Optimizing an Image Coding Framework with Deep Learning-based Pre- and Post-Processing

Paulo Eusébio
Instituto Superior Técnico – Universidade de Lisboa
Instituto de Telecomunicações
Lisbon, Portugal
paulo.eusebio@tecnico.ulisboa.pt

João Ascenso
Instituto Superior Técnico – Universidade de Lisboa
Instituto de Telecomunicações
Lisbon, Portugal
joao.ascenso@lx.it.pt

Fernando Pereira
Instituto Superior Técnico – Universidade de Lisboa
Instituto de Telecomunicações
Lisbon, Portugal
fp@lx.it.pt

***Abstract*— Convolutional neural networks (CNN) are a popular machine learning architecture used to address multiple image-based tasks from understanding to coding. This paper targets improving image compression efficiency by designing and optimizing an image coding framework where a standard image codec, e.g. JPEG, is combined with deep neural network based pre- and post-processing. While the pre-processing CNN targets simplifying the image to make it more amenable to compression, notably involving its down-sampling, the post-processing CNN targets enhancing the decoded image, also involving its up-sampling. To optimize the compression performance, the processing CNNs are trained involving a third CNN, so-called CNN-FakeCodec, which targets modeling the image codec output, since the encoder-decoder pair is not differentiable, thus not allowing any training. Since the available alternative coding solutions focus on minimizing the image distortion, this paper proposes a new loss function which also considers a rate component, thus allowing to jointly minimize the rate and distortion. The performance results show that the proposed coding solutions can outperform the selected benchmarks, both classical and CNN-based.**

Keywords*— *Image coding, deep neural networks, rate-distortion optimization, pre-processing, post-processing.

I. INTRODUCTION

Image coding is nowadays a fundamental technology in our society, used billions of times per day, by a very large percentage of the world population. This includes not only personal pictures, many widely diffused in social networks such as Instagram and Facebook, but also professional applications and services, such as the movie covers used in Netflix, YouTube, etc. Since the image resolution and target quality have been growing, their uncompressed size is also growing, thus critically asking for efficient image coding solutions to facilitate transmission and storage. In this context, efficient lossy image coding solutions are essential; these solutions achieve high compression, and thus large rate savings, notably by removing perceptually irrelevant image components, e.g. high-frequency details which are less relevant to the human visual system.

With the emergence of deep neural networks, notably adopting the convolutional neural network (CNN) architecture, and its success in solving complex image-based tasks, e.g. image classification [1], it is reasonable to expect this type of solutions to also bring benefits in terms of image coding. This may happen by using the deep neural networks as the codecs themselves or by improving standard image coding solutions, notably increasing their rate-distortion (RD) performance. The latter approach allows to benefit from compatibility with large standard image eco-systems such as the one associated to the JPEG standard. In this context, three main paths may be followed nowadays to improve the image compression performance, notably:

Improving conventional image coding technology: This path consists in further improving the conventional, signal processing-oriented image coding solutions 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]. This is the path currently followed with the emerging JPEG XL standard which targets "significant compression efficiency improvement over commonly used coding standards and solutions at equivalent subjective quality, and superior subjective quality at equivalent bitrates, across a wide range of perceptual qualities in common use." [4]. In the context of these image coding solutions, deep learning-based tools may be used for the optimization of specific modules, e.g. in terms of coding parameters or prediction modes [5].

Developing end-to-end deep learning-based image codecs: This path implies adopting a novel coding paradigm where the conventional coding architecture is replaced by an end-to-end deep learning-based architecture. This is a rather disruptive path which adopts an encoder-decoder architecture mostly based on neural networks, notably where analysis and synthesis transforms are learned based on a large amount of training data and an appropriate loss function, see examples in [6][7][8][9][10][11][12][13][14][15].

Combining conventional image coding with deep learning tools: This path targets combining the better of the two paths above, this means conventional image coding architectures with deep learning tools. One of the possible ways to perform this combination implies using a standard image codec, e.g. JPEG, thus still creating compliant coding streams, which is surrounded by CNN-based pre- and post-processing modules; these CNN modules may involve, first, image downsampling and simplification and, after, image upsampling and enhancement. The idea is to simplify the image to be coded with the standard codec using some appropriate deep learning-based pre-processing, thus saving rate, assuming that the decoded simplified image may be later enhanced with some appropriate deep learning-based post-processing. This type of coding solutions does not require changes in the selected image coding standard, thus still producing compliant streams, see examples in [16][17][18].

The image coding framework proposed in this paper follows the third path above by surrounding a JPEG codec (by far the most used image codec and the largest image eco-system) with two CNNs, notably a pre-processing, down-sampling CNN (CNN-DW) and a post-processing, upsampling CNN (CNN-UP). To include the image codec in the training process, a third CNN, so-called CNN-FakeCodec, is designed to simulate its distortion effects; this is needed since the real codec (in this case the JPEG codec) is not differentiable and thus would not allow any training. The second novelty is a new loss function which considers not only

the image distortion but also the coding rate in order the learned models are aware not only of the distortion (as usual) but also of the coding rate. While the first novelty targets a more complete training procedure by also considering a CNN that models the coding process in terms of its output decoded images, the second novelty aims at a complete RD optimization where both the distortion and rate are considered. This coding solution creates a JPEG compliant stream which may be decoded by the large number of JPEG decoders around as well as enhanced by the proposed CNN-UP deep neural network at no rate cost. The performance results show that the proposed image coding framework outperforms the selected benchmarks, both classical and CNN-based.

The remainder of this paper is organized as follows. Section 2 reviews the related work in the literature and points out its limitations. Section 3 describes the proposed image coding framework and the novel tools, notably the CNN-FakeCodec and RD loss function, which target addressing limitations of the relevant solutions in the literature. Section 4 reports the performance assessment regarding the relevant benchmarks and, finally, Section 5 closes the paper with final remarks and future work perspectives.

II. Related Work

There are very few solutions in the literature following the image coding approach adopted in this paper. In [16], Jiang *et al.* propose the first deep-learning framework targeting to optimize classical coding solutions, notably using a simple pre-processing CNN for image down-sampling, and a more complex post-processing CNN for image enhancement. While the two CNNs are jointly trained, the image codec is not explicitly considered in the overall optimization and the loss function does not consider the rate, just minimizing the distortion. In [17], Yue Li *et al*. propose a more sophisticated deep learning-based image coding framework, following the same principles as [16], while significantly improving the down-sampling CNN. This solution includes a regularization term in the loss function to guarantee that the down-sampled image to be coded is 'similar' to the image obtained with a classical down-sampling filter, notably a bicubic filter. The CNN-DW and CNN-UP are trained together in a first stage and then the CNN-UP is re-trained with decoded images to maximize the reconstruction performance and also perform artefact reduction. This solution (labelled as CNN-YL in the following) is reported to outperform the very efficient HEVC standard RD performance, notably for lower bitrates. In theory, these two solutions above can be used with any classical image codec, notably standard codecs, similarly to the coding framework proposed in this paper. In [18], Zhao *et al.* propose a similar coding framework using the JPEG standard but now also including a CNN to model the coding distortion, this means a virtual codec CNN; however, this solution has limited model training and overall assessment. This coding solution is JPEG specific since the coding distortion CNN, in practice a fake codec, has been trained to simulate the JPEG distortion.

III. Proposed Deep learning-based image Coding Framework

This section proposes a deep learning-based image coding framework where CNN-based pre- and post- processing modules surround a classical image codec. This solution improves the state-of-the-art by addressing the limitations of similar solutions already available in the literature. These limitations are basically twofold:

- **Image codec unawareness** - The CNN training process is not fully aware of the coding process since it is never trained end-to-end due to the non-differentiable nature of the quantization and entropy coding modules. For example, in the CNN-YL solution, the CNN-DW is trained unaware of the existence of an image codec which will code the images it creates. This problem mainly affects the CNN-DW, since it is never trained aware of the decoding distortion. For CNN-UP, this problem is minimized by training it with decoded images as input.
- **Loss function rate unawareness** – The training process minimizes a loss function that only considers the image distortion between the input and output images but not the coding rate. In this case, CNN-DW may learn a model that is not rate friendly, i.e. adding too much detail to the down-sampled image, which may require a significant rate to be coded, not worth in terms of RD performance.

The proposed solution addresses the limitations above by modelling the image codec during training with a CNN-FakeCodec network, thus allowing some end-to-end training of all CNNs involved. Moreover, the rate is considered during the learning process, i.e. the simplification CNN-DW model is aware of the rate cost for coding.

A. Overall Framework Architecture

Figure 1 shows the overall image coding framework architecture, notably its three main modules: the CNN-DW and CNN-UP pre- and post-processing modules surrounding the standard image codec, in this case a JPEG codec. In the training process, the standard image codec is substituted by the CNN-FakeCodec to allow training considering the coding artifacts.

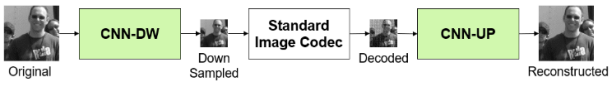

Figure 1 – Overall image coding framework architecture.

The two key technical novelties of this paper, the CNN-Fake-Codec and RD loss function are presented in the next sub-sections.

B. Proposing a CNN-based Fake Codec

Lossy image codecs remove some image information to allow larger rate savings, thus causing distortion, which may be largely perceptually undetectable. However, since this coding process is not differentiable, neither the CNN-DW nor the CNN-UP may be trained end-to-end with the image codec to achieve a better optimization. To address this limitation, a third CNN, so-called CNN-FakeCodec, is introduced to model the coding distortion during the training process; naturally, during the coding process a 'real' image codec is used. The idea is to perform the training in a way that the pre- and post-processing CNNs are both aware that some image information is lost in the coding process.

The proposed CNN-FakeCodec architecture is presented in Figure 2; this CNN has to simulate the effect of a real image codec and thus its output (decoded) image must be similar to the decoded image from a real image codec, e.g. JPEG. In this solution, the CNN-FakeCodec processes the entire input image to create the decoded image. Other CNN-FakeCodec architectures have also been tested by the authors, see [19], notably deeper CNNs and also a block-based

approach where the input image is divided into 8×8 blocks, as in JPEG coding; however, all those architectures did not improve the overall image compression performance compared to the solution proposed here.

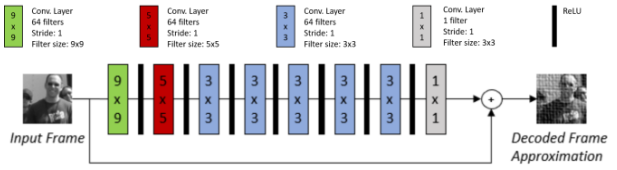

Figure 2: Proposed CNN-FakeCodec architecture.

For training, the CNN-FakeCodec input is the original image and the target is the same image decoded with a standard codec. The model is trained to minimize the mean-squared error (MSE) between the output (a decoded image estimation) and the corresponding target (a real decoded image); in practice, the CNN-FakeCodec learns the coding noise which is finally added to the original image to create the decoded image estimation, see Figure 2. After training, the CNN-FakeCodec can be integrated in the overall image coding framework and used to perform differentiable training and optimization. The new training process improves the CNN-YL training process coding with an additional training step as follows (assuming JPEG coding):

1. *CNN-UP training on images down-sampled with a bicubic filter; in this step a first CNN-UP model is obtained assuming a specific and common type of downsampling.*
2. *CNN-DW training with fixed CNN-UP; in this step a first CNN-DW adjusted to CNN-UP is obtained.*
3. *CNN-DW and CNN-UP joint training; this step targets better matching the two CNNs.*
4. *CNN-UP re-training with JPEG decoded images after down-sampling with CNN-DW; in this step a CNN-UP model aware of the coding process is obtained.*
5. *CNN-DW re-training with fixed CNN-FakeCodec and fixed CNN-UP; in this step, a coding aware CNN-DW model is finally obtained since by integrating the CNN-FakeCodec the loss function gradients may be now end-to-end back-propagated.*

The loss function to minimize 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 CNN-UP, *Fake(.)* is CNN-FakeCodec and *f(.)* is CNN-DW. 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-DW down-sampled image and the corresponding bicubic filtered image. A key difference regarding the CNN-YL solution is Step 5 where CNN-DW is trained together with CNN-FakeCodec and CNN-UP to consider the distortion effect of the image codec in the down-sampling process. The loss functions used here are the same as in CNN-YL [17], thus allowing a fair comparison.

C. Proposing a RD Loss Function

Since the solutions in the literature only minimize the image distortion, a new loss function is here proposed including a term also allowing to minimize the rate, thus implementing a full RD-based approach. This rate term replaces the bicubic filter term in (1) which is typically used as regularization term, a coarse way to control the bitrate. Regarding the CNN-YL training process described in the previous sub-section, this change only concerns the CNN-DW model training since this is the CNN computing the image to be coded and thus which should be rate aware. More precisely, here the CNN-YL training procedure is adopted while including a new loss function as follows:

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

where L_{REC} is the reconstruction loss (i.e. MSE) between the original image and the framework output image (after down-sampling, decoding and up-sampling); as usual in Lagrangian optimization, λ is a hyper-parameter controlling the rate term (R) weight. Ideally, R should be the rate required to encode the CNN-DW down-sampled image; however this value is only known after real coding, thus involving quantization and entropy coding. Since both these operations are non-differentiable, it is not possible to optimize a CNN as a function of the real rate. The solution is thus to develop a mechanism able to estimate the coding rate (i.e. the term R) used to encode an image that is compatible with CNN training.

Since JPEG is a rather simple codec, the JPEG rate to encode an image largely depends on the number of DCT coefficients which are not zero valued in an 8×8 block (or also zero valued since the total is 8×8 =64) after quantization [2]. Thus, to estimate the JPEG coding rate, the following method is used:

1. *Compute the DCT coefficients for all the down-sampled image blocks (obtained with CNN-DW).*
2. *Set to zero all the obtained DCT coefficients which are below the quantization thresholds recommended in the JPEG standard for transparent coding after scaling by the relevant QF (Quality Factor) which defines the target quality/rate.*
3. *Count the total number of non-zero DCT coefficients for all blocks and use it as JPEG rate estimation for the image.*

Since all these steps are differentiable operations, they can be used for training. To confirm that this mechanism offers a good estimation of the JPEG coding rate for an image, Figure 3 shows the real number of bytes per JPEG coded image as well as the number of non-zero DCT coefficients estimated with the procedure defined above, for a Quality Factor of 25 (QF25), using 100 images with size 48×48; the linear variation in this chart and equivalent charts for other QP confirm that the estimation mechanism above performs rather well, thus allowing to adopt it to estimate the JPEG rate in the CNN-DW RD optimization.

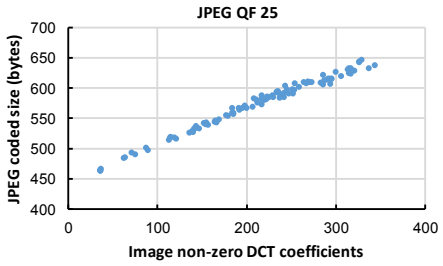

Figure 3: Real number of bytes per JPEG coded image and number of non-zero DCT coefficients for a specific set of images.

IV. Performance Assessment

This section assesses the proposed CNN-based image coding solutions, notably in comparison with some relevant benchmarks. For fair comparison, the CNN-DW and CNN-UP networks are the same as in [17].

A. Coding Solutions under Comparison

The coding solutions under comparison, both proposed and benchmarks, are:

- ***JPEG benchmark:*** Corresponds to a plain JPEG standard coding solution, i.e. without any pre-processing or post-processing.
- ***CNN-YL benchmark [17]:*** Corresponds to the most relevant alternative coding solution in the literature where deep learning-based tools are used for pre- and post-processing, surrounding a classical image codec, here the JPEG standard.
- ***Proposed CNN-FC solution:*** Corresponds to a coding solution including the CNN-FakeCodec proposed in Section III.B, where the CNN-YL benchmark training process is improved by adding a novel step integrating the proposed CNN-FakeCodec.
- ***Proposed CNN-RD solution:*** Corresponds to a coding solution including the RD loss function proposed in the CNN-YL benchmark.

B. Experimental Settings

For the CNN training and validation, the DIV2K [20] train and validation datasets have been used since they offer a large content diversity suitable to train more robust models, as reported in [20]. The test images have been selected from those commonly used by JPEG and also from the DIV2K validation dataset (naturally, those images were never used for training nor validation). These test images have been coded with the adopted benchmarks and the proposed image coding solutions, CNN-FC and CNN-RD.

All CNNs (benchmark and proposed solutions) have been trained for 30 epochs, with 117905 96×96 patches cropped (without overlapping) from the DIV2K training dataset, with batch-size 16. For gradient descent, the Adam optimizer was used with β_1,β_2=0.9 and default parameters. The learning rate was initialized at 0.0001 and halved after every 10 epochs.

A different CNN model was trained for each JPEG QF (and thus RD point in the charts) to maximize the RD performance. For the JPEG benchmark, the compression performance is reported for QFs = [5, 10, 15, 25, 35]; for the CNN-based benchmark and proposed solutions, the compression performance is reported for QFs = [25, 55, 75, 90], manually adjusted to achieve a similar rate range.

The adopted quality metric is the PSNR between the input original image and the output up-sampled image while the rate is measured as the average number of bits per pixel (bpp) required to encode the original (those coded with the JPEG benchmark) or the down-sampled images (those coded with the CNN-based codecs). These metrics allow plotting RD performance curves, and compute BD-Rate and BD-PSNR metrics, which are rather popular performance assessment mechanisms to compare the compression power of alternative image codecs. The BD-Rate reports the rate savings (or increases) in percentage of one codec regarding a reference codec to achieve the same quality; negative values regard rate savings and vice-versa. The BD-PSNR reports the PSNR increase (or reduction) of one codec regarding a reference codec while spending the same rate; positive values regard quality increases and vice-versa.

C. Performance Results and Analysis

Table 1 reports the BD-Rate and BD-PSNR performances for several images with different spatial resolutions. The obtained results show that, generally, the proposed CNN-FC and CNN-RD solutions outperform the benchmarks; the gains are rather substantial regarding the JPEG standard and, naturally, slightly smaller regarding the alternative CNN-YL solution.

Table 1 – BD-Rate and BD-PSNR performance results. The spatial resolution is 2048×2560 for Bike, 1920×1080 for Cactus, 1280×720 for Johnny, 2592×1944 for p08, and 2040×1356 for 0811, 0824, 0825, 0873, 0896 and 0898.

		Versus JPEG		Versus CNN-YL	
Proposed coding solution	**Image name**	**BD-Rate (%)**	**BD-PSNR (dB)**	**BD-Rate (%)**	**BD-PSNR (dB)**
CNN-FC	*Bike*	2.07	-0.273	-8.86	0.211
	Cactus	-25.18	1.609	-4.05	0.124
	Johnny	-40.76	2.868	-6.08	0.187
	p08	-17.08	1.278	-3.82	0.161
CNN-RD	*Bike*	-12.55	0.299	-24.86	0.739
	Cactus	-25.26	1.523	-1.33	0.059
	Johnny	-41.42	2.972	-6.16	0.263
	p08	-14.93	1.053	1.88	-0.042
	0811	-15.83	1.001	-0.00	0.038
	0824	-30.62	1.779	-9.90	0.414
	0825	-21.52	1.334	-9.93	0.446
	0873	-14.41	0.723	-15.44	0.60
	0896	-56.99	4.383	2.40	-0.111
	0898	-43.23	2.737	1.39	-0.063

Figure 4 shows that the CNN-based coding solutions have better performance than JPEG for the lower rates. This was expected since, in this case, the pre and post-processing CNN models allow to simplify (saving rate) and enhance (removing artifacts in an efficient way. For higher rates, the opposite may occur since the information lost in the down-sampling process may be rather relevant (despite saving rate) and is more difficult to be recovered by the up-sampling process; for these rates, the JPEG distortion is lower, thus reaching a better RD performance. Figure 4 shows also shows that the CNN-RD solution may outperform the JPEG benchmark for all rates, at least for some images.

Among the CNN-based solutions, the CNN-RD solution has the best RD performance, on average. The main reason is that the CNN-YL and CNN-FC solutions constrain the down-sampled images to be similar to a bicubic filtered image and thus to obtain an image after down-sampling that is amenable to compression and has good quality. On the other hand, the CNN-RD solution adopts a RD target which minimizes the rate with a suitable differentiable estimator. Moreover, 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. The CNN-FC solution achieves better RD performance compared to CNN-YL benchmark since the CNN-DW was trained jointly with CNN-UP while also including the CNN-FakeCodec. In this case, the simplification pre-processing step tries to preserve more details, naturally at

a higher rate cost which, on average, is worth for the overall RD performance.

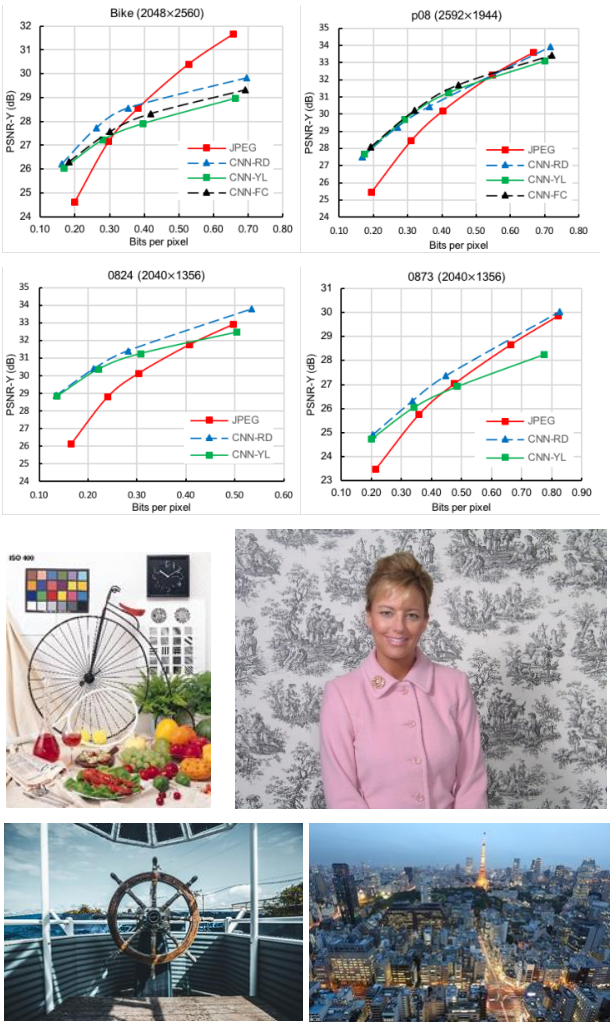

Figure 4: RD performance results for four images, *Bike, p08, 0824* and *0873*.

V. Final Remarks

This paper proposes a deep learning-based image coding framework where the overall compression performance is improved by surrounding a classical image codec, in this case a JPEG standard codec, with deep learning-based matching pre- and post-processing modules. The compression performance improvement is achieved in two ways: one by designing a CNN-FakeCodec able to simulate the image coding distortion, which may be used for more complete CNN training; a second by adopting a more complete loss function where the rate is combined with the image distortion. The performance results show that both proposed CNN-based solutions offer compression performance gains, although depending on the image content and rate range.

Acknowledgments

This work is funded by FCT/MCTES through national funds and when applicable co-funded EU funds under the project UIDB/EEA/50008/2020.

References

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

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

[3] G. J. Sullivan, J.-R. Ohm, W.-J. Han, T. 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] JPEG Committee, "JPEG XL Use Cases and Requirements", Doc. N82009, Lisbon JPEG meeting, January 2019.

[5] T. Dumas, A. Roumy, C. Guillemot, "Context-Adaptive Neural Network based Prediction for Image Compression", IEEE Transactions on Image Processing, vol. 29, pp. 679 – 693, August 2019.

[6] J. Ballé, V. Laparra, E. P. Simoncelli, "End-to-end Optimization of Nonlinear Transform Codes for Perceptual Quality," Picture Coding Symposium, Nuremberg, Germany, December 2016.

[7] G. Toderici, S.M. O'Malley, S.J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, R. Sukthankar, "Variable Rate Image Compression with Recurrent Neural Networks," International Conference on Learning Representations, San Juan, Puerto Rico, May 2016.

[8] J. Ballé, V. Laparra, E. P. Simoncelli, "Density Modeling Of Images using a Generalized Normalization Transformation," International Conference on Learning Representations, Toulon, France, April 2017.

[9] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor and M. Covell, "Full Resolution Image Compression with Recurrent Neural Networks," IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, July 2017.

[10] D. Minnen, J. Ballé, G. Toderici, "Joint Autoregressive and Hierarchical Priors for Learned Image Compression," Advances in Neural Information Processing Systems, no. 31, 2018.

[11] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, N. Johnston, "Variational Image Compression with a Scale Hyperprior," International Conference on Learning Representations, Vancouver, Canada, April 2018.

[12] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. Chinen, S. J. Hwang, J. Shor, G. Toderici, "Improved Lossy Image Compression with Priming and Spatially Adaptive Bit Rates for Recurrent Networks," International Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, June 2018.

[13] J. Ballé, "Efficient Nonlinear Transforms for Lossy Image Compression," Picture Coding Symposium, San Francisco, CA, USA, June 2018.

[14] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, L. Van Gool, "Conditional Probability Models for Deep Image Compression," IEEE International Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, June 2018.

[15] D. Minnen, G. Toderici, S. Singh, S. J. Hwang, M. Covell, "Image-Dependent Local Entropy Models for Learned Image Compression," International Conference on Image Processing, Athens, Greece, October 2018.

[16] F. Jiang, W. Tao, S. Liu, J. R., X. Guo, and D. Zhao, "An End-to-End Compression Framework Based on Convolutional Neural Networks," Data Compression Conference, Snowbird, UT, USA, April 2017.

[17] Y. Li, D. Liu, H. Li, Li Li, Z. Li, F. Wu, "Learning a Convolutional Neural Network for Image Compact-Resolution," IEEE Transactions on Image Processing, vol. 28, no. 3, pp. 1092 - 1107, March 2019.

[18] L. Zhao, H. Bai, A. Wang, Y. 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.

[19] P. Eusébio, "End-to-End Image Compression Optimization with Deep Neural Networks". M.Sc. Thesis, Instituto Superior Técnico, October 2019.

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