Deep Learning-Based Adaptive Image Compression System for a Real-World Scenario

Vito Walter Anelli, Yashar Deldjoo *Member, IEEE,* Tommaso Di Noia and Daniele Malitesta
Department of Electrical and Information Engineering
Politecnico di Bari
Bari, Italy
Email: name.surname@poliba.it

***Abstract*—Deep Learning-based (DL) image compression has shown prominent results compared to standard image compression techniques like *JPEG*, *JPEG2000*, *BPG* and *WebP*. Nevertheless, neither DL nor standard techniques generally can cope with critical real-world scenarios, with stringent performance constraints. In order to explore the nature of this gap, we first introduce an industrial scenario, which contemplates real-time compression of high-resolution images, with strict requirements on a number of quality-performance indicators, namely: the output image quality, the hardware, and the compression complexity. Next, we propose a DL-based image compression model, i.e. a Convolutional Residual Autoencoder (CRAE). In particular, CRAE integrates some structural benefits of a deep neural network, including PReLU activation function and sub-pixel convolution, which have proven to be especially suitable for image compression tasks. We analyze the performance of the proposed CRAE approach by adopting two types of processing: (i) global and, (ii) patch-based processing of image data. To test the models, we exploit a dataset composed of high-resolution images provided by the MERMEC company composed of consecutive images of the railway track captured by a machine vision system called V-CUBE. Furthermore, the company provided strict compression requirements that needed to be met by the developed system. Preliminary results of an ongoing study indicates that the proposed image compression system can meet the requirements by MERMEC with reasonable performance, with a mild advantage observed for full-based CRAE. The obtained outcomes suggests that CRAE can adapt to the specific structure of the given dataset and extracts the salient recurrent patterns inside an image. In summary, this line of research represents the core of the future plug-and-play DL architecture for constrained image compression.**

***Index Terms*—image compression, deep learning, convolutional autoencoders, real-world scenario**

I. Introduction

In the last years, we have observed a massive generation of digital information, e.g., text, audio, video, and images. Despite the spread of these technologies, storage and transmission of that information are a hard challenge for companies and researchers. In this respect, representing data in a more *compact* form is becoming crucial. Under a research perspective, image compression stands for the set of all those algorithms aimed at reducing information within images, for example by leveraging on: (i) *spatial* redundancy of neighboring pixels; (ii) *statistical* redundancy, addressed by *entropy coding* techniques such as *Huffman* [1] and *arithmetic* [2] coding; (iii) *subjective* redundancy addressed by transformation techniques such as *Discrete Cosine Transform* (DCT) [3], which discards unnecessary impercetible information by exploiting the frequency domain. Currently, image compression standards like JPEG [4], JPEG2000 [5], BPG [6] and WebP [7] are the mainstream of academic and industrial research for image compression.

Deep Convolutional Neural Networks (DCNNs) have shown remarkable accuracy in a number of predictive tasks dealing with complex unstructured data such as images. Examples of these tasks include image classification [8] and object detection [9]. Deep Learning has also begun to make its contribution to the image compression domain. In such context, Convolutional Autoencoders (CAEs) are one of the most adopted architectures (e.g., [10]–[12]), since they have proven to perform equal or even better than traditional image coding, by producing higher quality output images at high compression levels.

However, we believe that these research works are not generally tailored to solve specific *real-world* problems with several constraints to meet. In fact, they are often tested on thumbnail datasets, without considering any performance requirements (e.g., compression speed). What is more, they do not even provide the flexibility to extend/enhance the system and meet the aforementioned performance requirements.

In this paper, we define a real-world industrial task, i.e., the real-time compression of high-resolution images, and we set specific project requirements to undergo. To address these requirements, we propose a Convolutional Residual Autoencoder (CRAE), which combines recent advances on DL literature (e.g., PReLU activation function and sub-pixel convolution) specifically tailored to the task of image compression. Given the large size/resolution of images the system has to cope with, we specifically design two variants of the CRAE, the first system using a full-image processing of the content and the second conducting a patch-based processing. Then, we test and benchmark them. We have trained the two variants on a dataset of high-resolution images provided by MERMEC [13], whose ultimate purpose is the detection of cracks found on the railway track. The similarity between the images, i.e., the consecutive acquisitions of the railway track, is a further reason to think that our proposed approach can adequately match the task. In such a situation, an AI-driven image compression system may adapt to the specific hidden

978-1-7281-4384-2/20/$31.00

nature of the dataset. Indeed, it can recognize and extract the recurrent patterns found within the images to a high level of abstraction. Regarding the project requirements, not only does the approach have to preserve essential image details, i.e., cracks, but it must also take into account specific hardware and compression constraints. Finally, the two CRAE variations are compared with JPEG in terms of visual quality and compression performance, to find the best solution showing room for improvement.

The rest of this paper is organized as follows. Section II provides a literature review on learned image compression. Section III describes the real-world scenario and presents the two variants of CRAE. In Section IV, we describe the adopted dataset, the project requirements and the experimental setting with the evaluation metrics. Section V deals with the obtained results and a final discussion about the outputs. In Section VI, we sum up the whole work and suggest possible future directions for improvement.

II. Related work

Deep Autoencoders (DAEs) [14, chap. 14] are widely employed in literature, since they are simple but powerful architectures to perform tasks like learned *lossy* data compression. The general procedure consists of a joint training of: (i) an *encoder*, whose purpose is non-linear dimensionality reduction of the input; (ii) a *decoder*, which tries to reconstruct the input from the hidden representation, i.e. the *embedding*.

For images, encoders and decoders usually consist of stacked convolutional layers, which are mostly suited to work on 2-dimensional (2D) inputs. The actual dimensionality reduction is often addressed by *strided* convolutional layers [11], [12], [15], [16], since they allow to add more *trainable* weights to the overall model. On the decoder side, it is common practice to perform upscaling by means of *sub-pixel* convolutional layers [17], as they have shown better visual quality outputs then traditional upscaling methods [11], [12], [16]. Residual layers [8] can be placed within both the encoder and the decoder, with the purpose of speeding up the convergence [11], [12], [15]. Moreover, PReLU activation function [18] has proven to perform better than traditional activation functions for this specific task [10], [12].

In addition to this, learned image compression frameworks include a *quantizer* and an *entropy coder* between the encoder and the decoder, following the same scheme adopted in image compression standards like JPEG. This introduces the possibility of minimizing a *rate-distortion* loss function and controlling: (i) the average bitrate of the embeddings, (ii) the information loss between input and output. The problem with quantization is that it is not differentiable and so it could not be applied during the training phase for the model weights update. This situation is traditionally solved by removing the quantizer block and simulating its effect with a uniform noise added to the embedding [10], [19], [20], but also by different approaches such as universal quantization [21] or a smooth approximation of the round function [11]. In this work, we did not include any quantizer nor entropy coder, since our main goal was to compress to the maximum possible compression ratio by avoiding loss of significant details.

The Mean Squared Error (MSE) is the standard way to calculate the distortion term inside the rate-distortion loss function, but it often fails to catch differences among images that only humans might perceive. Consequently, some works tend to replace it with a more complex weighted combination of different visual quality metrics (e.g., MS-SSIM [22], see the work described in [15]) or different types of losses, e.g., perceptual [23] and adversarial [24] loss (see the works presented in [12], [20]). In this work, we decided to train the model only on MSE and finally evaluate the quality results of the demonstrated Convolutional Autoencoder in terms of MSE, PSNR and SSIM in order to obtain a comprehensive opinion about the performance of the system.

Finally, an interesting trend in literature is the one proposing a *content-based* image compression, i.e. encoding each image region with different bitrates according to their relative *importance*. For example, this can be achieved with the implementation of learned *importance maps* that allocate more bits to the areas with more details [15], [16], or with models *semantically* aware of the image content which compress specific *objects* with more quality [25].

III. Proposed approach

Given a set of training images $\mathcal{X}$, our goal is to build an image compression method with a convolutional autoencoder (image $\rightarrow$ encoder $\rightarrow$ compressed image $\rightarrow$ decoder $\rightarrow$ reconstructed image). The encoder $E : \mathbb{R}^n \rightarrow \mathbb{R}^f$ transforms a given image $x \in \mathcal{X}$ into a latent representation $\boldsymbol{z} = E(\boldsymbol{x})$. The decoder $D : \mathbb{R}^f \rightarrow \mathbb{R}^n$ reconstructs the original image from the latent representation $\hat{\boldsymbol{x}} = D(\boldsymbol{z})$. Before presenting our proposed approach, we recall and formalize the main concepts of image compression and Convolutional Neural Networks.

Let us define an image $x \in \mathbb{R}^n$ as an array of n-dimensional pixel intensities, and a function $b : \mathbb{R}^n \rightarrow \mathbb{R}$, which returns the average number of bits used to represent the pixel intensities of an image. We may define an image compression algorithm as the process aimed at finding a representation $\boldsymbol{x}_c$ for $\boldsymbol{x}$ such that $b(\boldsymbol{x}_c) < b(\boldsymbol{x})$. By applying the inverse process of compression to $\boldsymbol{x}_c$, i.e., decompression, we get a reconstructed version $\hat{\boldsymbol{x}}$ of $\boldsymbol{x}$ with $b(\hat{\boldsymbol{x}}) = b(\boldsymbol{x})$. When $\boldsymbol{x} \equiv \hat{\boldsymbol{x}}$, we say the image compression is *lossless*, otherwise we say it is *lossy*.

Given a 3-dimensional (3D) input $\boldsymbol{x}$ (i.e., an RGB image or the output of another convolution) and a 4-dimensional (4D) kernel $\boldsymbol{k}$, a convolutional layer produces a 3D output $\boldsymbol{h}$ (often referred to as *feature map*):

$$\left[h_{i,j,f} = \sum_{a,b,c} k_{a,b,c,f} \cdot x_{i+a,j+b,c} \right]_{f=1,\dots,F} \tag{1}$$

The kernel (whose height and width are usually much smaller than input's ones) acts like a *sliding-window* on the input. Traditionally, it moves horizontally and vertically by one step, i.e. with unitary *stride* (as in (1)). However, when the kernel moves by a step of s pixels ($s > 1$), the convolution is called

strided, and it *downsamples* input's width and height by the factor s:

$$\left[h_{i,j,f}^{(s)} = \sum_{a,b,c} k_{a,b,c,f} \cdot x_{i \times s+a, j \times s+b, c} \right]_{f=1,\dots,F} \quad (2)$$

A. Convolutional Autoencoders

Convolutional Autoencoders (CAEs) are autoencoders that include *convolutional* layers within the encoder and the decoder. For this reason, they are particularly suitable for performing *lossy* image compression.

1) Encoder: the encoder produces a compressed version of the input image, i.e., the *embedding*. We introduce some notations as follows:

- the input dimension $[H, W, C]$, where H, W, C are the *height*, the *width* and the *channels* respectively;
- S_i is the i-th strided convolutional layer with stride s_i;
- F_e is the number of feature maps of the last layer in the encoder;
- D_f is the total *downscaling* factor of the encoder, computed as:

$$D_f = \prod_{i=1}^{n_e} s_i$$

where n_e is the number of strided convolutions within the encoder.

Therefore, the dimension of the embedding is $[H/D_f, W/D_f, F_e]$. Consequently, the final *compression ratio* depends only on D_f and F_e.

2) Decoder: on the other side, the decoder learns to reconstruct the input image from its compressed representation, which means that it gradually *upsamples* embedding's height and width. Though there exist several traditional methods to perform such upscaling, the prevailing tendency in literature is to exploit *sub-pixel* convolution. Given an input with dimensions $[H, W, S^2 \times C]$, where H is the height, W is the width and $S^2 \times C$ is the number of feature maps, a sub-pixel convolutional layer rearranges all the values in a way that its output has dimensions $[S \times H, S \times W, C]$, i.e., input's height and width rescaled by a factor S.

B. CRAE: a specialized Convolutional Residual Autoencoder

Convolutional Residual Autoencoder (CRAE) is one of the most commonly adopted models for the *unsupervised*, *lossy* compression of images. We believe that a specialized CRAE may address a hard-constrained image compression task bringing together the merits of the unsupervised approach. For the architectural choices, we mainly refer to [11], [12]. On the one hand, the *encoder* consists of convolutional layers and *strided* convolutions to compress the input images. On the other hand, the decoder takes advantage of convolutional layers and *sub-pixel* convolutions. Indeed, it is common practice to exploit strided, and sub-pixel convolutions to improve the visual quality of reconstructed outputs. Additionally, residual layers speed up the convergence, whereas PReLU activation replaces the conventional ReLU (as in the literature).

C. A portable variant: patches compression with CRAE

The previous solution might show some limitations due to the strict *constraints* we set at the beginning. Since the compression system is supposed to take very *high-resolution* images as input, we may depict two possible scenarios: (i) adopting a *deep* network with a high number of *trainable* weights. For bigger images, it means renouncing *compression speed* (and failing to match the requirements); (ii) adopting a simple architecture to speed up the compression phase. It usually results in an *underfitted* model, i.e., the model is too simple to learn how to compress/decompress complex input images (failing to match the requirements).

To solve the dilemma, we propose a CRAE variant that re-interprets the previous architecture. We may train CRAE on smaller, non-overlapping, *patches* extracted from the input images (this is a known approach when we deal with high-resolution images, e.g., [26]). Given an image $\boldsymbol{x}$ with height H and width W, we can extract P non-overlapping patches with height h_p and width w_p. If we set:

$$\lambda_h = \frac{H}{h_p}, \lambda_w = \frac{W}{w_p}$$

P can be calculated as $P = \lambda_h \times \lambda_w$. In this case, we also suppose $\mathrm{mod}\lambda_w = 0$ and $\mathrm{mod}\lambda_h = 0$. The main advantage here lies in the fact that we preserve the *complexity* of the model. In fact, we train the model on *finer* regions. Thus, we enable the model to learn *smaller details* of the original images.

IV. Experimental setup

This section describes the experimental setup used to validate the usefulness of the demonstrated CRAE image compression system.

A. A high-resolution images Dataset

The images provided by MERMEC come from a machine vision system called V-CUBE. It consists of three sub-systems: *Track Inspection System* (TIS), *Rail Head Inspection System* (RHIS) and *Track Measurement System* (TMS). Since there are two possible acquisition zones called *Ballast* and *Slab*, each image is eventually identified by the couple (Zone, Sub-System).

The images come in raw format described in the following: TIS images are 8-bit grayscale image (256 intensity values) and stored with 1 byte per pixel (BPP), while TMS images are depth images with 1024 intensity levels and stored with 2 BPP; finally, RHIS are similar to TIS, as they are 8-bit grayscale images stored with 1 BPP.

The three sub-systems collect and compress images into smaller consecutive *slices* at time, whose dimensions are: TIS (4000 $\times$ 1024), TMS (1000 $\times$ 1536), RHIS (10000 $\times$ 512). Since each image from the original dataset was actually composed by *consecutive* slices acquired by the systems, the first preliminary step was to extract all the slices, dropping the ones which were smaller than the standard dimensions. The characteristic of the final dataset used in this work is presented in Table I.

TABLE I
SUMMARY OF THE DATASET. IT CONTAINS: ZONE, SYSTEM, AND NUMBER OF ACQUIRED IMAGES/SLICES.

Zone	System	Number of images	Number of slices
Ballast	RHIS_LEFT	6	1850
	VCC_TIS	21	1540
	VCC_TMS	21	1511
	VCL_TIS	4	1396
	VCL_TMS	4	1390
Slab	RHIS	8	2908
	VCC_TIS	50	3750
	VCC_TMS	50	3751
	VCL_TIS	11	4125
	VCL_TMS	11	4126

B. Project requirements for critical maintenance

Unlike previous approaches, we have designed (and tested) CRAE in a *real-world* scenario, on a dataset and with project requirements provided by the company MERMEC. Specifically, CRAE is asked to perform *real-time* compression of particularly *high-resolution* images, with limitations on the *hardware*, the computational *load* and with strict constraints on the output *quality* and *compression ratio*. In the following, we describe these requirements:

- TIS and RHIS slices: it is required to perform a *lossy* compression and target the maximum required ratio of 17%.
- TMS slices: it is required to perform a *lossless* compression with a maximum compression ratio of 20%.

Additionally, it is expected that the information loss due to the lossy compression does not affect the effectiveness of crack detection system for railway tracks, where this latter serves as the main objective of the overall project.

Ultimately, our system was expected to meet a number of *performance* constraints. For instance, it is required to optimize the compression phase so as to limit CPU load, mainly because the CPU is also used to perform other image processing tasks. In Table II, we report the performance requirements provided by MERMEC. Particularly, we may notice the throughput, defined as the average rate at which slices are acquired and compressed, and the CPU load, defined as the percentage of used CPU with respect to its maximum frequency.

C. Evaluation metrics

As a good practice in the literature, we have decided to minimize the *Mean Squared Error* (MSE) as a loss function. Let us define a training set of M images (patches) of P_X pixels

$$\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_M\}$$

split into N_B batches of N_I images (patches) each,

$$\mathcal{B} = \{B_1, B_2, ..., B_{N_B}\}$$

$$B_1 \cap B_2 \cap ... \cap B_N = \varnothing, \bigcup_{B_i \in \mathcal{B}} = \mathcal{X}$$

Hence, since the encoder function $E_{\Theta 1}$ has Θ_1 as trainable weights, and the decoder function $D_{\Theta 2}$ has Θ_2 as trainable weights, we may optimize the model as follows:

$$\left[\min_{\Theta_1, \Theta_2} \frac{1}{N_I} \sum_{i=1}^{N_I} \frac{1}{P_X} \sum_{j=1}^{P_X} (x_{n,i,j} - D_{\Theta_2}(E_{\Theta_1}(x_{n,i,j})))^2 \right]_{n=1,...,N_B} \tag{3}$$

D. Experimental setting

According to MERMEC's project requirements, we defined the experiment setting described in Table III. The considered dataset consists only of TIS slices, split into *train* and *test* sets (80%/20%). Before feeding the network with the slices, each of them is casted to `float32` and normalized in range $[0, 1]$. As for the patches-based approach, the input slices are split into 125 $\times$ 128 non-overlapping *patches*, i.e. each TIS slice of 4000 $\times$ 1024 contains 256 patches.

The Convolutional Residual Autoencoder (for both, the full slice-based approach and the patches-based approach) was trained by *shuffling* the input images at the beginning of each epoch (to avoid *overfitting*). Even then, we grouped them into *batches* (each consisting of 4 slices, i.e., 1024 patches) to speed up the training. We used Adam optimizer [27] with an initial learning rate of 0.0001 and the number of epochs was set to 70. The architectures designed and implemented in the two variations are exactly the same, with only two differences: (i) they work on slice and patch levels respectively and, (ii) the patch architecture shows an additional final *cropping* layer, which allows the output to have the same *height* and *width* as the input.

Encoder's and decoder's architectures are shown in Tables IV - V. Also, we mentioned some references that motivate the architectural choices. Moreover, it is worth noticing that, by default, the convolutional stride is intended as unitary. To preserve a compression ratio constant at 17% for both full-slice and patches-based approaches, we adopted the following configuration: (i) the total encoder downscaling factor is 32 (see III-A), (ii) the final encoder layer has 44 feature maps (see again III-A) and (iii) the embedding is stored in `float32` precision. The code was written in Python, and we exploited TensorFlow 2.0 for the implementation of the CRAE.

V. RESULTS AND DISCUSSION

We evaluated the visual and performance results on the test set by comparing the two variants of CRAE against JPEG. To provide a *fair* comparison, highlighting some key aspects may be beneficial. CRAE provides a constant *compression ratio* of 17%, while JPEG compresses at different compression ratios depending on the input image and the chosen *quality* factor, ranging from 0 to 100, with better quality implying higher the memory occupation. Hence, we first need to find the JPEG quality factor that approximates a 17% compression ratio for each slice in the test set ($\sim$ 84 on average). We evaluated the quality and performance results on *full-size* slices from the test set. Consequently, with the patches-based approach, we reconstructed the original slice, and then we evaluated

TABLE II
PERFORMANCE CONSTRAINTS PROVIDED BY MERMEC.

Dataset	Throughput [MB/sec]	CPU load [GHz]
TIS	347.22	0.83
RHIS	65.76	0.20
TMS	214.98	0.64

TABLE III
EXPERIMENTAL SETTINGS CONSTRAINTS.

Constraint	Description
Quality	The found cracks must be still visible after the lossy compression.
Performance	CPU load must be minimized, while throughput and compression speed must be maximized.
Hardware	**CPU:** 6 × single core hyper threaded (6 core, 2 threads), Intel(R) Core(TM) i7-5820K CPU @ 3.30GHz **GPU:** 1 × GeForce GTX 970, 4 GB. **RAM:** 8 × DDR4, 8 GB.
Images	TIS slices (4000 × 1024, 1 BPP).
Compression	Lossy compression with a maximum compression ratio of 17%.

the metrics. It is straightforward that the primary goal of this technique, learning the essential features for the specific task, cannot emerge with the standard but coarse evaluation. As for the visual quality of the output, we take into account three quality metrics traditionally adopted for image comparison, i.e., MSE, PSNR, and SSIM [28]. On the other side, we measure the performance of the methods considering time, CPU load, throughput, and GPU load during the compression phase. The compression consists of the following sub-steps: (i) reading of the slice; (ii) [optional] extraction of the patches for the patches-based approach; (iii) compression; (iv) storing the compressed image. Tables VI - VII show the results for quality and performance computed on the test set. Table VI shows the results for average quality metrics on the test set: Mean Squared Error (MSE), Peak Signal-To-Noise Ratio (PSNR), Structural Similarity Index (SSIM), Compression Ratio (CR), bits per pixel (bpp) and jpeg quality (Q). Table VII shows the results for average performance metrics on the test set: compression time, percentage of CPU load, CPU load, throughput and percentage of GPU load. The experimental evaluation shows that CRAE trained on full slices performs better than the one trained on non-overlapping patches both on quality and performance. Visually, the most significant difference lies in the *blocking artifacts* caused by the patches-based approach (see Fig. 1).

Since we have integrated the different blocks separately, we may provide some useful design comments. It is noteworthy that adding *residual blocks* to the network (both in the encoder and the decoder) actually helps to speed up the convergence. Moreover, we have experienced the different behavior when adding *batch normalization* [29]. Indeed, it seems not to improve convergence speed. It may be due to

TABLE IV
ENCODER'S ARCHITECTURE.

Layer	Parameters
Conv2D ↓ (as in [11], [12], [15], [16])	ksize = [5, 5] feature maps = 64 stride = [4, 4]
Activation PReLU (as in [10], [12])	
Conv2D ↓	ksize = [5, 5] feature maps = 64 stride = [4, 4]
Activation PReLU	
Res Block [Conv2D PReLU Conv2D] (as in [11], [12], [15])	Conv2D_1: ksize = [3, 3] feature maps = 64 Conv2D_2: ksize = [3, 3] feature maps = 64
Conv2D ↓	ksize = [3, 3] feature maps = 44 stride = [2, 2]

TABLE V
DECODER'S ARCHITECTURE.

Layer	Parameters
SubPixelConv ↑ (as in [11], [12], [16])	rescaling = 2
Res Block [Conv2D PReLU Conv2D]	Conv2D_1: ksize = [3, 3] feature maps = 64 Conv2D_2: ksize = [3, 3] feature maps = 64
SubPixelConv ↑	rescaling = 4
Conv2D	ksize = [3, 3] feature maps = 16
Activation PReLU	
SubPixelConv ↑	rescaling = 4

the presence of the residual blocks. However, we are interested in investigating this aspect further. On the quality side, *sub-pixel convolution* has shown to be an efficient way to perform upsampling, compared to other techniques commonly applied in *deconvolutional* networks (see [30], [31]). Additionally, we have exploited neither *quantization* nor *entropy coding*. It allows CRAE to ease the compression phase and keep a constant compression ratio at 17%. In this respect, it is known that quantization leads to an inevitable loss of the original information. We have chosen to be conservative in this specific configuration. However, we plan to introduce such blocks within the overall pipeline to effectively evaluate the impact of these blocks.

All in all, we may draw some general remarks. I.e., training the CRAE on full slices is more effective than training it on non-overlapping patches. It may be due to several reasons:

1) extracting the patches from each input slice represents an additional step before performing compression, so compression time increases;

TABLE VI
QUALITY METRICS EVALUATION ON THE TEST SET.

Method	MSE	PSNR	SSIM	CR	bpp	Q
CRAE (Full slices)	0.0009905	33.6015	0.8379	0.17	1.375	/
CRAE (Patches)	0.0009979	33.5425	0.8374	0.17	1.408	/
JPEG	0.0001198	41.1165	0.9642	0.17	1.336	84

TABLE VII
PERFORMANCE METRICS EVALUATION ON THE TEST SET.

Method	Time [sec]	CPU [%]	CPU [GHz]	Throughput [MB/sec]	GPU [%]
CRAE (Full-slices)	0.0269	13.14	0.47	152.29	79
CRAE (Patches)	0.0341	20.19	0.73	123.18	53
JPEG	0.0161	10.44	0.38	248.08	0

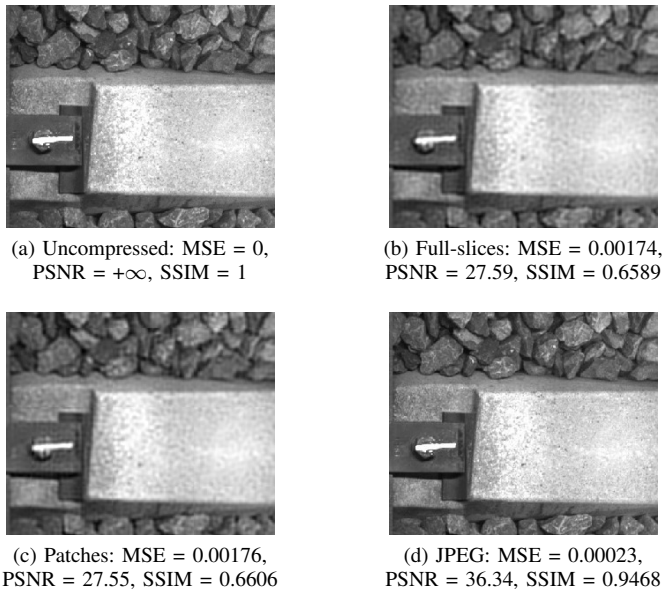
2) the GPU load is lower (since we are working with 125×128 images). Nevertheless, the patches-extraction step impacts on the CPU load.

Moreover, under a qualitative perspective, the patches-based approach performs worse than full-slices CRAE. The reason might be twofold:

1) the *blocking artifacts* may worsen the output quality of the reconstructed image (e.g., the crack is split into two consecutive patches);
2) the CRAE trained on the single patches is not able to perceive the *whole* slice context (the bigger picture). In fact, we feed the network with very *heterogeneous* scenes taken from the same slice (i.e., the patches).

Finally, results also show that JPEG is still far, both on quality and performance level. Even though the patches-based approach shows worse performance in the specific configuration, we believe it may have much more room for improvement than the full-slice solution. In detail, two observations led to this consideration:

- The MSE values calculated on patch-level are not uniformly distributed all over the areas within a slice. Indeed, there are regions with *higher* quality reconstruction, and others with *lower* quality reconstruction (see Fig. 2). A model working on full slices (especially with *high-resolution* images) cannot catch such details properly.
- In an *anomaly* detection task on the railway track, we will take into account every single region and assign a relative *importance* to it. Consequently, we may compress regions with cracks with higher quality than ones with futile details.



(a) Uncompressed: MSE = 0, PSNR = $+\infty$, SSIM = 1

(b) Full-slices: MSE = 0.00174, PSNR = 27.59, SSIM = 0.6589

(c) Patches: MSE = 0.00176, PSNR = 27.55, SSIM = 0.6606

(d) JPEG: MSE = 0.00023, PSNR = 36.34, SSIM = 0.9468

Fig. 1. A detail from (Ballast, VCC_TIS). The full-slice approach slightly outperforms the patches-based one, both visually and quantitatively. In addition to this, the patches-based output shows some blocking artifacts.

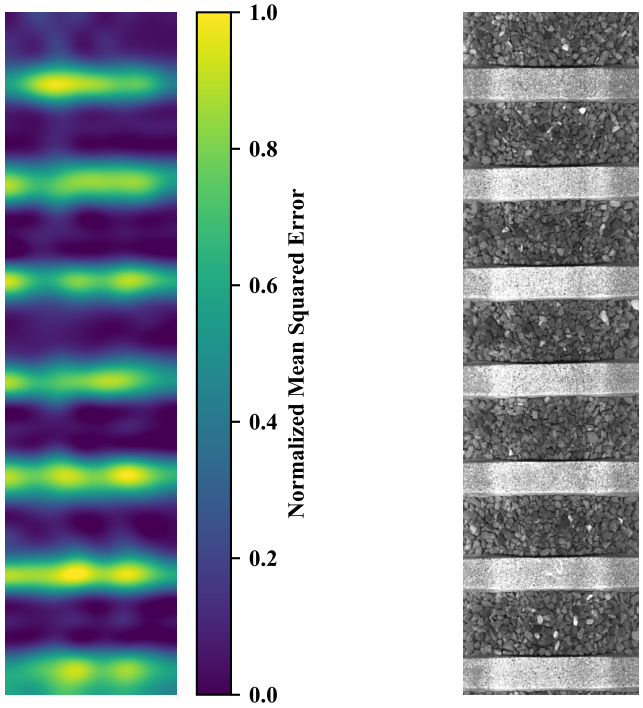


Fig. 2. Heatmap of the normalized MSE, calculated with the patches-based approach. Yellow and blue regions show high and low distortion respectively.

VI. CONCLUSION AND FUTURE WORK

In this work, we have designed a Deep Learning-based (DL) image compression system, i.e., Convolutional Residual Autoencoder (CRAE) for a real-world problem: the real-time compression of high-resolution images with strict project requirements. We have integrated into CRAE some effective solutions from the literature like PReLU activation function and sub-pixel convolution. To handle the complexity of input images, we have implemented two variants of CRAE. The first one processes full images, while the other exploits non-overlapping patches extracted from them. Hence, we

have trained and tested the model on a dataset provided by MERMEC, which consists of high-resolution, consecutive acquisitions of the railway track. In this respect, the ultimate purpose is to detect cracks for critical maintenance. The overall system has to compress those images in real-time, with additional project requirements on the computational load and on the quality of reconstructed images - which must necessarily avoid the loss of essential details (such as the detected cracks). Finally, we have evaluated quality and performance metrics (e.g., MSE, and compression speed), and we have set JPEG as a baseline. The results show that CRAE applied on full slices outperforms the one applied on patches, especially from a performance point of view. However, the patches-based approach shows more room from improvement since it brings the opportunity of treating different regions inside the image (which are not fixed-size patches) in different ways. It may let CRAE exploit their relative importance (i.e., regions with or without cracks). Finally, the patches-based approach eases the compression of very high-resolution images, and it may represent the right direction for beating image compression standards. Indeed, this opens the doors to a conditional compression technique in which the algorithm understands what images contain. The model may smartly distinguish a dataset composed by similar images showing recurrent patterns. Finally, it could be also aware of the absence (or presence) of cracks in each image and compress it accordingly.

Acknowledgments. The authors wish to thank MERMEC Inc. for their support and for the dataset of images. The authors acknowledge partial support of the following projects: Innonetwork CONTACT, Innonetwork APOLLON, ARS01_00821 FLET4.0, PON OK-INSAID.

REFERENCES

[1] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[2] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, no. 6, p. 520540, Jun. 1987. [Online]. Available: https://doi.org/10.1145/214762.214771

[3] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90–93, Jan 1974.

[4] G. K. Wallace, "The JPEG still picture compression standard," *Commun. ACM*, vol. 34, no. 4, pp. 30–44, 1991. [Online]. Available: https://doi.org/10.1145/103085.103089

[5] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The jpeg 2000 still image compression standard," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36–58, Sep. 2001.

[6] F. Bellard. (2015) Bpg image format. [Online]. Available: https://bellard.org/bpg/

[7] Google. (2001) Webp. a new image format for the web. [Online]. Available: https://developers.google.com/speed/webp

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 770–778. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90

[9] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 2015, pp. 91–99.

[10] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Deep convolutional autoencoder-based lossy image compression," in *2018 Picture Coding Symposium, PCS 2018, San Francisco, CA, USA, June 24-27, 2018*. IEEE, 2018, pp. 253–257. [Online]. Available: https://doi.org/10.1109/PCS.2018.8456308

[11] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. [Online]. Available: https://openreview.net/forum?id=rJiNwv9gg

[12] H. Liu, T. Chen, Q. Shen, T. Yue, and Z. Ma, "Deep image compression via end-to-end learning," in *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 2575–2578.

[13] Mermec group. [Online]. Available: http://www.mermecgroup.com/

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[15] D. Alexandre, C. Chang, W. Peng, and H. Hang, "An autoencoder-based learned image compressor: Description of challenge proposal by NCTU," in *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 2539–2542.

[16] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang, "Learning convolutional networks for content-weighted image compression," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 3214–3223.

[17] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 1874–1883. [Online]. Available: https://doi.org/10.1109/CVPR.2016.207

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015, pp. 1026–1034. [Online]. Available: https://doi.org/10.1109/ICCV.2015.123

[19] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. [Online]. Available: https://openreview.net/forum?id=rJxdQ3jeg

[20] J. Wang, X. Tao, M. Xu, and J. Lu, "Semantic perceptual image compression with a laplacian pyramid of convolutional networks," in *2019 IEEE International Conference on Image Processing (ICIP)*, Sep. 2019, pp. 699–703.

[21] Y. Choi, M. El-Khamy, and J. Lee, "Variable rate deep image compression with a conditional autoencoder," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[22] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, vol. 2, Nov 2003, pp. 1398–1402 Vol.2.

[23] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II*, ser. Lecture Notes in Computer Science, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., vol. 9906. Springer, 2016, pp. 694–711. [Online]. Available: https://doi.org/10.1007/978-3-319-46475-6_43

[24] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., 2014, pp. 2672–2680.

[25] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. A. Storer, "Semantic perceptual image compression using deep convolution networks," in *2017 Data Compression Conference, DCC 2017, Snowbird, UT, USA, April 4-7, 2017*, 2017, pp. 250–259. [Online]. Available: https://doi.org/10.1109/DCC.2017.56

[26] S. Wu, M. Zhang, G. Chen, and K. Chen, "A new approach to compute cnns for extremely large images," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, E. Lim, M. Winslett, M. Sanderson, A. W. Fu, J. Sun, J. S. Culpepper, E. Lo, J. C. Ho, D. Donato, R. Agrawal, Y. Zheng, C. Castillo, A. Sun, V. S. Tseng, and C. Li, Eds. ACM, 2017, pp. 39–48. [Online]. Available: https://doi.org/10.1145/3132847.3132872

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[28] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.

[29] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015, pp. 448–456. [Online]. Available: http://proceedings.mlr.press/v37/ioffe15.html

[30] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, ser. Lecture Notes in Computer Science, D. J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., vol. 8689. Springer, 2014, pp. 818–833. [Online]. Available: https://doi.org/10.1007/978-3-319-10590-1_53

[31] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, D. N. Metaxas, L. Quan, A. Sanfeliu, and L. V. Gool, Eds. IEEE Computer Society, 2011, pp. 2018–2025. [Online]. Available: https://doi.org/10.1109/ICCV.2011.6126474