

# Boosting Image and Video Compression via Learning Latent Residual Patterns

Yen-Chung Chen\*<sup>1</sup>  
yenc.cs06g@nctu.edu.tw

Keng-Jui Chang\*<sup>1</sup>  
adplz53.cs06g@nctu.edu.tw

Yi-Hsuan Tsai<sup>2</sup>  
ytsai@nec-labs.com

Wei-Chen Chiu<sup>1</sup>  
walon@cs.nctu.edu.tw

<sup>1</sup> National Chiao Tung University

<sup>2</sup> NEC Laboratories America

---

## Abstract

Reducing compression artifacts is essential for streaming videos with a better quality under a limited bandwidth. To tackle this problem, existing methods aim to directly recover details from the compressed video but do not consider learning rich information in uncompressed videos to aid this process. In this paper, we focus on utilizing the residual information, which is the difference between a compressed video and its corresponding original/uncompressed one, and propose a fairly efficient way to transmit the residual with the compressed video in order to boost the quality of video compression. Our proposed method is realized by learning to discover the patterns in the residual and storing them offline as dictionary-like patterns. During the testing stage, e.g., for video streaming, the residual is transmitted in the form of pattern indexes to reduce the cost of communication, and thus the original residual information can be easily retrieved back from the dictionary of learned residual patterns. We show the effectiveness of our framework on numerous datasets under various video compression coding methods. In addition, the proposed pipeline can be widely applied to the image compression task and reduce artifacts produced from conventional and CNN-based compression algorithms.

## 1 Introduction

Video and image have become two of the most popular media for people to communicate, share knowledge, record events, and have entertainment in our daily life. Along the development of recording technology and expectation of better visual experience, video and image data become higher resolution which naturally leads to larger size. However, as the network or communication channel is usually with limited bandwidth, video and image compression is essential to maintain the efficiency of transmission. Various coding standards have been proposed to perform compression, such as HEVC [23], MPEG-4 [15], and H.264 [28] for video compression, and JPEG [26], JPEG-2000 [11], and BPG [8] for image compression. Generally, most of the coding methods nowadays belong to lossy compression scheme, in which the file size is reduced by eliminating redundant information or some details. Ideally,

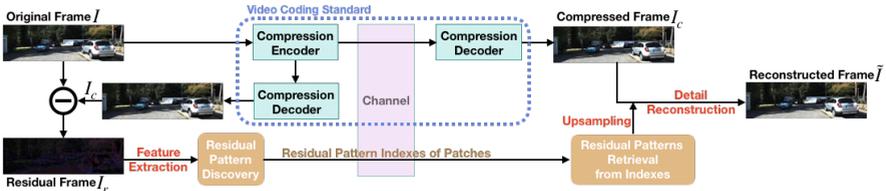


Figure 1: Overview of the proposed pipeline of frame-by-frame video compression enhancement. Our method first takes the residual frame as input, which is the difference between the original and compressed frame. Second, we extract features from the residual information and perform residual pattern discovery on the patch level, where the residual features are stored as a form of pattern indexes for reducing the required bandwidth during transmission through the channel, e.g., video streaming. On the client side, we first retrieve corresponding residual patterns from the received indexes, and then further use them to reconstruct the residual information and improve the quality of compressed frames.

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 generate obvious artifacts on data received by the user, e.g., block boundary, mosquito noise, and blur.

In order to reduce the influence of compression artifacts for a better viewing quality, several general strategies are investigated to reduce the visual difference between a compressed image (or video frame) and its corresponding original/uncompressed one, which is known as *residual*. These methods include: 1) developing a new compression procedure to minimize the residual, 2) estimating the residual from the compressed image and performing reconstruction, and 3) transmitting the residual together with the compressed image. 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 [25] of this strategy utilizes an autoencoder framework to encode the residual of a video sequence frame-by-frame into binary streams for transmission from the server side to the client one. Although this method provides better video quality than H.264 by employing binary residual representations, the bandwidth it needs to transmit the residual 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 visual 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 that are stored offline like a dictionary, in which the mean feature representation of each group is named as one *residual pattern*. As a result, given each patch in an image (or a video frame), its residual information can thus be retrieved by finding “which” pattern for reconstruction. Given an image (or 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 image. Based on 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 (see Figure 1 as an illustration). Specifically, since the residual pattern can be learned and sent to the client beforehand, during running the application (e.g., video streaming), we simply need to feed our image or video frames to the pre-trained model to generate the indexes, in which each index corresponds to one residual pattern. Then, based on another reconstruction network, once the client receives the indexes, the image 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 order to show the flexibility and generalizability of our proposed approach, we utilize different compression methods for both video and image data, including H.264, HEVC, VP9, JPEG, BPG, and [19], where the last one is a deep learning-based compression model. We conduct extensive experiments on several datasets, including KITTI [8] and Kinetics [22] for video compression, as well as ILSVRC [24] and Kodak [6] for image compression, with various settings for quantitative evaluation. The results in comparison to the state-of-the-art baselines verify the effectiveness of our proposed method in improving the visual quality without significant increase in the cost of residual transmission.

## 2 Related Works

**Image/Video Compression** Conventional image/video encoding procedures (e.g., JPEG [16], MPEG-4 [15], H.264 [28], and HEVC [23]) are built upon several important components, such as transformation coding, quantization, and entropy coding, where the transformation coding is particularly responsible for extracting features from image/video data. However, the common choices for transformation coding (e.g. Fourier transform, discrete cosine transform) are mostly linear operations, and most importantly, need careful hand-crafted design. Therefore, it can not be well generalized to different cases to have simultaneously efficient compression and good image/video compression quality. For instance, H.264 [28], which is a widely used video compression codec based on DCT, can provide good video quality under the case of high bitrate but suffer from blocking artifacts for the low bitrate.

**Learning-based Image/Video Compression** Previous works [9, 18, 22] have explored to automatically learn the transformation or feature extraction from visual data in order to assist the compression process. Particularly, the recent advance of deep learning brings a great progress along this direction, where most of proposed models [9, 24] inherit from autoencoder frameworks [10], in which the encoding and decoding processes in an autoencoder is analogous to the concept of compression. For instance, [24] propose to train a convolutional autoencoder for learning feature representations of images, and then apply typical quantization and entropy encoding on the features to perform compression. Instead of having a stage-by-stage pipeline as [24], [9] propose a deep architecture for learning compression, which integrates a uniform quantizer into the autoencoder. The discontinuous loss function for learning quantizer is approximated by a continuous proxy and thus the overall model is end-to-end trainable. In addition to dealing with quantization, [19] advance to use entropy coding techniques based on 3D-CNN context model for balancing the trade-off between the reconstruction error and the entropy of latent representation in the autoencoder. On the other hand, [29] recently propose an end-to-end trained deep video codec, where the video compression is formulated as image interpolation with motion compensation.

**Image/Video Artifact Removal** Another strategy of improving the quality of compressed output is to enhance the decoder or perform post-processing to reconstruct a better output [16, 17, 30]. For example, [20] enable the decoder to synthesize compressed output with photo-realistic quality by utilizing adversarial learning [9]; [5] propose a post-processing model based on the super-resolution network for removing artifacts on the compressed image. The work from [7] has a similar idea but the artifact removal network is implemented as a generative adversarial network (GAN) [9]. Recently, [25] addresses the video compression task from a different perspective: the compressed data is coupled with the corresponding residual, which is encoded in a binary form. Therefore, the quality of compressed video

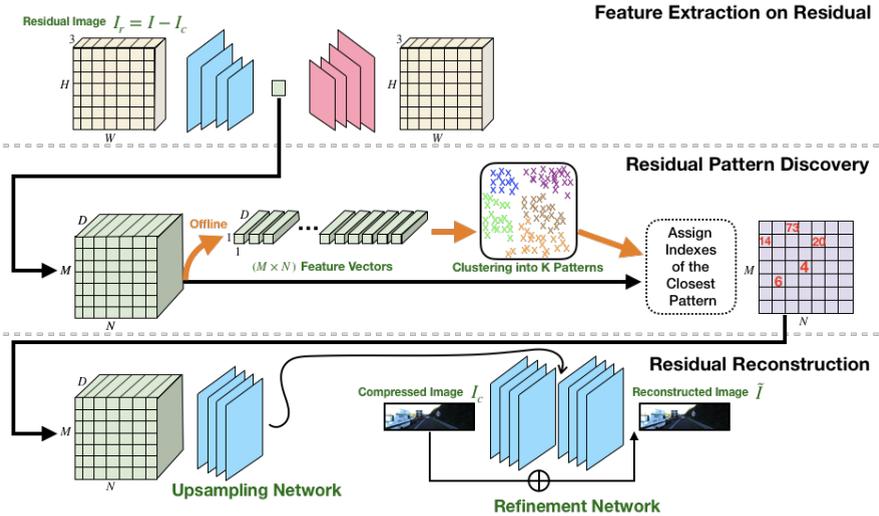


Figure 2: Overview of our proposed architecture to efficiently transmit the residual for improving compression, which contains three main components: feature extraction on residual, residual pattern discovery, and residual reconstruction. Please refer to Section 3 for details.

frames can be improved by adding back the decoded residual information accordingly. Our proposed method is closely related to [25], but advances to propose a holistic framework for improving the quality of compressed data with a more efficient way to encode the residual information by discovering patterns of residual. Moreover, the work from [25] only targets at domain-specific videos, whereas our method is designed for a general usage and can be applicable to reduce artifacts generated from various images/videos compression methods, including conventional and learning-based ones.

### 3 Proposed Method

Our goal is to learn efficient representations of residual information by discovering its patterns and approximating them, such that the residual can be transmitted with a minimal occupation of bandwidth and boost the quality of compressed videos or images. Our proposed framework consists of three components, as shown in Figure 2: 1) feature extraction on residual, 2) residual pattern discovery, and 3) residual reconstruction. We detail each component and the learning workflow in the following, where the network architectures and implementation details are provided in the supplement. The code and model will be released to the public.

#### 3.1 Feature Extraction on Residual

Given a compression method such as H.264 to compress a video 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 widely used for unsupervisedly learning feature representations of data. 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  $E(I_r)$ , then the decoder maps it back to reconstruct the original residual  $\tilde{I}_r = D(E(I_r))$ . The reconstruction error minimized over  $\mathcal{N}$  frames is defined as:

$$\mathcal{L}_{ae} = \sum_{i=1}^{\mathcal{N}} \|\tilde{I}_r^i - I_r^i\|_1. \quad (1)$$

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.

### 3.2 Residual Pattern Discovery

As shown in Figure 2, 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  $P_r^{mn}$  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  $P_r$  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. Here we adopt  $k$ -means as our clustering algorithm. 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 extracted features  $E(I_r)$ , the quality of such approximation and the clustering outcome also varies accordingly. Therefore, we try to update the autoencoder during model training such that  $E(I_r)$  better fits our hypothesis and reflects a more compact structure in the distribution of  $\mathcal{P}$ , by minimizing the objective:

$$\mathcal{L}_{cen} = \sum_{r=1}^{|\mathcal{P}|} \|P_r - C_{\kappa(r)}\|_1, \quad (2)$$

where  $|\mathcal{P}|$  denotes the number of feature vectors in  $\mathcal{P}$ , and  $\kappa(r) \in \{1, \dots, K\}$  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 [14] on  $\tilde{P}$  and is empirically able to reduce around 97% of the bitrate.

### 3.3 Residual Reconstruction

When a client receives from the server a compressed video frame  $I_c$  together with its residual represented as  $\tilde{P}$ , the ensuing task is 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,  $\tilde{P}$  can 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 frame  $I$  based on  $P_c$  and  $I_c$ , where  $T$  is composed of two sub-networks: an upsampling network  $U$  and a refinement network  $F$ . As shown in Figure 2, the upsampling network  $U$  first maps  $P_c$  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 reconstruction network is to minimize the difference between  $\tilde{I}$  and  $I$ :

$$\mathcal{L}_{rec} = \sum_i^{\mathcal{N}} \|\tilde{I}^i - I^i\|_1. \quad (3)$$

### 3.4 Iterative Training Procedure

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 the 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:

$$\mathcal{L}_{rec}^* = \sum_i^{\mathcal{N}} \|F(I_c^i, U(P_r^i)) - I^i\|_1. \quad (4)$$

---

**Algorithm 1:** Training procedure of our proposed framework.

---

**Data:** Input frames  $I$  and the corresponding compression  $I_c$  in training videos.

```

1 for Each Epoch do
2   if index of current epoch < L then
3      $\forall I : \theta_E, \theta_D \stackrel{\pm}{\leftarrow} -\Delta_{\theta_E, \theta_D} \mathcal{L}_{ae}$ ;
4      $\forall I : \theta_E, \theta_U, \theta_F \stackrel{\pm}{\leftarrow} -\Delta_{\theta_E, \theta_U, \theta_F} \mathcal{L}_{rec}^*$ ;
5     clustering on  $\mathcal{P}$  to get  $\{C_k \mid k = 1 \cdots K\}$ ;
6   else
7      $\forall I : \theta_U, \theta_F \stackrel{\pm}{\leftarrow} -\Delta_{\theta_U, \theta_F} \mathcal{L}_{rec}$ ;
8      $\theta_E \stackrel{\pm}{\leftarrow} -\Delta_{\theta_E} \mathcal{L}_{cen}$ ;
9      $\forall I : \theta_E, \theta_D \stackrel{\pm}{\leftarrow} -\Delta_{\theta_E, \theta_D} \mathcal{L}_{ae}$ ;
10    clustering on  $\mathcal{P}$  to get  $\{C_k \mid k = 1 \cdots K\}$ ;

```

---

## 4 Experiments

**Datasets and Metrics.** Four datasets are considered, including the tracking benchmark on **KITTI** [8] and **Kinetics** [14] for video compression, **ILSVRC** [23] and **Kodak** [6] for the image part, where the first three are used for both training and testing while the last one (i.e. Kodak) is for testing only. The PSNR and SSIM [24] metrics are adopted for evaluation.

- **KITTI:** Tracking benchmark on KITTI includes 50 various sequences of street scenes. We resize the videos into resolution of  $360 \times 1200$  for our feature extraction, which contains three downsampling operations. Our training and testing split of KITTI dataset is exactly the same as [23], in which the training split contains 14688 frames from 42 videos. The remaining 8 videos (i.e., 3590 frames) are used for the testing split.
- **Kinetics:** Kinetics is a large-scale dataset collected from Youtube which has a diverse range of human activities. Two subsets are collected from Kinetics: 10K frames in 1093

videos over 391 classes for training, and 3253 frames in 345 videos over 345 classes for testing respectively. Note that the video classes are overlapped between training and testing sets, but the video sequences are not. In order to remove compression artifacts introduced by prior codecs on YouTube, we follow the same procedure as in [29] to downsample high resolution videos into the ones of  $352 \times 288$ px.

- **ILSVRC:** The ImageNet dataset from ILSVRC2012 is used. Both training and testing sets (i.e., 10996 and 2500 images respectively) are obtained by randomly sampling from around 1000 categories. We apply  $128 \times 128$  center crops for image preprocessing.
- **Kodak:** Kodak dataset consists of 24 lossless images, and is commonly used for testing.

**Number of Residual Pattern Group  $K$ .** In order to determine  $K$  (i.e., the number of groups for residual patterns), we conduct an ablation study on KITTI, which is detailed in the supplementary material. We empirically find that the performance saturates when  $K$  goes up to a certain amount, e.g.,  $K = 2048$ . In practice, we use  $K = 1024$  for all the other experiments to account for both the accuracy and efficiency.

## 4.1 Video Compression Performance

We conduct experiments on several coding standards (e.g., HEVC, H.264, and VP9) under various bitrate settings, and introduce two baseline models for the comparison: 1) a state-of-the-art video compression pipeline from [25] which originally is proposed for domain-specific video streaming, and 2) an artifact removal network which is widely used to eliminate the distortions caused by compression, denoted as DRN+. For the former baseline from [25], as discussed in the section of related works, it aims to transmit the binary-encoded residual together with the compressed data. We adopt the best experimental setting in their paper, i.e., {number of channels, number of layers} = {32,3}. For the latter baseline, it is an 8-layer convolutional neural network based on [13, 30] with residual blocks, in which each conv-layer is followed by a ReLU activation function except for the last one. The network is fed by the compressed image as input in a frame-by-frame process and outputs the enhanced image. Note that there is no any stride in the DRN+ model in order to keep the resolution of frames. Both baselines are trained on KITTI and Kinetics datasets with different codecs and bitrates.

Note that, apart from the bandwidth used by video coding standards, both our model and the baseline from [25] require additional bandwidth to transmit the residual information: our proposed model consumes  $\sim 0.043/0.01$  Mbps for KITTI/Kinetics datasets while [25] needs  $\sim 0.96/0.23$  Mbps respectively. Hence, in order to have fair comparison, during training baselines and our model, we take these extra bitrate into consideration by adjusting the bandwidth of video coding (e.g., deduce the bitrate of video coding used for the [25] method), such that each method consumes the overall bandwidth equally.

**Quantitative Results.** In Table 1 we show that our method generally outperforms both baseline models on KITTI and Kinetics, achieving improvement by at most 1.9dB in PSNR (under H.264 codec on Kinetics). This attributes from our design that 1) learns patch patterns from the input residual, which makes the following reconstruction task easier, and 2) transmits the residual information in a more efficient manner. In contrast, the DRN+ baseline focuses on reconstruction directly from the compressed frame, which may suffer from the overfitting issue and is not aware of the residual pattern, while [25] requires much higher bitrate to transmit the residual information such that it can only use less bitrate for video coding under the same overall bandwidth. We note that, the Kinetics dataset is a more challenging one than KITTI, as it contains a diverse set of object categories with motions.

		Kinetics									KITTI		
Coding Standard		H.264			HEVC			VP9			H.264		
BitRate (bits/sec)		1M	2M	5M									
PSNR	Original	31.638	34.680	37.271	29.209	33.255	37.512	33.152	35.008	36.445	26.039	27.692	28.568
	Tsai <i>et al.</i> [14]	31.156	34.694	37.775	<b>30.116</b>	33.167	38.064	33.318	35.093	36.967	26.702	28.298	29.047
	DRN+ [15, 16]	32.776	36.264	39.432	29.944	34.550	39.536	34.276	36.296	38.162	26.664	28.342	29.273
	Ours	<b>33.044</b>	<b>36.384</b>	<b>39.651</b>	30.030	<b>34.570</b>	<b>39.754</b>	<b>34.425</b>	<b>36.555</b>	<b>38.292</b>	<b>27.000</b>	<b>28.527</b>	<b>29.679</b>
SSIM	Original	0.895	0.940	0.967	0.814	0.887	0.968	0.911	0.940	0.958	0.750	0.813	0.849
	Tsai <i>et al.</i> [14]	0.884	0.935	0.965	<b>0.854</b>	0.886	0.964	0.915	0.935	0.958	0.765	0.829	0.847
	DRN+ [15, 16]	0.909	0.952	0.976	0.827	0.895	0.972	0.922	0.947	0.966	0.768	0.827	0.860
	Ours	<b>0.913</b>	<b>0.953</b>	<b>0.977</b>	0.831	<b>0.898</b>	<b>0.976</b>	<b>0.925</b>	<b>0.950</b>	<b>0.967</b>	<b>0.778</b>	<b>0.832</b>	<b>0.869</b>

Table 1: Results on the Kinetics and KITTI datasets with several coding standards (i.e., H.264, HEVC, and VP9) under various bitrates.

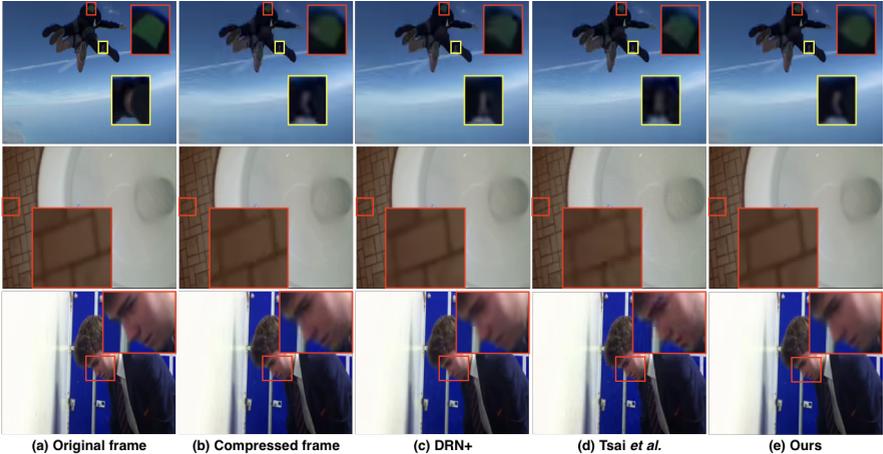


Figure 3: Example results on the Kinetics dataset. We show that under severe condition such as low bit-rate, DRN+ and Tsai *et al.* are likely to produce undesirable results. For example, in the second row of (c), the tile line in the bottom-left corner of the zoom-in patch is missing.

**Qualitative Results.** We present some example results with original frame, compressed frame, results from baselines and our model in Figure 3. The results show that our reconstruction from the H.264, VP9 and HEVC standard is able to recover more details than baselines. More results are provided in the supplementary material.

## 4.2 Image Compression Performance

In addition to video compression, our framework is also applicable to the image case. We apply our method based on various codecs such as JPEG2000, BPG and a deep learning-based model [19], which is the state-of-the-art image compression method. We adopt the same baselines as the video compression part. For JPEG2000 and BPG, we set the bpp (bits/pixel) to be 0.15 as reference to the recent challenge on learned image compression, hold in CVPR 2019. For [25] baseline, we use {number of channels, number of layers} = {8, 5} for having the same extra bitrate as ours, since the original setting {32, 3} will exceed more than 0.15bpp. The results on ILSVRC are presented in Table 2, and our method performs favorably against both baseline methods under different codecs. In Table 3, we further use the Kodak dataset, which is considered as a common benchmark for image compression task due to its high quality. Since there is no training set available on this dataset, we directly

ILSVRC	Coding Standard	JPEG2000	BPG	Coding Standard	DL-based		
	bpp (bits/pixel)	0.15	0.15	Quality	low	medium	high
PSNR	Original	22.989	28.625	Original	25.274	26.003	26.786
	Tsai <i>et al.</i> [15]	23.157	28.538	Tsai <i>et al.</i> [15]	25.290	26.150	26.917
	DRN+ [16, 17]	24.876	29.112	DRN+ [16, 17]	25.550	26.448	27.295
	Ours	<b>25.327</b>	<b>29.299</b>	Ours	<b>25.902</b>	<b>26.843</b>	<b>27.885</b>
SSIM	Original	0.570	0.755	Original	0.758	0.788	0.808
	Tsai <i>et al.</i> [15]	0.571	0.752	Tsai <i>et al.</i> [15]	0.758	0.787	0.806
	DRN+ [16, 17]	0.656	0.772	DRN+ [16, 17]	0.765	0.797	0.820
	Ours	<b>0.658</b>	<b>0.779</b>	Ours	<b>0.773</b>	<b>0.809</b>	<b>0.835</b>

Table 2: Results on the ILSVRC dataset. For JPEG2000 and BPG, 0.15 bpp is adopted. For deep-learning based compression method, we follow their release code to have three levels.

Kodak	Coding Standard	JPEG2000	BPG	Coding Standard	DL-based		
	bpp (bits/pixel)	0.15	0.15	Quality	low	medium	high
PSNR	Original	12.912	28.537	Original	27.179	28.775	30.147
	Tsai <i>et al.</i> [15]	13.071	28.538	Tsai <i>et al.</i> [15]	27.204	28.908	30.296
	DRN+	13.062	29.062	DRN+	27.186	28.802	30.058
	Ours	<b>13.513</b>	<b>29.448</b>	Ours	<b>27.830</b>	<b>29.084</b>	<b>30.402</b>
SSIM	Original	0.423	0.755	Original	0.845	<b>0.887</b>	<b>0.918</b>
	Tsai <i>et al.</i> [15]	0.442	0.755	Tsai <i>et al.</i> [15]	0.845	<b>0.887</b>	0.917
	DRN+ [16, 17]	0.448	0.773	DRN+ [16, 17]	0.836	0.879	0.908
	Ours	<b>0.468</b>	<b>0.789</b>	Ours	<b>0.854</b>	0.885	0.914

Table 3: Results on the Kodak dataset. For JPEG2000 and BPG, 0.15 bpp is adopted. For deep-learning based compression method, we follow their release code to have three levels.

use the model trained on ILSVRC for evaluation. Similarly we observe that our results are mostly improved from the original image and outperforms both baselines, which demonstrates the generalizability of our learned residual patterns. Note that, in the DL-based setting, we observe that SSIM is not always improved, and one possible reason is that our reconstruction network is not optimized for SSIM, given that the SSIM score is already high in this case. Qualitative results are shown in Figure 4 and our method is able to recover details of the heavily compressed region.

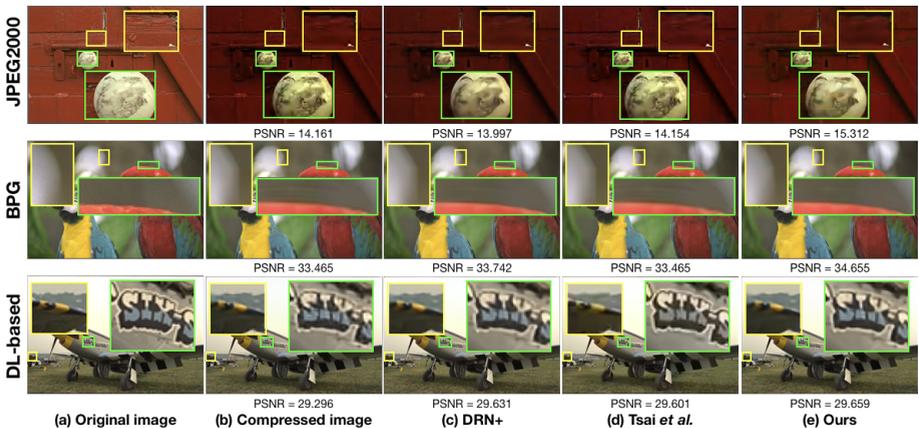


Figure 4: Example results on the Kodak dataset. We show that our method produces more clear reconstruction compared to the DRN+ method and [15], especially on textured regions.

## 5 Conclusion

In this paper, we present a deep learning-based method to reduce the influence of compression artifacts. We perform clustering on the patch-wise latent residual to find residual patterns, such that only patch indexes are required for transmission and corresponding residual information can be retrieved on the client side. As a result, our method simultaneously reduces the cost of transmitting residual information and boosts video or image quality. We conduct extensive experiments and show that our method outperforms baseline models consistently under several coding standards, not only for video compression (e.g., H.264, HEVC, and VP9) but also for the image case (e.g., JPEG2000, BPG, and CNN-based model).

**Acknowledgement.** This project is supported by the Ministry of Science and Technology of Taiwan under grant MOST-109-2634-F-009-015, MOST-109-2634-F-009-020, and MOST-109-2636-E-009-018, and we are grateful to the National Center for High-performance Computing of Taiwan for computer time and facilities.

## References

- [1] Tinku Acharya and Ping-Sing Tsai. *JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures*. 2004.
- [2] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. In *International Conference on Learning Representations (ICLR)*, 2017.
- [3] Fabrice Bellard. BPG Image Format. <http://bellard.org/bpg/>, 2014.
- [4] Li Cheng and SVN Vishwanathan. Learning to compress images and videos. In *International Conference on Machine Learning (ICML)*, 2007.
- [5] Chao Dong, Yubin Deng, Chen Change Loy, and Xiaoou Tang. Compression artifacts reduction by a deep convolutional network. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.
- [6] Rich Franzen. Kodak PhotoCD dataset. <http://r0k.us/graphics/kodak/>, 2007.
- [7] Leonardo Galteri, Lorenzo Seidenari, Marco Bertini, and Alberto Del Bimbo. Deep generative adversarial compression artifact removal. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.
- [8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.
- [10] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 2006.

- [11] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 1952.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] Weiping Li. Overview of fine granularity scalability in MPEG-4 video standard. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 2001.
- [16] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Zhiyong Gao, and Ming-Ting Sun. Deep kalman filtering network for video compression artifact reduction. In *European Conference on Computer Vision (ECCV)*, 2018.
- [17] Danial Maleki, Soheila Nadalian, Mohammad Mahdi Derakhshani, and Mohammad Amin Sadeghi. Blockcnn: A deep network for artifact removal and image compression. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018.
- [18] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *Conference of the ACM Special Interest Group on Data Communication (SIGCOMM)*, 2017.
- [19] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [20] Oren Rippel and Lubomir Bourdev. Real-time adaptive image compression. In *International Conference on Machine Learning (ICML)*, 2017.
- [21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 2015.
- [22] Karl Skretting and Kjersti Engan. Image compression using learned dictionaries by RLS-DLA and compared with K-SVD. 2011.
- [23] 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.
- [24] Wen Tao, Feng Jiang, Shengping Zhang, Jie Ren, Wuzhen Shi, Wangmeng Zuo, Xun Guo, and Debin Zhao. An end-to-end compression framework based on convolutional neural networks. In *Data Compression Conference (DCC)*, 2017.

- [25] 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.
- [26] Gregory K. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 1991.
- [27] 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.
- [28] 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.
- [29] Chao-Yuan Wu, Nayan Singhal, and Philipp Krähenbühl. Video compression through image interpolation. In *European Conference on Computer Vision (ECCV)*, 2018.
- [30] Jaeyoung Yoo, Sang-ho Lee, and Nojun Kwak. Image restoration by estimating frequency distribution of local patches. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [31] 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.