Paper ID 168:

Deep Learning-based Image Compression with Trellis Coded Quantization

Binglin Li[1], Mohammad Akbari[1], Jie Liang[1], and Yang Wang[2]

[1]Simon Fraser University

{binglinl, akbari, jiel}@sfu.ca

[2]University of Manitoba

ywang@cs.umanitoba.ca

Introduction

- Optimal image codec:

$$min\ D(x,\hat{x}) + \lambda H(\hat{z}). \quad (\lambda > 0)$$

- Scalar quantizer (SQ) is commonly applied in deep learning based image compression models.
- Trellis Coded Quantizer (TCQ) is an efficient vector quantizer (VQ).

TABLE I
TRELLIS CODED QUANTIZATION PERFORMANCE FOR THE MEMORYLESS UNIFORM SOURCE USING RATE-$(R+1)$ LLOYD–MAX OUTPUT POINTS. (VALUES ARE LISTED AS SNR IN dB)

Rate (bits)	Trellis Size (States) 4	8	16	32	64	128	256	Lloyd-Max Quantizer	Distortion Rate Function
1	5.78	5.96	6.06	6.13	6.19	6.29	6.33	6.02	6.79
2	12.47	12.60	12.69	12.76	12.83	12.90	12.93	12.04	13.21
3	18.77	18.90	18.98	19.04	19.10	19.16	19.20	18.06	19.42
4	24.95	25.05	25.13	25.19	25.24	25.30	25.34	24.08	N/A
8	49.16	49.24	49.32	49.38	49.44	49.49	49.53	48.16	49.69
12	73.24	73.35	73.40	73.48	73.53	73.58	73.61	72.25	73.78

↑0.78dB

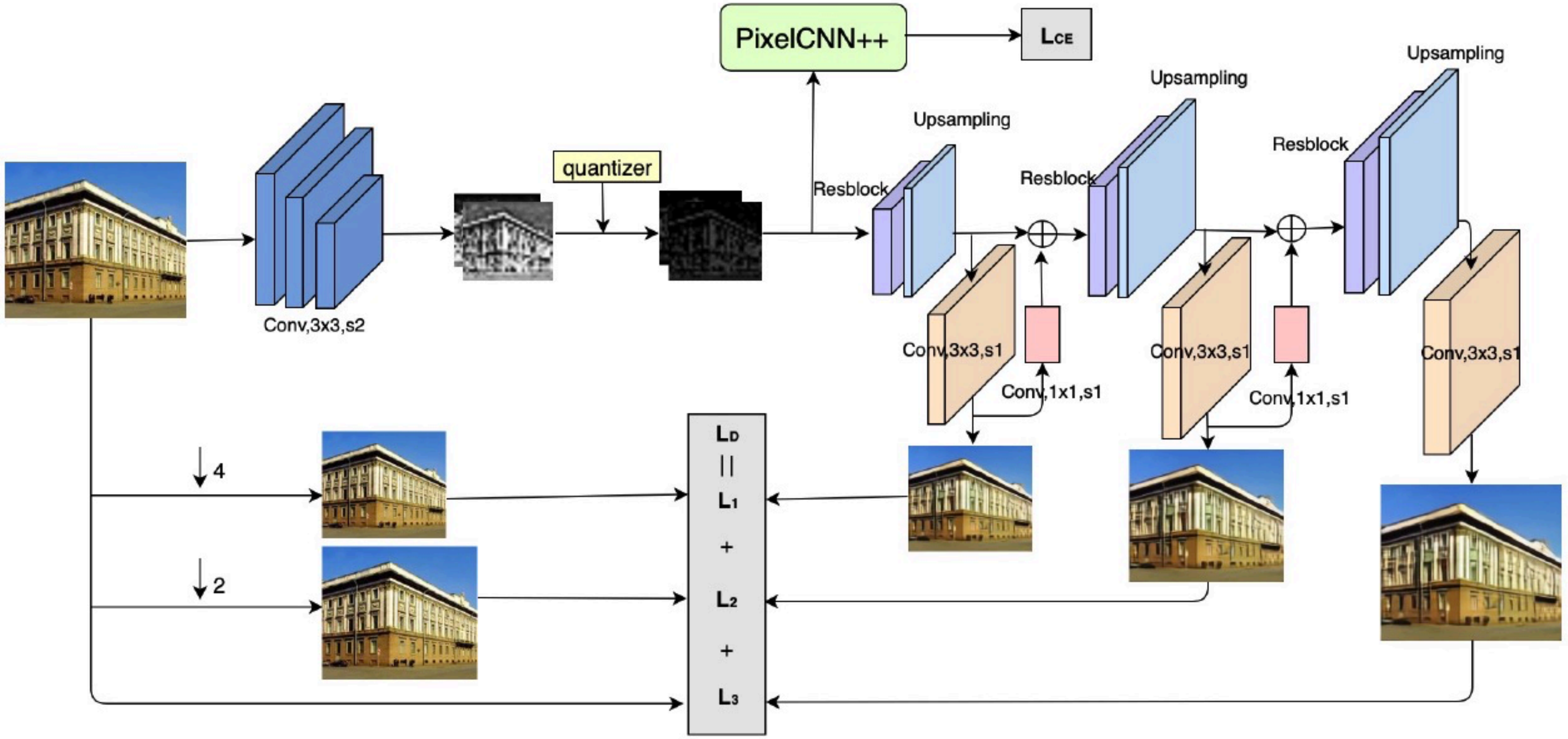
[1] Michael W. Marcellin and Thomas R. Fischer, "Trellis Coded 1uantization of Memoryless and Gauss-Markov Sources," IEEE transactions on Communications, vol. 38, no. 1, pp. 82-93, 1990.

SFU

Contributions

- Incorporate TCQ into a deep learning based image compression framework.
- All components (encoder, TCQ, decoder, entropy estimator) are trained end-to-end.
- Test on two datasets and show superior image compression performance at low bit rates compared with previous works.
- Compare TCQ and SQ with MSE and MS-SSIM loss during training and demonstrate the advantage of TCQ.

Overview



$$L_{\text{MS-SSIM}} = 100(1 - L_D(\text{MS-SSIM}(\tilde{X}, X))) + \lambda L_{CE}$$

Trellis Coded Quantization (TCQ)

- Forward pass: similar to implementation in JPEG2000 [2].
  - $V_{max} = 1$, $V_{min} = -1$, step size: $\Delta = \frac{2}{2^{R+1}}$
  - Reconstruction point

$$c_j \ (j = 1, 2, \cdots, 2^{R+1}) \ = \ -1 + \Delta/2 + (j-1) \times \Delta.$$

  - Partition to form D_0, D_1, D_2, D_3 subsets.
  - Indexing method I : $q b_1 b_2 \cdots b_{R-1}$

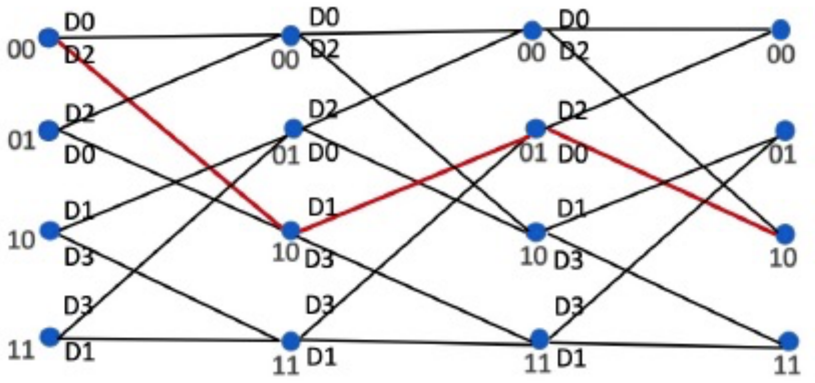


Figure 2: An example of 4 state trellis structure.

[2] "Information technology “ Jpeg 2000 image coding system: Core coding system," Standard, International Organization for Standardization, Dec. 2000.

SFU

Trellis Coded Quantization (TCQ)

- Backward pass:
  - Straight-through estimator (derivative is set to 1): converges slowly for TCQ.

  - Differentiable soft quantization [3]:
    Given reconstruction points $C = \{c_1, c_2, \cdots, c_L\}$ $(L = 2^{R+1})$,

$$\tilde{Q}(z) = \Sigma_{j=1}^{L} \frac{exp(-\sigma||z - c_j||)}{\Sigma_{l=1}^{L} exp(-\sigma||z - c_l||)} c_j$$

where σ is a hyperparameter to adjust "softness" of quantization.

[3] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool, "Soft-to-hard vector quantization for end-to-end learning compressible representations," in Advances in Neural Information Processing Systems, 2017, pp. 1141-1151.

Discussions

- Time and memory complexity are both proportional to number of symbols.

  $\Rightarrow$ reshape feature maps $B\times C\times H\times W$ to $BC\times HW$,

  BC is batch size for TCQ and HW is number of symbols in a feature map.

- Indexing method II [4] vs. Indexing method I
  - $A0 = D_0 \cup D_2,\ A1 = D_1 \cup D_3$
  - a node codeword can be chosen either from A0 or A1
  - Use all R bits to represent indices for A0, same for A1

Figure 3: (a) indexing method I for TCQ, (b) indexing method II for TCQ, (c) SQ.

[4] Michael W Marcellin, “On Entropy-constrained Trellis Coded Quantization," IEEE Transactions on Communications, vol. 42, no. 1, pp. 14-16, 1994.

Entropy Coding Model

- Employ PixelCNN++ [5] to model the probability density function on an image x $p(x) = \prod_i p(x_i|x_{<i})$, where the conditional probability only depends on pixels above and to the left of the pixel.

- Encoding:
  - Assume R bits/symbol, for feature map $F(C\times H\times W)$, PixelCNN++ outputs $2^R\times C\times H\times W$ probability matrix.
  - Adaptive Arithmetic Coding (AAC) is used to compress F.

[5] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma, “PixelCNN++: Improving the Pixelcnn with Discretized Logistic Mixture Likelihood and Other Modifications," arXiv preprint arXiv:1701.05517, 2017.

Entropy Coding Model

- Decoding:
  - Input PixelCNN++ model with all zeros, decode indices and recover symbols at position $(c = 1{:}C, i = 1, j = 1)$.
  - Continue decoding based on output probability below:

$$p(z_{c=1:C,i=u,j=v} \mid Context) = \mathrm{PixelCNN{+}{+}}(T_{\{1:C,1:u,1:(v-1)\}\bigcup\{1:C,1:(u-1),v:W\}})_{c,i,j}$$

where $T_{1:x,1:y,1:z}$ is a tensor with decoded symbol at location $\{(c, i, j) \mid 1 \leq c \leq x, 1 \leq i \leq y, 1 \leq j \leq z\}$ and zeros otherwise.

When $\mu = 1, \{1{:}C, 1{:}(u-1), v{:}W\} = \emptyset$

When v $= 1, \{1{:}C, 1{:}u, v{:}(v-1)\} = \emptyset$

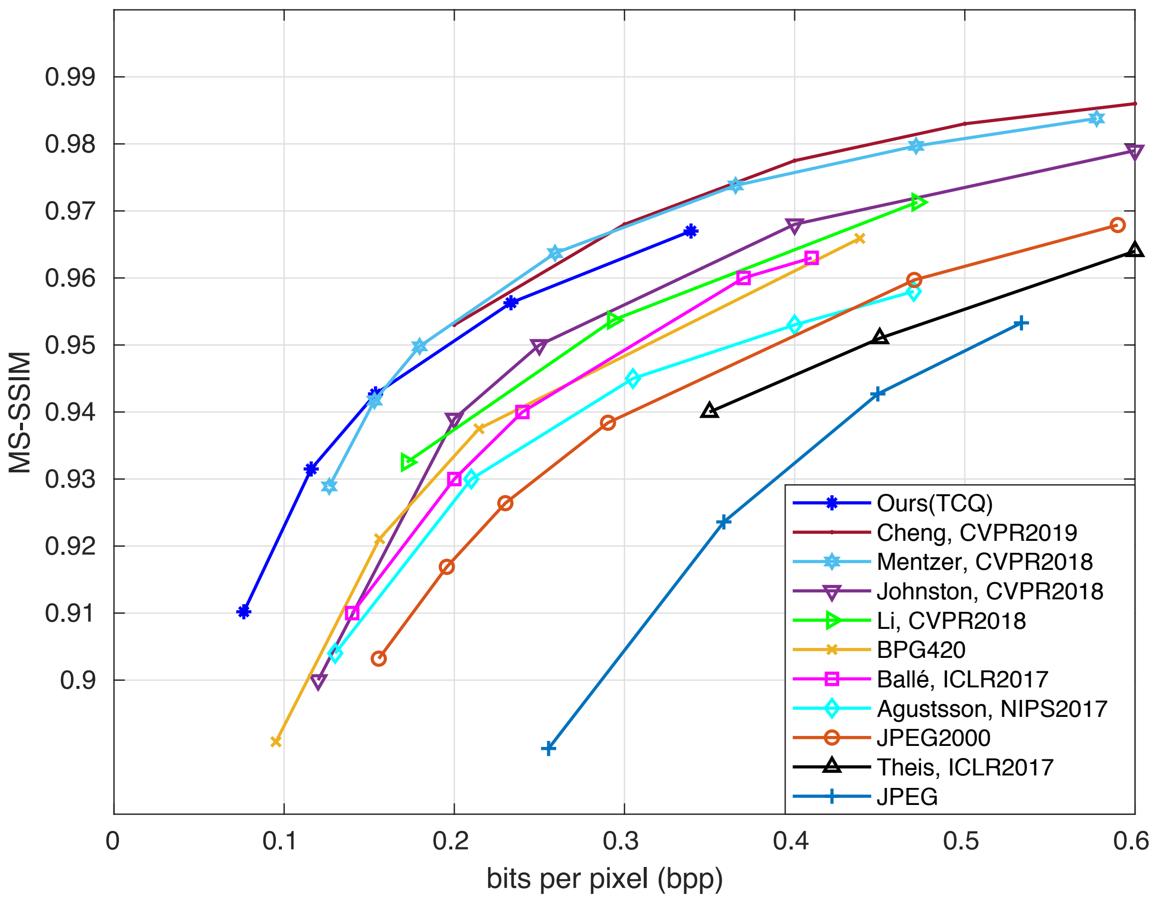
Experimental Results

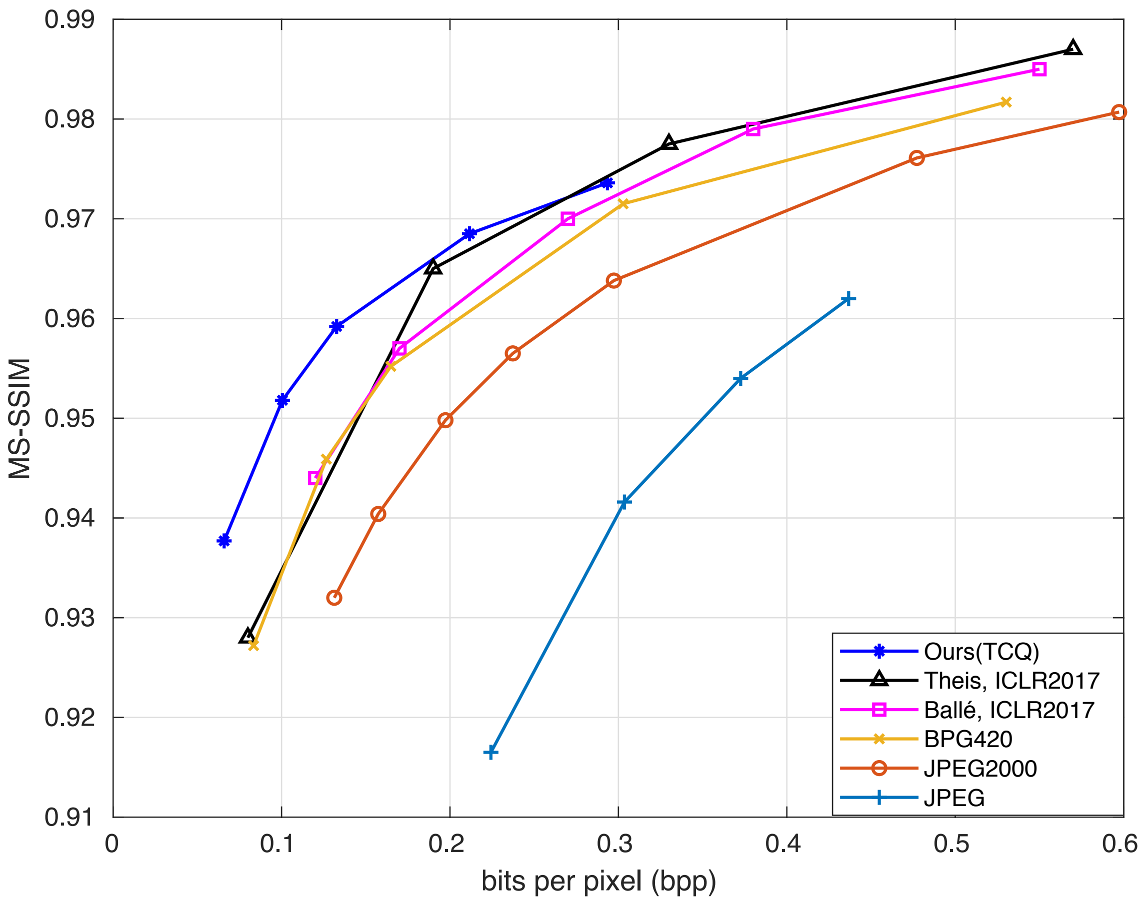
- Dataset
  - ADE20K dataset for training (20K) and validation (2K).
  - Test on Kodak PhotoCD dataset (24 512×758 images) and Tecnick SAMPLING dataset (100 1200×1200 images).

- Training details:
  - Crop images by 256x256 during training and test on whole images.
  - Run on one 12G GTX TITAN GPU with Adam optimizer
  - R=2, 4-state TCQ, R=2 SQ. Both use differentiable soft quant for backward.
  - Increase channel size with $\{4,6,8,12,16\}$ to control bitrate.

Experimental Results

- Comparison with previous works





Experimental Results

- Comparison between TCQ and SQ (JPEG2000)

Table 3.6. TCQ performance for IID uniform data (SNR in dB).

Rate (bits)	*Trellis Size (States)* 4	8	16	32	64	128	256	*Lloyd-Max Quantizer*	*Distortion-Rate Function*
1	6.22	6.33	6.39	6.44	6.48	6.55	6.58	6.02	6.79
2	$\boxed{12.62}$	12.73	12.80	12.85	12.91	12.97	13.00	$\boxed{12.04}$	13.21
3	18.83	18.94	19.01	19.08	19.13	19.18	19.23	18.06	19.42

↑0.58 dB

Table 3.7. TCQ performance for IID Gaussian data (SNR in dB).

Rate (bits)	*Trellis Size (States)* 4	8	16	32	64	128	256	*Lloyd-Max Quantizer*	*Distortion-Rate Function*
1	5.00	5.19	5.27	5.34	5.43	5.52	5.56	4.40	6.02
2	$\boxed{10.56}$	10.70	10.78	10.85	10.94	10.99	11.04	$\boxed{9.30}$	12.04
3	16.19	16.33	16.40	16.47	16.56	16.61	16.64	14.62	18.06

↑1.26 dB

Credit to Taubman, David and Marcellin, Michael, "JPEG2000 Image Compression Fundamentals, Standards and Practice" 2002.

Experimental Results

- Comparison between TCQ and SQ (MS-SSIM loss)

Table 1: Performance comparisons between TCQ and SQ using MS-SSIM loss for training

quantizer	Kodak dataset		Tecnick dataset	
	PSNR(dB)/bpp	MS-SSIM/bpp	PSNR(dB)/bpp	MS-SSIM/bpp
SQ	24.54/0.077	0.9028/0.077	26.14/0.068	0.9326/0.068
TCQ	↑0.41 24.95/0.076	↑0.007 0.9102/0.076	↑0.68 26.82/0.066	↑0.005 0.9377/0.066
SQ	25.66/0.117	0.9259/0.117	27.63/0.104	0.9493/0.104
TCQ	↑0.19 25.85/0.116	↑0.006 0.9315/0.116	↑0.23 27.86/0.101	↑0.003 0.9518/0.101
SQ	26.23/0.157	0.9386/0.157	28.32/0.139	0.9572/0.139
TCQ	↑0.24 26.47/0.154	↑0.004 0.9427/0.154	↑0.13 28.45/0.133	↑0.002 0.9592/0.133

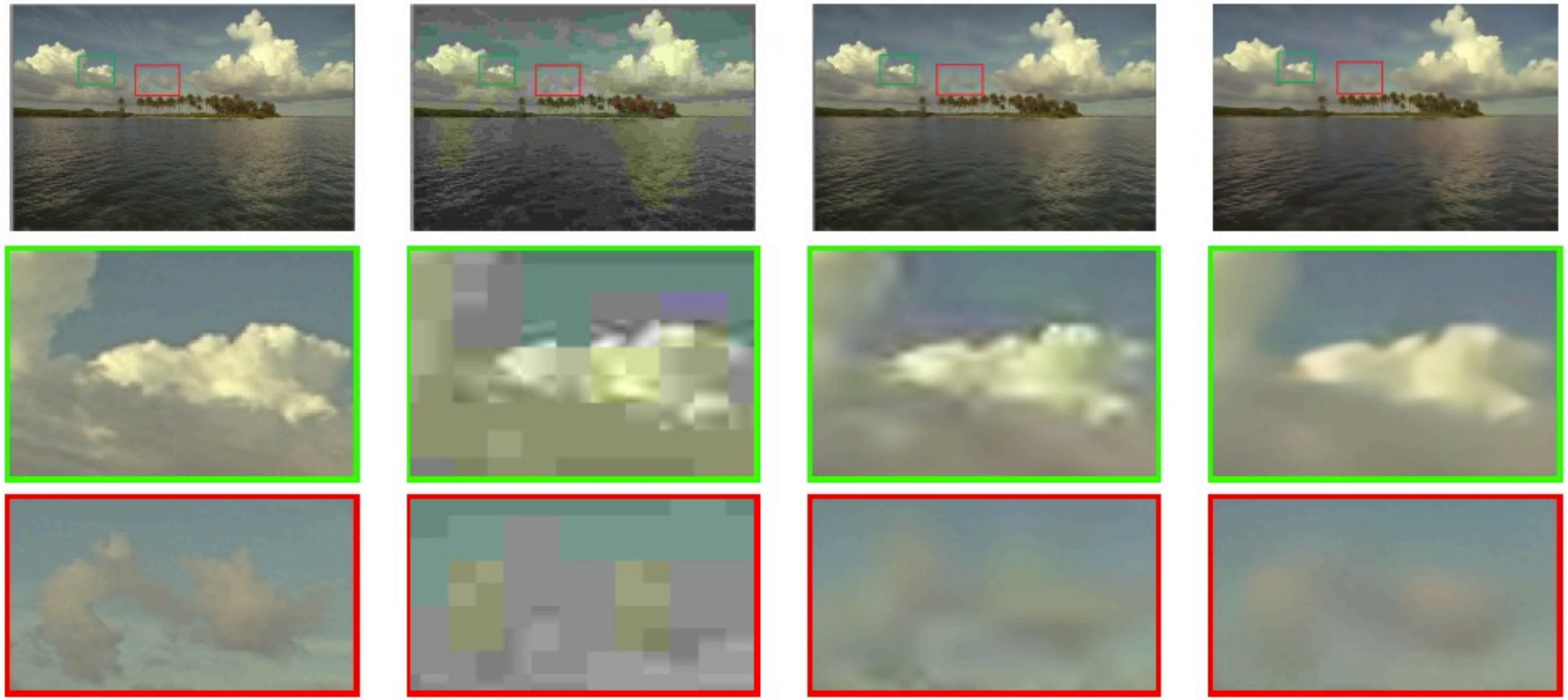
Experimental Results

- Comparison between TCQ and SQ (MSE loss)

Table 2: Performance comparisons between TCQ and SQ using MSE loss for training

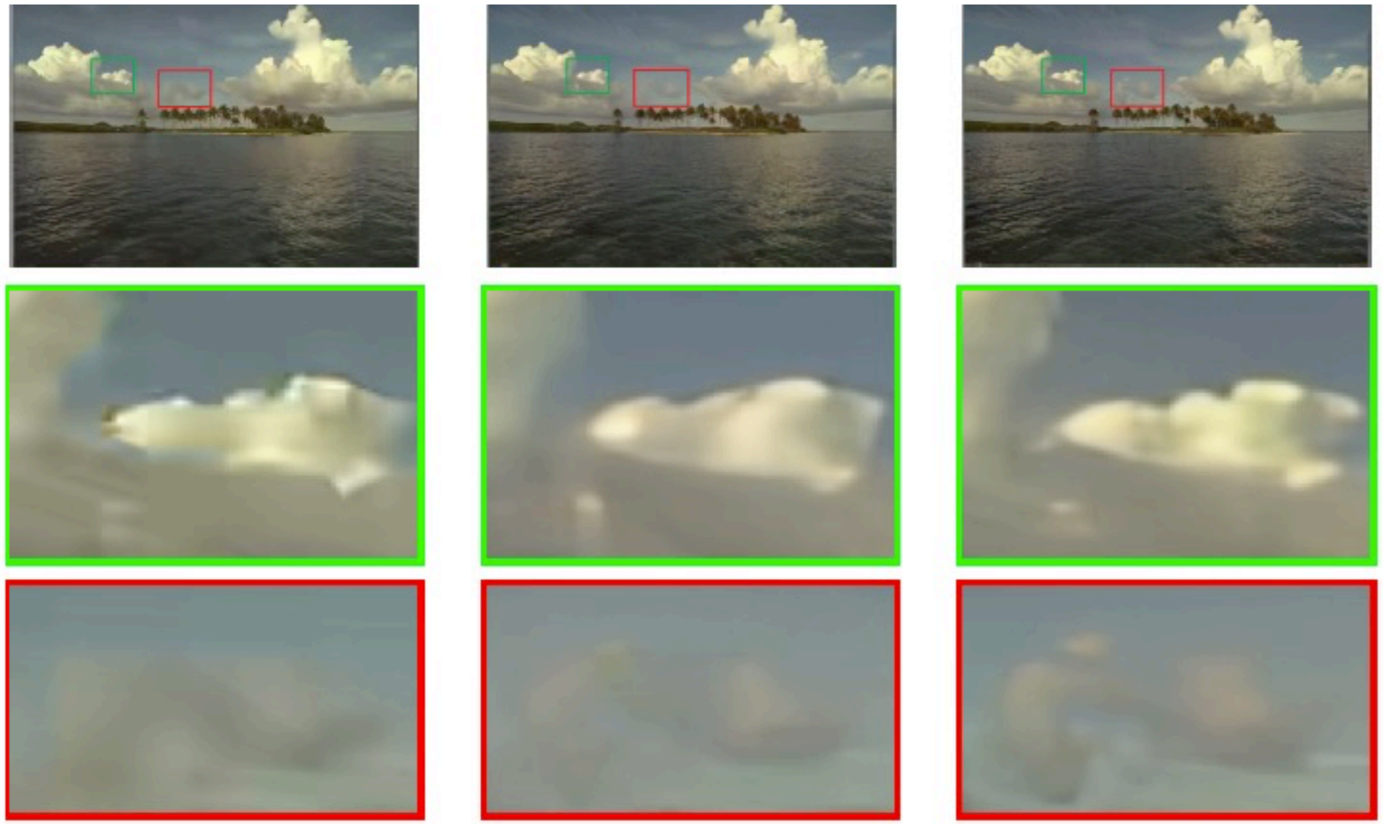
quantizer	Kodak dataset		Tecnick dataset	
	PSNR(dB)/bpp	MS-SSIM/bpp	PSNR(dB)/bpp	MS-SSIM/bpp
SQ	24.86/0.064	0.8715/0.064	25.94/0.054	0.9091/0.054
TCQ	↑0.37 25.23/0.062	↑0.011 0.8824/0.062	↑0.72 26.66/0.052	↑0.010 0.9189/0.052
SQ	25.81/0.098	0.8992/0.098	27.35/0.081	0.9308/0.081
TCQ	↑0.54 26.35/0.096	↑0.010 0.9090/0.096	↑0.61 27.96/0.078	↑0.007 0.9373/0.078
SQ	26.56/0.133	0.9178/0.133	28.18/0.112	0.9427/0.112
TCQ	↑0.33 26.89/0.130	↑0.005 0.9232/0.130	↑0.47 28.65/0.110	↑0.005 0.9473/0.110

Experimental Results



	0.8177/0.128	0.9028/0.118	0.9139/0.117
(a) Original	(b) JPEG	(c) JPEG2000	(d) Ballé [1]

Experimental Results



0.9189/0.112
(e) BPG

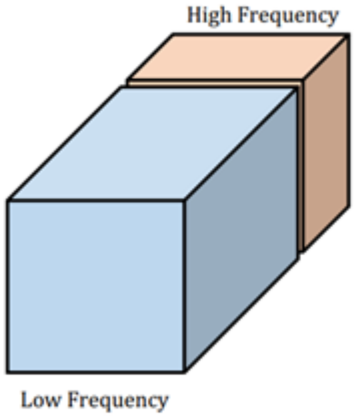
0.9257/0.112
(f) Ours (SQ)

0.9323/0.109
(g) Ours (TCQ)

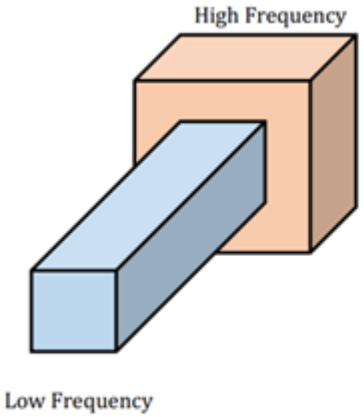
Our Latest Work

- Mohammad Akbari, Jie Liang, Jingning Han, Chengjie Tu, "Generalized Octave Convolutions for Learned Multi-Frequency Image Compression," arXiv:2002.10032, Feb. 2020.

- Based on Chen et al., "Drop an Octave: Reducing Spatial Redundancy in Convolutional Neural Networks with Octave Convolution, " arXiv:1904.05049, Apr. 2019.



Conventional Convolution output:
Same resolution for all frequencies

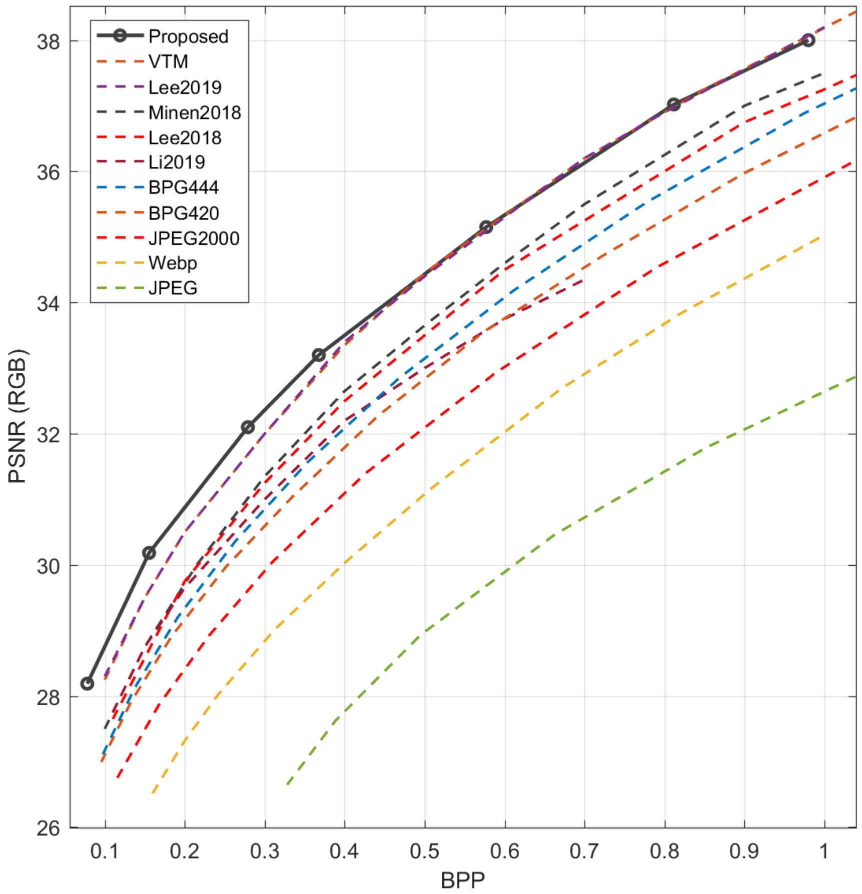


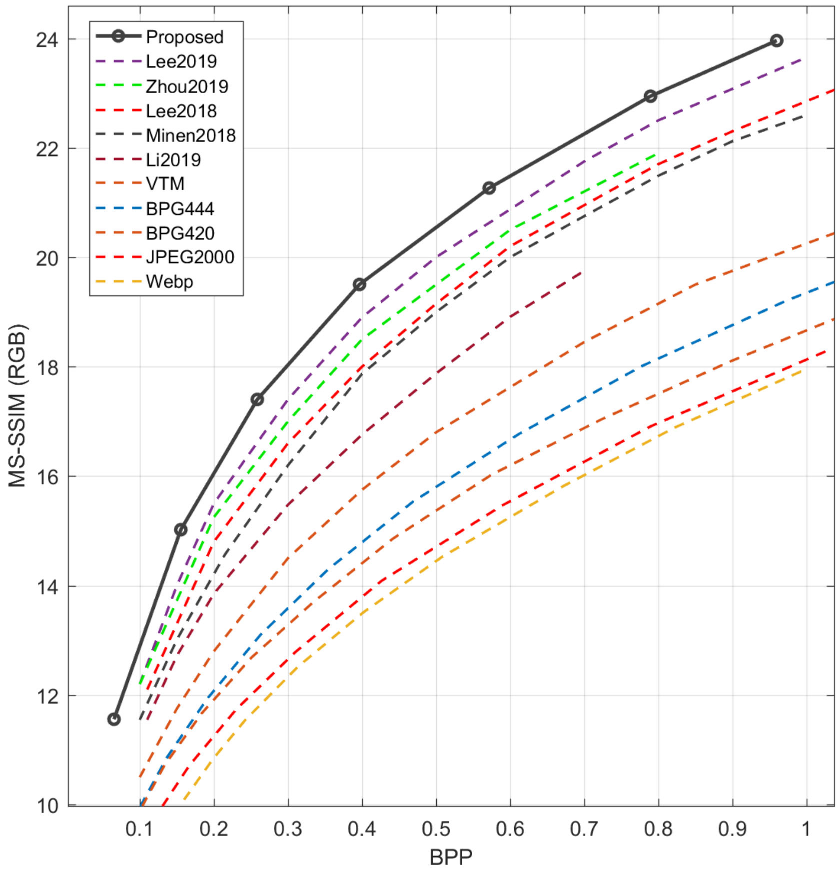
Octave Convolution output:
Lower resolution for lower frequencies.
Similar to Wavelet Transform
Suitable for image compression

Experimental Results (Kodak Dataset)

SFU

- By generalizing octave convolution and applying to image compression, our scheme can outperform H.266/VVC Test Model (VTM) in both PSNR and MS-SSIM
- Best result in the literature





https://arxiv.org/abs/2002.10032

Deep Learning-based Image Compression with Trellis Coded Quantization

Binglin Li[1], Mohammad Akbari[1], Jie Liang[1], and Yang Wang[2]

[1]Simon Fraser University
{binglinl, akbari, jiel}@sfu.ca

[2]University of Manitoba
ywang@cs.umanitoba.ca

SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

University of Manitoba

Thank You