Learning to Hide Residual for Boosting Image Compression
Supplementary Materials

Yi-Lun Lee[1]
vongola850704@gmail.com
Yen-Chung Chen[1]
yenc.cs06g@nctu.edu.tw
Min-Yuan Tseng[1]
piews482zt@gmail.com
Yi-Hsuan Tsai[2]
wasidennis@gmail.com
Wei-Chen Chiu[1]
walon@cs.nctu.edu.tw

[1] National Yang Ming Chiao Tung University
[2] Phiar Technologies, Inc.

1 Training Details

We describe the training process of Strategy I in Algorithm 1. In each training iteration t, we sample a pair of uncompressed image I^H and its compressed version I^L from the training set. After adopting the pretrained weights θ_{sim} of the simulation network and randomly initialize the weights θ_e and θ_d for the encryption encoder E and the decryption decoder D, we then start to update these parameters using our objective function. Specifically, we first use the loss $\mathcal{L}_{sim}$ (Eq.(2) in the main paper) to update the parameters in the simulation network, namely:

$$\theta_{sim}^{t+1} = \theta_{sim}^{t} - \alpha \bigtriangledown_{\theta_{sim}^{t}} \mathcal{L}_{sim}, \tag{1}$$

where α is the learning rate. We then use the overall loss function $\mathcal{L}_{total}$ (Eq.(6) in the mian paper) to update the parameters in the encryption encoder E and the decryption decoder D, namely:

$$\left(\theta_e^{t+1}, \theta_d^{t+1}\right) = \left(\theta_e^{t}, \theta_d^{t}\right) - \alpha \bigtriangledown_{\theta_e^t, \theta_d^t} \mathcal{L}_{total}. \tag{2}$$

As for the training process of Strategy II and the one for deep-learning basedcodec, we do not need the parameters of the simulation network θ_{sim} and thus skip the update in Eq. (1).

Algorithm 1: Training process of our Strategy I.

for *Each Iteration* **do**
  Sample an image pair (I^H, I^L)
  //Update simulation network
  Obtain θ_{sim}^{t+1} via (1)
  //Update encryption encoder and decryption decoder
  Obtain θ_e^{t+1} and θ_d^{t+1} via (2)

2 Additional Results

2.1 Effectiveness of Hidden Residual

To show the effectiveness of hidden residual in the proposed method, we investigate the performance gain from the hidden residual encoded in I_{enc}^L. In Table 1 and 2, we show the average PSNR of the original compressed image I^L, the encoded compressed image I_{enc}^L, and the final outputs I'^H of our method (with respect to the original input image I^H), on both Kinetics and Kodak datasets and with different coding standards. Since the final output of our method I'^H is equal to the summation of encoded compressed image I_{enc}^L and the decoded hidden information I'^R, we can tell from the PSNR gap between I_{enc}^L and I'^H that most of the contribution in boosting image compression is attributed to the hidden information I'^R.

2.2 Ablation Study

We conduct an ablation study on the hyper-parameter λ, which helps to balance between loss terms in the overall objective function, $\mathcal{L}_{total} = \lambda \mathcal{L}_{rec} + \mathcal{L}_{con}$. We adopt the JPEG coding and set the quality value to 50. As shown in Fig. 1, we observe that increasing the weight λ would increase both the bpp of encoded compressed images I_{enc}^L and the PSNR of the final result I'^H. It is not surprising as a higher λ would encourage the Encoder E to hide more information for minimizing the distance between I'^H and I^H, while leading to the increased bpp. In addition, we observe that the trend of fixed-bpp setting is different, which is already saturated when λ is larger than 4. This is because the bpp of encoded compressed image I_{enc}^L

Table 1: Evaluation on the intermediate and final results, including original compressed image I^L, our encoded compressed image I_{enc}^L, and our final output I'^H, on both Kinetics and Kodak datasets with different traditional codecs.

	Coding	BPG		JPEG		JPEG2000
	Mode	Fix-Q	Fix-BPP	Fix-Q	Fix-BPP	Fix-Q/BPP
	Quality / Bpp	30 / 0.702	- / 1.0	50 / 0.889	- / 1.0	48 / 0.50
Kinetics	I^L	36.404	39.271	32.640	33.826	29.292
	I_{enc}^L	36.555	37.010	31.310	31.760	26.349
	I'^H	39.099	39.989	36.370	36.121	30.945
Kodak	I^L	35.373	37.084	32.059	32.749	28.575
	I_{enc}^L	35.232	35.018	30.230	30.213	25.504
	I'^H	37.917	37.578	34.960	34.243	30.108

Table 2: Evaluation on the intermediate and final results, including original compressed image I^L, our encoded compressed image I^L_{enc}, and our final output I'^H, on both Kinetics and Kodak datasets with different deep-learning-based codecs.

	Coding	CAE-B		CAE-P		RNN	
	Bpp	0.5	2.0	0.8	2.5	0.5	1.0
Kinetics	I^L	26.162	29.835	30.978	34.108	27.990	30.822
	I^L_{enc}	26.611	30.792	31.330	34.731	28.210	31.052
	I'^H	27.258	31.584	31.911	35.875	28.935	31.914
Kodak	I^L	25.494	28.774	29.677	32.797	26.890	29.512
	I^L_{enc}	25.934	29.508	30.046	33.388	27.111	29.708
	I'^H	26.636	30.441	30.584	34.799	27.761	30.337

is fixed, such that the capacity of hidden information is limited.

2.3 Analysis on Low Bit-Rate Compression

In Table 3 and 4, we show quantitative comparisons between our method and several baseline approaches under low bpp setting on Kinetics and Kodak datasets. For fully differentiable deep-learning-based codecs, we show favorable performance over other baselines, which demonstrates that our method is able to recover severe distortion under low bpp setting via end-to-end optimization. On the other hand, the proposed framework performs competitively when applying to traditional codecs under the low bpp setting. One limitation is that the simulation network for traditional codecs is not perfect (i.e., when there exists large discrepancy between I^L_{enc} and I^{SL}_{enc}), which could cause the inaccurate gradients back-propagated through our skip-component. Nevertheless, improving the simulation network for traditional codecs is out of the scope of this work and we will include it in the future work.

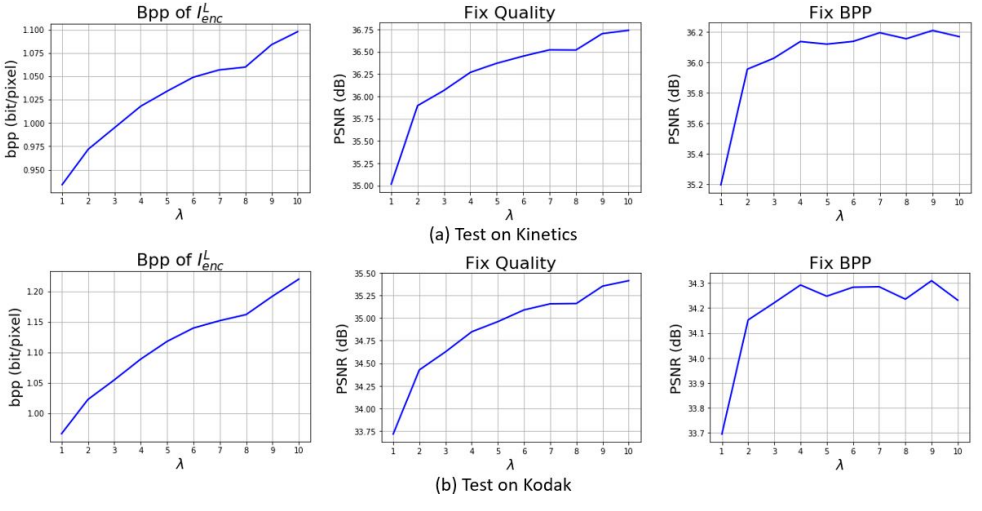

Figure 1: Ablation study on the hyper-parameter λ.

Table 3: Result on the Kinetics dataset with different compression codecs under low bit-rate setting. We compare our proposed model with respect to the baselines from DnDNN [5] and MemNet [3].

Kinetics	Coding Standard	JPEG2000	CAE-B [6]	RNN [4]
	Bpp (bits/pixel)	0.15	0.125	0.125
PSNR	Original	24.643	23.220	22.660
	MemNet [3]	25.279	23.612	23.412
	DnCNN [5]	**25.459**	23.636	23.353
	Ours	25.213	**23.940**	**24.072**
SSIM	Original	0.680	0.680	0.690
	MemNet [3]	0.711	**0.707**	0.719
	DnCNN [5]	0.720	0.704	0.709
	Ours	**0.734**	**0.707**	**0.724**

Table 4: Results on the Kodak dataset with different compression codecs under low bit-rate setting. We compare our proposed model with respect to the baselines from DnDNN [5] and MemNet [3].

Kodak	Coding Standard	JPEG2000	CAE-B [6]	RNN [4]
	Bpp (bits/pixel)	0.15	0.125	0.125
PSNR	Original	25.323	22.936	22.888
	MemNet [3]	25.415	23.088	23.229
	DnCNN [5]	**25.922**	23.322	23.441
	Ours	25.624	**23.674**	**24.078**
SSIM	Original	0.631	0.603	0.612
	MemNet [3]	0.646	0.617	0.633
	DnCNN [5]	0.658	0.618	0.632
	Ours	**0.685**	**0.625**	**0.642**

2.4 Comparison with Jiang *et al.* [1]

Jiang *et al.* [1] propose a compression framework which is compatible with existing image codecs, including JPEG, JPEG2000, and BPG. However, we point out two differences between Jiang *et al.* [1] and our proposed framework: 1) Our main novelty is to hide the residual and our framework is general for both traditional and deep learning-based codecs, which are not explored in Jiang *et al.*; 2) the model from Jiang *et al.* employs iterative training to handle the non-differential issue in traditional codecs, whereas our method allows end-to-end backpropagation. In Table 5, we show the quantitative comparison between our proposed method and the model provided by Jiang *et al.*, where we follow the experimental setting (i.e., JPEG codec, bpp=0.28) in their released code for fair comparisons. We find both models have comparable performance but our method has advantages as described above (i.e.,

Table 5: Quantitative comparison with Jiang *et al.* [1] on the Kodak dataset under JPEG (bpp=0.28).

	PSNR	SSIM
Original	26.308	0.702
Jiang *et al.* [1]	28.945	**0.825**
Ours	**29.102**	0.812

Table 6: Application of our proposed method to Liu *et al.* [2] on the Kodak dataset under different bpps (0.79 and 1.45).

	Bpp=0.79		Bpp=1.45	
	PSNR	SSIM	PSNR	SSIM
Original (Liu *et al.* [2])	34.577	0.927	38.920	0.965
DnCNN [5]	34.816	0.933	39.067	**0.968**
Ours	**34.942**	**0.934**	**39.231**	**0.968**

general for different codecs and end-to-end learnable).

2.5 Application to SOTA deep learning-based codec Liu *et al.* [2]

To verify our efficacy on the recent deep learning-based codecs, we apply our hiding residual framework to one of the state-of-the-art learning-based codecs, Liu *et al.* [2], as shown in Table 6. The results show that our proposed method still can bring improvement to the SOTA compression codec and outperform the baseline (i.e. DnCNN) under different settings of bpp.

2.6 Bits-per-pixel versus PSNR

We provide the rate-distortion curves (bits-per-pixel versus PSNR) based on different codecs, such as JEPG, JPEG2000, BPG, and RNN, in Fig. 2 for our proposed method and DnCNN (the most competitive baseline). We find that for the traditional codecs (JPEG, JPEG2000, and BPG), in terms of PSNR, our method has comparative performance w.r.t. DnCNN at low bpp and achieves better PSNR when the bpp increases. On the other hand, for the deep-learning-based method (i.e. RNN here), our method outperforms DNCNN by a margin no matter the bpp is low or high. To be more detailed, we find that there are two different phenomenons according to the type of codecs (traditional or deep-learning-based), as shown in Fig. 2. First, in the case of traditional codecs, as the bpp grows, the performance gain is increasing. However, when the bpp is very low, our model downgrades to DnCNN and has a similar performance. That is because both our strategy I and II do not perfectly remove the influence of non-differentiable compression functions under the limitation of low bpp. It needs more effort to encode the residual. Therefore, when the bpp is set to a very low number, these learnt encoded messages are still destroyed by the compression and consequently downgrade to the post-processing image restoration model (i.e. DnCNN). On the other hand, in the case of learnt codecs (e.g. RNN), the compression procedure can be differentiated so the gradient can flow back perfectly, encouraging the encoder to hide the residual effectively

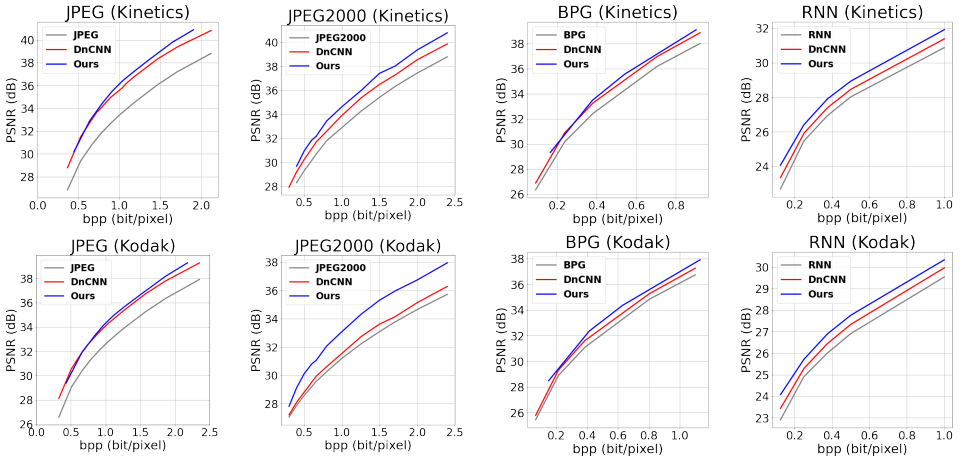

Figure 2: Rate-distortion curves of ours, DnCNN, and codecs (JPEG, JPEG2000, BPG and RNN) on both Kinetics and Kodak datasets.

and correctly. Hence, we can see that the performance gain is stable even if the bpp decreases to a very low value.

2.7 Qualitative Results

In this section, we provide some observations on both the original residual I^R and the decoded hidden information I'^R, where the latter is decrypted from the compressed encoded image I^L_{enc}. Moreover, we present more qualitative results of our method for improving the visual quality of compressed images.

We visualize the decrypted hidden information I'^R, its corresponding original residual I^R, and the difference between its corresponding I^H and I^L_{enc} in Fig. 3. Since the compressed image I^L and the encoded compressed image I^L_{enc} are not exactly the same, we can observe that the difference between I^H and I^L_{enc} is quite similar to the original residual I^R yet still different in some details. Moreover, our decrypted hidden information I'^R has the same contour as the original residual I^R, but has similar details w.r.t. the difference between I^H and I^L_{enc}. As a result, we do not constraint the Decoder D to learn the reconstruction of residual directly (i.e. forcing $I'R$ to be consistent with I^R), but only utilize the concept of residual learning to adaptively produce the residual for I^L_{enc}.

We provide more example results on Kinetics, as shown in Figure 4 with **JPEG** coding, Figure 5 with **JPEG-2000** coding, and Figure 6 with several traditional and deep-learning-based codings respectively.

3 Network Architecture

Here we provide the details for the network architecture of each component used in our proposed method.

1) Encoder E. As shown in Table 7, the encoder E for the encryption process is composed of 6 convolution layers without down-sampling on the image size, where the uncom-

pressed/original image I^H and its residual I^R are concatenated and then fed into E.

2) Decoder D**.** The decoder D for the decryption process is an 8-layer convolutional neural network (as shown in Table 8) which attempts to decrypt the hidden information from I^L_{enc}. The architecture of our decoder is identical to the baseline model DnCNN [5].

3) Simulation Network. The simulation network follows an autoencoder architecture (as shown in Table 9) which learns to mimic the compression function by reproducing the outputs of traditional codec.

Table 7: Architecture of Encoder E.

layer	channel	kernel	stride	activation
conv1_1	64	3×3	1	ReLU
conv1_2	64	3×3	1	ReLU
conv1_3	64	3×3	1	ReLU
conv1_4	64	3×3	1	ReLU
conv2_1	64	3×3	1	ReLU

Table 8: Architecture of Decoder D.

layer	channel	kernel	stride	activation
conv1_1	64	3×3	1	ReLU
conv1_2	64	3×3	1	ReLU
conv1_3	64	3×3	1	ReLU
conv1_4	64	3×3	1	ReLU
conv1_5	64	3×3	1	ReLU
conv1_6	64	3×3	1	ReLU
conv2_1	64	3×3	1	ReLU
conv2_2	3	3×3	1	

Table 9: Architecture of Simulation Network.

layer	channel	kernel	stride	activation
conv1_1	32	3×3	1	ReLU
conv1_2	64	3×3	1	ReLU
conv1_3	128	3×3	1	ReLU
conv1_4	128	3×3	1	
deconv1_1	128	3×3	1	ReLU
deconv1_2	64	3×3	1	ReLU
deconv1_3	32	3×3	1	ReLU
deconv1_4	3	3×3	1	

References

[1] Feng Jiang, Wen Tao, Shaohui Liu, Jie Ren, Xun Guo, and Debin Zhao. An end-to-end compression framework based on convolutional neural networks. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 2017.

[2] Jiaheng Liu, Guo Lu, Zhihao Hu, and Dong Xu. A unified end-to-end framework for efficient deep image compression. *arXiv preprint arXiv:2002.03370*, 2020.

[3] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[4] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full resolution image compression with recurrent neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

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

[6] Haimeng Zhao. Cae-p: Compressive autoencoder with pruning based on admm. *ArXiv:1901.07196*, 2019.

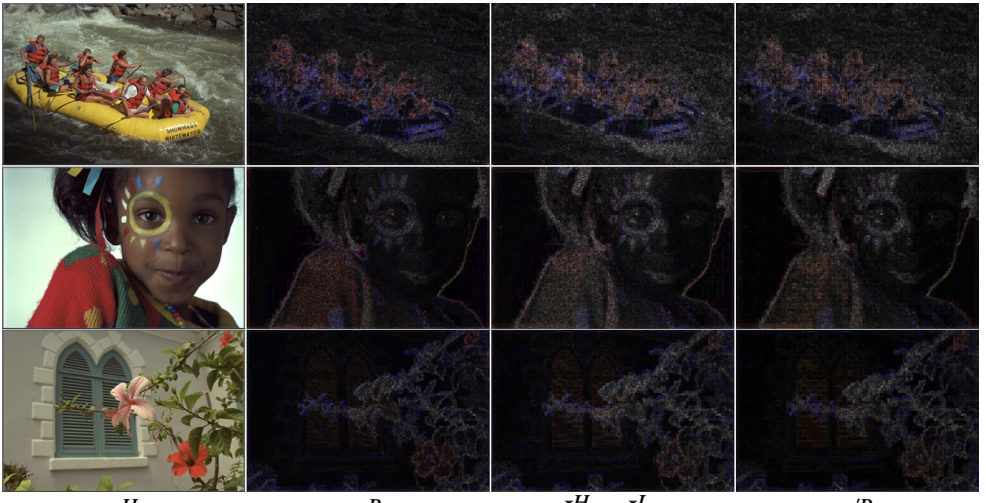

Figure 3: Example results of visualizing the original image I^H, the original residual I^R, the decrypted hidden information I'^R, and the difference between I^H and the encoded compressed image I^L_{enc}. For I^R, $I^H - I^L_{enc}$, and I'^R, we multiply each pixel value by 5 in order to have better visualization.

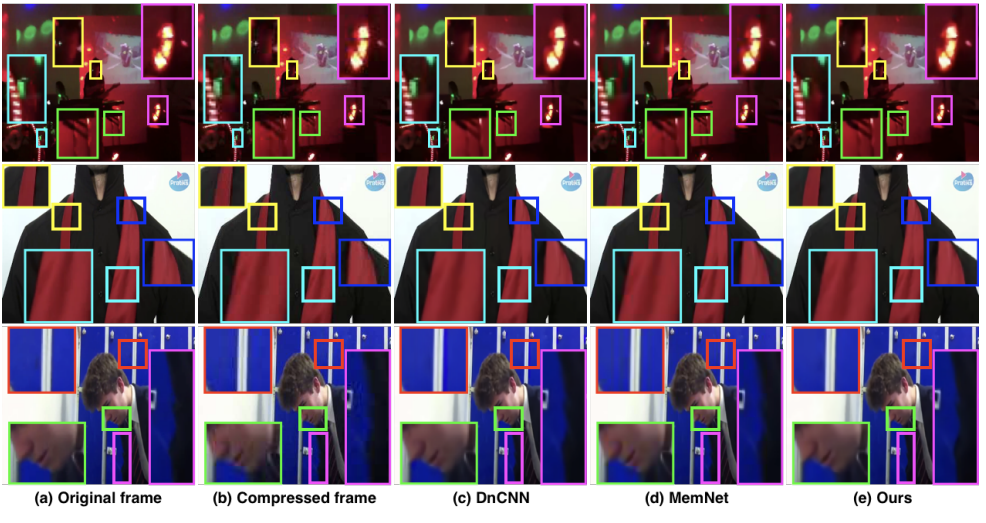

Figure 4: Example results for improving visual quality of compressed images. We show that our method produces clearer reconstruction, in comparison to the baselines from DnCNN and MemNet.

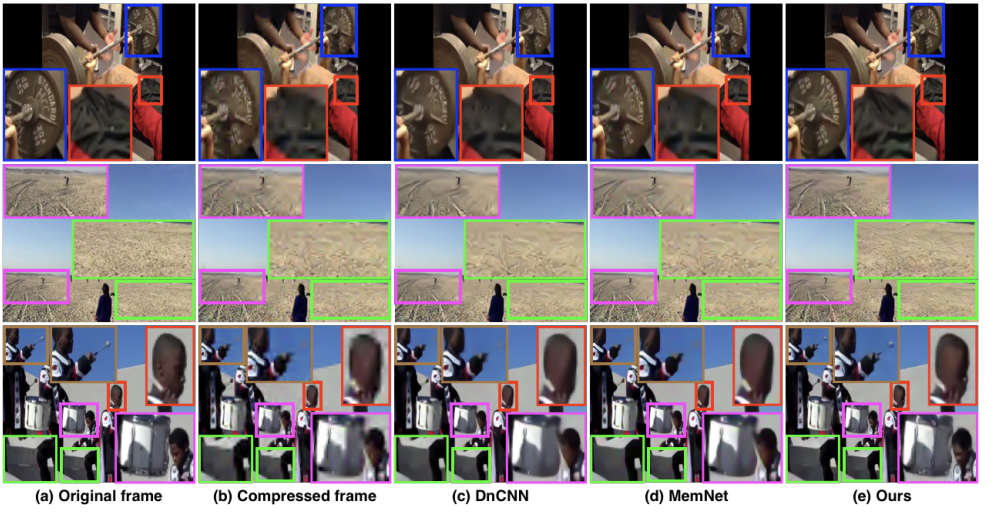

Figure 5: Example results for improving visual quality of compressed images (bpp $= 0.5$ here). We show that under such severe condition of low bpp, DnCNN and MemNet are likely to produce undesirable results. For example, in the third row of column (c) and (d), the man's head in the region annotated in red is oversmoothed.

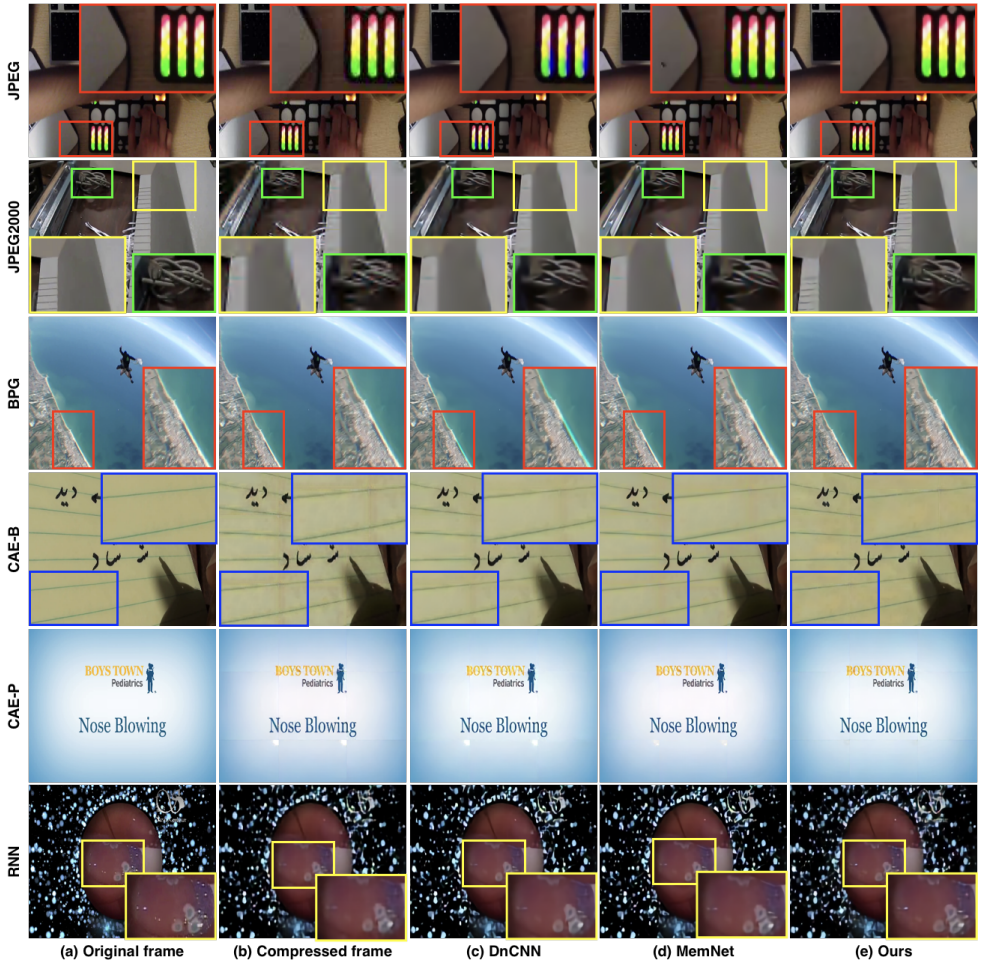

Figure 6: More example results for improving the visual quality of compressed images. We show that our proposed method is more likely to recover the details from compressed images, while other baselines may cause blurs or unnatural color blocks in some regions. For instance, in the first row of column (c) DnCNN, there is a blue artifact on the panel.