END-TO-END IMAGE COMPRESSION OPTIMIZATION WITH DEEP NEURAL NETWORKS

Paulo Miguel Ribeiro Eusébio

Electrical and Computer Engineering
Instituto Superior Técnico
Lisbon, Portugal
paulo.eusebio@tecnico.ulisboa.pt

ABSTRACT

Convolutional neural networks (CNNs) are a popular deep learning architecture used to address image-based tasks and have also generated a growing interest to address low-level tasks such as image coding. Thus, in this paper is proposed a CNN-based coding framework, where a CNN is used for image simplification (i.e. down-sampling) before the encoder and other CNN for image enhancement (i.e. down-sampling) after the decoder, this is based on similar frameworks. Also, a third CNN, so-called CNN-FakeCodec, is introduced, to model coding distortion, for an improved training procedure. Since related solutions, mostly focus on minimizing distortion, here it is also defined a new loss function that uses the discrete cosine transform to estimate the rate required to encode an image, thus allowing to minimize both rate and distortion. The assessment results show that the proposed solutions can outperform the selected benchmarks, although depending on the image content and target quality.

***Index Terms*—** Deep learning, convolutional neural networks, image coding, rate-distortion optimization, discrete cosine transform.

1. INTRODUCTION

In today's world, image coding is critical in multiple contexts, notably for the large companies (e.g. *Instagram*, *Facebook* or *Google*) which need to store and transfer large amounts of images. Without image coding, the storage and bandwidth requirements would be much larger since storing an image using its original sample-based representation is very rate costly. Efficient lossy image coding solutions are critical to meet those requirements since they achieve large rate savings by removing some perceptually irrelevant image components, notably high-frequency details which are less relevant to the human visual system.

With the emergence of deep neural networks, notably the convolutional neural network (CNN) architecture, and its success solving complex image-based tasks (e.g. image classification [1]), it is reasonable to expect these architectures to do well in the image coding field and improve standard coding tools, i.e. to improve its rate-distortion (RD) performance. In this context, three main paths may be followed nowadays to improve image coding, notably:

- **Conventional image coding technology improvement:** This path consists in further improving the conventional, signal processing-oriented solutions for image coding as developed in the past, notably the Joint Photographic Experts Group (JPEG) standard [2] or the High Efficiency Video Coding (HEVC) (intra coding mode) standard [3].
- **Pure deep learning-based image codecs:** This path implies replacing the conventional codecs architecture with a fully deep learning-based architecture. This is a totally disruptive path which adopts an encoder-decoder architecture only based on neural networks, such as in [4] [5].
- **Conventional image coding complemented with deep learning techniques:** This path tries to somehow combine the best of the two worlds above. Usually achieved by surrounding the conventional codec with a down-sampling CNN before the encoder and an up-sampling CNN after the decoder. The idea is to simplify the image to be coded with the conventional codec with some appropriate deep learning pre-processing, thus saving rate, assuming that the simplified decoded image may be enriched/enhanced with some appropriate deep learning post-processing. This type of solutions does not require changes in the selected coding standard such as in [6] [7].

The coding framework proposed here, which belongs to the last category, uses a CNN before the encoder for image down-sampling (CNN-CR) and another CNN after the encoder for image up-sampling (CNN-SR). A third CNN, so-called CNN-FakeCodec, and a new loss function, are also proposed here. The former aiming at a more complete training procedure, by modeling coding distortion, and the latter aiming at a more complete approach to the RD problem, allowing to minimize both distortion and rate.

The remainder of this paper is organized as follows. Section 2 reviews similar solutions in the literature, in Section 3, the limitations in the literature are identified and addressed. Section 4 reports the performance assessment for benchmarks and the solutions proposed here. Finally, in Section 5 are the main conclusions and a path for the future work.

2. RELATED WORK

In the literature three relevant solutions have the same approach as proposed here. In [6], *Jiang et al.* proposed the first deep-learning framework to optimize conventional coding solutions, consisting in a simple pre-processing CNN for image down-sampling, and more complex post-processing CNN for image enhancement. In [7] *Yue Li et al.* propose a more refined framework but with the same principles. It also has a pre-processing CNN for down-sampling and a post-processing CNN for up-sampling, with significant work to improve and study the down-sampling CNN. Both referred solutions can be used without interfering with the selected codec standard, similarly to the coding framework proposed in this paper. In [8], *Zhao et al.* propose a framework with a post-processing CNN that also models coding distortion targeting to improve a pre-processing CNN.

3. ADOPTED CODING SOLUTION: ANALYSIS AND IMPROVEMENTS

Yue Li et al. is the benchmark for deep learning-based solutions that improve conventional coding solutions, since in [7] it is reported to outperform the very efficient HEVC, notably for low bitrates. In this context, this solution is taken as a foundation of this work, aiming to develop an improved coding solution. Hereafter, this reference solution will be called CNN-YL for 'CNN-based framework proposed by *Yue Li et al.*'.

The CNN-YL is made with two CNNs, a down-sampling CNN (CNN-CR) before the encoder, and an up-sampling CNN (CNN-SR) after the decoder. The CNNs are trained together in a first stage and then the CNN-SR is re-trained with decoded images to maximize reconstruction performance and learn artefact reduction.

3.1. Adopted Coding Solution Limitations

In order to develop an improved coding framework, first it is necessary to identify CNN-YL's limitations. They are the following:

- **1st limitation, CNN-CR unawareness to coding:** The down-sampling CNN is trained unaware of the existence of a codec. More precisely, conventional codecs (such as JPEG or HEVC) involve a quantization function which is not differentiable, thus it is not possible to train both CNNs together with the codec. This problem affects mainly the CNN-CR, since it is never trained aware of the distortion that will follow it. For the up-sampling CNN this problem is minimized by re-training it with decoded images.
- **2nd limitation, CNN-YL unawareness to rate:** The CNNs are trained to minimize a loss function that measures distortion between target and input images. Since the only goal is to minimize distortion, the down-sampling CNN may learn a model that is not coding friendly, i.e. that adds too much details to an image, thus requiring much more rate, which may not be worth in RD performance.

3.2. CNN-FakeCodec: A Trained CNN to Reproduce Coding Artefacts

Lossy codecs remove some image information to allow larger rate savings, thus causing distortion. However, this process is not differentiable (due to quantization and rounding) and neither the CNN-CR nor the CNN-SR can be trained together with the codec. To address this limitation, a third CNN, so-called CNN-FakeCodec, is introduced to model coding distortion. The idea is to train the CNN-CR together with the CNN-FakeCodec and the CNN-SR, thus allowing it to be aware that some image information will be lost. The main difference to [8] is that the CNN-FakeCodec is trained and used independently of the other CNNs, thus allowing to assess how useful is to model coding distortion.

The CNN-FakeCodec is in Figure 1. Other architectures were also tested in [9], notably more complex CNNs and a different approach where the input image was divided in 8x8 blocks, as in JPEG, however, for those architectures, the framework overall performance did not improve.

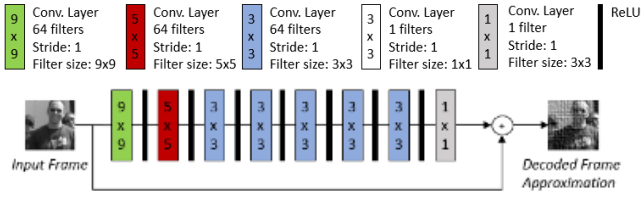

Figure 1: CNN-FakeCodec architecture.

For training, the CNN-FakeCodec input is the original image and the target is the same image decoded with a conventional codec. The model is trained to minimize the mean-squared error (MSE) between output (that approximates a decoded image) and target. After training, the CNN-FakeCodec can be used within the complete framework. Thus, the new complete training procedure is the following:

1. CNN-SR training on images down-sampled with a bicubic filter.
2. CNN-CR training with fixed CNN-SR.
3. CNN-CR and CNN-SR joint training.
4. CNN-SR re-training with JPEG decoded images after down-sampling with CNN-CR.
5. CNN-CR re-training with fixed CNN-FakeCodec and fixed CNN-SR.

The loss function to minimize in the new step 5 is

$$\|g\left(Fake(f(x))\right) - y\|_2^2 + \lambda\|f(x) - F(x)\|_2^2, \qquad (1)$$

where g(.) is the CNN-SR, Fake(.) is the CNN-FakeCodec, f(.) is the CNN-CR. x and y are the input and target images for training, F(.) is a bicubic filter and $\lambda = 0.7$ controls how similar are the CNN-CR computed image and the image obtained with the bicubic filter. This expression is the same used for CNN-YL with the addition of the CNN-FakeCodec. Until step 4, the procedure and loss functions are the same as used in CNN-YL [7], thus allowing a fair comparison

3.3. Improved Loss Function: Including a Rate Term

Since the related solutions only focus is to minimize distortion, here is proposed a new loss function term that allows minimizing rate along with distortion. Naturally, this only concerns the CNN-CR model since it computes the image to be coded. More precisely, the CNN-YL training procedure is repeated, however with a new loss function:

$$L = L_{REC} + \lambda R \quad (2)$$

where L_{REC} is the reconstruction loss (i.e. MSE) between target and the framework output (after down-sampling, decoding and up-sampling) λ is a hyper-parameter to control the strength of the rate term (R). Ideally, R would be the rate required to encode each CNN-CR output image, however this value is only known after quantization and entropy coding. Both involve rounding operations which are not differentiable, thus it is not possible optimize a neural network as a function of this rate value. It is necessary to develop a mechanism to estimate the rate spent (i.e. the term R) to encode an image that is compatible with neural network training.

Since JPEG is a rather simple codec, it is easier to estimate the bitrate to encode an image. In JPEG, the rate to encode an image largely depends on the number of DCT coefficients which are not zero valued (or also zero valued) after quantization [2]. Thus, to estimate the rate for JPEG the following method is used:

1. Compute the DCT coefficients for all the down-sampled image blocks (obtained with CNN-CR).

2. If the obtained coefficients are below the JPEG quantization table, this means they will be set to zero after quantization.

Since both steps can be executed with differentiable matrix operations, they can be used for training. After finding the DCT coefficients that will be set to zero, the non-zero coefficients have to be counted, this can be done in a differentiable way with the following expression for all image blocks:

$$R_k = \sum_{u=0}^{8}\sum_{v=0}^{8} \frac{Y(u,v)}{Y(u,v)+\epsilon} \quad (3)$$

where R_k is the estimated count of non-zero DCT coefficients for block k, Y(u,v) is the non-zero coefficient at row u and column v and ϵ is a small value approximate each division to 1 or 0, in this case $\epsilon = 0.0001$. Then the estimated rate R for an image with N 8x8 blocks is

$$R = \sum_{k=0}^{N} R_k. \quad (4)$$

Figure 2 shows an example validating the correlation between estimated count of non-zeros and the rate required to encode the same image. Where for 100 images with size 48x48 it was estimated the number of non-zeros with eq. (**4**) and then coded with JPEG QF 25.

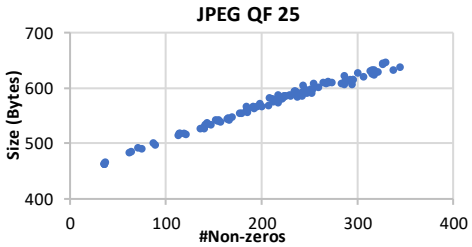

Figure 2: Correlation between the estimated count of non-zero DCT coefficients after compression and the image rate for a specific image.

Despite some oscillations in Figure 2, there is a good correlation between the rate and the estimated count of non-zero coefficients, thus allowing to adopt the proposed rate estimation mechanism for the CNN-CR.

4. PERFORMANCE ASSESSMENT

Since this paper tries to improve the related solutions training procedure, the CNN-CR and CNN-SR architectures are the same in [7] to allow a fair comparison. The selected benchmarks for the performance assessment were:

- **Benchmark 1 – JPEG:** Corresponds to a plain standard coding solution, i.e. without any pre-processing or post-processing. Here the selected standard coding solution is JPEG due to its popularity and simplicity.
- **Benchmark 2 – CNN-YL**: Corresponds to a standard coding solution complemented with deep learning-based tools for down- and up-sampling. The CNN-YL solution has been selected as the benchmark since it is reported in [7] to improve standard image coding, notably the very efficient HEVC Intra image coding solution.

Having the benchmarks defined, the solutions defined and assessed in this paper are:

- **CNN-FC:** Defined in Section 3.2, where the CNN-YL is trained with the CNN-FakeCodec.
- **CNN-RD:** Defined in Section 3.3, where the CNN-YL loss function is replaced with eq. (**1**).

4.1. Experimental Settings

For neural network training and validation, the DIV2K [10] train and validation datasets were used since they offer a large content diversity suitable to train more robust models, as reported in [10]. The test images correspond to the test conditions for the Compression Grand Challenge organized in the International Conference on Image Processing 2016 (ICIP 2016) [11] context and the Moving Picture Experts Group (MPEG) [12] test video sequences (only each sequence's 1st frame). These images are used for the final performance assessment codec benchmarking and for neural network testing (simulating realistic conditions), since they are typically used for testing coding solutions.

All neural networks (benchmark and proposed solutions) were trained for 30 epochs, with 117905 96x96 patches cropped (without overlapping) from the DIV2K training

dataset, with batch-size 16. For gradient descent, the Adam optimizer was used with $\beta_1, \beta_2 = 0.9$ and default parameters. The learning rate was initialized at 0.0001 and halved after every 10 epochs.

A new model was trained for each JPEG QF to maximize performance. For JPEG, performance is reported for QFs = [5, 10, 15, 25, 35] and the benchmark and proposed solutions performance is reported for QFs = [25, 55, 75, 90] manually adjusted to achieve similar rate ranges.

The adopted quality metric is the PSNR between the input (original) and output (decoded) images while the rate is measured as the average bits per pixel (bpp) required to encode the input (original) image with JPEG. These metrics allow plotting RD curves, a popular assessment approach for the compression power of image codecs.

4.2 RD Performance for the Proposed Solutions

Figure 3 shows the RD performance for the proposed solutions (CNN-FC and CNN-RD) and the reference benchmarks (JPEG and CNN-YL). λ in eq. (**2**) was experimentally set to maximize RD performance (see [9] for details) to $\lambda = 1E-8$ for QF90 and $\lambda = 1E-7$ for QFs = 25, 55, 75.

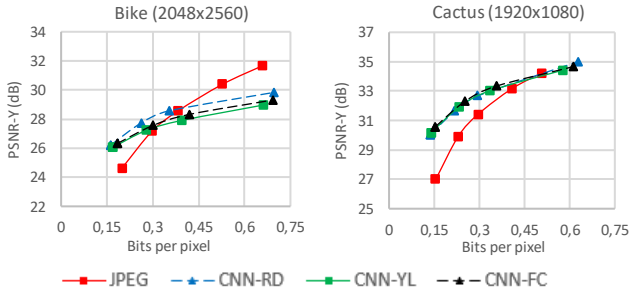

Figure 3: RD performance for the proposed coding framework with a method to minimize both the rate and distortion.

Since in Figure 3, the RD performance is very similar for the CNN-based solutions, in Table 1 are the BD-rate and BD-PSNR performances for the proposed solutions with respect to the benchmarks, e.g. for *Bike* the CNN-RD has 12.55 % rate saving and a 0.299 dB PSNR (i.e. better quality) difference with respect to JPEG.

Table 1: BD-rate and BD-PSNR for the proposed solutions.

Solution	Image	JPEG (% \ dB)	CNN-YL (% \ dB)
CNN-RD	*Bike*	*-12.55 \ 0.299*	*-24.86 \ 0.739*
	Cactus	*-8.51 \ 0.345*	*-14.93 \ 0.536*
CNN-FC	*Bike*	*2.07 \ -0.273*	*-8.57 \ 0.202*
	Cactus	*-30.69 \ 2.093*	*-3.44 \ 0.113*

Figure 3 shows that all CNN-based solutions have better performance than JPEG at low rates. For larger rates (around 0.4 bpp), the information lost with down-sampling is too large (despite saving rate), where for JPEG the distortion is lower, thus having a better RD performance.

Among the CNN-based solutions, the CNN-RD, on average, has the best performance. The main reason is that CNN-YL/FC constrain the down-sampled images to be similar to a version obtained with a bicubic filter (to avoid noisy images). The CNN-RD on the other hand, has a softer constraint: to minimize rate. A softer constraint allows different transformations, which leads to a better performance in most cases. Also, the CNN-RD is better for images with more details (such as *Bike*) since bicubic down-sampling is not adequate to preserve high-frequency details. In [9] 8 other images were tested with similar peformance.

The CNN-FC achieves better PSNR values at the cost of more bpp, thus with a similar RD performance to the benchmark. This means the down-sampling CNN-CR tries to preserve more details, naturally costing more rate which, on average, is worth for the overall RD performance in comparison to the benchmark CNN-YL.

Figure 4 shows a visual example that allows to understand the behavior at each stage for the studied frameworks.

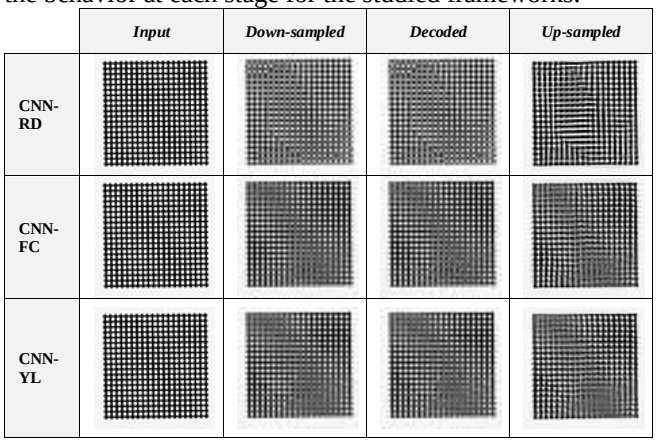

Figure 4: Obtained images along the coding pipeline. The showed image is a 90x90 patch cropped from the test image *Bike*. Images were coded with JPEG with QF 75.

Visually, since the CNN-FC was trained with a similar loss function as the CNN-YL, the high-frequency details are very close, only at the bottom right corner it is noticeable a slightly higher contrast for the CNN-FC showing its effort to preserve more details. For the CNN-RD the details are clearly different, due to the new loss function, with this approach, the CNNs have more freedom to explore other transformations which may result in an improved RD performance.

5. CONCLUSIONS AND FUTURE WORK

After assessing the performance for the two solutions proposed in this paper, the main contributions are:

- **CNN-FakeCodec:** A CNN to model coding distortion that allowed the CNN-CR to preserve more details, however at the cost of more bpp.
- **Loss function with a rate term:** A new loss function that successfully allowed to optimize, for some cases, the tradeoff between rate and distortion.

For future work, the new loss function should be tested and studied for other frameworks (e.g. without down-sampling) for a better understanding of its impact.

6. REFERENCES

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in Advances in Neural Information Processing Systems, vol. 25, Curran Associates, Inc., 2012, pp. 1097-1105.

[2] G. Wallace, "The JPEG still picture compression standard," IEEE Transactions on Consumer Electronics, vol. 38, no. 1, pp. 18-34, February 1992.

[3] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, Thomas Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1649-1668, December 2012.

[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, Honolulu, HI, USA, July 2017.

[5] Lucas Theis, Wenzhe Shi, Andrew Cunningham, Ferenc Huszár, "Lossy Image Compression with Compressive Autoencoders," https://arxiv.org/abs/1703.00395, 2017.

[6] Feng Jiang, Wen Tao, Shaohui Liu, Jie Ren, Xun Guo, and Debin Zhao, "An End-to-End Compression Framework Based on Convolutional Neural Networks," in Data Compression Conference (DCC), Snowbird, Utah, USA, April 2017.

[7] Yue Li, Dong Liu, Houqiang Li, Li Li, Zhu Li, Feng Wu, "Learning a Convolutional Neural Network for Image Compact-Resolution," IEEE Transactions on Image Processing, vol. 28, no. 3, pp. 1092 - 1107, March 2019.

[8] Lijun Zhao, Huihui Bai, Anhong Wang, Yao Zhao, "Learning a virtual codec based on deep convolutional neural network to compress image," Journal of Visual Communication and Image Representation, vol. 63, August 2019.

[9] Paulo Eusébio, “End-to-End Image Compression Optimization with Deep Neural Networks”. Master’s Thesis, Instituto Superior Técnico, October 2019.

[10] R. T. E. Agustsson, "NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study," in IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, July 2017.

[11] “Overview and Benchmarking Summary for the ICIP 2016 Compression Challange," [Online]. Available: https://jpeg.org/downloads/aic/wg1n73041_icip_2016_grand_challenge.pdf. [Accessed October 2019].

[12] F. Bossen, "Common test conditions and software reference configurations," in Joint Collaborative Team on Video Coding (JCT-VC) 2nd Meeting, Geneva, CH, July 2010.