Boosting Image and Video Compression via Learning Latent Residual Patterns *Supplementary Materials*

Yen-Chung Chen*[1]
yenc.cs06g@nctu.edu.tw
Keng-Jui Chang*[1]
adplz53.cs06g@nctu.edu.tw
Yi-Hsuan Tsai[2]
ytsai@nec-labs.com
Wei-Chen Chiu[1]
walon@cs.nctu.edu.tw

[1] National Chiao Tung University
[2] NEC Laboratories America

1 Additional Results

In this supplementary material, we present more qualitative results of our method for improving the visual quality of compressed video and image. Additional example results for improving compressed videos are shown in Figure 1. Please also check the short video clip provided in the supplementary material to see our results, where we directly stack the frames sampled from the original and resultant videos to form it. The left part of the video clip is the compressed frames, and the right part is the reconstructed results produced by our proposed method. In terms of image compression, we also provide more example results on ILSVRC, as shown in Figure 2 for **JPEG2000**, Figure 3 for **BPG** and Figure 4 for **DL-based compression** [2] respectively.

2 Implementation Details

All our models in the following experiments are trained from scratch for 60 epochs using the Adam [1] optimizer. The initial learning rate is set to 10^{-4} and divided by 2 for every 6 epochs. Our implementation is based on PyTorch [3] and runs on a single Tesla V100 with 16 GB memory. In addition, during the stage of residual pattern discovery, due to memory limitation, it is not feasible to perform clustering on all the patch-wise residual patterns collected from the entire training set. Therefore, we randomly sample patches from every video frame to extract feature vectors for patch-wise residuals. The code and model will be released to the public.

The symbol ∗ indicates equal contribution.

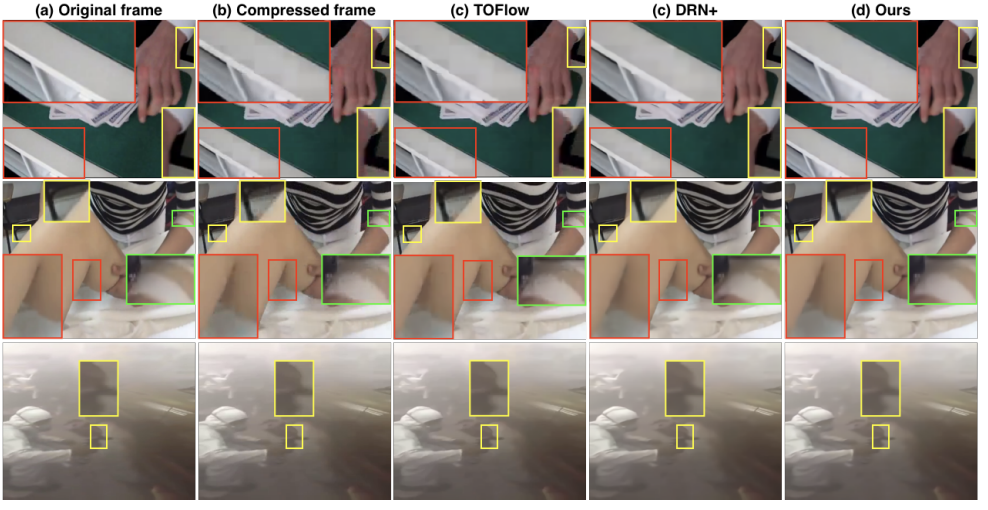

Figure 1: Example results for improving visual quality of compressed videos. Take the Tsai *et al*. baseline as an example, in the first row of column (d), we can observe some artifacts in the region annotated in yellow. Compression method: H.264; Bit-rate: 1Mbps.

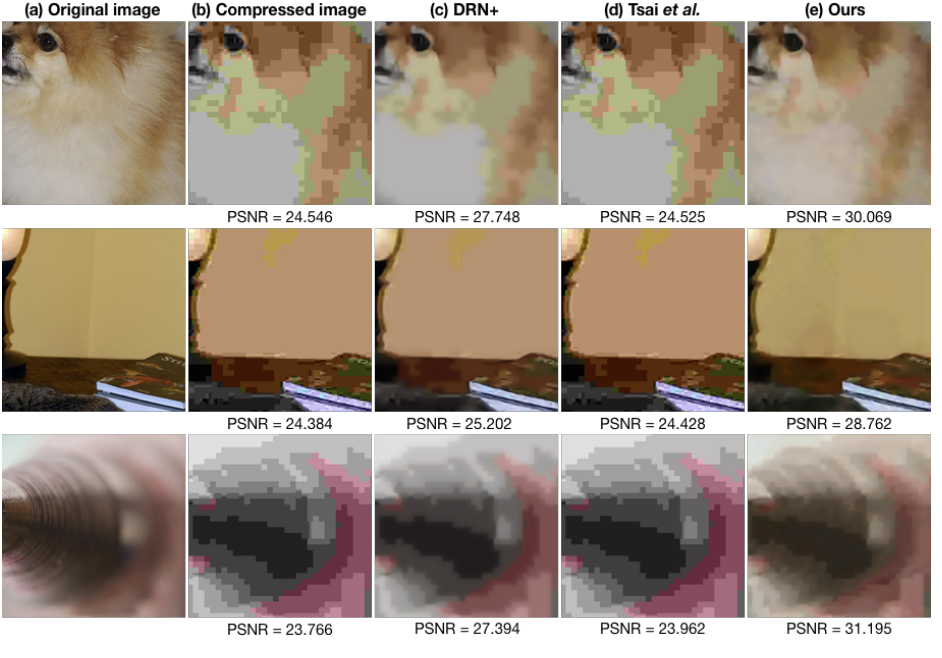

Figure 2: Example results for improving visual quality of compressed images. Compression method: JPEG2000; Bpp: 0.15

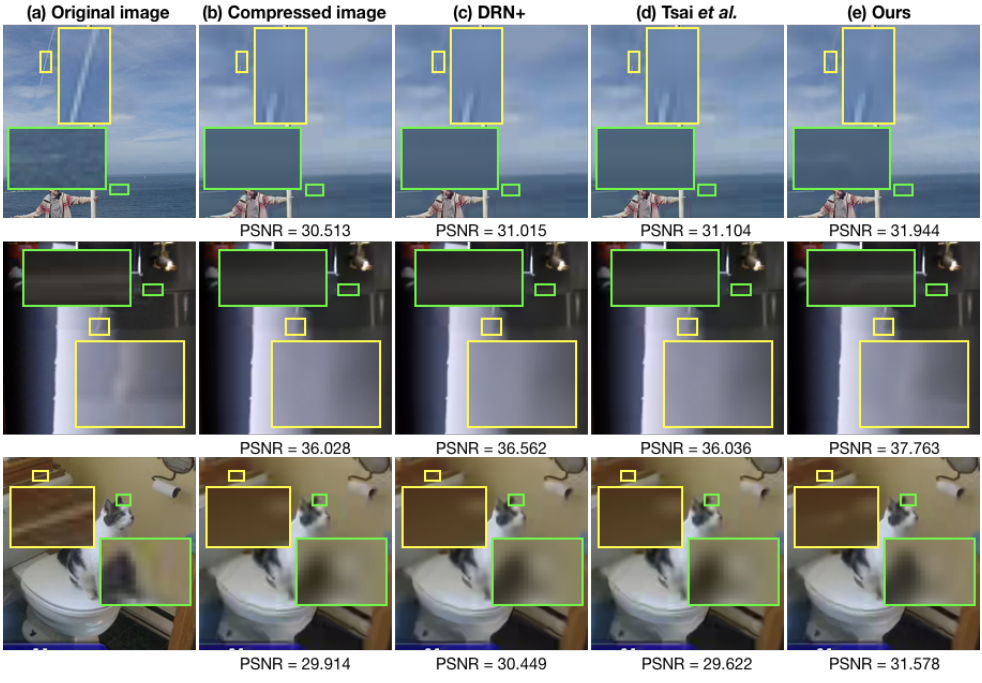

Figure 3: Example results for improving visual quality of compressed images. Compression method: BPG; Bpp: 0.15

layer	channel	kernel	stride
conv1_1	32	3×3	1
conv1_2	32	3×3	2
conv2_1	64	3×3	1
conv2_2	64	3×3	2
conv3_1	128	3×3	1
conv3_2	128	3×3	2
conv4_1	256	3×3	1
conv4_2	256	3×3	2
deconv1_1	256	3×3	1
deconv1_2	128	3×3	1
upsample1		2×2	2
deconv2_1	128	3×3	1
deconv2_2	64	3×3	1
upsample2		2×2	2
deconv3_1	64	3×3	1
deconv3_2	32	3×3	1
upsample3		2×2	2
deconv4_1	32	3×3	1
deconv4_2	3	3×3	1

Table 1: Architecture of feature extraction network.

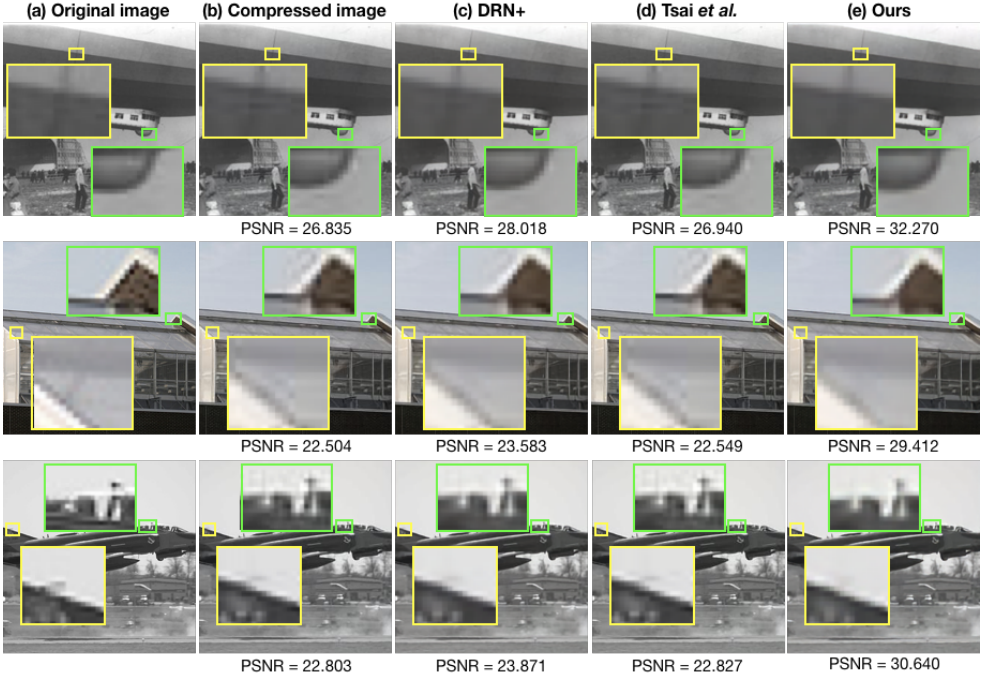

Figure 4: Example results for improving visual quality of compressed images. Compression method: Mentzer *et al*. [7]; Setting: officially-released model weights which is for high quality compressions.

layer	channel	kernel	stride
conv1	64	3×3	1
conv2	64	3×3	1
conv3	64	3×3	1
conv4	64	3×3	1
conv5	64	3×3	1
conv6	64	3×3	1
conv7	64	3×3	1
conv8	3	3×3	1

Table 2: Architecture of refinement network F.

3 Network Architecture

Here we provide the details for the network architecture of each component followed by the inference time of our proposed framework.

1) Feature extraction on residual. As shown in Table 1, the encoder and the decoder for the feature extraction on residual are composed of 4 blocks of convolution and deconvolution layers respectively.

2) Upsampling network. The upsampling network U is simply responsible for mapping the reconstructed feature map of residual P_c into $U(P_c)$ of the same size as the compressed image. The architecture of U follows ESPCN [4].

3) Refinement network. The refinement network F is an 8-layer convolutional neural network (as shown in Table 2), where the compressed frame I_c and the residual information $U(P_c)$ are fed into F through the first and fifth layer respectively. The architecture of our refinement network F is almost identical to the baseline model DRN+.

The inference time of our model on the client side is 323 (frames/sec) under Ubuntu 18.04.2 LTS with a single GTX 1080TI GPU, which is sufficient for the video streaming usage.

We will release the source code, trained models, and all the implementation details for reproducibility and clarification.

4 Experimental Settings

Following Tsai *et al*. [5], we focus on video streaming setting for the experiments related to video compression in our main manuscript. All videos are compressed under constrained variable bit-rate mode, in which the maximally-allowed bit-rate is specified and the bit-rate will be allocated according to the complexity of each video frame. In other words, the bitrate we specified will not be necessarily equal to the average bitrate along the entire compression process. For instance, when we set the maximum bitrate to 5Mbps, the resulting average bitrates for H.264 and HEVC are 3.41Mbps and 3.06Mbps respectively. For the experiments on image compression, We empirically set the number of residual pattern group K to be 1024. The compressed images of **JPEG2000** and **BPG** are yielded under constrained bpp (bits per pixel) 0.15 while DL-based compression approach are obtained by its officially-released model weights which is for high, medium, and low quality compressions.

5 Ablation Study

Here we conduct a study on the KITTI dataset to experiment the sensitivity of our model performance in boosting video quality of compressed video with respect to the the number of residual patterns group K. For this study, the bandwidth of H.264 is set to 5Mbps for generating our training and testing videos. We train our model under different settings of K, which is related to the number of patterns used in our model for approximating residual feature vectors. In Table 3, as K increases, we observe that the performance is gradually improved due to more residual patterns are used in the model. However, we empirically find that the performance saturates when K goes up to 4096, and thus we use $K = 2048$ in experiments on KITTI dataset, and $K = 1024$ in the experiments on all the other datasets with taking the trade-off between efficiency and accuracy into consideration.

KITTI	Coding Standard	H.264
PSNR	$K = 512$	29.453
	$K = 1024$	29.478
	$K = 2048$	**29.679**
	$K = 4096$	29.621
SSIM	$K = 512$	0.865
	$K = 1024$	0.864
	$K = 2048$	**0.869**
	$K = 4096$	0.867

Table 3: Ablation study on numbers of residual patterns K.

References

[1] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ArXiv:1412.6980*, 2014.

[2] 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.

[3] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems (NIPS) Workshops*, 2017.

[4] 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.

[5] 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.