Description of Challenge Proposal by NCTU: An Autoencoder-based Image Compressor with Principle Component Analysis and Soft-Bit Rate Estimation

Chih-Peng Chang[1] David Alexandre[2] Wen-Hsiao Peng[1,3] Hsueh-Ming Hang[2,3]
cpchang.cs08g@nctu.edu.tw davidalexandre.eed05g@nctu.edu.tw wpeng@cs.nctu.edu.tw hmhang@nctu.edu.tw

[1]Computer Science Dept., [2]Electronics Engineering Dept.,
[3]Pervasive AI Research (PAIR) Labs, National Chiao Tung University, Taiwan

Abstract

This paper describes the technology proposal by NCTU for learning-based image compression. The selected technologies include an autoencoder that incorporates (1) a principal component analysis (PCA) layer for energy compaction, (2) a uniform, scalar quantizer for lossy compression, (3) a context-adaptive bitplane coder for entropy coding, and (4) a soft-bit-based rate estimator. The PCA layer includes 1×1 eigen kernels derived from the sample covariance of co-located feature samples across channels. The bitplane coder compresses PCA-transformed feature samples based on their quantized, fixed-point representations, of which the soft bits provide a differentiable approximation for context-adaptive rate estimation. The training of our compression system proceeds in two alternating phases: one for updating the rate estimator and the other for fine tuning the autoencoder regularized by the rate estimator. The proposed method outperforms BPG in terms of both PSNR and MS-SSIM. Several bug fixes have been made since the submission of our decoder. This paper presents the up-to-date results.

1. Introduction

Learning-based image compression has recently attracted lots of attention due to the renaissance of deep learning. Unlike the traditional methods, the learning-based schemes can be adapted to any differentiable objective, opening up many optimization possibilities.

Most learning-based methods [1, 3, 4, 6, 8, 9, 10, 11] rely on training an autoencoder end-to-end with the aim of striking a good balance between distortion and rate losses. Two challenges arise. First, the quantization process for lossy feature map compression causes zero gradients during the back-propagation process. Second, the rate loss is often painful to estimate accurately, as it is highly coupled with entropy coding, the operation of which is not differentiable.

Several prior arts are proposed to address these issues. Li *et al.* [6] overcome the zero gradients by a straight-through mechanism, which simply considers the quantizer to be an identity function during the back-propagation process. Mentzer *et al.* [8] introduce a non-uniform soft quantizer with a smooth mapping function as a surrogate of the hard quantizer. Ballé *et al.* [3, 4] and Theis *et al.* [11] adopt an additive noise model for the quantizer.

The rate estimation is even more challenging. Theis *et al.* [11] estimate the rate from the upper-bound of non-differentiable number of bits. For better estimation, Ballé *et al.* [3, 4] and Minnen *et al.* [9] compute the differential entropy of the quantizer output based on the additive noise model. To bind the rate estimation tightly to the actual entropy coding, Mentzer *et al.* [8] use the context probability model implemented by PixelRNN [12] to compute the self-information of each coding symbol. Their scheme is, however, complicated due to the use of PixelRNN [12] and the non-binary arithmetic coding.

In this paper, we propose a learned image compression system. It has the striking feature of combining effective coding tools from modern image codecs (*e.g.* PCA, uniform quantization, binary bitplane coding with on-the-fly probability updating, and simple context models) with the strong suit of deep learning (*e.g.* non-linear autoencoder). Moreover, we introduce the notion of soft bits to represent quantization indices of feature samples so that the rate loss can be estimated accurately in a context-adaptive, differentiable manner. Experimental results show that our method outperforms BPG in terms of PSNR and MS-SSIM.

This work is an extension of our previous work[2], we introduce PCA into the original framework.

2. Proposed Method

This section details the framework of our image compression system, including the overall architecture, the operation of each component, and the modeling of compression rate and distortion for end-to-end training. Notation-wise, we use a bold letter (*e.g.* $\boldsymbol{x}$) to refer collectively to a

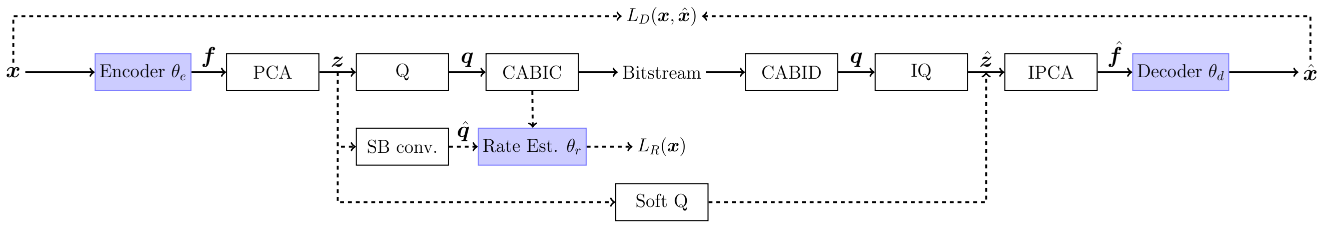

Figure 1. The overall architecture of the proposed image compression system.

high-dimensional tensor and a Roman letter (*e.g.* x) to denote its element in some order.

2.1. Overall Architecture

Fig. 1 illustrates our proposed framework. There are two data paths shown, one for operating the model in test mode (that is, for putting it into use in practice) and the other for its training (*i.e.* training mode).

The data path in test mode, as indicated by the solid arrow lines, begins with encoding an image $\boldsymbol{x} \in \mathbb{R}^{W\times H\times 3}$ of size $W \times H$ in 4:4:4 RGB format through a convolutional encoder $E(\boldsymbol{x}; \theta_e)$ into a compact set of feature maps $\boldsymbol{f} \in \mathbb{R}^{W/8\times H/8\times C}$. Subsequently, the co-located feature samples in $\boldsymbol{f}$ are convolved with 1×1 kernels derived from Principle Component Analysis (PCA) (Section 2.2) to arrive at de-correlated and energy-compacted feature maps $\boldsymbol{z} \in \mathbb{R}^{W/8\times H/8\times C}$. For lossy compression, the magnitude $|z|$ of each PCA-transformed feature sample z is uniformly quantized by a b-bit scalar quantizer Q, leading to a fixed-point binary representation $q = \lfloor |z|/2^{-b} \rfloor$, where 2^{-b} is the quantization step size. That is, the quantization (output) index q is the first b significant bits of $|z|$ in its binary representation (*e.g.* $q = 1100$ for $|z| = 0.81, b = 4$). Note that the sign of z is signaled separately. Like most image compression systems, either learning-based or conventional, the quantization indices $\boldsymbol{q}$ are compacted further by lossless arithmetic coding. Specifically, we arrange $\boldsymbol{q}$ as bitplanes and perform context-adaptive bitplane encoding/decoding (CABIC/CABID) (Section 2.4). To reconstruct the input $\boldsymbol{x}$ approximately, the feature sample is first recovered via inverse quantization (IQ) $\hat{z} = q/2^b \times sign(z)$, followed by the inverse PCA and the convolutional decoding $\hat{\boldsymbol{x}} = D(\hat{\boldsymbol{f}}; \theta_d)$. Currently, our encoder and decoder comes from the ones proposed in [8]; the parameters θ_e, θ_d are however learned by our framework, which aims to strike a good trade-off between rate $L_R(\boldsymbol{q})$ and distortion $L_D(\boldsymbol{x}, \hat{\boldsymbol{x}})$ by minimizing the following objective function with respect to θ_e, θ_d:

$$\lambda \times L_R(\boldsymbol{q}) + L_D(\boldsymbol{x}, \hat{\boldsymbol{x}}), \quad (1)$$

where $L_D(\boldsymbol{x}, \hat{\boldsymbol{x}})$ is defined to be a sum of mean squared errors between the RGB components of $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$.

The data path in training mode, as outlined by the dashed arrow lines, is designed for end-to-end model training. Training a learning-based compression system is often faced with two issues: (1) the quantization effect, which describes a stair-like mapping from the input (*e.g.* $\boldsymbol{z}$) to the output (*e.g.* $\hat{\boldsymbol{z}}$), gives rise to zero gradients almost everywhere, and (2) the rate cost needed to achieve a rate-distortion optimized design is painful to estimate accurately.

To address these issues, we model the quantizer with a differentiable function formed by a superposition of sigmoid functions whenever applicable. As an example, we introduce the notion of *soft bits* $\tilde{\boldsymbol{q}}$ as an alternative to the *hard bit* representation of the quantization indices $\boldsymbol{q}$. Instead of rendering q into "1","1","0","0" for $|z| = 0.81, b = 4$ as was done previously, we express these binary hard bits as real-valued soft bits, *e.g.* "0.91", "0.95", "0.1", "0.07", by the soft bit conversion (SB Conv.) module (Section 2.3). In doing so, each of these soft bits is formulated as a differentiable function of z. They can thus be utilized together with a differentiable rate estimator, implemented by a learnable neural network with parameter θ_r in Fig. 1, to give an accurate estimate of the coding cost (Section 2.5). The same principle of soft quantization is also applied to the mapping from $\boldsymbol{z}$ to $\hat{\boldsymbol{z}}$ (see Soft Q in Fig. 1) for a differentiable approximation of the quantizer.

To sum up, our framework has three networks to be learned end-to-end: the encoder, the decoder, and the rate estimator. Among these, only the encoder and the decoder will actually operate in test mode, while the rate estimator is activated for training only. In addition, the PCA kernels are learned in a separate training step and remain fixed during the end-to-end training for the three networks (Section 3).

2.2. Principal Component Analysis (PCA)

We apply PCA to feature maps $\boldsymbol{f}$ for the sake of achieving energy compaction and decorrelating co-located feature samples across channels. Both properties are desirable to enable efficient bit allocation among transformed feature maps. In our system, PCA (and similarly its inverse) is implemented by a convolutional layer with 1×1 kernels derived from the eigenvectors of the sample covariance matrix of co-located feature samples, *i.e.* $\{f_{i,j,k}\}_{k=1}^{C}$ where (i, j) and k are their spatial coordinates and channel indices, respectively. The computation of the sample covariance matrix is performed in a separate training step, in which the autoencoder is optimized based on minimizing solely the distortion $L_D(\boldsymbol{x}, \hat{\boldsymbol{x}})$. Moreover, mean removal (respectively, mean addition) is done for each feature map before the PCA

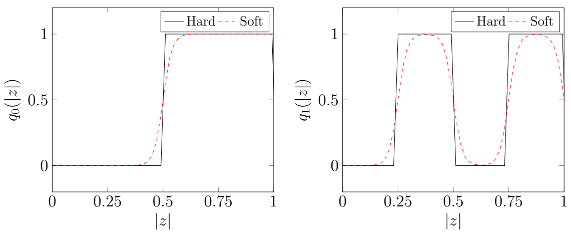

Figure 2. Soft bit versus hard bit mappings.

(respectively, after the IPCA). With enough precision, the PCA and IPCA perform an identity transformation.

2.3. Soft Bit Conversion

The soft bit conversion plays a central role in enabling our compression system end-to-end trainable. It is to convert the binary, hard-bit representation of the quantization index q of a PCA-transformed feature sample z into a differentiable function of z, namely the soft-bit representation. In the previous example, the binary fixed-point representation of q for a feature sample $|z| = 0.81$ is "1100" when $|z|$ is quantized uniformly with a step size of 2^{-4}. We observe that each of these hard bits $q_0 = 1$, $q_1 = 1$, $q_2 = 0$, $q_3 = 0$ is in fact a function of $|z|$. For instance, the first bit q_0 equals to 1 when $|z|$ is in the interval $[0.5, 1)$ and 0 when in the interval of $[0, 0.5)$. The mappings for the first two bits q_0, q_1 are visualized in Fig. 2 (see the hard-bit curves). Apparently, due to their rectangular waveforms, the derivative with respect to $|z|$ is zero almost everywhere, making the training with back-propagation impossible.

To circumvent this difficulty, we approximate these hard-bit mappings by a superposition of sigmoid functions (see the soft-bit curves in Fig. 2). This is motivated by the fact that any rectangular waveform can be expressed as a superposition of step functions, which in turn can be approximated by sigmoid functions with a adequately chosen hyper-parameter α:

$$u(s) := \begin{cases} 1 & \text{if } s \geq 0 \\ 0 & \text{if } s < 0 \end{cases} \approx \sigma_\alpha(s) := \frac{1}{1+e^{-\alpha s}}. \quad (2)$$

As an example, it is seen that:

$$\begin{aligned} q_1(|z|) &\approx \tilde{q}_1(|z|) \\ &:= \sigma_\alpha(|z| - 0.25) - \sigma_\alpha(|z| - 0.5) \\ &+ \sigma_\alpha(|z| - 0.75) - \sigma_\alpha(|z| - 1). \end{aligned} \quad (3)$$

Likewise, in training mode, the mapping from $\boldsymbol{z}$ to $\hat{\boldsymbol{z}}$ (Soft Q in Fig. 1) follows the same modeling approach.

2.4. Context-adaptive Bitplane Coding (CABIC)

For entropy coding, we organize $\boldsymbol{q}$ into bitplanes. A bitplane is formed collectively by the same binary digits of quantization indices. For example, the most significant bitplane consists of all the q_0 of feature samples. Bits are then coded starting from the most significant bitplane to the least significant one, with different feature maps processed in the same manner yet separately.

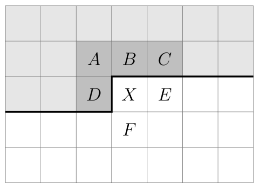

Figure 3. Illustration of context template for context-adaptive bitplane coding. X denotes the location of a coding sample.

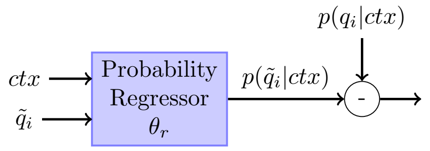

Figure 4. Training of our probability regressor.

To encode bitplanes, we adopt context-adaptive binary arithmetic coding. Inspired by JPEG2000, we classify every bit into a significant bit or a refinement bit. Using Fig. 3 for illustration, for coding a significant bit of the quantization index at X, we refer to the binary significant status of the surrounding indices at B, D, E and F. This yields a total of 16 context patterns (or ctx values for short), each corresponding to a binary probability model that is updated on-the-fly. The context model for sign bits follows the same design but with reference to the signs of surrounding feature samples. For coding a refinement bit, the ctx value is computed based on the bit values of quantization indices at B, D, E, F in the previous bitplane along with those of A, B, C, D in the current bitplane. Since refinement bits are less predictable, we reduce the number of their ctx values to 9 only.

Note that we adopt a traditional hand-crafted design for arithmetic coding because (1) it allows simple adaptation of the context probability model to learn local image statistics and (2) it avoids the need to perform neural network inference at bit level, which introduces extra processing latency in the highly sequential arithmetic decoding process.

2.5. Rate Estimator

To estimate the code length needed to represent an input bit at training time, we refer to its *self-information*. The self-information of a probabilistic event $\mathcal{E}$ is defined to be the negative logarithm $-\log p(\mathcal{E})$ of its probability $p(\mathcal{E})$. In our case, the probability of a coding bit q_i is maintained in a context probability model, which keeps track of $p(q_i|ctx)$, where ctx denotes its context pattern/value. It is however noted that $p(q_i|ctx)$ is approximated by the relative frequency of q_i given the ctx, *e.g.* how many times the event $q_i = 1$ occurs given the present ctx, which is a statistics quantity not differentiable with respective to q_i.

To overcome this problem, we train a rate estimator that includes a neural network as a probability regressor to fit $p(q_i|ctx)$ collected from the training data, as illustrated in Fig. 4. In particular, the probability regressor takes as input the soft bits version $\tilde{q}_i$ of q_i so that it generates non-zero gradient of the estimated rate (computed to be $-\log p(\tilde{q}_i|ctx)$) with respect to the encoder parameter θ_e:

$$\nabla_{\theta_e}(-\log p(\tilde{q}_i|ctx)) = -\frac{1}{p(\tilde{q}_i|ctx)}\frac{\partial p(\tilde{q}_i|ctx)}{\partial \tilde{q}_i}\frac{d\tilde{q}_i}{df}\nabla_{\theta_e} f. \tag{4}$$

It can be seen that if the hard bit mapping is used, the term $d\tilde{q}_i/df$ would be replaced with dq_i/df, which vanishes.

Eq. (4) gives us some important insights into how the estimated rate cost of an input bit q_i would influence the update of the encoder parameter θ_e. Its contribution to the change of θ_e in a gradient update step will be more significant if q_i is in its less probable state, *i.e.*, $p(\tilde{q}_i|ctx) \leq 0.5$, or if its conditional probability distribution $p(\tilde{q}_i|ctx)$ is more biased, *i.e.*, $\partial p(\tilde{q}_i|ctx)/\partial \tilde{q}_i$ is larger. The latter occurs when $p(\tilde{q}_i = 1|ctx) \gg p(\tilde{q}_i = 0|ctx)$ or vice versa.

3. Training

The encoder, decoder, and rate estimator are trained in two alternating phases, with the PCA layer pre-determined following the approach in Section 2.2. In the first phase, we collect the statistics of the context probabilities $p(q_i|ctx)$ from the feature maps, and update the rate estimator θ_r by minimizing the regression error between $p(q_i|ctx)$ and $p(\tilde{q}_i|ctx)$. In the second phase, we incorporate the rate estimator to give an estimate of the rate cost $L_R(\boldsymbol{q})$ and update both the encoder and decoder by minimizing $\lambda \times L_R(\boldsymbol{q}) + L_D(\boldsymbol{x}, \hat{\boldsymbol{x}})$ with respect to their network parameters θ_e, θ_d. During training, we set the batch size to 8 and the learning rate to $1e^{-4}$.

Our model is trained on the COCO dataset[7]. They are randomly cropped into 160x160 patches, and the horizontal and vertical flipping is performed for data augmentation.

4. Experimental results

This section compares the rate-distortion performance of the proposed method with the other codecs. The comparison is conducted on Kodak dataset [5] by compressing test images at several rates with a varying number of feature maps. For every test image, we first calculate the average PSNR and MS-SSIM over its three color components. We then present the average values over the entire dataset as a single quality indicator. The distortion loss $L_D(\boldsymbol{x}, \hat{\boldsymbol{x}})$ is set to mean squared error and negative MS-SSIM, respectively. From Fig. 5, we see that in terms of PSNR, our method performs slightly better than BPG while outperforming the other baselines by a large margin. In terms of MS-SSIM, its superiority over the competing methods is obvious.

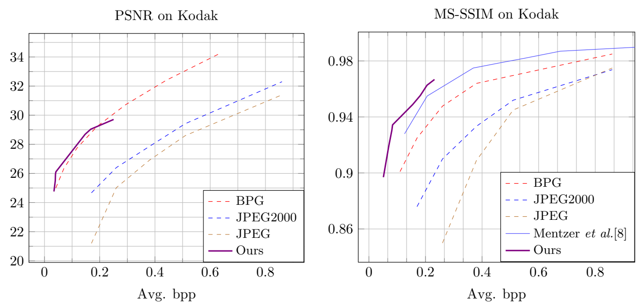

Figure 5. Rate-distortion comparison on Kodak dataset.

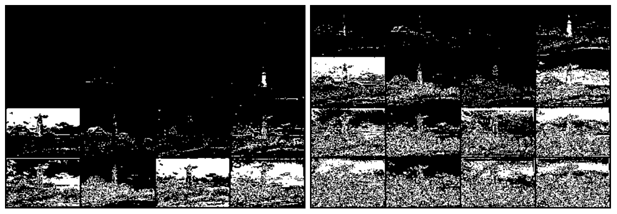

Figure 6. Visualization of the quantized feature maps $\hat{\boldsymbol{z}}$ at different rates. Each column displays the contents of the four bitplanes for representing a feature map.

Fig. 6 shows the bit allocation among quantized feature maps $\hat{\boldsymbol{z}}$ due to our soft-bit-based rate-distortion optimization. Three observations can be made: (1) the dynamic range of feature samples is adjusted automatically by the encoder depending on the compression rate, as evidenced by the presence of nearly all zero bitplanes at lower bpp's; (2) some feature maps are more important than the others in the rate-distortion sense, as evidenced by the uneven bit distribution across feature maps; and (3) the bit allocation is spatially varying, as indicated by the uneven bit distribution across different regions. These together produce a net effect similar in spirit to the importance map mechanism [6].

5. Conclusion

This paper introduces a learned image compression system with PCA and soft bits-based rate-distortion optimization. PCA is found effective for de-correlating feature maps and achieving energy compaction. The soft bits representation allows the rate estimation to be coupled tightly with entropy coding, giving an accurate rate estimate. We also show that learning-based compression methods can leverage well-designed coding tools from modern image codecs for a more cost-effective compression solution.

6. Acknowledgement

This work is partially supported by the MOST, Taiwan under Grant MOST 108-2634-F-009-007 through Pervasive AI Research (PAIR) Labs, National Chiao Tung University, Taiwan.

References

[1] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L.V. Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Advances in Neural Information Processing Systems*, pages 1141–1151, 2017.

[2] David Alexandre, Chih-Peng Chang, Wen-Hsiao Peng, and Hsueh-Ming Hang. Learned image compression with soft bit-based rate-distortion optimization. *arXiv preprint arXiv:1905.00190*, 2019.

[3] J. Ballé, V. Laparra, and E. P Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

[4] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

[5] Kodak PhotoCD dataset. `http://r0k.us/graphics/kodak/`.

[6] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang. Learning convolutional networks for content-weighted image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3214–3223, 2018.

[7] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[8] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4394–4402, 2018.

[9] D. Minnen, J. Ballé, and G. D. Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pages 10794–10803, 2018.

[10] O. Rippel and L. Bourdev. Real-time adaptive image compression. In *International Conference on Machine Learning*, pages 2922–2930, 2017.

[11] L. Theis, W. Shi, A. Cunningham, and F. Huszár. Lossy image compression with compressive autoencoders. In *International Conference on Learning Representations*, 2017.

[12] A. Van Oord, N.l Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756, 2016.