Non-Local ConvLSTM for Video Compression Artifact Reduction

Yi Xu[1*]   Longwen Gao[2]   Kai Tian[1]   Shuigeng Zhou[1†]   Huyang Sun[2]

[1]Shanghai Key Lab of Intelligent Information Processing, and School of
Computer Science, Fudan University, Shanghai, China

[2]Bilibili, Shanghai, China

{yxu17,ktian14,sgzhou}@fudan.edu.cn   {gaolongwen,sunhuyang}@bilibili.com

Abstract

Video compression artifact reduction aims to recover high-quality videos from low-quality compressed videos. Most existing approaches use a single neighboring frame or a pair of neighboring frames (preceding and/or following the target frame) for this task. Furthermore, as frames of high quality overall may contain low-quality patches, and high-quality patches may exist in frames of low quality overall, current methods focusing on nearby peak-quality frames (PQFs) may miss high-quality details in low-quality frames. To remedy these shortcomings, in this paper we propose a novel end-to-end deep neural network called non-local ConvLSTM (NL-ConvLSTM in short) that exploits multiple consecutive frames. An approximate non-local strategy is introduced in NL-ConvLSTM to capture global motion patterns and trace the spatiotemporal dependency in a video sequence. This approximate strategy makes the non-local module work in a fast and low space-cost way. Our method uses the preceding and following frames of the target frame to generate a residual, from which a higher quality frame is reconstructed. Experiments on two datasets show that NL-ConvLSTM outperforms the existing methods.

1. Introduction

Video compression algorithms are widely used due to limited communication bandwidth and storage space in many real (especially mobile) application senarios [34]. While significantly reducing the cost of transmission and storage, lossy video compression also leads various compression artifacts such as blocking, edge/texture floating, mosquito noise and jerkiness [48]. Such visual distortions often severely impact the quality of experience (QoE). Consequently, video compression artifact reduction has
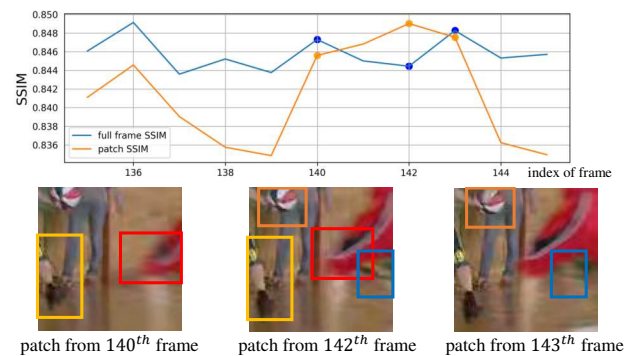

Figure 1. An example of high-quality patches existing in low-quality frames. Here, though the 140th and 143th frames have better full-frame SSIM than the 142th frame, the cropped patch from the 142th frame has the best patch SSIM. The upper part shows the SSIM values of the fames and cropped patches; the lower images are the cropped patches from three frames. Also, comparing the details in the boxes of the same color, we can see that the cropped patches from the 142th frame are of better quality.

emerged as an important research topic in multimedia and computer vision areas [26, 43, 45].

In recent years, significant advances in compressed image/video enhancement have been achieved due to the successful applications of deep neural networks. For example, [11, 12, 36, 49] directly utilize deep convolutional neural networks to remove compression artifacts of images without considering the characteristics of underlying compression algorithms. [16, 38, 43, 44] propose models that are fed with compressed frames and output the enhanced ones. These models all use a single frame as input, do not consider the temporal dependency of neighboring frames. To exploit the temporal correlation of neighboring frames, [26] proposes the deep kalman filter network, [42] employs task oriented motions, and [45] uses two motion-compensated nearest PQFs. However, [26] uses only the preceding frames of the target frame, while [42, 45] adopt only a pair of neighboring frames, which may miss high-quality details of some other neighbor frames (will be explained later).

---

*This author did most work during his internship at Bilibili.
†Corresponding author.

Video compression algorithms have intra-frame and inter-frame codecs. Inter coded frames (P and B frames) significantly depend on the preceding and following neighbor frames. Therefore, extracting spatiotemporal relationships among the neighboring frames can provide useful information for improving video enhancement performance. However, mining detailed information from one/two neighboring frame(s) or even two nearest PQFs is not enough for compression video artifact reduction. To illustrate this point, we present an example in Fig. 1. Frames with a larger *structural similarity index measure* (SSIM) are usually regarded as better visual quality. Here, though the 140th and 143th frames have better overall visual quality than the 142th frame, the cropped patch of highest quality comes from the 142th frame. The high-quality details in such patches would be ignored if mining spatiotemporal information from videos using the existing methods.

Motivated by the observation above, in this paper we try to capture the hidden spatiotemporal information from multiple preceding and following frames of the target frame for boosting the performance of video compression artifact reduction. To this end, we develop a non-local ConvLSTM framework that uses the non-local mechanism [3] and ConvLSTM [41] architecture to learn the spatiotemporal information from a frame sequence. To speed up the non-local module, we further design an approximate and effective method to compute the inter-frame pixelwise similarity. Comparing with the existing methods, our method is advantageous in at least three aspects: 1) No accurate motion estimation and compensation is explicitly needed; 2) It is applicable to videos compressed by various commonly-used compression algorithms such as H.264/AVC and H.265/HEVC; 3) The proposed method outperforms the existing methods.

Major contributions of this work include: 1) We propose a new idea for video compression artifact reduction by exploiting multiple preceding and following frames of the target frame, without explicitly computing and compensating motion between frames. 2) We develop an end-to-end deep neural network called non-local ConvLSTM to learn the spatiotemporal information from multiple neighboring frames. 3) We design an approximate method to compute the inter-frame pixel-wise similarity, which dramatically reduces calculation and memory cost. 4) We conduct extensive experiments over two datasets to evaluate the proposed method, which achieves state-of-the-art performance for video compression artifact reduction.

2. Related Work

2.1. Single Image Compression Artifact Reduction

Early works mainly include manually-designed filters [3, 9, 30, 51], iterative approaches based on the theory of pro-

jections onto convex sets [29, 47], wavelet-based methods [40] and sparse coding [5, 25].

Following the success of AlexNet [21] on ImageNet [31], many deep learning based methods have been applied to this low-level long-standing computer vision task. Dong *et al.* [11] firstly proposed a four-layer network named ARCNN to reduce JPEG compression artifacts. Afterwards, [22, 28, 35, 36, 49, 50] proposed deeper networks to further reduce compression artifacts. A notable example is [49], which devised an end-to-end trainable denoising convolutional neural network (DnCNN) for Gaussian denoising. DnCNN also achieves a promising result on JPEG deblocking task. Moreover, [7, 14, 46] enhanced visual quality by exploiting the wavelet/frequency domain information of JPEG compression images. Recently, more methods [1, 6, 8, 12, 15, 24, 27] were proposed and got competitive results. Concretely, Galteri *et al.* [12] used a deep generative adversarial network and recovered more photorealistic details. Inspired by [39], [24] incorporated non-local operations into a recursive framework for quality restoration, it computed self-similarity between each pixel and its neighbors, and applied the non-local module recursively for correlation propagation. Differently, here we adopt a non-local module to capture global motion patterns by exploiting inter-frame pixel-wise similarity.

2.2. Video Compression Artifact Reduction

Most of existing video compression artifact reduction works [10, 16, 38, 43, 44] focus on individual frames, neglecting the spatiotemporal correlation between neighboring frames. Recently, several works were proposed to exploit the spatiotemporal information from neighboring frames. Xue *et al.* [42] designed a neural network with a motion estimation component and a video processing component, and utilized a joint training strategy to handle various low-level vision tasks. Lu *et al.* [26] further incorporated quantized prediction residual in compressed code streams as strong prior knowledge, and proposed a deep Kalman filter network (DKFN) to utilize the spatiotemporal information from the preceding frames of the target frame. In addition, considering that quality of nearby compressed frames fluctuates dramatically, [13, 45] proposed multi-frame quality enhancement (MFQE) and utilized motion compensation of two nearest PQFs to enhance lowquality frames. Comparing with DKFN[26], MFQE is a post-processing method and uses less prior knowledge of the compression codec, but still achieves state-of-the-art performance on HEVC compressed videos.

In addition to compression artifact removal, spatiotemporal correlation mining is also a hot topic in other video quality enhancement tasks, such as video super resolution (VSR). [4, 18, 19, 23, 32, 37, 42] estimated optical flow and warped several frames to capture the hidden spa-
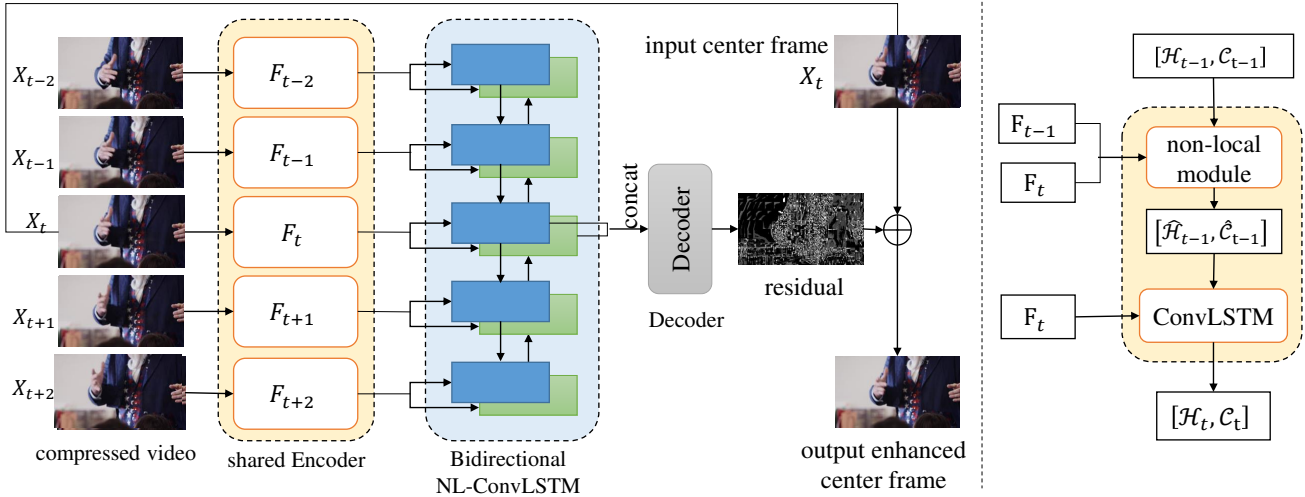
Figure 2. The framework of our method (left) and the architecture of NL-ConvLSTM (right)

tiotemporal dependency for VSR. Although these methods work well, they rely heavily on the accuracy of motion estimation. Instead of explicitly taking advantage of motion between frames, [17] utilized a 3D convolutional network as a dynamic filter generation network to generate dynamic upsampling filter and fine residual for VSR.

In summary, most end-to-end CNN based visual quality enhancement methods consider only either a single frame or a pair of neighboring frames, thus may miss important details of other neighboring frames. Unlike these works, here we employ the NL-ConvLSTM mechanism to utilize multiple frames and capture spatiotemporal variations in a frame sequence without explicit motion estimation and compensation, With an approximate strategy for non-local similarity computation, our method can effectively reduce artifacts and achieves state-of-the-art performance.

3. Method

The goal of video compression artifact reduction is to infer a high quality frame $\hat{Y}_t$ from a compressed frame X_t of the original frame (ground truth) Y_t, where $X_t \in \mathbb{R}^{C \times N}$ is the compressed frame at time t. Here, C is the number of channels of a single frame. For the sake of notation clarity, we collapse the spatial positions (width W and height H) into one dimension, $N = HW$. Let $\mathcal{X}_t = \{X_{t-T}, \ldots, X_{t+T}\}$ denote a sequence of $(2T+1)$ consecutive compressed frames, our method takes $\mathcal{X}_t$ as input and outputs $\hat{Y}_t$.

3.1. The Framework

Our method is an end-to-end trainable framework that consists of three modules: Encoder, NL-ConvLSTM module and Decoder, as shown in Fig. 2. They are respectively responsible for extracting features from individual frames,

learning spatiotemporal correlation across frames, and decoding high-level features to residuals, with which the high-quality frames are reconstructed eventually.

Encoder. It is designed with several 2D convolutional layers to extract features from $\mathcal{X}_t$. With $\mathcal{X}_t$ as input, it outputs $\mathcal{F}_t = \{F_{t-T}, \ldots, F_{t+T}\}$. Here, $F_t \in \mathbb{R}^{C_f \times N}$ is the corresponding feature extracted from X_t, C_f is the channel size of output features. It processes each frame individually.

NL-ConvLSTM. To trace the spatiotemporal dependency in a frame sequence, we put a ConvLSTM [41] module between the Encoder and the Decoder. ConvLSTM is able to capture spatiotemporal information from a frame sequence of arbitrary length, but it is not good at handling large motions and blurring motions well [18]. To tackle this problem, we embed the non-local (NL) [3] mechanism into ConvLSTM, and develop the NL-ConvLSTM module. Here, the non-local similarity is used for pixels from different frames rather than pixels within a frame [3]. The NL-ConvLSTM module $\mathcal{N}$ can be described as

$$[\mathcal{H}_t, \mathcal{C}_t] = \mathcal{N}(F_{t-1}, F_t, [\mathcal{H}_{t-1}, \mathcal{C}_{t-1}]). \quad (1)$$

Different from ConvLSTM in [37, 41] that is fed with only feature F_t at time t, NL-ConvLSTM takes additional feature F_{t-1} at time $(t\text{-}1)$ as input, and outputs the corresponding hidden state and cell state $\mathcal{H}_t, \mathcal{C}_t \in \mathbb{R}^{C_h \times N}$. Here, C_h is the number of channels of hidden state and cell state. Moreover, hidden state $\mathcal{H}_{t-1}$ and cell state $\mathcal{C}_{t-1}$ are not fed into gate operation directly in NL-ConvLSTM. In contrast, we calculate inter-frame pixel-wise similarity S_t between F_{t-1} and F_t, then perform a weighted sum over $\mathcal{H}_{t-1}$ and $\mathcal{C}_{t-1}$ with S_t as weight. In addition, bi-directional ConvLSTM is used in our paper to learn spatiotemporal dependency from both preceding and following frames. In the following sections, we only mention the operation of forward NL-ConvLSTM

for simplicity. Details of the NL-ConvLSTM module can be referred to Fig. 2 (right), Fig. 3, Section 3.2 and 3.3.

Decoder. It decodes the hidden state from both directions of NL-ConvLSTM module into residual, with which the high-quality frame is reconstructed. Specifically, we first combine the hidden states by a convolutional layer whose kernel size is 1×1, then use several stacked convolutional layers to generate the residual.

3.2. Non-local ConvLSTM

ConvLSTM can be described as follows [37]:

$$[\mathcal{H}_t, \mathcal{C}_t] = ConvLSTM(F_t, [\mathcal{H}_{t-1}, \mathcal{C}_{t-1}]). \quad (2)$$

To learn robust spatiotemporal dependency, we adopt the non-local mechanism into ConvLSTM to help estimate motion patterns in frame sequences. As an extension of ConvLSTM, NL-ConvLSTM can be formulated as

$$S_t = NL\left(F_{t-1}, F_t\right),$$
$$\left[\hat{\mathcal{H}}_{t-1}, \hat{\mathcal{C}}_{t-1}\right] = NLWarp([\mathcal{H}_{t-1}, \mathcal{C}_{t-1}], S_t), \quad (3)$$
$$[H_t, C_t] = ConvLSTM(F_t, \left[\hat{\mathcal{H}}_{t-1}, \hat{\mathcal{C}}_{t-1}\right]),$$

where $S_t \in \mathbb{R}^{N \times N}$ denotes the similarity matrix between the pixels of the current frame and all pixels of the preceding frame. NL is the non-local operator for calculating the similarity matrix between features of two frames, $NLWarp$ is the warping operation for the hidden state and cell state at time $(t\text{-}1)$ with a weighted sum form.

Following the non-local operation [3], the inter-frame pixel-wise similarity and non-local warping operation in our work are as follows:

$$D_t\left(i, j\right) = \|F_{t-1}\left(i\right) - F_t\left(j\right)\|_2,$$
$$S_t\left(i, j\right) = \frac{\exp\left(-D_t\left(i, j\right)/\beta\right)}{\sum_{\forall i} \exp\left(-D_t\left(i, j\right)/\beta\right)}, \quad (4)$$
$$\left[\hat{\mathcal{H}}_{t-1}, \hat{\mathcal{C}}_{t-1}\right] = [\mathcal{H}_t \cdot S_t, \mathcal{C}_t \cdot S_t],$$

where $i, j \in \{1, \cdots, N\}$ are indices of pixels in a feature map, $F(i)$ and $\mathcal{H}(i)$ are the corresponding feature and state at position i. $D_t\left(i, j\right)$ and $S_t\left(i, j\right)$ are the Euclidean distance and similarity throughout all channels between pixel i in the preceding feature map at time t-1 and pixel j in the current feature map at time t. $S_t\left(i, j\right)$ satisfies $\sum_i S_t\left(i, j\right) = 1$. Thus, the non-local method can be seen as a special attention mechanism [39].

3.3. Two-stage Non-local Similarity Approximation

Directly computing $S_t \in \mathbb{R}^{N \times N}$ and the warping operation will incur extremely high computation and memory cost for high-resolution videos. Therefore, we propose a two-stage non-local method to approximate D_t as $\hat{D}_t$ and
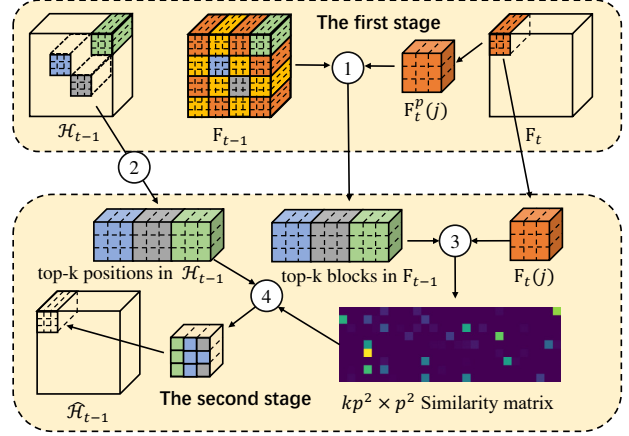

Figure 3. The workflow of two-stage similarity approximation. ① finding the top-k most similar blocks in F_{t-1} with respect to block $F_t^p(j)$ from F_t; ② extracting blocks in $\mathcal{H}_{t-1}$ from the corresponding positions of the top-k most similar blocks in F_{t-1}; ③ calculating pixel-wise similarity between the selected blocks from F_{t-1} and $F_t^p(j)$; ④ NLWarp operation for $\mathcal{H}_t$.

S_t as $\hat{S}_t$, which can reduce computation and memory while maintaining accuracy. The key idea of our approximate method is to pre-filter image blocks according to the deep feature learned by the Encoder before calculating pixel-wise similarity. The details are as follows:

In the 1st stage, we use average pooling to downsample the feature map from the Encoder and reduce the block matching sensitivities of geometry transformations (shifts and rotations). Denote the kernel size of the average pooling as p and the downsampled feature map as F_t^p. Then, the resolution of the feature map is reduced to N/p^2, i.e., $1/p^2$ of the original resolution. Each super-pixel in the downsampled feature map F_t^p corresponds to a block consisting of p^2 pixels in the original feature map. Thus, the downsampled distance matrix $D_t^p \in \mathbb{R}^{(N/p^2)^2}$ can be calculated by

$$D_t^p\left(i, j\right) = \left\|F_{t-1}^p\left(i\right) - F_t^p\left(j\right)\right\|_2. \quad (5)$$

For each pixel in any block b_t of F_t^p, we consider only $k \times p^2$ pixels in the k blocks of F_{t-1}^p that are nearest to b_t.

In the 2nd stage, we compute and store the similarities between each pixel of F_t and the corresponding $k \times p^2$ pixels of F_{t-1}. While for the other pixels in the preceding frame, the elements of $\hat{D}_t$ and $\hat{S}_t$ are set to $+\infty$ and 0 respectively. As similar pixels are sparse and a pixel can be represented by a few pixels from the neighboring frame, the quality loss of top-k blocks approximation is negligible. Fig. 3 illustrates the workflow of two-stage similarity approximation. The NLWarp operation for $\mathcal{C}_t$ is similar to that of $\mathcal{H}_t$. We show only the operation for $\mathcal{H}_t$ in Fig. 3 for simplicity.

Complexity analysis. Tab. 1 compares the complex-

ity of our approximate method with that of the original method. Since $\log k \ll C$ in our experiments, we can neglect the item related to searching the top-k nearest blocks in Tab. 1 for simplicity. We denote the complexity of the original non-local method as ψ, and the complexity of our approximate method as ϕ, which can be rewritten to $\mathcal{O}((N/p^2)^2 C + 2kNCp^2)$. By properly choosing the values of k and p so that $kp^2 \ll N$, we have $\phi/\psi = 1/(2p^4) + kp^2/N \ll 1$, which means that our method dramatically reduces the computation cost of the original method. And for a given k, ϕ/ψ achieves the minimum $1.5(k/N)^{2/3}$ with $p=(N/k)^{1/6}$. Similar conclusion can be drawn for memory cost. More specifically, by setting p=10, k=4, C=64, and f=4[1], ϕ is close to $(\mathcal{O}(NC^2 f^2))$, which is the computational complexity of a convolutional layer with a $f \times f$ kernel.

Non-local operation *vs*. motion compensation. There are similarities and differences between the two operations. *Similarities*: 1) Both can be adopted to capture spatiotemporal relationships and motion patterns in consecutive frames. 2) Both can be seen as an attention mechanism. In non-local operation, the warped state $\hat{\mathcal{H}}_{t-1}$ is calculated with the states of all pixels in $\mathcal{H}_{t-1}$ in a weighted sum form; while in motion compensation, each pixel in $\hat{\mathcal{H}}_{t-1}$ is evaluated by interpolation with some neighboring pixels in $\mathcal{H}_{t-1}$, also in a weighted sum form. *Differences*: 1) In non-local operation, each pixel is warped from multiple positions in $\mathcal{H}_{t-1}$, and the motion is not restricted by a fixed flow magnitude, which is different from motion compensation where a fixed flow magnitude has to be set. Thus, non-local operation can capture global motion patterns more effectively. 2) In non-local operation, the similarity is determined once the feature is extracted; while in motion compensation, we need to train additional layers for motion field generation.

3.4. Implementation Details

In our implementation, following existing methods [26, 45], we use L_2 norm as the loss function:

$$l\left(\mathcal{X}_t, Y_t\right) = \left\| \hat{Y}_t - Y_t \right\|_2. \quad (6)$$

Due to the advantage of NL-ConvLSTM, global motion can be captured with a small kernel in ConvLSTM. So our NL-ConvLSTMs is implemented with a 3×3 kernel. For all datasets, the networks are trained using the ADAM [20] optimizer with an initial learning rate of 10^{-4} and a mini-batch size of 32. In training, raw and compressed sequences are sampled with a patch size of 80×80 for NL-ConvLSTM. In contrast, the full resolution video sequences are fed into our model during testing. We use k=4 and p=10 in all experiments for balancing efficiency and effectiveness, and set T=3 for all datasets.

---

[1] f is the kernel size of a convolutional layer

Table 1. Complexity comparison of the original non-local approach and ours. Here, N and C are the numbers of positions and channels, k and p are the number of pre-filtered blocks and the downsampling scale. By setting k=4 and p=10, our method cuts the time and space to about 1/1000 of that consumed by the original non-local method in 1080P videos.

	Original non-local	NL-ConvLSTM
Time	$\mathcal{O}(2N^2 C)$	$\mathcal{O}((N/p^2)^2 (C + \log k) + 2kNCp^2)$
Space	$\mathcal{O}(2N^2)$	$\mathcal{O}((N/p^2)^2 + kN/p^2 + 2kNp^2)$

To further accelerate the non-local operator, vectorization is adopted in calculating distance matrix D_t. Although vectorization does not reduce the number of floating-point operations, it enables acceleration via parallel computing. By expanding D_t in Equ. (4), we have

$$D_t{}^2 = \sum^{C_f} F_{t-1}^2 \cdot \mathbf{1}^\top + \mathbf{1} \cdot \sum^{C_f} F_t^{2\top} - 2F_{t-1}^\top \cdot F_t, \quad (7)$$

where $\mathbf{1} \in \mathbb{R}^{N \times 1}$ is a vector whose elements are 1. We adopt Equ. (7) for calculating D_t^p in the 1st stage, and implement a sparse version of Equ. (7) to compute distances between each pixel in the current frame and $k \times p^2$ pre-filtered pixels in the preceding frame.

4. Performance Evaluation

To evaluate our method, we conduct extensive experiments on two datasets: Vimeo-90K [42] and Yang *et al*.'s dataset [45]. Our evaluation consists of five parts: 1) Ablation study; 2) Quantitative evaluation with two performance metrics (PSNR and SSIM); 3) Qualitative evaluation by comparing the visual effect of compression artifact reduction of different methods; 4) Run time comparison; 5) Checking the effectiveness of our method on videos compressed by another algorithm.

4.1. Datasets and Settings

Vimeo-90K. It is a recently-built large-scale video dataset for low-level video processing. All frames are resized to a fixed resolution 448×256. We follow the settings in [26] and refer the interested readers to [26] for details. In short, compressed clips are generated by x265 in FFmpeg with quantization parameter QP=32 and 37. Loop filter, SAO [34] and B-frames are disabled in codec. We follow [26] and only evaluate the 4th frame of each clip.

Yang *et al*.'s dataset. It consists of 70 video sequences selected from the datasets of Xiph.org[2] and JCT-VC [2]. Resolutions of these video sequences vary from 352×240 to $2,560 \times 1,600$. For a fair comparison, we follow the settings in [45]: 60 sequences are taken for training and the remaining 10 for testing. All sequences are encoded in HEVC LDP mode, using HM 16.0 with QP=37 and 42.

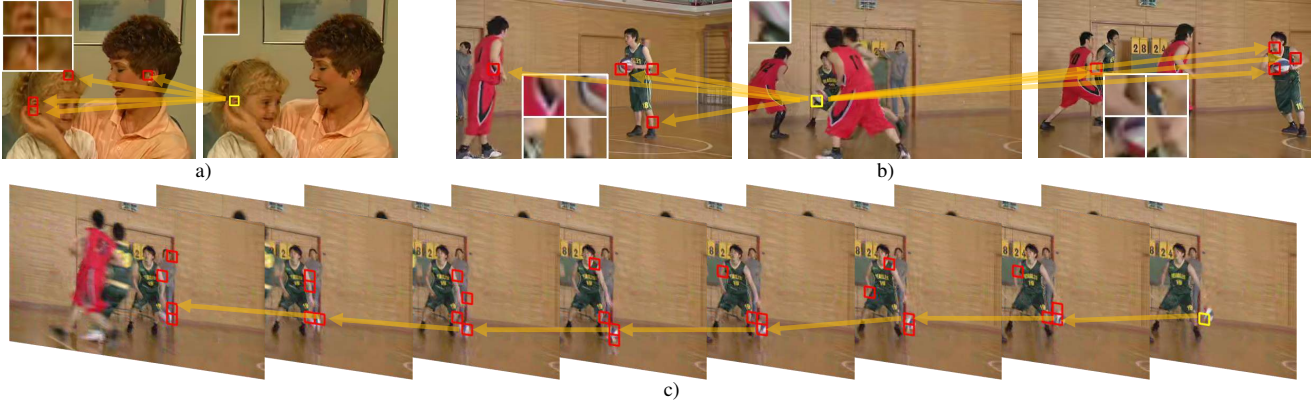[2] https://media.xiph.org/video/derf/

Figure 4. Examples of blocks after pre-filtering in the first stage of our method. Images are from Yang *et al.*'s dataset. The red blocks are the top-4 most similar blocks with respect to the yellow block in another frame.

Table 2. Ablation study of the proposed NL-ConvLSTM on Yang *et al.*'s dataset with QP=37. The results of PSNR improvement ΔPSNR (db) are reported in the 1st row. The results of SSIM improvement ΔSSIM ($\times 10^{-2}$) are listed in the 2nd row.

	Encoder-Decoder with 1 frame	ConvLSTM with 7 frames	ME-ConvLSTM with 7 frames	Our method with 7 frames
ΔPSNR	0.395	0.456	0.503	**0.601**
ΔSSIM	0.684	0.723	0.827	**0.897**

4.2. Ablation Study

Effect of multiple frames. Here we evaluate the effectiveness of utilizing multiple frames. First, we compare the performance between Encoder-Decoder (using 1 frame) and ConvLSTM (using 7 frames). The results of the 2nd and 3rd columns in Table 2 show that utilizing multiple frames in ConvLSTM obviously boosts performance. Then, we further verify the effectiveness of our method using more frames as input on Yang *et al.*'s dataset with QP=37. Each of the input clips consists of 20 consecutive frames: the target frame, 15 preceding frames and 4 following frames. We use the pre-trained model on the same settings with T=3, and then fine-tune the model on 20-frame inputs using almost similar training settings, except for a smaller batch size. We find that our model tuned on longer sequences gets 0.604dB/0.00923 of ΔPSNR/ΔSSIM, which are better than the results of the 5th column in Table 2. The results above indicate that multiple frames and longer sequences do improve artifact reduction performance.

Effect of the non-local mechanism. Here we investigate the effectiveness of non-local mechanism. The non-local module aims to learn spatiotemporal dependency between two consecutive frames. Usually, motion estimation and compensation module can do such a role. Thus, we use a motion estimation and compensation module to replace the non-local module, and name such a method as ME-

ConvLSTM. We follow the motion generation architecture in [33]. The results on Yang *et al.*'s dataset with QP=37 are listed in the 4th column of Table 2. Compared with ME-ConvLSTM, our method performs better, 19.48% higher in terms of ΔPSNR and 8.46% higher in terms of ΔSSIM, which demonstrates that ConvLSTM with non-local mechanism utilizes temporal information better.

Effect of the Two-Stage Non-local Method. We propose a two-stage non-local method to learn the spatiotemporal dependency between two neighboring frames. In the first stage, it tries to find the top-k most similar blocks in F_{t-1}^p for each block of F_t^p. This pre-filtering impacts spatiotemporal dependency learning. Here, we visualize some pre-filtering results in the first stage to show that our NL-ConvLSTM method can learn the spatiotemporal dependency among consecutive frames.

Fig. 4 shows some examples of blocks after pre-filtering. The red blocks are the top-4 most similar blocks in a frame after pre-filtering with respect to the yellow block in another frame. In Fig. 4(a) and Fig. 4(b), these blocks are enlarged, bounded with white boxes, and shown in the full image. Fig. 4(a) shows two consecutive frames. The yellow block in the right frame contains the daughter's ear. Our method finds the daughter's ear and her mother's ear in the left frame. This shows that our method can capture similar patterns at various locations, which could provide additional information for enhancement. In Fig. 4(b), we manually construct a 3-frame sequence with a large time gap. It is difficult for motion estimation based methods to handle such large motion. However, our method can still capture the spatiotemporal dependency of basketball. In Fig. 4(c), we thread the top-1 block of each frame from the yellow block in the right-most frame to left iteratively. Such a path reflects the robust spatiotemporal relationship built among frames by our method.

Table 3. Average PSNR/SSIM on Vimeo-90K.

QP	32	37
HEVC [34]	34.19 / 0.950	31.98 / 0.923
ARCNN [11]	34.87 / 0.954	32.54 / 0.930
DnCNN [49]	35.58 / 0.961	33.01 / 0.936
DSCNN [44]	35.61 / 0.960	32.99 / 0.938
DKFN [26]	35.81 / 0.962	33.23 / 0.939
3D CNN	35.81 / 0.961	33.25 / 0.938
Our method	**35.95 / 0.965**	**33.39 / 0.943**

Table 4. Average ΔPSNR (dB) and ΔSSIM ($\times 10^{-2}$) on Yang *et al.*'s dataset.

QP	Seq.	ARCNN [11]	DnCNN [49]	DSCNN [44]	MFQE [45]	Our method
37	1	0.241 / 0.51	0.448 / 0.83	0.492 / 0.87	0.772 / 1.15	**0.827 / 1.21**
	2	0.115 / 0.30	0.439 / 0.52	0.458 / 0.58	0.604 / 0.63	**0.971 / 0.92**
	3	0.161 / 0.49	0.276 / 0.76	0.271 / 0.74	0.472 / 0.91	**0.483 / 0.99**
	4	0.183 / 0.35	0.377 / 0.55	0.393 / 0.54	0.438 / 0.48	**0.576 / 0.66**
	5	0.150 / 0.30	0.333 / 0.48	0.356 / 0.53	0.550 / 0.52	**0.598 / 0.74**
	6	0.161 / 0.23	0.415 / 0.50	0.435 / 0.49	0.598 / 0.51	**0.658 / 0.67**
	7	0.128 / 0.29	0.284 / 0.44	0.277 / 0.45	0.390 / 0.45	**0.394 / 0.58**
	8	0.125 / 0.37	0.276 / 0.61	0.230 / 0.63	0.484 / 1.01	**0.563 / 1.18**
	9	0.149 / 0.38	0.299 / 0.71	0.271 / 0.66	0.394 / 0.92	**0.439 / 1.03**
	10	0.146 / 0.24	0.289 / 0.58	0.274 / 0.54	0.402 / 0.80	**0.501 / 0.99**
	Ave.	0.156 / 0.35	0.344 / 0.59	0.346 / 0.60	0.510 / 0.74	**0.601 / 0.90**
42	Ave.	0.252 / 0.83	0.301 / 0.96	0.364 / 1.06	0.461 / —	**0.614 / 1.47**

1: *PeopleOnStreet* 2: *TunnelFlag* 3: *Kimono* 4: *BarScene* 5: *Vidyo1*
6: *Vidyo3* 7: *Vidyo4* 8: *BasketballPass* 9: *RaceHorses* 10: *MaD*

4.3. Quantitative Comparison

For a fair comparison, we use the same data processing method and training data. Thus, to demonstrate the advantage of our method, we compare it with five existing methods: ARCNN [11], DnCNN [49], DSCNN [44], DKFN [26] and MFQE [45]. For ARCNN, DnCNN and DSCNN, we get better results after they were retrained on Yang *et al.*'s dataset than that reported in [45]. For DKFN [26], we directly cite performance results from the original paper where it was evaluated only on Vimeo-90K. For MFQE [45], we cite the results of PSNR improvement, and compute SSIM improvement via our manually labeled PQFs and its published model[3]. Besides, considering that 3D CNN is able to capture spatiotemporal information from video frames, we also implement it for performance comparison by adopting the same architecture as [17] that aims at video super-resolution. Due to its large memory cost, we train and evaluate it only on Vimeo-90K.

Quality enhancement. Results of PSNR/SSIM on two datasets are in Table 3 and Table 4 respectively. In Table 3, we use HEVC with loop filters [34] as the baseline. From Table 3, we can see that our method outperforms the 2nd best methods (DKFN and 3D CNN) by about 0.14 dB in terms of PSNR and 25% improvement on ΔSSIM. These methods that utilize spatiotemporal information, including ours, DKFN and 3D CNN, all achieve better performance

---

[3] only publishes the model with QP=37.
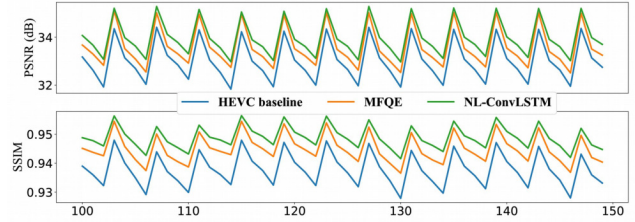
Figure 5. PSNR/SSIM curves of HEVC baseline, MFQE and NL-ConvLSTM on the sequence *TunnelFlag* with QP=37.

Table 5. Run-time (*ms per frame*) comparison among six methods.

Resolution	180x180	416x240	640x360	1280x720	1920x1080
ARCNN [11]	1.73	4.58	9.19	36.06	80.70
DnCNN [49]	6.30	15.84	35.51	139.77	315.83
DSCNN [44]	15.26	36.88	82.31	322.92	731.21
MFQE[4] [45]	20.28+	51.01+	112.87+	443.82+	1009.00+
original NL	4391.75	-	-	-	-
ours	102.13	304.11	621.94	2607.60	6738.00

than the remaining methods.

Yang *et al.*'s dataset has less training data than Vimeo-90K, and these two datasets are processed with different compression settings. All methods perform a little worse on Yang *et al.*'s dataset than on Vimeo-90K. However, from Table 4 we can still get similar conclusion: methods exploiting spatiotemporal information of neighboring frames perform better than these do not exploit. Our method outperforms all the other methods on all test sequences, and on average its PSNR/SSIM improvement is 17.8%/21.6% higher than that of MFQE. Specifically, for QP=37, our method achieves the highest PSNR/SSIM improvement on the 2nd sequence, which is 60.7% / 46% higher than that of MFQE. For QP=42, our method gets 33.19% and 68.68% increase of ΔPSNR over MFQE and DSCNN, respectively.

Quality fluctuation. Quality fluctuation is an index to evaluate the quality of a whole video [13, 45]. Drastic quality fluctuation often leads to severe temporal inconsistency and degradation of QoE. We evaluate fluctuation by Standard Deviation (STD) and Peak-Valley Difference (PVD)[5] of the PSNR/SSIM curves for each sequence as in [13, 45]. Here, we present only the STDs and PVDs of HEVC baseline, MFQE and our method for simplicity. For PSNR, the STD values of HEVC baseline, MFQE and our method are 1.130dB, 1.055dB and 1.036dB, and their PVD values are 1.558dB, 1.109dB, and 1.038dB, respectively. For SSIM, we notice similar trends. Fig. 5 shows the PSNR curves of HEVC baseline, MFQE and NL-ConvLSTM on the sequence *TunnelFlag*. In Fig. 5, comparing with MFQE, our method gets similar improvement on PQFs, but achieves much higher PSNR and SSIM improvement on non-PQFs. All these results show that our method performs more stably

---

[4]time for detecting PQFs are not included
[5]PVD calculates the average difference between peak values and their preceding/following nearest valley values

7049

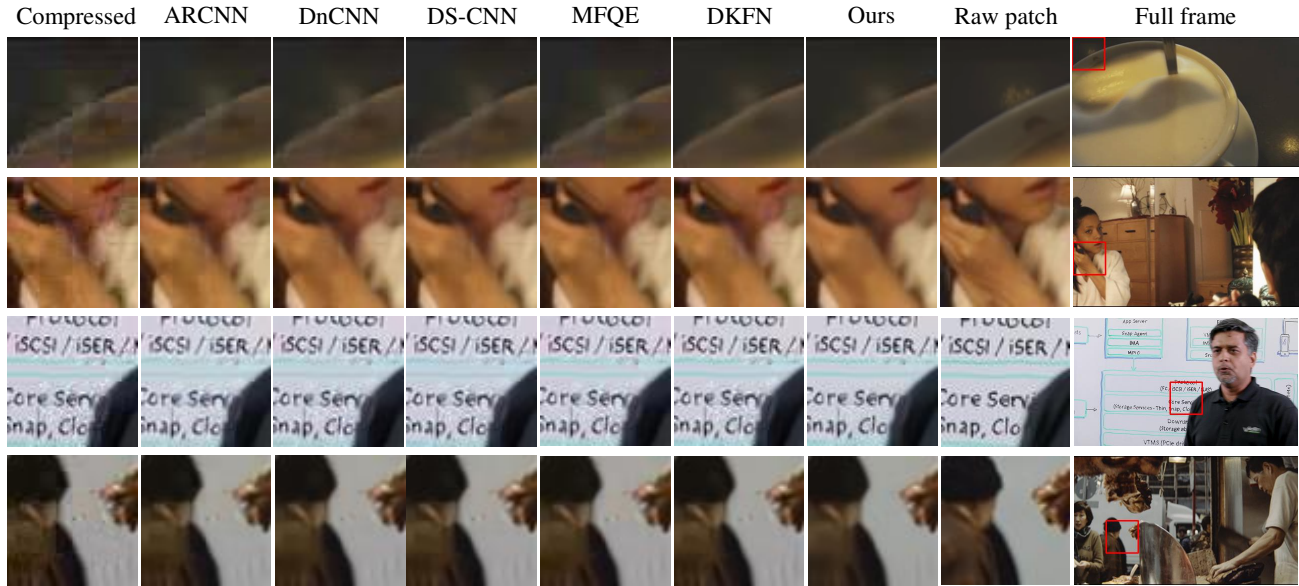| Compressed	ARCNN	DnCNN	DS-CNN	MFQE	DKFN	Ours	Raw patch	Full frame

Figure 6. Comparison of reduction effect on four kinds of compression artifacts.

than the baseline and MFQE.

4.4. Qualitative Comparison

Fig. 6 compares the reduction effect of different methods on four kinds of compression artifacts happened to images from Vimeo-90K with qp=37. The four artifacts are blocking, color bleeding, mosquito noise and ringing [48]. Each row stands for an image with a kind of compression artifacts. Concretely, in the 1st row, the cup edge is blurred due to blocking artifacts; and in the 2nd row, black color of hair overlapping the face results in a black stripe on the face. Words in the 3rd image are surrounded by mosquito noise, and the 4th image suffers from a silhouette-like shade parallel to the outline of the person. The last column shows the original frames, and the 8th column presents a cropped part from each original frame. The 1st column lists the compressed image of each cropped part. From the 2nd column to the 7th column, the cropped parts after artifact reduction by different methods are illustrated. Checking the images in Fig. 6, we can see that the cropped images after artifact reduction by our method (in the 7th column) have higher quality than these processed by the other methods, and are more similar to the original images (the 8th column). This means that our method can handle these distortions better than the five existing methods.

4.5. Run Time Comparison

In Tab. 5 we present run-time comparison results. As our method has to process more frames, it consumes more time than the other methods. However, the run-time of our method is acceptable, and our method uses much less time

than the original NL method (the NL-ConvLSTM without the approximation mechanism). Here, we give only the original NL's run-time at resolution 180×180 (for higher resolutions, it consumes too much time and GPU-memory). Our method can be further sped up by cudnn-accelerated ConvLSTM and highly-tuned implementation.

4.6. Applying to Other Compression Standard

Finally, we check whether our method is effective for compressed video sequences generated by other compression algorithms, such as H.264. The models are initialized with the corresponding models trained on video clips compressed by HEVC, and then are fine-tuned on video clips compressed by x264 in FFmpeg with QP=37. The resulting PSNR/SSIM improvements of our method on Vimeo-90K and Yang et al.'s dataset are 1.43dB/0.011 and 0.693dB/0.0085. The enhancement performance of our method on H.264 compressed videos is comparable to that on HEVC compressed videos. These results indicate that our method is also effective for H.264 compressed videos.

5. Conclusion

In this paper, we propose a novel end-to-end non-local ConvLSTM for video compression artifact removal by modeling spatial deformations and temporal variations simultaneously. Experiments on two datasets show that our method can enhance the quality of compressed videos considerably, remove artifacts effectively and outperform the existing methods. In the future, we plan to extend our method to other low-level video enhancement tasks, such as video super-resolution, interpolation and frame prediction.

References

[1] Siavash Arjomand Bigdeli, Matthias Zwicker, Paolo Favaro, and Meiguang Jin. Deep mean-shift priors for image restoration. In *NIPS*, pages 763–772, 2017.

[2] Frank Bossen. Common test conditions and software reference configurations, joint collaborative team on video coding (jct-vc) of itut sg16 wp3 and iso. Technical report, IEC JTC1/SC29/WG11, Doc. JCTVC-J1100, Stockholm, Sweden, 2012.

[3] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *CVPR*, volume 2, pages 60–65. IEEE, 2005.

[4] Jose Caballero, Christian Ledig, Andrew Aitken, Alejandro Acosta, Johannes Totz, Zehan Wang, and Wenzhe Shi. Real-time video super-resolution with spatio-temporal networks and motion compensation. In *CVPR*, pages 4778–4787, 2017.

[5] Huibin Chang, Michael K Ng, and Tieyong Zeng. Reducing artifacts in jpeg decompression via a learned dictionary. *IEEE Transactions on Signal Processing*, 62(3):718–728, 2014.

[6] Jen-Hao Rick Chang, Chun-Liang Li, Barnabas Poczos, BVK Vijaya Kumar, and Aswin C Sankaranarayanan. One network to solve them all-solving linear inverse problems using deep projection models. In *ICCV*, pages 5889–5898, 2017.

[7] Honggang Chen, Xiaohai He, Linbo Qing, Shuhua Xiong, and Truong Q Nguyen. Dpw-sdnet: Dual pixel-wavelet domain deep cnns for soft decoding of jpeg-compressed images. In *CVPR Workshops*, pages 711–720, 2018.

[8] Kai Cui and Eckehard Steinbach. Decoder side image quality enhancement exploiting inter-channel correlation in a 3-stage cnn: Submission to clic 2018. In *CVPR Workshops*, pages 2571–2574, 2018.

[9] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE TIP*, 16(8):2080–2095, 2007.

[10] Yuanying Dai, Dong Liu, and Feng Wu. A convolutional neural network approach for post-processing in hevc intra coding. In *International Conference on Multimedia Modeling*, pages 28–39. Springer, 2017.

[11] Chao Dong, Yubin Deng, Chen Change Loy, and Xiaoou Tang. Compression artifacts reduction by a deep convolutional network. In *ICCV*, pages 576–584, 2015.

[12] Leonardo Galteri, Lorenzo Seidenari, Marco Bertini, and Alberto Del Bimbo. Deep generative adversarial compression artifact removal. In *ICCV*, pages 4826–4835, 2017.

[13] Zhenyu Guan, Qunliang Xing, Mai Xu, Ren Yang, Tie Liu, and Zulin Wang. Mfqe 2.0: A new approach for multi-frame quality enhancement on compressed video. *arXiv preprint arXiv:1902.09707*, 2019.

[14] Jun Guo and Hongyang Chao. Building dual-domain representations for compression artifacts reduction. In *ECCV*, pages 628–644. Springer, 2016.

[15] Jun Guo and Hongyang Chao. One-to-many network for visually pleasing compression artifacts reduction. In *CVPR*, pages 4867–4876, 2017.

[16] Zhipeng Jin, Ping An, Chao Yang, and Liquan Shen. Quality enhancement for intra frame coding via cnns: An adversarial approach. In *ICASSP*, pages 1368–1372. IEEE, 2018.

[17] Younghyun Jo, Seoung Wug Oh, Jaeyeon Kang, and Seon Joo Kim. Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation. In *CVPR*, pages 3224–3232, 2018.

[18] Armin Kappeler, Seunghwan Yoo, Qiqin Dai, and Aggelos K Katsaggelos. Video super-resolution with convolutional neural networks. *IEEE Transactions on Computational Imaging*, 2(2):109–122, 2016.

[19] Tae Hyun Kim, Mehdi SM Sajjadi, Michael Hirsch, and Bernhard Schölkopf. Spatio-temporal transformer network for video restoration. In *ECCV*, pages 111–127. Springer, Cham, 2018.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[22] Ke Li, Bahetiyaer Bare, and Bo Yan. An efficient deep convolutional neural networks model for compressed image deblocking. In *ICME*, pages 1320–1325. IEEE, 2017.

[23] Ding Liu, Zhaowen Wang, Yuchen Fan, Xianming Liu, Zhangyang Wang, Shiyu Chang, and Thomas Huang. Robust video super-resolution with learned temporal dynamics. In *ICCV*, pages 2526–2534. IEEE, 2017.

[24] Ding Liu, Bihan Wen, Yuchen Fan, Chen Change Loy, and Thomas S Huang. Non-local recurrent network for image restoration. In *NIPS*, pages 1680–1689, 2018.

[25] Xianming Liu, Xiaolin Wu, Jiantao Zhou, and Debin Zhao. Data-driven sparsity-based restoration of jpeg-compressed images in dual transform-pixel domain. In *CVPR*, pages 5171–5178, 2015.

[26] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Zhiyong Gao, and Ming-Ting Sun. Deep kalman filtering network for video compression artifact reduction. In *ECCV*, pages 568–584, 2018.

[27] Danial Maleki, Soheila Nadalian, Mohammad Mahdi Derakhshani, and Mohammad Amin Sadeghi. Blockcnn: A deep network for artifact removal and image compression. In *CVPR Workshops*, pages 2555–2558, 2018.

[28] Xiaojiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *NIPS*, pages 2802–2810, 2016.

[29] Shigenobu Minami and Avideh Zakhor. An optimization approach for removing blocking effects in transform coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(2):74–82, 1995.

[30] Bhaskar Ramamurthi and Allen Gersho. Nonlinear space-variant postprocessing of block coded images. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(5):1258–1268, 1986.

[31] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, San-jeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[32] Mehdi SM Sajjadi, Raviteja Vemulapalli, and Matthew Brown. Frame-recurrent video super-resolution. In *CVPR*, pages 6626–6634, 2018.

[33] Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Deep learning for precipitation nowcasting: A benchmark and a new model. In *NIPS*, pages 5617–5627, 2017.

[34] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, Thomas Wiegand, et al. Overview of the high efficiency video coding(hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, 2012.

[35] Pavel Svoboda, Michal Hradis, David Barina, and Pavel Zemcik. Compression artifacts removal using convolutional neural networks. *arXiv preprint arXiv:1605.00366*, 2016.

[36] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Mem-net: A persistent memory network for image restoration. In *CVPR*, pages 4539–4547, 2017.

[37] Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. Detail-revealing deep video super-resolution. In *ICCV*, pages 22–29, 2017.

[38] Tingting Wang, Mingjin Chen, and Hongyang Chao. A novel deep learning-based method of improving coding efficiency from the decoder-end for hevc. In *Data Compression Conference, 2017*, pages 410–419. IEEE, 2017.

[39] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaim-ing He. Non-local neural networks. In *CVPR*, pages 7794–7803, 2018.

[40] Shuanhu Wu, Hong Yan, and Zheng Tan. An efficient wavelet-based deblocking algorithm for highly compressed images. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(11):1193–1198, 2001.

[41] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Ye-ung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NIPS*, pages 802–810, 2015.

[42] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *IJCV*, 127(8):1106–1125, 2019.

[43] Ren Yang, Mai Xu, Tie Liu, Zulin Wang, and Zhenyu Guan. Enhancing quality for hevc compressed videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(7):2039–2054, 2018.

[44] Ren Yang, Mai Xu, and Zulin Wang. Decoder-side hevc quality enhancement with scalable convolutional neural net-work. In *ICME*, pages 817–822. IEEE, 2017.

[45] Ren Yang, Mai Xu, Zulin Wang, and Tianyi Li. Multi-frame quality enhancement for compressed video. In *CVPR*, pages 6664–6673, 2018.

[46] Jaeyoung Yoo, Sang-ho Lee, and Nojun Kwak. Image restoration by estimating frequency distribution of local patches. In *CVPR*, pages 6684–6692, 2018.

[47] Avideh Zakhor. Iterative procedures for reduction of block-ing effects in transform image coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 2(1):91–95, 1992.

[48] Kai Zeng, Tiesong Zhao, Abdul Rehman, and Zhou Wang. Characterizing perceptual artifacts in compressed video streams. In *Human Vision and Electronic Imaging XIX*, vol-ume 9014, page 90140Q. International Society for Optics and Photonics, 2014.

[49] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE TIP*, 26(7):3142–3155, 2017.

[50] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning deep cnn denoiser prior for image restoration. In *CVPR*, volume 2, 2017.

[51] Xinfeng Zhang, Ruiqin Xiong, Xiaopeng Fan, Siwei Ma, and Wen Gao. Compression artifact reduction by overlapped-block transform coefficient estimation with block similarity. *IEEE TIP*, 22(12):4613–4626, 2013.