence.

TABLE 6 shows the FS of the videos, where it can be seen that our baseline network (without temporal branches) reduces the FS, and the ARTN further reduces the FS as expected. It can also be seen that the flickering artifacts tend to increase as the QP increases. Focusing on high QPs, which produce low quality sequences, our baseline model reduces the FS approximately 3 to 5 percents and the ARTN by 4 to 9 percents.

[2]If the original video has flickering, then it would be meaningless to measure the flickering of the compressed video.

TABLE 6. FS results for the test set.

Sequence		BasketballDrive	BQTerrace	GTAV	Kimono	Pedestrian
QP 34						
HEVC	FS	24.298	27.952	26.554	15.478	8.726
Baseline	FS	19.725	27.397	25.933	15.078	8.509
	ΔFS	-18.8%	-2.0%	-2.3%	-2.6%	-2.5%
ARTN	FS	19.616	27.242	25.773	14.887	8.372
	ΔFS	-19.3%	-2.5%	-2.9%	-3.8%	-4.1%
QP 37						
HEVC	FS	27.978	30.180	31.727	18.697	10.215
Baseline	FS	22.657	29.338	30.913	18.184	9.870
	ΔFS	-19.0%	-2.8%	-2.6%	-2.7%	-3.4%
ARTN	FS	22.410	29.149	30.643	17.864	9.725
	ΔFS	-19.9%	-3.4%	-3.4%	-4.5%	-4.8%
QP 42						
HEVC	FS	36.070	35.034	40.508	25.216	13.893
Baseline	FS	34.786	33.633	39.256	24.366	13.361
	ΔFS	-3.6%	-4.0%	-3.1%	-3.4%	-3.8%
ARTN	FS	34.509	33.457	38.845	23.874	12.995
	ΔFS	-4.3%	-4.5%	-4.1%	-5.3%	-6.5%
QP 47						
HEVC	FS	46.738	41.301	48.834	32.920	19.275
Baseline	FS	44.708	39.491	47.239	31.508	18.333
	ΔFS	-4.3%	-4.4%	-3.3%	-4.3%	-4.9%
ARTN	FS	44.215	39.162	46.577	30.504	17.568
	ΔFS	-5.4%	-5.2%	-4.6%	-7.3%	-8.9%

E. DISCUSSION

1) EFFECTIVENESS OF TEMPORAL BRANCH

The effect of our temporal branch is also analyzed by visualizing the feature maps as shown in FIGURE 8. Upper-left 32 feature maps are the ones from the upper branch (from the previous frame), lower-left 32 from the lower-branch (next frame), and the right 64 are from the center branch (current frame). The depth of convolution layers is in the ascending order from the top to bottom. As can be seen, the features from the previous/next frames are correlated with the current

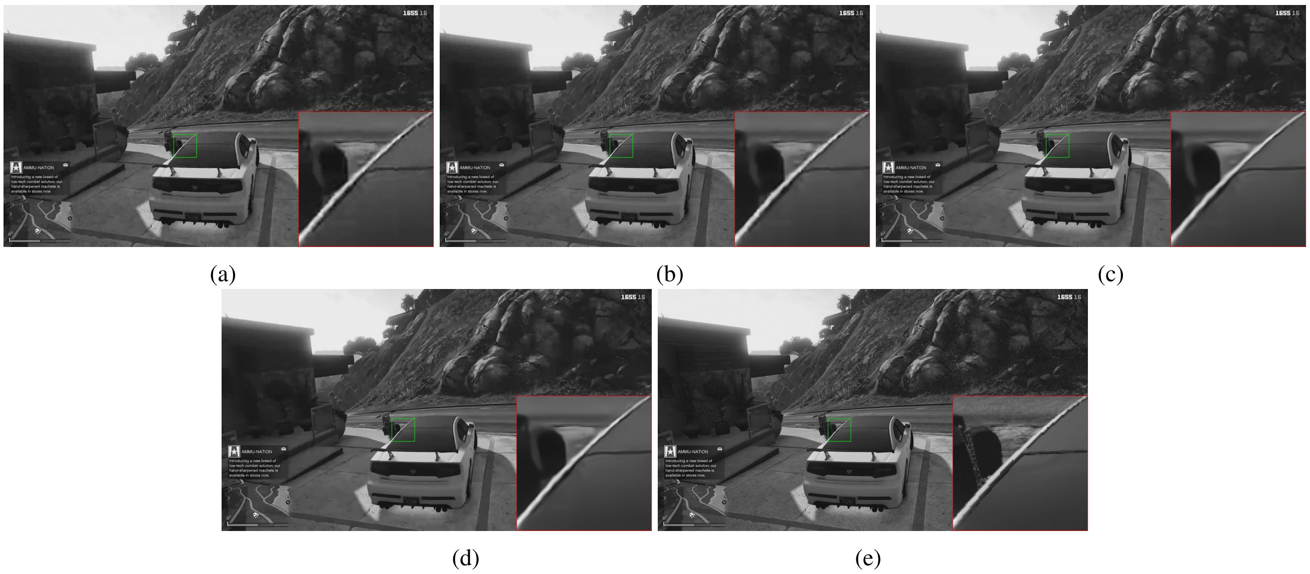

FIGURE 9. Visualization results on the sequence "GTAV" of QP 42 (AVC). (a) H.264/AVC. (b) AR-CNN. (c) Baseline. (d) ARTN. (e) Ground Truth.

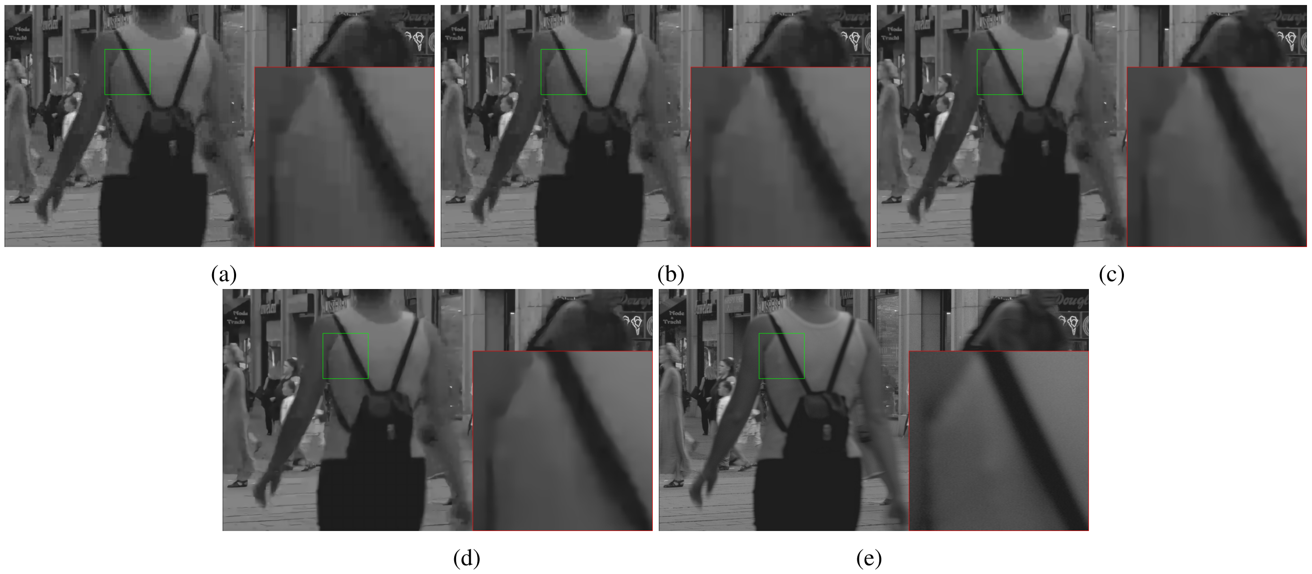

FIGURE 10. Visualization results on the sequence "Pedestrian" of QP 47 (AVC). (a) H.264/AVC. (b) AR-CNN. (c) Baseline. (d) ARTN. (e) Ground Truth.

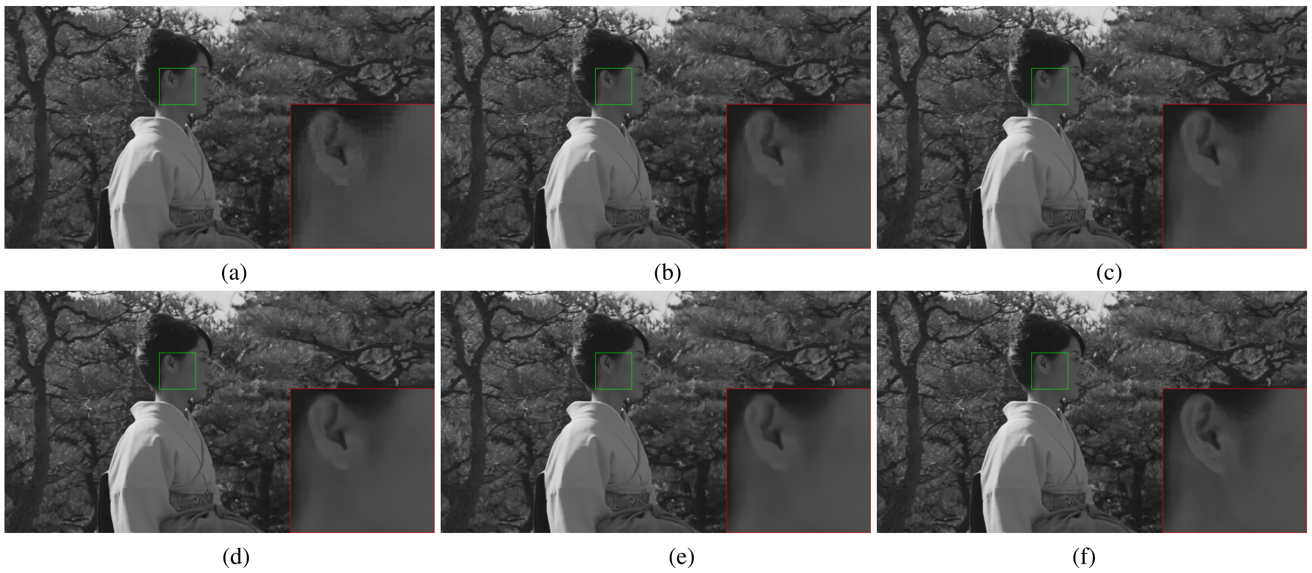

FIGURE 11. Visualization results on the sequence "Kimono" of QP 20 (MPEG-2). (a) MPEG-2. (b) AR-CNN. (c) VSRNet. (d) Baseline. (e) ARTN. (f) Ground Truth.

one, which means that the temporal branch networks successfully find additional features that help to enhance the current frame further.

To analyze the effect of motion estimation (ME) quantitatively, we conduct an additional experiment that we feed three successive patches in the same spatial position without ME. We tested on the case of QP 20 with MPEG-2 and obtained 33.92 dB, while the result with the ME is 34.23 dB as shown in TABLE 4. Hence, the ME brings about 0.3 dB gain in this case, and thus the ME has some effect on the performance. When comparing this result (33.92dB, when using temporal branch without ME) with others in TABLE 4, the ARTN without ME is is still better than the AR-CNN (33.83dB) but worse than our baseline model (34.07dB).

By referring to TABLE 4, "using temporal branch without ME" is still better than the AR-CNN but worse than our baseline model. Hence, we can conclude that when using the patches from different frames, it is required to align them by the ME.

As addressed in CNN-based non-local means denoising [45], [46], using more related patches enhances the restoration performance, and we believe this is not the exception with the compression artifacts reduction problem. Also, we believe that using the temporally correlated features (not

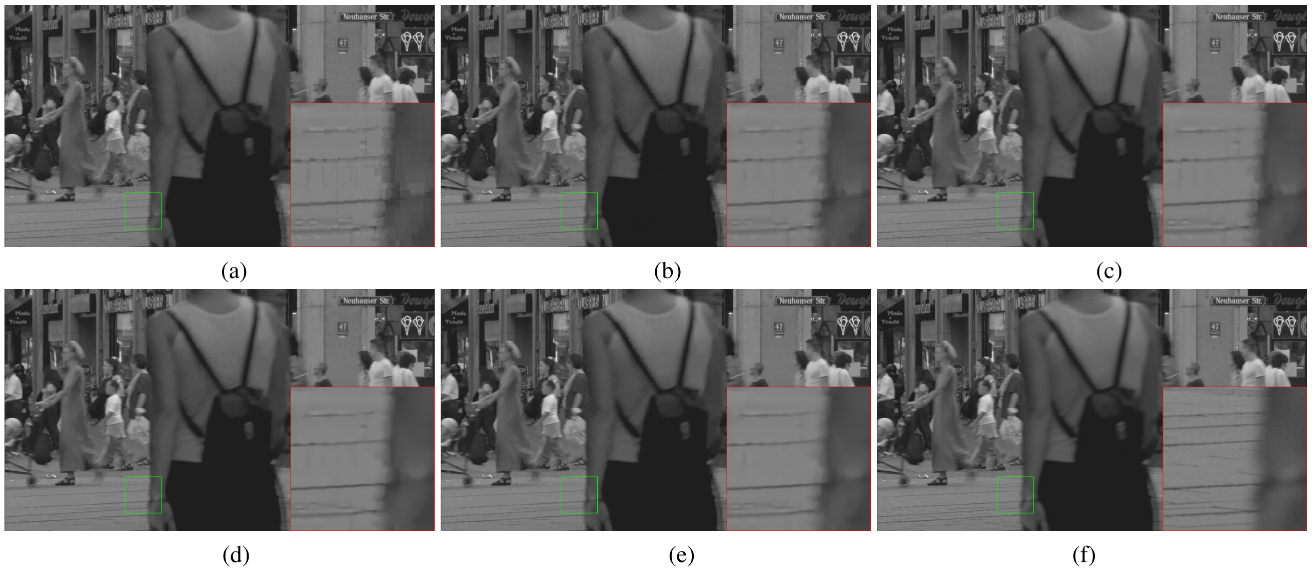

FIGURE 12. **Visualization results on the sequence "Pedestrian" of QP 30 (MPEG-2). (a) MPEG-2. (b) AR-CNN. (c) VSRNet. (d) Baseline. (e) ARTN. (f) Ground Truth.**

the spatially related ones within a frame as in the non-local means denoising methods) mitigates the temporal differences in enhancement performances, which leads to reduced flickering artifacts.

2) RUNNING TIME

We implement our ARTN and all the other networks with MatCaffe [40] on a computer with NVIDIA Titan X GPU and Intel i7-7700K CPU. The AR-CNN, VRCNN, and our baseline network take the computation time of 0.11, 0.12, and 0.25 seconds respectively for processing an FHD (1920×1080) frame. In the case of our ARTN, it takes about 2 seconds per frame where most of the computation time is taken for the motion estimation (ME). For the VSRNet [35] which also needs ME in the pre-processing step, it takes about 55 seconds per one FHD frame in the same environment. In both ARTN and VSRNet, the ME is executed on a CPU without considering the parallelism. Hence, if we use the parallel code for the ME and run it on the GPU, we believe that the computation time of ARTN and VSRNet can be much reduced.

The results of original size images and videos are available at http://github.com/JWSoh/ARTN for the comparisons of image quality and flickering, where we will also make our codes publicly available.

V. CONCLUSION

We have proposed a deep temporal network for reducing the artifacts in video compression. The network consists of three temporal branches of convolution layers which are merged to a single Inception-based network. Each of the temporal branches receives a frame, where we set three branches to process three consecutive frames in a video sequence. The temporal branches extract the features from the corresponding frames which are then concatenated and fed to the single network. The experiments show that the proposed network yields higher gain over the conventional networks for MPEG-2, AVC, and HEVC, by about 1.27 dB, 0.47 dB, and 0.23 dB respectively. It is also verified that using the temporal information brings further gains. The flickering artifacts are also measured in terms of our own measure, and it is shown that the proposed method also reduces the flickering that is commonly found in compressed videos.

Finally, although there have been many compression artifacts removal methods after the AR-CNN, we compared our method mainly with the AR-CNN by setting our network's complexity as similar as the AR-CNN. We believe that we can have more gains by using the recent deeper networks as our main and/or temporal branches. Specifically, using the temporal branches enabled more gains than just using a single deeper network with the same amount of parameters, which also reduced the flickering artifacts.

REFERENCES

[1] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, pp. 18–34, Feb. 1992.

[2] *Generic Coding of Moving Pictures and Associated Audio Information—Part 2: Video*, 1995.

[3] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[4] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, Y.-K. Wang, and T. Wiegand, *High Efficiency Video Coding (HEVC) Text Specification Draft 10*, document JCTVC-L1003, 2013, vol. 1.

[5] A. Foi, V. Katkovnik, and K. Egiazarian, "Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images," *IEEE Trans. Image Process.*, vol. 16, no. 5, pp. 1395–1411, May 2007.

[6] C. Dong, Y. Deng, C. Change Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 576–584.

[7] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[8] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[12] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3431–3440.

[13] L. Cavigelli, M. Magno, and L. Benini, "Accelerating real-time embedded scene labeling with convolutional networks," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2015, pp. 1–6.

[14] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2014, pp. 184–199.

[15] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1646–1654.

[16] W. Shi *et al.*, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1874–1883.

[17] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017.

[18] P. Svoboda, M. Hradis, D. Barina, and P. Zemcik. (2016). "Compression artifacts removal using convolutional neural networks." [Online]. Available: https://arxiv.org/abs/1605.00366

[19] J. Guo and H. Chao, "Building dual-domain representations for compression artifacts reduction," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2016, pp. 628–644.

[20] A. Norkin *et al.*, "HEVC deblocking filter," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1746–1754, Dec. 2012.

[21] C.-M. Fu *et al.*, "Sample adaptive offset in the HEVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1755–1764, Dec. 2012.

[22] Y. Dai, D. Liu, and F. Wu, "A convolutional neural network approach for post-processing in HEVC intra coding," in *Proc. Int. Conf. Multimedia Modeling*. Springer, 2017, pp. 28–39.

[23] W.-S. Park and M. Kim, "CNN-based in-loop filtering for coding efficiency improvement," in *Proc. IEEE 12th Image, Video, Multidimensional Signal Process. Workshop (IVMSP)*, Jul. 2016, pp. 1–5.

[24] R. Yang, M. Xu, and Z. Wang, "Decoder-side HEVC quality enhancement with scalable convolutional neural network," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2017, pp. 817–822.

[25] L. Cavigelli, P. Hager, and L. Benini, "CAS-CNN: A deep convolutional neural network for image compression artifact suppression," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2017, pp. 752–759.

[26] Y. Kim, I. Hwang, and N. I. Cho. (2017). "A new convolutional network-in-network structure and its applications in skin detection, semantic segmentation, and artifact reduction." [Online]. Available: https://arxiv.org/abs/1701.06190

[27] Y. Kim, I. Hwang, and N. I. Cho, "Convolutional neural networks and training strategies for skin detection," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 3919–3923.

[28] A. Diba, V. Sharma, and L. Van Gool, "Deep temporal linear encoding networks," in *Proc. Comput. Vis. Pattern Recognit.*, 2017, pp. 1541–1550.

[29] L. Wang *et al.*, "Temporal segment networks: Towards good practices for deep action recognition," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2016, pp. 20–36.

[30] C. Godard, K. Matzen, and M. Uyttendaele. (2017). "Deep burst denoising." [Online]. Available: https://arxiv.org/abs/1712.05790

[31] X. Chen, L. Song, and X. Yang, "Deep RNNs for video denoising," *Proc. SPIE*, vol. 9971, p. 99711T, Sep. 2016.

[32] C. Lu, M. Hirsch, and B. Schölkopf, "Flexible spatio-temporal networks for video prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 6523–6531.

[33] J. Guo and H. Chao, "Building an end-to-end spatial-temporal convolutional network for video super-resolution," in *Proc. AAAI*, 2017, pp. 4053–4060.

[34] C. Jia, S. Wang, X. Zhang, S. Wang, and S. Ma. (2017). "Spatial-temporal residue network based in-loop filter for video coding." [Online]. Available: https://arxiv.org/abs/1709.08462

[35] A. Kappeler, S. Yoo, Q. Dai, and A. K. Katsaggelos, "Video super-resolution with convolutional neural networks," *IEEE Trans. Comput. Imag.*, vol. 2, no. 2, pp. 109–122, Jun. 2016.

[36] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

[37] J. Lu and M. L. Liou, "A simple and efficient search algorithm for block-matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 2, pp. 429–433, Apr. 1997.

[38] S. Bosse, D. Maniry, T. Wiegand, and W. Samek, "A deep neural network for image quality assessment," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 3773–3777.

[39] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. (2015). "Loss functions for neural networks for image processing." [Online]. Available: https://arxiv.org/abs/1511.08861

[40] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 675–678.

[41] D. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980

[42] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[43] J. X. Yang and H. R. Wu, "A non-linear post filtering method for flicker reduction in H.264/AVC coded video sequences," in *Proc. IEEE 10th Workshop Multimedia Signal Process.*, Oct. 2008, pp. 181–186.

[44] S. S. Chun, J.-R. Kim, and S. Sull, "Intra prediction mode selection for flicker reduction in H.264/AVC," *IEEE Trans. Consum. Electron.*, vol. 52, no. 4, pp. 1303–1310, Nov. 2006.

[45] D. Yang and J. Sun, "BM3D-Net: A convolutional neural network for transform-domain collaborative filtering," *IEEE Signal Process. Lett.*, vol. 25, no. 1, pp. 55–59, Jan. 2018.

[46] B. Ahn and N. I. Cho. (2017). "Block-matching convolutional neural network for image denoising." [Online]. Available: https://arxiv.org/abs/1704.00524

JAE WOONG SOH received the B.S. degree in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2016, where he is currently pursuing the Ph.D. degree in electrical and computer engineering. His research interests include image processing, computer vision, and machine learning.

JAEWOO PARK received the B.S. degree in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2017, where he is currently pursuing the Ph.D. degree in electrical and computer engineering. His research interests include image processing, computer vision, and machine learning.

YOONSIK KIM received the B.S. degree in electronics engineering from Chung-Ang University, Seoul, South Korea, in 2014. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Seoul National University, Seoul. His research interests include image processing, computer vision, and machine learning.

BYEONGYONG AHN received B.S. and Ph.D. degrees in electrical and computer engineering from Seoul National University in 2011 and 2017, respectively. In 2017, he joined Samsung Electronics as a Senior Engineer. His current research interests include image enhancement, neural network, and document image processing.

HYUN-SEUNG LEE received the B.S. and M.S. degrees in electronic engineering from Sogang University in 2005 and 2007, respectively. He is currently pursuing the Ph.D. degree in electronic engineering with Seoul National University. He joined the Multi-Media Platform Laboratory, Digital Media and Communications R&D Center, Samsung Electronics Co., Ltd., in 2007, where he has been a Senior Engineer with the Video Display Division since 2017. His research interests include video processing and compression.

YOUNG-SU MOON received the B.S. degree in mechanical engineering from Seoul National University in 1988 and the M.S. degree in mechanical engineering and the Ph.D. degree in automation and design engineering from KAIST in 1991 and 2001, respectively. He joined the R&D Center, Samsung Electronics Co., (SEC) Ltd., in 1988, and rejoined the Multimedia computing Laboratory, SAIT, in 2001. In the DMC R&D Center, SEC, he was appointed as a Samsung Master in 2015, where he has been with the Visual Display Division since 2016. His current research interests include advanced video processing, AI technology, new form-factor display processing, and perceptual display processing.

NAM IK CHO received the B.S., M.S., and Ph.D. degrees in control and instrumentation engineering from Seoul National University, Seoul, South Korea, in 1986, 1988, and 1992, respectively. From 1991 to 1993, he was a Research Associate with the Engineering Research Center for Advanced Control and Instrumentation, Seoul National University. From 1994 to 1998, he was with University of Seoul as an Assistant Professor of electrical engineering. In 1999, he joined the Department of Electrical and Computer Engineering, Seoul National University, where he is currently a Professor. His research interests include image processing, adaptive filtering, digital filter design, and computer vision.