Learning Patterns of Latent Residual for Improving Video Compression

Yen-Chung Chen[1*] Keng-Jui Chang[1*] Yi-Hsuan Tsai[2] Wei-Chen Chiu[1]
[1]National Chiao Tung University, Taiwan [2]NEC Laboratories America

Abstract

We tackle the problem of reducing compression artifacts. Specifically, we focus on transmitting the residual from the original video, i.e. difference between a compressed video and its corresponding original/uncompressed one, together with the compressed video during video transmission. Our video compression pipeline is capable of diminishing the overall cost of transmitting the residual and simultaneously achieving comparable video quality with respect to a state-of-the-art baseline. We provide experimental results based on various compression methods on video datasets with great diversity, to substantiate the capacity of our pipeline in improving video compression.

1. Introduction

Video has become one of the most popular medium to communicate, share knowledge, and record events recently in our daily life. Along the development of recording technology and expectation of better visual experience, video data becomes higher resolution which naturally leads to larger size. However, as the communication channel or network is usually with limited bandwidth, video compression is essential to maintain the efficiency of video transmission.

Various video coding standards have been proposed for video compression, such as HEVC [7], MPEG-4 [5], and H.264 [10]. Generally, most of the video coding methods used by consumers belong to the lossy compression case, in which the file size of a video is reduced by eliminating redundant information or some details. Ideally, the loss caused by compression should be undetectable by end-users, but with the demand of transmitting more data grows (e.g., video streaming), it is inevitable to require more data reduction that would generate obvious compression artifacts, e.g., block boundary, mosquito noise, and blur.

The visual difference between a compressed video and its corresponding original/uncompressed one is known as *residual*. In order to reduce the influence of compression artifacts for a better viewing quality, several general strategies are investigated, including: 1) developing a new compression procedure to minimize the residual, 2) estimating the residual from the compressed video and performing reconstruction, and 3) transmitting the residual from the original image together with the compressed video. In this paper, we particularly focus on the third one since it directly extracts useful residual information from the original image (i.e., server side). A representative work [8] of this strategy utilizes an autoencoder framework to encode the residual frame-by-frame into binary representations for transmission from the server side to the client one. Although this method can provide better video quality than H.264 by employing the binary residual representations, the bandwidth it needs to transmit the residual information is still high and thus the bitrate can be significantly increased.

In this paper, to reduce such issue, we propose a holistic framework which is able to simultaneously eliminate the overall cost of transmitting the residual and achieve comparable video quality. The main advance in our proposed method is based on a hypothesis that the variety of patch-wise residual can be quantized into certain groups, in which the mean representation of these groups is named as *residual pattern*. Given a video frame, its residual information can thus be easily encoded by using the indexes of the residual pattern, combined from all the patches within this video frame. With this operation, we no longer need to transmit the original residual through the channel but only require to remember the indexes of residual pattern, which leads to a more efficient way for residual transmission.

Since the residual pattern can be trained and sent to the client beforehand, during running the application (i.e., video streaming), we simply need to feed our video data to the pre-trained model to generate the indexes, in which each index corresponds to one residual pattern. Then, based on another pre-trained reconstruction network, once the client receives the indexes, the video frame can be reconstructed immediately using the corresponding residual pattern. To achieve this, we design a framework consisting of three components: feature extraction on the residual, residual pattern discovery, and residual reconstruction.

In the experiment, we use several compression methods (i.e. H.264, HEVC, and VP9) and conduct extensive ex-

*The symbol $*$ indicates equal contribution.

periments on a large-scale video dataset, i.e. Kinetics [2], with various settings for qualitative and quantitative evaluation. The results in comparison to the state-of-the-art baseline successfully verify the effectiveness of our proposed method in improving the video quality without significant increase in the cost of residual transmission.

2. Proposed Method

Our proposed method consists of three components, as shown in Figure 1: 1) feature extraction on residual, 2) residual pattern discovery, and 3) residual reconstruction. In the following, we first sequentially introduce the details of each component, and then present the workflow of the overall learning procedure.

2.1. Feature Extraction on Residual

Given a compression method to compress a video sequence, in a frame-by-frame basis, the first component of our model aims to find the high-level representations of the residual information I_r between a video frame I and its compressed version I_c, i.e., $I_r = I - I_c$. This is realized by an autoencoder, which is a popular algorithm of learning feature representations of data in an unsupervised manner.

A typical architecture of an autoencoder is composed of a pair of encoder E and decoder D, where in our case the encoder projects a residual information I_r into a feature vector $E(I_r)$, and then the decoder maps it back to reconstruct the original residual $\tilde{I}_r = D(E(I_r))$. We minimize the objective of reconstruction error over $\mathcal{N}$ frames, which is defined as:

$$\mathcal{L}_{ae} = \sum_{i}^{\mathcal{N}} \left\| \tilde{I}_r^i - I_r^i \right\|_1 . \tag{1}$$

As such, the autoencoder learns to retain the latent residual information of the input I_r in the feature space, while extracting useful knowledge $E(I_r)$ for the next step.

2.2. Residual Pattern Discovery

As shown in Figure 1, the feature map $E(I_r)$ extracted from the residual image I_r is of the size $M \times N \times D$, where $M = H/8, N = W/8, D$ are height, width, and channel number respectively. We denote a vector P_{mn} of length D obtained from each spatial location (m, n) to represent a corresponding patch in the residual image with respect to its receptive field. In other words, we obtain in total $R = M \times N$ patches from the residual image and extract features for these patch-wise residuals.

We hypothesize that the collection $\mathcal{P}$ of all feature vectors representing patch-wise residuals from training video frames is distributed with multiple modes, i.e., they can be grouped and each group shows a specific pattern of patch-wise residual. Based on this hypothesis, the second component in our model is to perform clustering on $\mathcal{P}$ in order to discover residual patterns. Assume there are K modes/groups in $\mathcal{P}$, the center $\{C_k \mid k = 1 \cdots K\}$ of each group can be treated as the representative one and used to approximate other members belonging to the same group.

However, as the distribution of $\mathcal{P}$ is dependent upon the feature representations $E(I_r)$ extracted in Section 2.1, the quality of such approximation and the clustering outcome also varies accordingly. Therefore, during training the proposed model, we try to update the autoencoder such that $E(I_r)$ better fits our hypothesis and reflects a more compact structure in the distribution of $\mathcal{P}$. This is achieved by minimizing the objective:

$$\mathcal{L}_{cen} = \sum_{r}^{|\mathcal{P}|} \left\| P_r - C_{\kappa(r)} \right\|_1 , \tag{2}$$

where $|\mathcal{P}|$ denotes the number of feature vectors in $\mathcal{P}$, and $\kappa(r) \in \{1, \cdots, K\}$ is a mapping function to obtain the group index of a feature vector P_r.

Given a video frame I, its residual information I_r can now be efficiently represented as the group indexes $\tilde{P} = \{\kappa(r) \mid r = 1 \cdots R\}$ of corresponding patch-wise residual patterns $\{P_r \mid r = 1 \cdots R\}$. As a result, it costs only $R \times \log_2(K)$ bits to transmit the residual, which is significantly lower than transmitting the real residual in double precision. To further reduce the cost during transmission, we apply Huffman coding [1] on $\tilde{P}$ and is empirically able to reduce around 97% of the bit-rate, where K is set to 2048 in our experiments.

2.3. Residual Reconstruction

When a client receives from the server a compressed video frame I_c together with its residual represented by $\tilde{P}$, we now consider how to use $\tilde{P}$ for improving the quality of I_c. Here, we assume that the client stores the database of K representative patch-wise residual patterns beforehand. Given $\{C_k \mid k = 1 \cdots K\}$ learned from the stage of residual pattern discovery, $\tilde{P}$ is able to be converted to approximated residual features via retrieving center patches from indexes as $P_c = \{C_{\kappa(r)} \mid r = 1 \cdots R\}$. We then propose a residual reconstruction network T to reconstruct the original video frame I based on the input P_c and I_c, in which T is composed of two sub-networks: an upsampling network U and a refinement network F.

As shown in Figure 1, the upsampling network U first takes P_c as input and map it to a higher-dimensional feature map $U(P_c)$, and then the refinement network F processes $U(P_c)$ and I_c to output the final result $\tilde{I} = F(I_c, U(P_c))$. The objective of training the reconstruction network is to minimize the difference between $\tilde{I}$ and I:

$$\mathcal{L}_{rec} = \sum_{i}^{\mathcal{N}} \left\| \tilde{I}^i - I^i \right\|_1 . \tag{3}$$

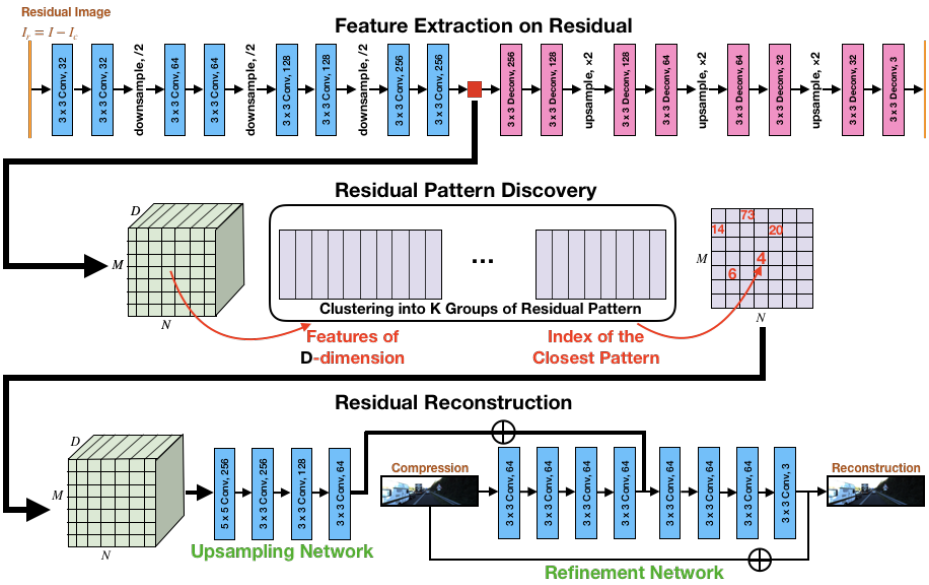

Figure 1. Architecture of the proposed frame-by-frame video compression enhancement method. There are three components in our model, we background the architecture of each component by three different colors.

To be detailed, the upsampling network U follows the architecture of ESPCN [6]. The refinement network F is an 8-layer convolutional neural network with each layer followed by a ReLU activation except for the last one, where I_c and $U(P_c)$ are fed into F through the first and fifth layer respectively.

Data: Input frames I and the corresponding compression I_c in training videos.
for *Each Epoch* **do**
  if *index of current epoch* $< L$ **then**
    $\forall I : \theta_E, \theta_D \overset{+}{\leftarrow} -\Delta_{\theta_E,\theta_D}\mathcal{L}_{ae}$;
    $\forall I : \theta_E, \theta_U, \theta_F \overset{+}{\leftarrow} -\Delta_{\theta_E,\theta_U,\theta_F}\mathcal{L}^*_{rec}$;
    clustering on $\mathcal{P}$ to get $\{C_k \mid k = 1 \cdots K\}$;
  else
    $\forall I : \theta_U, \theta_F \overset{+}{\leftarrow} -\Delta_{\theta_U,\theta_F}\mathcal{L}_{rec}$;
    $\theta_E \overset{+}{\leftarrow} -\Delta_{\theta_E}\mathcal{L}_{cen}$;
    $\forall I : \theta_E, \theta_D \overset{+}{\leftarrow} -\Delta_{\theta_E,\theta_D}\mathcal{L}_{ae}$;
    clustering on $\mathcal{P}$ to get $\{C_k \mid k = 1 \cdots K\}$;
  end
end

Algorithm 1: Training procedure of our method.

2.4. Training Procedure

To train the proposed model, we alternatively perform optimization on each loss function from each step (i.e., $\mathcal{L}_{ae}$, $\mathcal{L}_{cen}$, and $\mathcal{L}_{rec}$). We summarize the overall training procedure in Algorithm 1, where we denote the parameters of {encoder E, decoder D, upsampling network U, and refinement network F} as $\{\theta_E, \theta_D, \theta_U, \theta_F\}$ respectively.

In particular, for the first L epochs during training, we skip the stage of residual pattern discovery and focus on learning the autoencoder and reconstruction network, in order to stabilize the training at the early stage. Therefore, the input to the upsampling network becomes the real patch-wise residual features $\{P_r \mid r = 1 \cdots R\}$, while the objective function for reconstruction is thus re-written as:

$$\mathcal{L}^*_{rec} = \sum_{i}^{\mathcal{N}} \left\| F(I^i_c, U(P^i_r)) - I^i \right\|_1 . \tag{4}$$

3. Experiments

Dataset and Protocol. We use Kinetics dataset [2] in our experiments, which is a large-scale and high-quality dataset collected from Youtube, composed of a diverse range of human activities. We collect two subsets from Kinetics for training and testing respectively. The training subset contains 10K frames in 1093 videos over 391 classes, while the testing subset has 3253 frames in 345 videos over 345 classes. Note that the video classes in the testing set is overlapped with the ones in the training set, but the video sequences are not overlapped. In order to remove compression artifacts introduced by prior codecs on YouTube, we follow the same procedure as in [11] to downsample high resolution videos (with width and height greater than 720px) into the ones of 352×288px. We adopt the PSNR and SSIM [9] metrics for quantitative evaluation, which is typical in most of research works on video compression.

Table 1. Results on the Kinetics dataset with various coding standards and bitrates.

Kinetics	Coding Standard	H.264			HEVC			VP9		
	BitRate (bits/sec)	1M	2M	5M	1M	2M	5M	1M	2M	5M
PSNR	Original	31.638	34.680	37.271	29.209	33.255	37.512	33.152	35.008	36.445
	DRN+ [4, 13]	32.776	36.264	39.432	29.944	34.550	39.536	34.276	36.296	38.162
	Ours	**33.044**	**36.384**	**39.651**	**30.030**	**34.570**	**39.702**	**34.425**	**36.555**	**38.292**
SSIM	Original	0.939	0.967	0.984	0.878	0.927	0.983	0.951	0.968	0.979
	DRN+ [4, 13]	0.947	0.973	0.988	0.885	0.932	0.986	0.957	0.972	**0.984**
	Ours	**0.949**	**0.974**	**0.989**	**0.889**	**0.933**	**0.987**	**0.958**	**0.973**	**0.984**

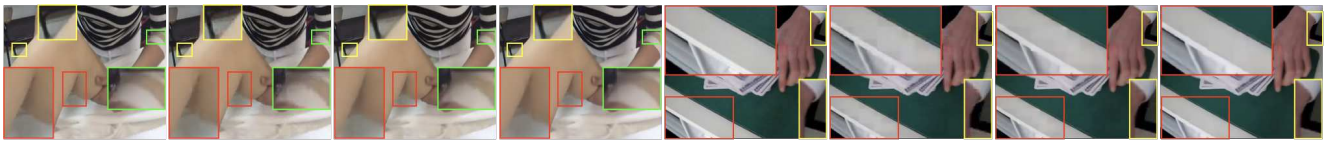

Figure 2. Example results on the Kinetics dataset, where DRN+ is likely to produce undesirable results. For each example, the first (leftmost) column is the original frame, the second column is the compressed frame, the third column is the result of DRN+, and the forth column is the result of our method.

3.1. Video Compression Performance

We conduct experiments on several coding standards and bit-rate settings. For comparison, we introduce our baseline model, an artifact removal network which is a deep learning-based method based on [3, 13] with residual blocks. Residual information is willing to be learned for removing compression artifacts by an 8-layer convolutional neural networks, in which each conv-layer is followed by a ReLU activation function except for the last one. Note that there is no any stride in our baseline model in order to keep the resolution of frames. The network is fed by the compressed image as input in a frame-by-frame process and outputs the enhanced image. We train this baseline model (referrerd as DRN+) on Kinetics with various bit-rates.

Note that, in our proposed method, additional bandwidth is required during transmitting group indexes of patch-wise residual patterns. That is, apart from the bandwidth used by video coding standards, an extra bandwidth is consumed by transmission of indexes. Hence, for a fair comparison, we also take this into consideration by adding this extra bandwidth to the one from video coding during training the baseline (i.e., allowing for more bitrate in video coding), so that the overall bandwidth consumed would be equal for both the baseline and our method.

Quantitative Results. In Table 1, we present comparisons on the Kinetics dataset under various coding standards, including H.264, HEVC, and VP9. On this dataset with diverse set of object categories and large motions, our results are consistently better than the DRN+ baseline. This attributes from our design that learns patch patterns from the input residual, which makes the following reconstruction task easier. In contrast, the baseline method focuses on reconstruction directly from the compressed frame, which may suffer from the overfitting issue and is not aware of the residual pattern. Additionally, we evaluate our model on test set (in total 7K frames) of Vimeo-90k dataset [12], the results are shown in Table 2, where our model is able to outperform the baseline under three different coding standards.

Table 2. Results on the Vimeo-90k dataset. Bit-rate: 5Mbps.

Vimeo-90k	Coding Standard	H.264	HEVC	VP9
PSNR	Original	38.33	39.01	39.62
	DRN+ [4, 13]	40.18	40.89	41.50
	Ours	**40.47**	**41.24**	**41.79**

Qualitative Results. We present some example results with original frame, compressed frame, and our reconstructed result in Figure 2. The results show that our reconstruction from the H.264, VP9 and HEVC standard is able to recover some details.

4. Conclusion

In this paper, we present a deep learning-based method for addressing the issue of reducing the influence of compression artifacts. We perform clustering on the patch-wise latent residual to find residual patterns. We only require to transmit pattern indexes of patches additionally, such that we can obtain the corresponding residual information on the client side. Our proposed method simultaneously reduces the cost of transmitting residual information and boosts video quality and experimentally show superior performance on various coding standards and bitrates.

References

[1] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 1952. 2

[2] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *ArXiv:1705.06950*, 2017. 2, 3

[3] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4

[4] Ogun Kirmemis, Gonca Bakar, and A Murat Tekalp. Learned compression artifact removal by deep residual networks. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018. 4

[5] Weiping Li. Overview of fine granularity scalability in MPEG-4 video standard. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 2001. 1

[6] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[7] 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 (TCSVT)*, 2012. 1

[8] Yi-Hsuan Tsai, Ming-Yu Liu, Deqing Sun, Ming-Hsuan Yang, and Jan Kautz. Learning binary residual representations for domain-specific video streaming. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018. 1

[9] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *Asilomar Conference on Signals, Systems, and Computers*, 2003. 3

[10] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 2003. 1

[11] Chao-Yuan Wu, Nayan Singhal, and Philipp Krähenbühl. Video compression through image interpolation. In *European Conference on Computer Vision (ECCV)*, 2018. 3

[12] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision (IJCV)*, 2017. 4

[13] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing (TIP)*, 2017. 4