Design space exploration for super-resolution using convolutional autoencoders

Rafael Gallardo-García
Computer Science Faculty
Meritorious Autonomous University of Puebla
Puebla, México
rafael.gallardo@alumno.buap.mx

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

Carlos Hernández-Gracidas
Physics and Mathematics Faculty
CONACYT-BUAP
Puebla, México
cahernandezgr@conacyt.mx

Ignacio Algredo-Badillo
Computer Science Department
CONACYT-INAOE
Puebla, México
email address or ORCID

6th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—Nowadays, in the context of digital image processing, there is a need for obtaining high-resolution images from low-resolution versions, to improve visualization and feature extraction from these images. This defines the open super-resolution problem. There are several methods that have been designed to solve this problem, among which there is deep learning (DL), which is one of the most promising in terms of results, partly due to the big amount of hardware resources available nowadays for their processing. Several DL algorithms have been used in a number of tasks; however, in this work we decided to explore the possibilities of comparing, in the super-resolution task, the models generated by the convolutional autoencoders (CAE), given their behavior of compressing and then reconstructing the input, which seems to be an interesting feature with potential for its use in super-resolution. In this work, we present quantitative and qualitative analysis that allow showing the behavior of these models based on AE. For the performance evaluation of these models in the super-resolution task, we use the DIV2K and Flickr2K image sets, comparing our results with respect to other methods, such as Nearest Neighbors, among others, showing that there is an improvement in terms of some metrics for some of the proposed architectures and that, although the performance of other state-of-the-art DL methods is not improved, there is a considerable reduction of hardware resources needed for processing these architectures.**

***Index Terms*—Autoencoders, Deep learning, Super-resolution, Qualitative Analysis, Quantitative Analysis**

I. Introduction

Image super-resolution (SR), specifically the single image SR, looks for retrieving a high-resolution (HR) image from a single low-resolution (LR) image [6]. This is an inherently ill-posed problem since there are multiple solutions for each of the pixels in the LR image or, in other words, this problems falls in the category of undetermined inverse problems, for which tere is no unique solution [6]. The image SR problem is very important in computer vision and image processing [27] since the solutions to this task have several real-world applications, going from medical image processing [8], [11], [13], surveillance and security [20], [30], and multimedia applications to movies and animation [22]–[24].

Finding a solution to the problem of image SR is an open problem, and several approaches to it have been proposed from very diverse areas, such as the statistical methods, based on predictions, based on borders, based on patches and based on scattered representations [27]. The newest approaches, which have also been the most successful so far, are mostly based on deep learning (DL). The earliest efforts for using DL for solving the image SR problem consisted in using convolutional neural networks. For example, [6] presents the super-resolution convolutional neural network (SRCNN) and, since then, several works have been published dealing with this problem, reaching promising results with the generative adversarial networks (GANs).

The complexity of the problem, added to the wide variety of potential applications, have given rise to the big family of SR techniques and algorithms, which in general differentiate from each other in one or more of these aspects: kind of network architecture, variations in the cost function, and variations in the learning principles or strategies. In this sense, and given the emergence of new knowledge, where no clear foundations are given on what kind of network should be used, nor which topology or configuration is the most adequate, it becomes necessary to propose studies that explore the design space for the solution of this task.

In the present work, we carry out an exploration of the aforementioned design space, with special emphasis in the use of convolutional autoencoders (CAEs) for solving the SR problem. The main motivation behind this proposal is the fact that, although in the state-of-the-art there are models that perform very well in SR, they are very complex and demanding, which makes them almost impossible to implement them in common computers (with limited hardware resources). Besides, there are size limitations (for the input or the output images) in most of these models. Conversely, the experiments in this

work intended to solve the SR problem were performed using commercial graphic processing units (GPUs), which facilitates applying the developed techniques in real-world problems.

The use of CAEs allows extracting several features from the input image and, at the same time, helps find a compact representation of that image. For that reason, a CAE is expected to find compact representations of the input image, and then to reconstruct it with the highest possible quality. This is the motivation in this work for exploring different designs and topologies as possible solutions covering a wide design space, which can be used as a guide mark for the design of deeper and more complex networks.

The main contributions of this work can be listed as follows:

- We present the first comprehensive exploration of the design space of fully CAEs to solve Super-Resolution tasks.
- We propose several new CAE-based topologies, with a focus on high-quality image reconstruction.
- Some of the proposed topologies can handle variable input size (due to their fully convolutional nature).
- Each proposal was evaluated using different performance metrics. Thus, we discuss some hypotheses to explain the results obtained.
- Some of our topologies outperformed the nearest neighbors method in the test set, being a good indicator of the viability of the CAEs for up-sampling tasks.
- The discussion presented has a special emphasis on the behavior of the combinations of each topology and a variety of learning strategies.
- We generated a dataset with a wide variety of objects, colors, and actions, with a total of 3,550 images in 2K resolution.

This article is organized as follows: in Section II, a review of concepts of interest is presented, Section III describes the state-of-the-art in SR, section IV introduces the methodology and gives details on how the experiments were performed, Section V describes the proposed architectures, Section VI reports the obtained results and, finally, section VIII shows the conclusions and the future work.

II. Fundamentals

Usually, an LR image I_x can be modeled as the output of a degradation function D in the following manner:

$$I_x = D(I_y, \delta), \tag{1}$$

where I_y is the HR image and δ corresponds to the parameters of the degradation function (such as the scaling factor or the amount of noise). In the image SR problem, it often happens that both process D and the parameters of degradation δ are unknown, so there is just access to the LR image I_x. In this situation, the problem consists of retrieving an approximation $\hat{I}_y$ to the ground truth HR image I_y, from the LR image I_x. This retrieval is carried out using an SR model F:

$$\hat{I}_y = F(I_x, \theta), \tag{2}$$

where θ are the parameters of model F.

Most of the works reviewed in [27] are focused in modelling the degradation process D as a simple downsampling operation and, although such a degradation process is unknown and affected by several factors, modelling can be carried out as follows:

$$D(I_y, \delta) = (I_y) \downarrow_s, \{s\} \subset \delta, \tag{3}$$

where $\downarrow_s$ is the downsampling operation with a scaling factor s. The datasets available in the literature and, especifically the one generated for the development of this work, follow this idea, using the Lanczos resampling function as $\downarrow_s$, with $s = 6$

For working with image SR there are traditional methods, (such as nearest neighbors, bilinear and bicubic interpolations; and Lanczos resampling); however, the quality of the retrieved images is not the best, therefore we propose using DL for an architecture to learn features, borders and colors in each image, and that way the output of the model is expected to be a bigger image with no information loss.

A. Evaluation metrics

In [1], it is mentioned that some of the methods that are regularly used for comparing two images are based on human perception and objective computational methods. Methods based on human perception are closer to our reach; however, analyzing several images comes with a big time cost and increase propensity to errors. Computational methods tend to be faster, hence the time cost is very low. However, the latter are not necessarily consistent among each other and, furthermore, they are incapable of capturing human visual perception with precision so these methods can lead to big differences in the evaluation results [16].

Image comparison methods can be divided into three types: a) complete reference methods, which perform evaluation using reference images; b) reduced reference methods, which are based on the comparison of extracted features; and c) methods with no reference.

In this work, we used the following complete reference computational methods to compare the model's output images $\hat{I}_y$:

- *Mean Squared Error (MSE).* The MSE metric is an estimator that measures the average of the square errors. If the MSE value is high it means that the images are less similar; on the contrary, if it equals zero, the images are identical. Given the way it is calculated, MSE cannot be negative.

$$MSE(I_y, \hat{I}_y) = \frac{1}{n} \sum_{i=1}^{n} (I_{y_i}, \hat{I}_{y_i})^2. \tag{4}$$

- *Peak Signal to Noise Ratio (PSNR).* The PSNR comparison method is very popular since it has been used to

measure the quality of an image that has gone through a reconstruction process. For the super-resolution problem, PSNR is defined as the ratio of the maximum possible pixel value L to the maximum MSE between the two images being compared (this value is calculated in logarithmic scale). Given the ground truth images I_y, and the images reconstructed with the model $\hat{I_y}$, PSNR is defined using the following equation:

$$PSNR(I_y, \hat{I}_y) = 10 * log_{10}(\frac{L^2}{MSE(I_y, \hat{I}_y)}). \quad (5)$$

- *Structural Similarity Index Measure (SSIM).* In [26], SSIM is proposed as a structure similarity metric for comparing images. To determine how similar the structure is, SSIM is based on contrast, luminosity, etc., using the following formula:

$$SSIM(I_y, \hat{I_y}) = \frac{(2 * \mu_x\mu_y + c_1)(2 * \sigma_{x,y} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \quad (6)$$

where μ_x is the average value of image I_y, μ_y is the average value of image $\hat{I_y}$, σ_x is the variance of image I_y, σ_y is the variance of image $\hat{I_y}$, $\sigma_{x,y}$ is the covariance between both images, and c_1 and c_2 are constants used to avoid division by 0.

B. Cost functions

In the area of super-resolution, several function are used to measure error in image reconstruction (presented in Section II-A). These methods are also used as cost functions, serving as a guide for the model to optimize its output [27], as we will explain next:

- *Mean Absolute Error (MAE) or L1 Norm.* L1 norm measures the loss of pixels or the difference between two images (see Eq. (7)), where n is the amount of pixels in the images. In practice, L1 norm shows a better performance and convergence; also, results are not so smooth as compared to the ones obtained using L2 norm.

$$L1(\hat{I_y}, I_y) = \frac{1}{n}\sum_{i=1}^{n} |\hat{I_{y_i}} - I_{y_i}|. \quad (7)$$

- *MSE or L2 Norm.* Even though MSE (defined in Eq. (4)) is more tolerant to small errors, it also penalizes more big errors and can be affected in a greater way by atypical results. The results when using MSE as cost function are blurred images, besides details in the image are also blurred, hence obtaining unsharp images which are non-pleasant to the human eye. Similar to norm L1, L2 measures loss or the difference of pixels between two images.
- *PSNR.* When PSNR is used, its definition is highly related to the difference in pixels and the minimization of this loss, which maximizes PSNR. Despite this, since pixel loss does not take into account image quality, results can lack some details and are perceptually bad, with too smooth textures. In the experiments we carried out using PSNR as the cost function, 2 variations were tried, the first one using MSE as in the original definition (see Eq. (5)), and the second one replacing MSE with L1 norm, as it is shown next:

$$PSNR(I_y, \hat{I}_y) = 10 * log_{10}(\frac{L^2}{L1}). \quad (8)$$

In Section VI, we will delve into how the variation based on L1 helped obtaining visually sharper results than when using the traditional version of PSNR.
- *SSIM.* SSIM is a perception metric based on the visible structures in the image. This metric, which is defined in Eq. (6), quantifies the image quality degradation caused by the processing. The structural information in an image considers that the pixels are strongly interdependent, specially those closer to each other. Interdependence in images provides relevant information about the structure of the objects in the visual scene.
- *Perceptual Loss.* Perceptual cost functions are employed when comparing two functions that are visually similar, for example, when they are the same image but with a displaced pixel. Perceptual cost functions are useful when high-level features must be measured, such as contents or discrepancies in the image style. These functions are very similar to the pixel-wise cost functions (such as L1 norm) an can be used for training feed-forward networks for tasks like image transformation. It can be said that from all the cost functions used in this work, perceptual cost is the most complex of them. These functions are based on the sum of all the errors between the pixels, taking later the mean. In [14], they argue that perceptual cost functions are not only the best alternative when generating higher-quality images, as compared to other alternatives, but also they make this up to three times faster if properly optimized. In general, perceptual cost functions first extract features from both images, using some pre-trained convolutional network, calculating later the difference in these features (for example, using L2 norm). These kinds of functions can be posed as follows:

$$D(I_y, \hat{I}_y) = d(F(I_y), F(\hat{I}_y)), \quad (9)$$

where D is the cost, I_y and $\hat{I}_y$ are the images to be compared, d is a dissimilarity measure (for example, $L1$ or $L2$) and F is a feature extraction network (such as VGG-16 pr VGG-19). To exemplify, in the following we give a possible definition of a perceptual cost function based on L1 norm.

$$D(I_{yf}, \hat{I}_{yf}) = \frac{1}{n}\sum_{i=1}^{n} |\hat{I}_{yf_i} - I_{yf_i}|, \quad (10)$$

where I_{yf} and $\hat{I}_{yf}$ are the features from images I_y and $\hat{I}_y$ extracted by the pre-trained network.

III. State-of-the-art

For the state-of-the-art review, we have focused in works using DL for the SR problem. This task, which is highly complex and even impossible for certain algorithms with less capacity, implies that a neural network must learn how to transform an LR image into an HR image. The different approaches that have been used in the state-of-the-art for image SR are described in the following.

A. Pre-upsampling SR

Some of the solutions that have been proposed for reducing the difficulty of this problem consist in employing traditional methods as a part of the SR process. In some of these architectures, before the training images are input to the DL architecture, they undergo a processing where they are scaled using a traditional method, as in [27], where they present an architecture based on pre-upsampling. Likewise, the super-resolution architecture SRCNN (Super-Resolution Convolutional Neural Network) [5] uses pre-upsampling, where the LR images are up-scaled using bicubic interpolation, so in the next step the neural network layers can sharpen the image by means of the convolution operation; this way, it is easier for the network to learn how to sharpen an image than if it had to learn how to transform an LR image into an HR version of it. Making this pre-upsampling also reduces computational cost.

B. Post-upsampling SR

One of the main frameworks for performing image SR tasks is post-upsampling. The difference between pre-upsampling and this is that the latter performs image resampling at the end of the model. Several authors suggest this with the purpose of improving computational efficiency and to make full use of DL to increase image resolution, computing most results in a low dimensionality space. Methods based on post-upsampling extract features directly from the input image, using subpixel convolution [22] or transposed convolution [6] to carry out upsampling.

The authors of [6] Suggest a reformulation of the SRCNN architecture, with the purpose of reducing the computational cost this architecture demands. In [31], they propose two image SR models based on Convolutional Sparse Coding (CSC), one of them using post-upsampling, with which they reach a superior performance as compared to the state-of-the-art methods in SR that year. Because the feature extraction process requires a high computational cost, the post-upsampling framework has become one of the most popular for this kind of tasks, given that, by performing feature extraction in a low dimensionality space, the computational cost is significantly reduced.

C. Progressive upsampling SR

Due to the different limitations in both pre-upsampling and post-upsampling, a progressive re-scaling approach has been adopted in several works. In contrast to the previously described approaches, re-scaling is not carried out in a single step, hence a cascade of Convolutional Neural Networks (CNN) is used, which progressively reconstructs the images. For multiple scales, this approach requires no individual model for each scaling factor desired, since in the different stages the image is scaled to a higher resolution and the CNN layers refine it. By dividing the image re-scaling into small tasks, the computational cost is reduced even more; furthermore, the learning difficulty is also reduced for the network. The spacial and temporal costs by re-scaling to multiple sizes are not excessive.

Architectures like the one described in [15] use the information contained in the reconstructed images in the intermediate layers as a basis for the next layer, while other architectures, like the one described in [28], maintain a principal information flow. Architectures following this approach are complicated to design when multiple stages are wanted since more guidelines are necessary during network modelling and, sometimes, more sophisticated training strategies are also needed, given that stability problems in the training may appear. By dividing the tasks into small parts, it is also simpler to use more specific learning strategies.

D. Iterative up-and-down upsampling

Iterative up-and-down upsampling is a rarely used methodology, but that has given interesting results when solving SR tasks using DL. In the aforementioned frameworks a single re-scaling direction is always followed (LR to HR); that is to say from a lower dimension to a higher dimension, and the only difference between both is the position that the re-scaling has in the SR process (at the beginning, at the end or progressive); however, in iterative up-and-down upsampling processes the direction in which the dimension is increased is varied.

SR methods scale images in an iterative bidirectional fashion, i.e., they carry out a repetitive process, where the input image is first scaled to a higher dimension, reducing it later to a lower scale, once and again until the goal dimension is obtained. Following this approach, the Deep Back-Projection Networks for Super-Resolution (DBPN) [10] architecture was the winner of the New Trends in Image Restoration and Enhancement (NTIRE) competition in 2018, in the re-scaling to 8x line.

The architectures that work using this framework search for capturing the mutual dependence that exists between the LR and the HR images using the so-called "back-projection" process [12]. In the SR methods working in this framework, a reconstruction process is iteratively applied, after which it goes back in a way that the intensity of the HR image can be fine tuned [27]. The Feedback Network for Image Super-Resolution (SRFBN) architecture, proposed in [17], employs the iterative up-and-down upsampling, adding dense connections and feedback blocks, in a way that richer and more complex representations of the relations among the images are learned.

Although these methods have reported good results, and have shown to learn deep, rich relations between the LR and the HR images, Wang et al. [27] mention that the design

criteria in the use of "back-projection" modules are not yet clear, and further investigation is needed in the area.

IV. Design space

CNNs are one kind of network which is specialized in processing data organized as grids (matrices). This work will focus in detailing the usefulness of CNNs in image processing. The basic operation, which gives its name to this kind of network is convolution. The convolution is one type of specialized linear operation on to functions with real arguments, where the first function corresponds to the input data, while the second function corresponds to the kernel.

In this manner, CNNs are composed by the input I (multidimensional array of data) and a kernel K (multidimensional array of parameters), where the parameters will be adapting according to the learning algorithm. Both multidimensional arrays are known as tensors. For this work, we consider that if we have two bidimensional input images I, then it is likely that we want the kernel K to be bidimensional as well (see Eq. (11)).

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n). \tag{11}$$

Autoencoders (AEs) are neural networks that are trained to replicate or to copy their input to their output. AEs contain an internal hidden layer h that describes a code which is used for the representation of their input [7]. These architectures can be decomposed into two parts: the encoder $h = f(x)$ and the decoder $r = g(h)$. AEs are designed to stop them from learning $g(f(x)) = x$ in each part of their output, which would make them perfect copiers, but little useful in learning tasks. Although some years ago AEs were used for dimensionality reduction, or for feature learning, recent works have established theoretical relationships between AEs and latent variable models, which has put them in front of modeling and generative tasks [7].

It is possible to obtain an AE, known as denoising autoencoder (DAE), capable of learning something useful when its reconstruction-error term in the cost function is changed. In general, we can say that AEs try to minimize a function like:

$$L(x, g(f(x))), \tag{12}$$

where l is a cost function that penalizes $g(f(x))$ for being little similar to x, which is similar to the $L2$ norm of their difference.The previous statement forces the AE to learn an identity function that does not work if the purpose is that the output is different from the input, but also it keeps certain characteristics. On the other hand, a DAE minimizes:

$$L(x, g(f(\tilde{x}))), \tag{13}$$

where $\tilde{x}$ is a copy of x that has been corrupted by some kind of noise (in this case, a degradation). DAEs should be capable of undoing such a corruption instead of simply copying their input.

On the other hand, CAEs are architecture that look for minimizing the reconstruction error by learning the best parameters or kernels K that compound their layers. This kind of architectures are easily distinguishable from CNNs since, while CNNs are capable of extracting and using features, with the purpose of classifying their input (acting as supervised learning algorithms), CAEs are trained to learn automatically filters capable of extracting important features, which can be used later for reconstructing their input.

One interesting characteristic of CAEs is that, being completely convolutional, the scale in an acceptable way the dimensionality of their inputs. This is because input dimensionality does not affect the amount of parameters needed for producing an output.

The hypothesis in the present article is that a CAE is capable of solving the super-resolution task, with robustness and with the capability to scale any image, no matter their dimensionality.

A. Proposed methodology

The objective of this work consists of evaluating a model F that is capable of reversing the degradation process $\downarrow_s$, improving the quality of image I. This way, it is necessary for the outputs $\hat{I}_y$ generated by F to obtain an optimal average score (from all the evaluated scaling methods) in MSE, PSNR and SSIM on a test dataset never seen before. To achieve this, the exploration of the design space consists of using a CAE (the model F) that is capable of reversing the degradation process $\downarrow_s$.

The main contribution of this work lies in that, so far, to the best of our knowledge, there are no works looking for solving the SR problem by using a single CAE, since although it is true that there exist approaches using paired AEs [21], [29] used for maping LR encodings to HR ones, and even some approaches make use of AEs for SR tasks [2], [3], [25], it can be observed that, in these proposals, the AE does not constitute the main architecture.

In this sense, in this work we explore the idea of using an AE as the unique architecture for solving the SR task.

The idea behind the use of an AE for this task is the intuition on DAEs (see Eq. (13)). If we interpret the corruption function that generates $\tilde{x}$ as the degradation function D and the corruption process as a degradation process $\downarrow s$, we can apply the principle of DAEs for the task of image scaling. This application of DAEs searches for reversing the effects of process $\downarrow s$, which implies re-scaling the image to its original image (ground truth), interpreting an input LR image I_x as $\tilde{x}$.The DAE would carry out the denoising process on D, generating this way an image $\hat{I}_y$, which is the same size as the original image (I_y). Our proposal is intended to apply the principle of DAEs by using a completely convolutional architecture, giving place to a CAE. The fundamental process in DAEs or CAEs is somehow different from the one proposed in this work, since the input I_x to the encoder is a lower dimensionality version of the output $\hat{I}_y$, where the error is measured as the difference between the generated output, with

respect to the ground truth image I_y. Equation (14) exemplifies this with MSE.

$$L(I_y, \hat{I}_y) = \frac{1}{2}||I_y - \hat{I}_y||_2^2, \tag{14}$$

where $\hat{I}_y$ is the result of the SR model F, in this case, a CAE $F = g(f(\hat{I}_y))$, where $f(x)$ is the encoder, $g(x)$ is the decoder, and x is the LR input ($\tilde{x}$ in the traditional DAEs). Hence, the complete SR operation can be defined as follows:

$$\hat{I}_y^2 = g(f(I_x)). \tag{15}$$

Based on this, the training process of the CAE proposed for the SR tasks will consist of minimizing the difference between images I_y and $\hat{I}_y$, being able to measure this difference by means of any of the cost functions previously described.

B. Datasets

Nowadays, there is a variety of datasets for SR, most of which contain HR images. Also, the images in these datasets contain objects from different classes, for example, animals, buildings, houses, cars, etc. Some of the most popular datasets are BSDS 300, BSDS 500, DIV2K, General-100, Manga109, Flickr2K, PIRM, among others.

For this work in SR, the datasets DIV2K (published in [1]) and Flickr2K (where the images were taken from the social network Flickr) were fused. The DIV2K dataset contain 1,000 images with a 2K resolution, while the Flickr2K dataset contains 2,550 images with a 2K resolution. The fusion of these two datasets was named DF2K and contains a total of 3,550 HR images. Figure 1 shows some examples of the images contained in the training, validation and test sets. in DF2K. In Fig. 2 we present an example from DIV2K, for both the LR (left) and the HR (right) images.

Although the sufix sK in the names of the two datasets used and described previously, in fact not all the images in them are the same size. To standardize the images to the same size we performed a trimming process, which consisted of applying a trimming at the center of the image, with dimensions of 600×600 pixels. The resulting image is considered as the ground truth I_y. Also, the ground truth images went through a downsampling process $\downarrow s$ to 100 pixels using the traditional method Lanczos. The resulting images of this process $\downarrow s$ are considered as the training images I_x for the proposed architectures, which are described in the following section.

V. Proposed AEs

The SR problem is an open one, so there is not a single way to solve it sine there are, in fact, different approaches that can be taken as a starting point. Considering this, we present several alternatives of DL approaches that are explored in this work, along with the reasoning behind these proposals.

A. Reference Architecture

Although the main goal of the architecture is to develop an AE capable of solving SR tasks, we always had in mind that this AE should be lightweight. The limitations in terms of the hardware that was available for the development of this work encouraged us to develop lightweight architectures (mainly in terms of memory used).

Based on the above mentioned, we made the decision of starting with a simple AE, consisting of a few layers, which could give us enough margin for enlargement or reduction (in terms of the capability of the architecture) for the experiments to be robust enough. The reference architecture is taken from [9], where the authors propose a simple CAE for MNIST. The architecture can be visualized in Fig. 3.

B. Base Architecture

Although the architecture mentioned in Section V-A is taken as reference for developing the intuiton about CAEs, our base architecture, which we call Super-Resolution Convolutional Autoenconder (SRCAE), does only inherit its general structure, i.e., the amount of layers in the encoder and the amount of layers in the decoder. SRCAE eliminates the internal layers that are completely connected in the base architecture. Besides, the dimensions of the output in SRCAE are, approximately, six times higher than the input dimensions.

Figure 3 will be used as the main resource to distinguish the main differences between the reference architecture and our SRCAE. Figure 4 shows the base architecture for this work, which is an adaption of the architecture in Fig. 3, but adding the capacity of scaling up the output size up to six times the input size; besides, the dense intermediate layers are eliminated with the purpose of building a completely convolutional AE.

The main differences between the reference architecture in Fig. 3 and our base architecture in Fig. 4 are listed next:

- The dense layers in the middle of the autoencoder are eliminated with the purpose of creating a completely convolutional architecture.
- The enconder in SRCAE has three layers, while the decoder has two of them; on the contrary, in the reference architecture there were three layers in both the encoder and the decoder, without considering the dense layers.
- The input and output dimensions are different in SRCAE since the output dimension is six times the input dimension (the sampling operator is the transposed convolution).

It is worth mentioning that, although Fig. 4 shows the base architecture, the proposals designed, trained and evaluated in this work can vary in one or more of the following components:

- Type of framework.
- Amount of input and output channels.
- Cost function.
- Use (or not) of batch normalization.
- Depth and characteristics of the encoder.

Fig. 1. Sample images from the DF2K dataset.

Fig. 2. Training sample from DIV2K: (left) LR image; (right) HR image

- Depth and characteristics of the decoder.
- Amount of channels in each layer.

C. RGB Super-Resolution Convolutional Autoencoder (RGB-SRCAE)

Taking as reference the architecture proposed in [18], we designed a novel base architecture, suggesting to eliminate the *skip-connections*. This proposal, which we call RGBSRCAE, is an AE that learns to reproduce the same information received as its input. For so doing, it compresses the input to a hidden variable space, and subsequently it reconstructs the output from the acquired information. RGBSRCAE works with color images in the three channes: Red, Green and Blue (RGB), for both the input I_x and the output I_x of model F.

A classical AE consists of two parts: i) an encoder, which compresses input data with the purpose of extracting the most important features in them, making use of the convolution for this; and ii) a decoder, which reconstructs the input from the features extracted in the encoding process, making use of the transposed convolution to reconstruct the images.

It is worth mentioning that, as opposed to a classical AE, RGBSRCAE does not code the input data; instead, it just makes use of the decoding, with which it increases the image size trying to improve its quality (see Fig. 5).

The encoding layer increases the amount of channels in the image to 16, while keeping the same dimensions from the input. The dimensions of the image are given by $Conv1 = 100$. The first decoding layer only increases the dimensions to 197×197 without modifying the channels. With the help of the second decoding layer, the number of channels in the image are reduced to 8, also increasing the image size to 587×587. In the last decoding layer, image dimension is reduced to the desired size, while the number of channels is also reduced to the ones in the input image. The dimensions of the image are given by $Deconv3 = 585$.

D. Gray-Scale Super-Resolution Convolutional Autoencoder (GSSRCAE)

As opposed to SRCAE, the GSSRCAE architecture only has a depth of 1 in the input filters and output layers (see Fig. 6). The variation between SRCAE and GSSRCAE exists only in the input and the output layers, being 3 for SRCAE (RGB color images) and one for GSSRCAE (gray-scale images).

E. Deeper Super-resolution Convolutional Autoencoder (DSRCAE)

To be clear, both SRCAE and GSSRCAE can be considered to be incomplete architectures if they are regarded as AEs, because there is not any data encoding process involved in them. Thinking of this, deeper architectures were proposed where encoding does take place (see Fig. 7).

The proposed DSRCAE architectures use one or more convolution layers, which reduce the size of the input I_x and increase the number of channels, while the transposed convolution layers reduce the number of channels and generate a larger size image.

F. Interpolation-based Super-resolution Convolutional Autoencoder (IbSRCAE)

Based on the idea behind SRCNN [4], we thought of an architecture that uses traditional methods for pre-processing the image before its being input to the network. Specifically, the proposed architecture, IbSRCAE, uses bicubic interpolation for re-scaling the input images (see Fig. 8).

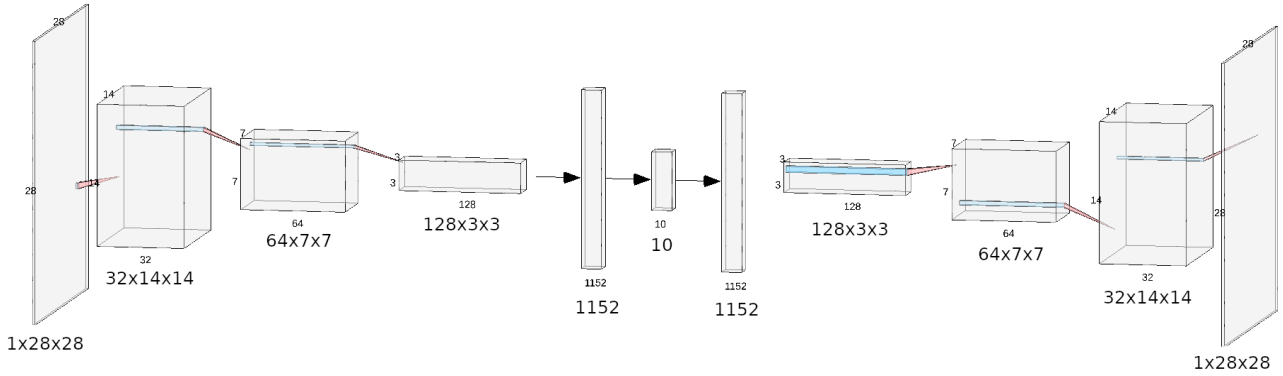


Fig. 3. Structure of the Convolutional Autoencoder used for the MNIST dataset, proposed by X. Guo et al.

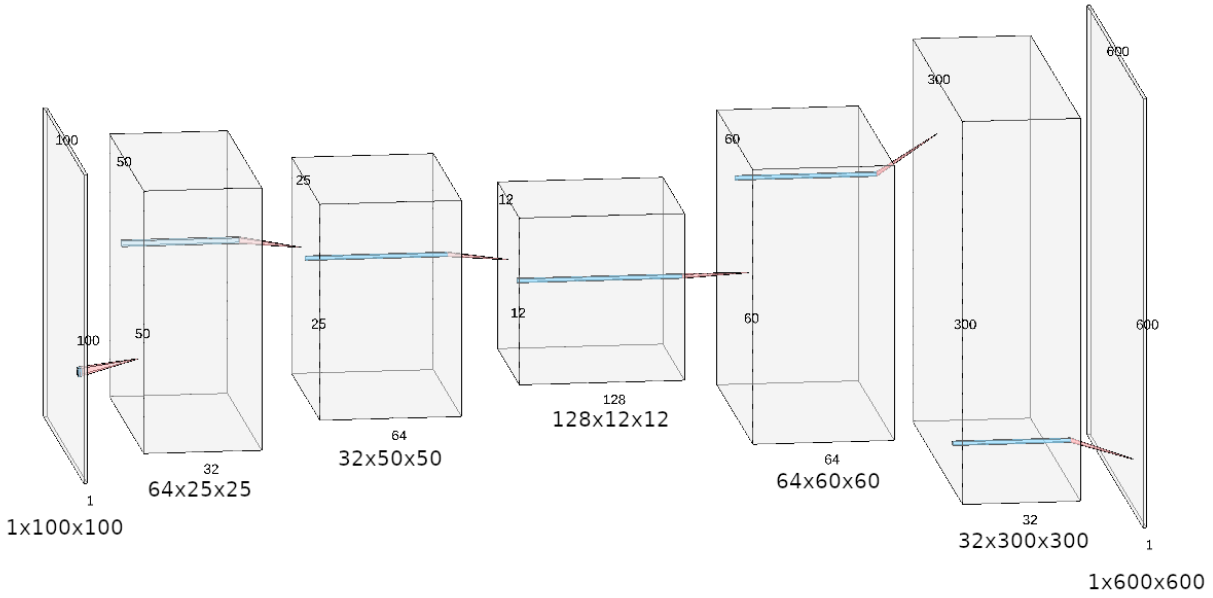


Fig. 4. Base architecture proposed in this work: SRCAE.

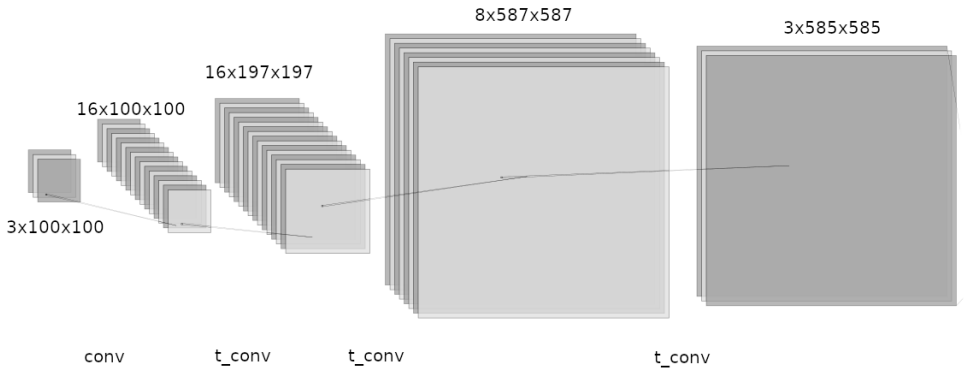


Fig. 5. Proposed RGBSRCAE architecture.

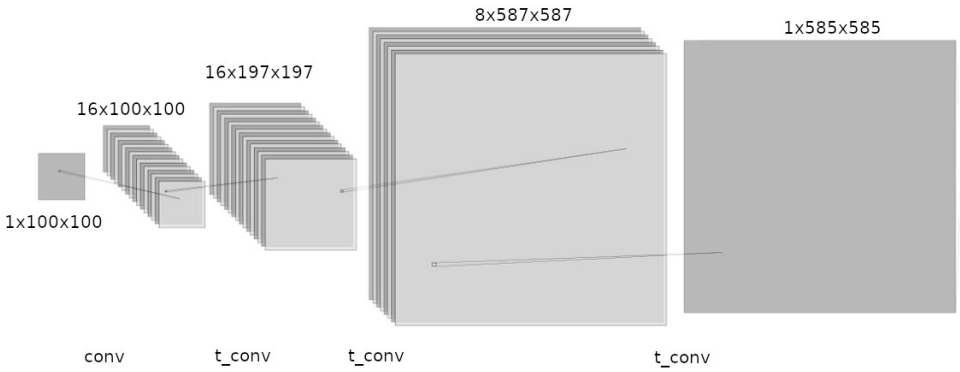


Fig. 6. Proposed GSSRCAE architecture.

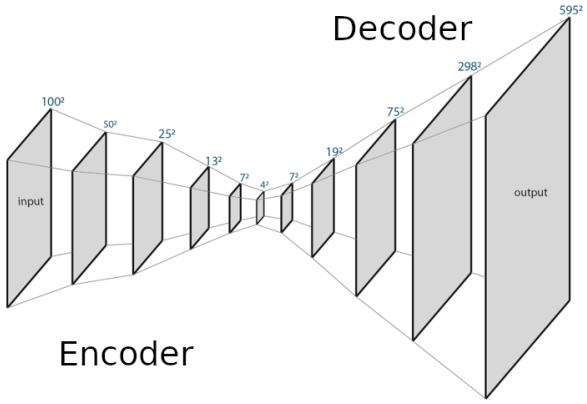


Fig. 7. Proposed DSRCAE architecture.

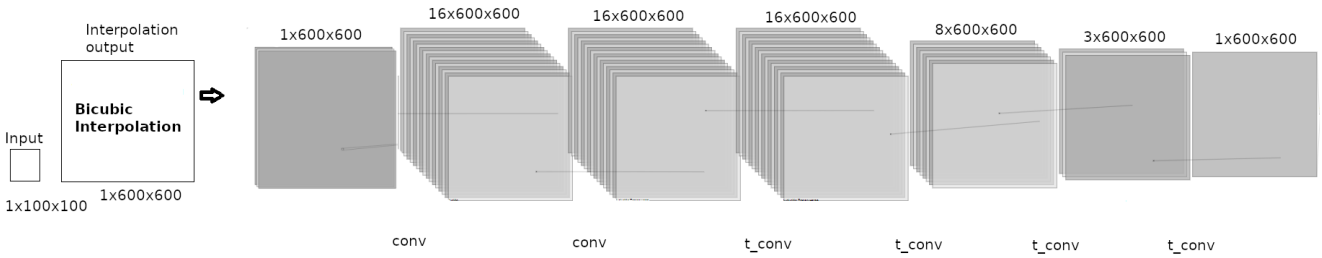


Fig. 8. Proposed IbSRCAE architecture.

Thus, IbSRCAE re-scales image I_x to the output size, and it is then that the network is used to improve and fine-tune the re-scaled image.

G. Larger Super-resolution Convolutional Autoencoder (LSRCAE)

As compared to other networks, SRCAE is relatively small. Considering this fact, we also considered the possibility that better results might be obtained by increasing both the number of transposed convolution layers, and the number of channels in the network, trying to extract more features from the images. The diagram corresponding to this architecture can be seen in Fig. 9.

There are different ways to increase the number of transposed convolution layers without exceeding the maximum output image size. The only requirement for LSRCAE is for it to have more layers and maybe more layers than GSSRCAE. LSRCAE explores the belief that more layers in a network are capable of extracting more features, and consequently to generate better results.

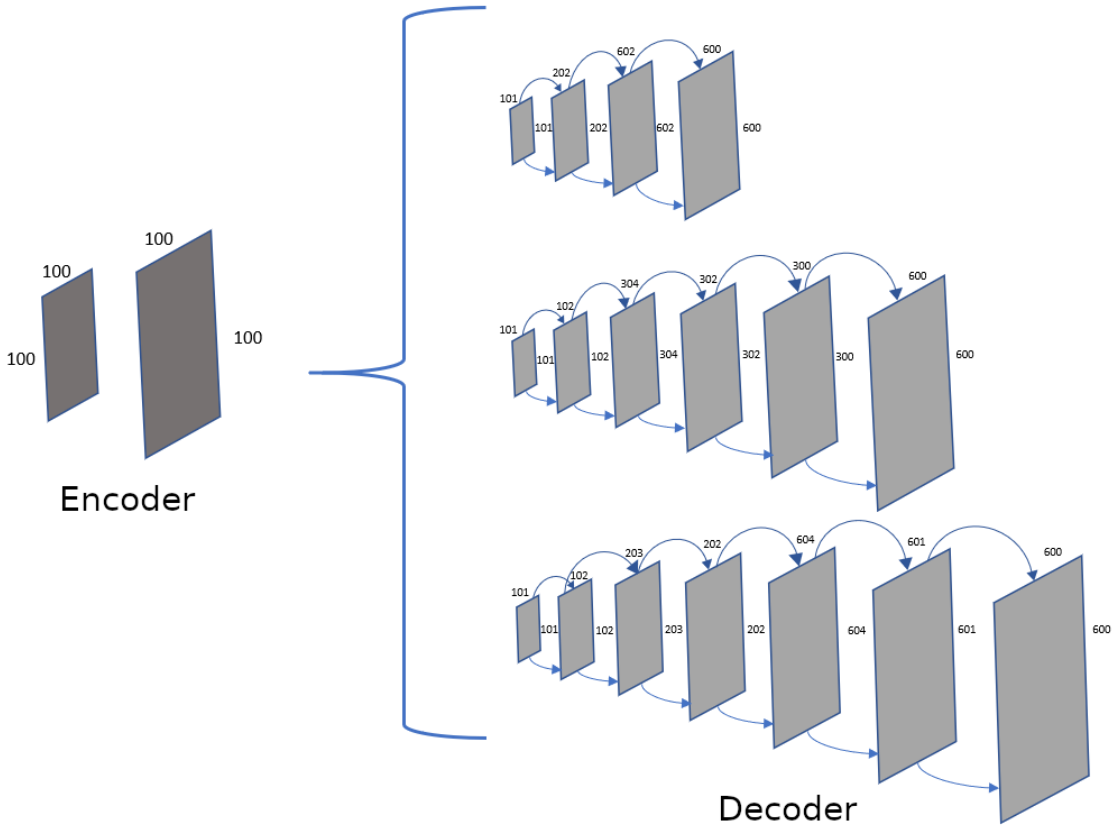


Fig. 9. Proposed LSRCAE architecture.

H. Non-blur Super-resolution Convolutional Autoencoder (NbSRCAE)

The proposed NbSRCAE is inspired by the intuition that a "normal" figure can be blurred by using a single filter; if that were the case, it would be possible to find an inverse form of such a filter, capable of removing somehow this blur. In principle, the problem with the base architecture is that the output image is the result of a combination of a series of filters and not of the application of a single one (for example, in GSSRCAE, four filters are applied) which operate o the input image and generate the output image, which complicates the implementation of a single filter to reverse the resulting blur with SRCAE and GSRCAE. Despite this, we decided to propose the architecture shown in Fig. 10.

The operation of this architecture can be understood from left to right, and the steps it follows are: 1) enter the 100x100 pixel input image I_n, 2) re-scale the image I_n using the GSSRCAE architecture, 3) the resulting image, O_n, will be blurred, 4) apply $f(O_n, k)$ according to the decision made: k can be a learned or a fixed kernel.

VI. Results

In this section, we describe in detail the experimentation, which is carried out in the following way:

1) The dataset is created by fusing DIV2K and Flickr2K, producing the DF2K dataset.
2) The dataset is randomized.
3) The training (80%, with a total of 2,840 images), validation and test (10% for both, with 355 images each) subsets are obtained.
4) The I_y images in all of the subsets are scaled to a $\frac{x}{6}$ factor, using Lanczos re-sampling as $\downarrow s$, to obtain the I_x images.
5) Each image I_x is cropped at its center, with a size of 100×100 pixels. A similar process is applied to each image I_y, with the difference that they are cropped to a size of 600×600 pixels at their center (the output size is six times the input).
6) The I_x images are scaled to a $6\times$ factor, using each of the traditional methods.
7) The outputs provided by the traditional methods are evaluated using MSE, PSNR, and SSIM measures, obtaining the baseline, whose performance is intended to be outperformed by the proposed architectures (F models).
8) The F models are proposed, implemented, and trained. They are capable of receiving an LR image I_x and scaling it to a $6\times$ factor.
9) All of the I_x images in the tet set are input to the F

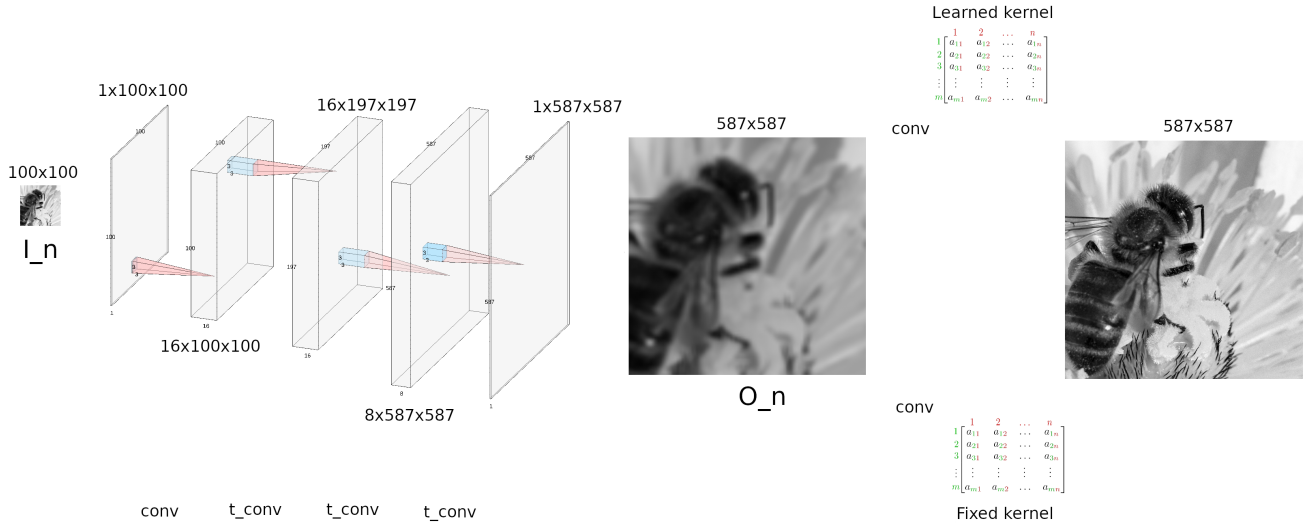


Fig. 10. Proposed NbSRCAE architecture.

model, so that the HR version $\hat{I}_y$ of the whole test set is generated.

10) The outputs of each F model are evaluated using the same measures described in step 7, so that it is possible to compare them to the baseline.
11) Results with traditional methods are compared to results with the methods proposed in this work, based on DL.
12) If the goal of the project is not reached, we repeat from step 8. As expressed in this step, each one of the architectures proposed in this work, and presented in Section V, is inspired by the literature review and the analysis of the results obtained by the previous architectures.

Figure 11 shows a block diagram about the process of generation and partition of our main datasets (including evaluation of the traditional methods). On the other hand, Figure 12 illustrates the process of training, evaluation and comparison against the baseline of each model.

Hence, the experiments herein presented compare the six proposed AE architectures to traditional scaling methods (in this work, we consider Lanczos re-sampling, bicubic interpolation, bilinear interpolation, and nearest neighbors, as traditional methods). We present as well qualitative and quantitative analyses for each architecture and their variations. Performance evaluation for each of the baseline methods was carried out using the test subset, containing 355 images. The results of these evaluations can be seen in Table I, where the first column corresponds to the name of the scaling method, while the next columns correspond to each evaluation measure (MSE, PSNR, and SSIM).

As it can be noticed, the baseline method with the lowest performance in the three measures was Nearest neighbors, while Lanczos resampling is the one that obtained the best results in the three measures. The implementation of Nearest Neighbors requires the least processing time, while Lanczos re-sampling is the one that requires the highest processing time.

TABLE I
EVALUATION OF THE BASELINE METHODS USING THE TEST SET.

Method	MSE	PSNR	SSIM
Lanczos re-samplinc	589.51	22.54	0.586
Bicubic interpolation	590.51	22.49	0.582
Bilinear interpolation	598.44	22.31	0.570
Nearest neighbors	692.97	21.41	0.529

A. *Qualitative analysis*

Considering all the tested variations of the proposed architectures presented in Section V, we implemented around 50 architectures, which were trained and evaluated, with a training time between 25 and 180 minutes (around 100 total training time for all the implementations).

Figure 13 shows an example of the qualitative (visual) comparison for the 20 best results using MSE, PSNR, and SSIM, where the images with the best results are shown for the three measures. It is clear, however, that a numeric result does not always guarantee clear images, as it is shown in image GSSRCAE_DIV2K_PerceptualLoss_50e (second row, second column), which is darker as compared to the others.

Figure 14 shows the qualitative comparison of the best result with our methods (IbSRCAE_L1_7Layers_50e), among all the ones shown in Fig. 13. This comparison is carried out with respect to the baseline methods, where it can be observed that there is an apparent improvement from our method as compared to the best baseline method (Lanczos). The sample image was processed in grayscale.

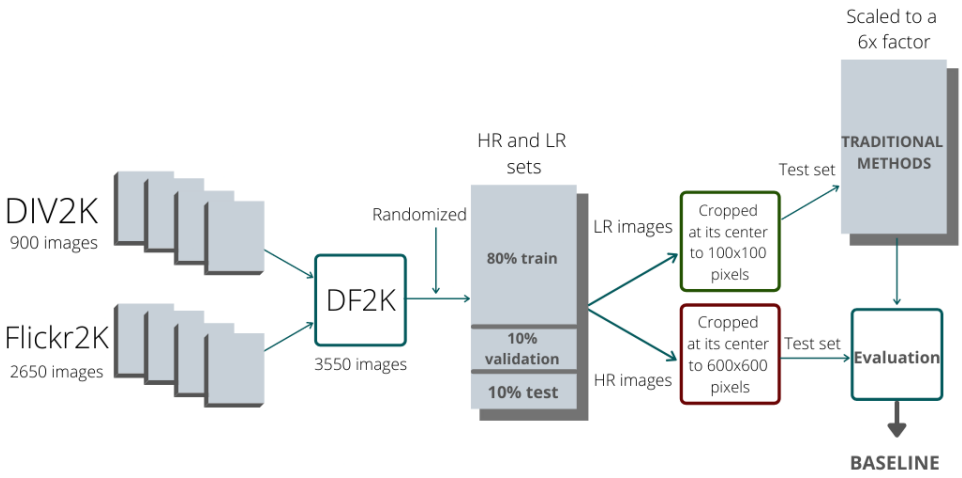


Fig. 11. Building blocks of the baseline .

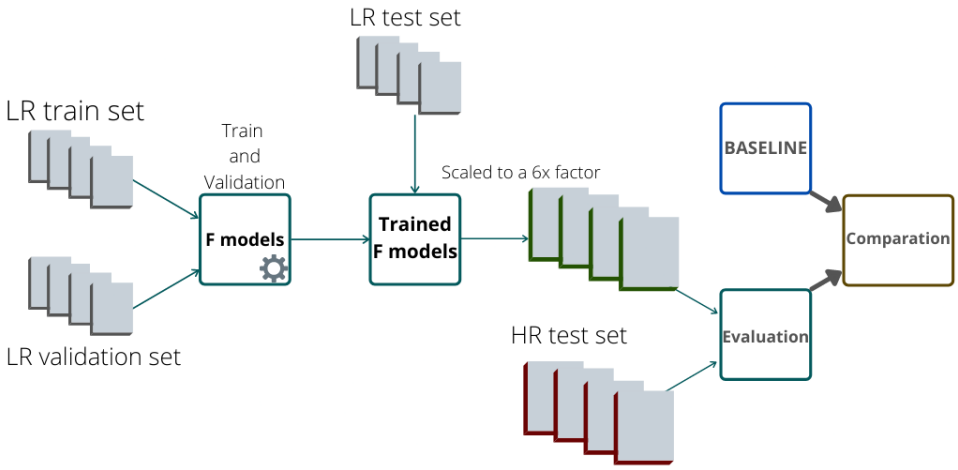


Fig. 12. Process to train and evaluate the F models.

B. Quantitative analysis

Table II presents the quantitative analysis, ordering results from the highest to the lowest with respect to SSIM, where the first column corresponds to the name of the architecture; the second and third rows describe the name of the training and test set used; the fourth column refers to the number of training epochs used; the fifth column describes the number of parameters in the architecture, i.e., the amount of trained weights; the sixth column shows which cost function was used for determining the error between the estimated value and the real value; finally, the last three columns show the scores obtained in each of the measures (MSE, SSIM, and PSNR) used to evaluate each architecture.

Figure 15 plots the scores obtained by the 10 top architectures presented in Table II.

VII. Discussion

A. About the results

Three of the proposed architectures surpassed the baseline method Nearest Neighbors, all of them belonging to the *Interpolation-based SR* category. IBSRCAE 7 Layers L1 is the best one of the proposed architecture, with an MSE of 447.31, an SSIM of 0.623, and a PSNR of 23.181. The structure of this architecture consists of 7 layers, where one of them corresponds to the encoder, while the 6 remaining layers correspond to the decoder. In this variation of the architecture, training consisted of 50 epochs, with a total of 8,527 trained weights, using as its cost function the $L1$ norm, using set DF2K for training and tests.

Another variation of IBSRCAE 7 Layers MSE is the second best architecture, with an MSE score of 454.03, an SSIM of 0.613, and a PSNR of 23.028. This variation had the same structure as before, with the exception that its cost function was MSE.

The third architecture capable of surpassing Nearest Neighbors is IBSRCAE 4 Layers L1. It must be mentioned that this is the architecture with the least number of layers, which means a lower computational cost (in the training stage) with respect to the other alternatives. Its structure is made up of four layers, one of them corresponding to the encoder, while

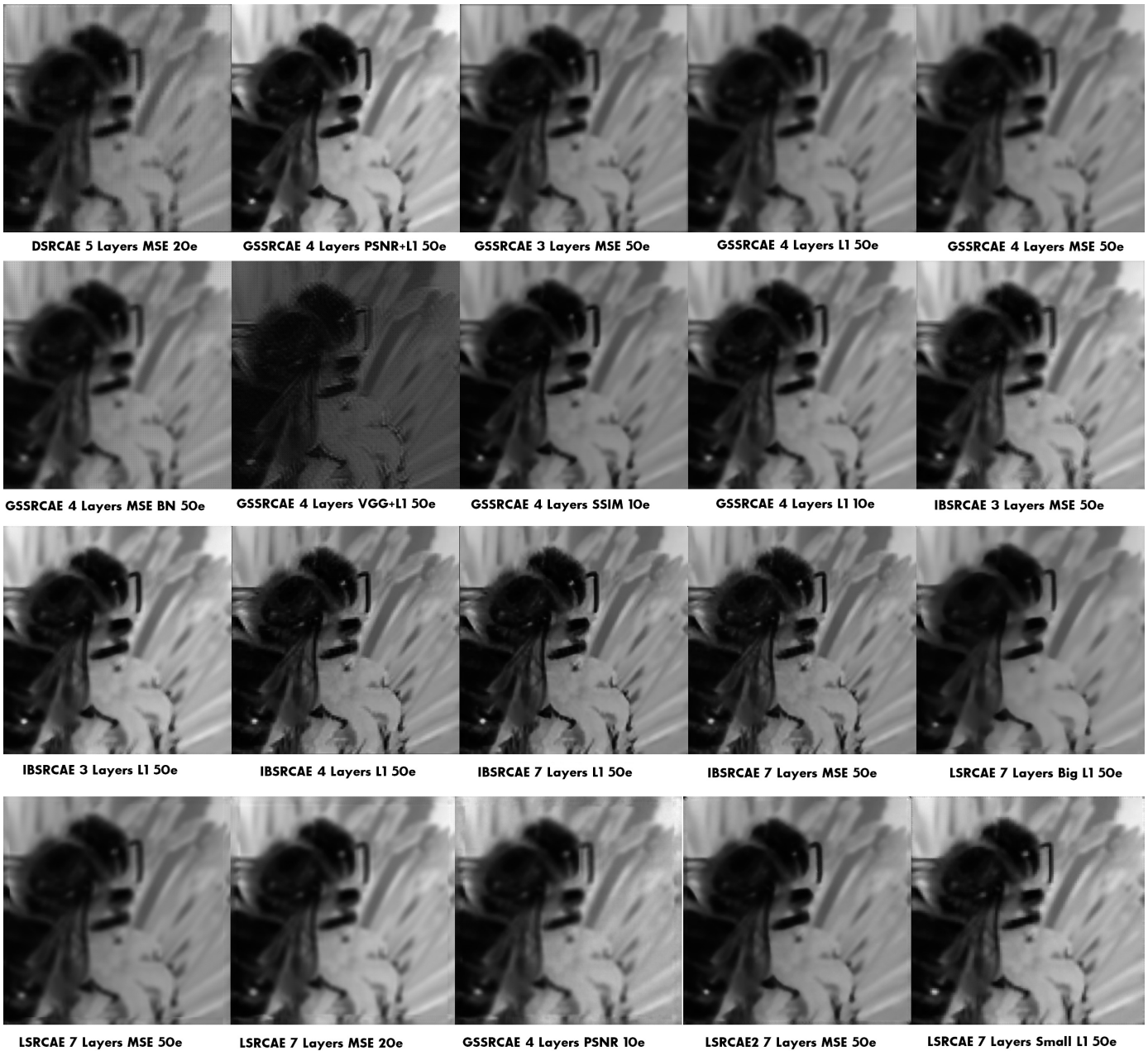


Fig. 13. Comparison of the 20 best results using MSE, PSNR, and SSIM.

Fig. 14. Visual comparison of our best result with the results obtained by the baseline methods. The first four images correspond to Nearest Neighbour, Bilinear interpolation, Bicubic interpolation, and Lanczos re-sampling, respectively; the fifth column corresponds to our best result; and, finally, the sixth image corresponds to the original HR image.

TABLE II
EVALUATION OF THE IMPLEMENTED ARCHITECTURES, ORDERED FROM THE LOWEST TO THE HIGHEST WITH RESPECT TO SSIM.

Architecture	Training data	Test data	# Epochs	# Parameters	Loss function	MSE	SSIM	PSNR
Lanczos Resampling	*	DF2K Test	*	*	*	359.736	0.646	24.335
IBSRCAE 7 Layers L1 50e	DF2K Train	DF2K Test	50	8527	L1	447.313	0.623	23.181
IBSRCAE 7 Layers MSE 50e	DF2K Train	DF2K Test	50	8527	MSE	454.030	0.613	23.028
IBSRCAE 4 Layers L1 50e	DF2K Train	DF2K Test	50	3096	L1	495.911	0.611	22.386
Nearest Neighbors	*	DF2K Test	*	*	*	464.234	0.594	22.839
LSRCAE 7 Layers Big L1 50e	DF2K Train	DF2K Test	50	47617	L1	932.337	0.535	19.594
LSRCAE 7 Layers MSE 50e	DF2K Train	DF2K Test	50	710689	MSE	881.678	0.534	19.847
GSSRCAE 4 Layers SSIM 10e	DF2K Train	DF2K Test	10	3751	SSIM	1047.100	0.533	18.821
GSSRCAE 4 Layers PSNR 10e	DF2K Train	DF2K Test	10	3751	PSNR	926.510	0.531	19.569
GSSRCAE 4 Layers L1 10e	DF2K Train	DF2K Test	10	3751	L1	946.570	0.530	19.480
GSSRCAE 4 Layers L1 50e	DF2K Train	DF2K Test	50	3751	L1	961.410	0.529	19.414
GSSRCAE 4 Layers 20e	DF2K Train	DF2K Test	20	9569	L1	981.104	0.528	19.311
LSRCAE2 7 Layers MSE 50e	DF2K Train	DF2K Test	50	36481	MSE	1256.460	0.524	18.071
LSRCAE 7 Layers Small L1 50e	DF2K Train	DF2K Test	50	41689	L1	1240.907	0.523	18.021
GSSRCAE 3 Layers MSE 50e	DF2K Train	DF2K Test	50	1393	MSE	962.580	0.523	19.323
GSSRCAE 4 Layers Data Augment 2 L1 10e	DIV2K Train	DF2K Test	10	3751	L1	953.980	0.521	19.217
LSRCAE 7 Layers MSE 20e	DF2K Train	DF2K Test	20	41689	MSE	1195.610	0.521	18.192
GSSRCAE 4 Layers MSE BN 50e	DF2K Train	DF2K Test	50	3751	MSE	1251.700	0.521	17.974
GSSRCAE 4 Layers PSNR+L1 50e	DF2K Train	DF2K Test	50	3751	PSNR + L1	1246.700	0.520	17.989
GSSRCAE 4 Layers Data Augment L1 10e	DIV2K Train	DF2K Test	10	3751	L1	963.770	0.520	19.345
LSRCAE2 7 Layers PSNR+L1 50e	DF2K Train	DF2K Test	50	36481	PSNR + L1	1243.120	0.519	17.990
IBSRCAE 3 Layers L1 50e	DF2K Train	DF2K Test	50	3713	L1	1261.130	0.519	17.930
LSRCAE 7 Layers MSE 50e	DF2K Train	DF2K Test	50	41689	MSE	1190.550	0.518	18.202
IBSRCAE 3 Layers L1 50e	DF2K Train	DF2K Test	50	3713	L1	1245.190	0.518	18.003
GSSRCAE 3 Layers L1 50e	DF2K Train	DF2K Test	50	1393	L1	989.641	0.517	19.239
IBSRCAE 4 Layers L1 50e	DF2K Train	DF2K Test	50	2793	L1	1248.091	0.516	17.981
LSRCAE 3 Layers L1 50e	DF2K Train	DF2K Test	50	2977	L1	1255.008	0.515	17.947
LSRCAE 3 Layers PSNR+L1 50e	DF2K Train	DF2K Test	50	2977	PSNR + L1	1259.250	0.515	17.941
IBSRCAE 3 Layers MSE 50e	DF2K Train	DF2K Test	50	3713	MSE	1209.310	0.514	18.103
GSSRCAE 4 Layers SSIM 50e	DF2K Train	DF2K Test	50	3751	SSIM	1392.600	0.510	17.489
LSRCAE 2 Layers L1 50e	DF2K Train	DF2K Test	50	753	L1	1265.850	0.509	17.905
LSRCAE 2 Layers PSNR+L1 50e	DF2K Train	DF2K Test	50	753	PSNR + L1	1269.670	0.509	17.895
IBSRCAE 7 Layers L1 50e	DF2K Train	DF2K Test	50	8527	L1	1265.870	0.508	17.925
DSRCAE 5 Layers MSE 20e	DF2K Train	DF2K Test	20	7657	MSE	958.123	0.504	18.406
Wider GSSRCAE 4 Layers L1 10e	DIV2K Train	DF2K Test	10	9569	L1	1004.900	0.503	19.030
GSSRCAE 4 Layers L1 10e	DIV2K Train	DF2K Test	10	3751	L1	1018.600	0.503	18.925
GSSRCAE 4 Layers PSNR 10e	DIV2K Train	DF2K Test	10	3751	PSNR	1056.700	0.499	18.879
GSSRCAE 4 Layers Data Augment 3 L1 10e	DIV2K Train	DF2K Test	10	3751	L1	1156.300	0.493	18.410
GSSRCAE 4 Layers SSIM 50e	DIV2K Train	DF2K Test	10	3751	SSIM	1394.800	0.472	17.556
SRCAE 4 Layers MSE 50e	DF2K Train	DF2K Test	50	3200	MSE	1700.000	0.440	16.850
GSSRCAE 4 Layers MSE 50e	DF2K Train	DF2K Test	50	3751	MSE	1020.000	0.430	19.031
RGBSRCAE 4 Layers PSNR 50e	DF2K Train	DF2K Test	50	3751	PSNR	1083.900	0.426	18.829
SRCAE 4 Layers MSE 50e	DF2K Train	DF2K Test	50	3200	MSE	1429.680	0.410	17.075
GSSRCAE 4 Layers VGG+L1 10e	DIV2K Train	DF2K Test	10	3751	VGG + L1	4802.360	0.389	11.705
GSSRCAE 4 Layers VGG+L1 50e	DIV2K Train	DF2K Test	50	3751	VGG + L1	3402.600	0.387	13.120
RGBSRCAE 4 Layers SSIM 50e	DF2K Train	DF2K Test	50	3751	SSIM	5951.800	0.353	11.664
NEWSRCAE 6 Layers MSE 50e	DF2K Train	DF2K Test	50	8231	MSE	1410.320	0.340	16.850
NEWSRCAE 6 Layers MSE 50e	DF2K Train	DF2K Test	50	8231	MSE	2950.000	0.321	14.093
DSRCAE 10 Layers MSE 50e	DF2K Train	DF2K Test	50	12369	MSE	2729.475	0.318	13.823
RGBSRCAE 4 Layers L1+VGG 50e	DF2K Train	DF2K Test	50	3751	L1 + VGG	1186.800	0.316	11.210
DSRCAE 10 Layers MSE 50e	DIV2K Train	DF2K Test	50	12369	MSE	3036.662	0.275	13.629

the three remaining correspond to the decoder. Its training and evaluation was carried out by 50 epochs, and as norm $L1$ was chosen as the cost function. Training and test set were obtained from DF2K. As we previously said, the computational cost of this architecture was reduced as compared to the first and second architectures (the number of trained weights was 3,096, which represents a reduction of 64%).

One of the main problems found during the development of this work, lies in the output image quality reduction when the decoding process is intensified. (i.e., the encoder depth is increased). One of the predominating problems was the appearing of a light *blur* in the output image, which, after some experiments, we attribute to the encoder. To prove this, we present Table III, showing a comparison among the results obtained by the trained architectures under the following conditions:

- Same training and test sets.
- Trained during 50 epochs with MSE as its cost function.
- Using three input and output channels.

The main variation among these architectures is in the encoder depth. These results are shown in two rows, where the first of them shows results when we train using an encoder, while the second one shows results when the encoder is suppressed. Bold values show the best scores for each topology. It is

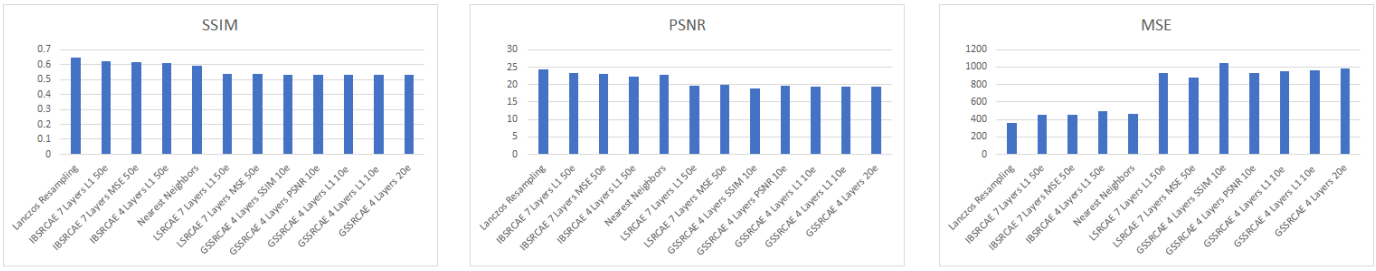


Fig. 15. Plot of the top 10 architectures for SSIM (left), PSNR (center), and MSE (right).

easy to notice that the architectures without encoder obtain much better results than the ones using encoder for all of the measures.

Figure 16 exemplifies the results for the SR task when the encoding process is intensified, which corresponds to the architectures with a value different from 0. in the encoder layers (second column of Table III). As opposed to this, Fig. 17 shows visually the results of these architectures when no encoder is used (value 0 in the second column of Table III).

B. About frameworks

For this SR work, we used pre-sampling as an alternative, since the available computational resources were not enough for a fair comparison with state-of-the-art methods. We considered that using pre-sampling techniques would lead the architectures to obtain results that could compare or even surpass the ones provided by traditional methods, so we proceeded to develop and implement SRCAE in the pre-sampling framework. As we described in Section VI, the architectures that reached comparable results are into the pre-sampling framework.

On the other hand, the SR progressive sampling framework offered, as its main advantage, the capability of visualizing and analyzing the output of each decoder layer, which allowed to progressively track the scaling process. By using progressive sampling in the decoder, the model was able to learn new parameters before increasing dimensions, increasing the architecture's own capability to adjust to the problem. However, the results obtained by the developed architectures in this framework are still far from the baseline methods.

The post-sampling framework was used in the variations mainly to compensate the output size; for example, if a given version of SRCAE reached an output size of 585×585 pixels, then a sampling function, based on interpolation, was used to increase the dimensions of the output image, so that it coincided with the size of the *ground truth*. There were even some cases where this sampling was inversely; for example, if the output of the architecture was 625×625 pixels, we used a sampling function to reduce the dimensions to 600×600 pixels. The biggest benefit obtained from this framework is in the freedom that it provide when experimenting with several architectures, reducing the responsibility of the decoder regarding the dimensions of the output image. The results obtained in this framework did not improve significantly as compared to progressive sampling.

C. About transposed convolution as scaling operator

Throughout the experiments we proved that the use of transposed convolution can lead to unequal superposition in some parts of the image, which can cause artifacts in the output image. In the implemented architectures, two or more transposed convolution layers are used to sample the image or to refine the outputs of some previous layers, and, in some cases, they were also used for reducing image resolution. For reducing the amount of artifacts in the output (similar to the texture in a chessboard) of the layers of the transposed convolution, a trick is used that consists of making the kernel size divisible by the convolution stride [19]. Nonetheless, this did not improve in a meaningful way the output quality.

Some of the architectures designed to suppress output image blur and artifacts, follow approaches like pre-sampling for preserving the spatial dimensions of the images, so that transposed convolution is avoided. However, we consider transposed convolution is not the cause of blur in the output images (see Section VII-A).

D. About cost functions

In literature, diverse learning strategies can be found, from which many of them have been fundamental in the success of state-of-the-art architectures. In this section, we discuss the obtained results and the main difference among the architectures optimizing different cost functions.

1) Pixelwise cost functions (Pixel Loss): Pixelwise cost functions measure the difference in pixels between two images, which can limit the quality of the resulting images. Pixelwise cost functions measure the pixelwise difference between two images, which can limit the quality of the resulting images. When measuring the difference between $\hat{I}_y$ and I_y the purpose is that the results are the most similar to I_y when performing a pixelwise comparison, i.e. these cost functions optimize a low level similarity. These cost functions include L1 and L2 norms.

While L1 norm adds up the pixelwise differences between two images, L2 norm adds up the square of these differences. The use of L2 penalizes the most biggest errors, but it is more tolerating to small errors, hence, resulting images tend to have softer textures when L2 norm is used; this difference is

TABLE III

COMPARISON OF THE ARCHITECTURES TRAINED UNDER THE SAME CONDITIONS. THE ONLY VARIATION IS OBSERVED IN THE SECOND COLUMN (INDICATING THE ENCODER DEPTH). THESE ARCHITECTURES WERE TRAINED WITH AND WITHOUT ENCODER, SO THAT WE COULD MEASURE THE ERROR INTRODUCED BY THE ENCODING TO THE OUTPUT IMAGES.

Architecture	# Layers in encoder	# Layers in decoder	MSE	SSIM	PSNR
Lanczos Ressampling	NA	NA	589.59	0.586	22.54
Nearest Neighbors	NA	NA	692.95	0.529	21.41
RGBSRCAE 8L	4	4	2808.0	0.123	13.89
	0	4	**833.96**	**0.476**	**19.40**
RGBSRCAE 6L	3	3	2727.0	0.128	14.01
	0	3	**1024.0**	**0.410**	**18.46**
RGBSRCAE 5L	2	3	1484.0	0.291	16.66
	0	3	**1224.0**	**0.301**	**17.03**

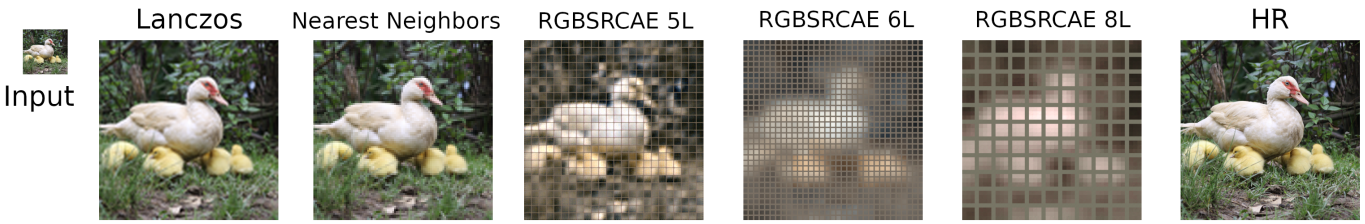


Fig. 16. Visual comparison of results using complete CAE.

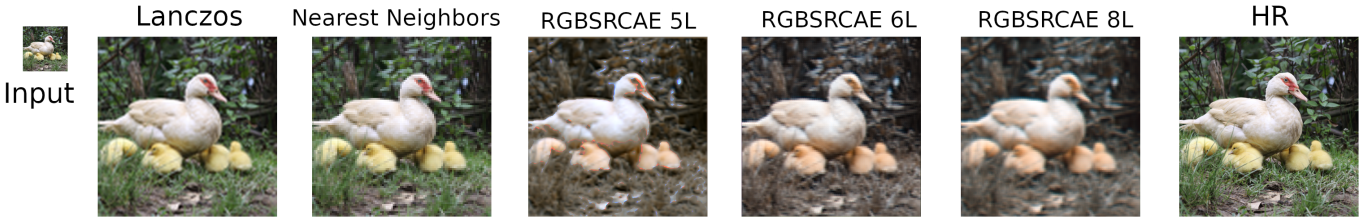


Fig. 17. Visual comparison of results using only convolutional decoder.

noticeable when the output are compared to images generated by architectures that optimize L1 norm (see Fig. 19). L1 showed a better performance and a faster convergence during training (as compared to L2).

Although the results are promising, pixelwise cost functions do not takie into account more complex aspects in an image (such as the structure, texture or contents), hence, the results obtained using them lack details present in image I_y. The above led us to try more complex cost functions which have proved to perform well with this kind of problem [14].

2) Perceptual cost functions (Perceptual Loss): Perceptual cost functions have been widely used since it was discovered that pixelwise cost functions fail at properly measuring visual quality of the reconstructions. Perceptual cost functions, besides measuring better the quality of the reconstruction, are capable of producing more realistic results with higher quality [27].

- SSIM: Since our visual perception seeks to find structures in the images, the use of SSIM seems to be a good alternative. The definition is about terms that are determinant in the representation of an image: luminance, contrast, and structure, are the basis of this cost function. The structure in an image is relevant because there exist interdependencies among pixels, even more when those

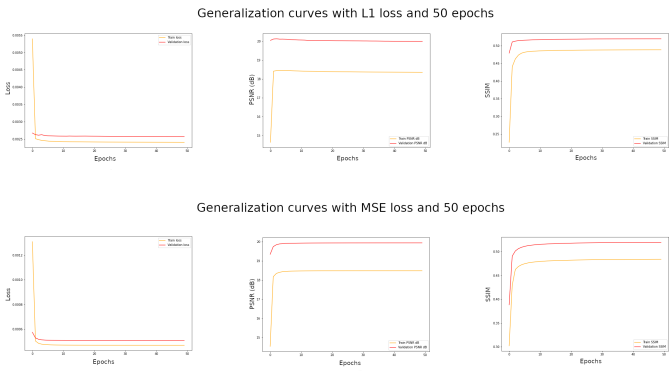
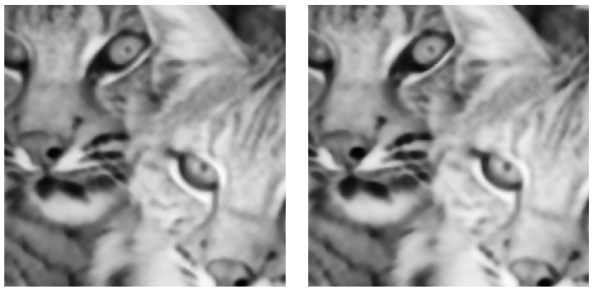


Fig. 18. SRCAE performance during training with different cost functions: (top) the architecture was trained using norm L1; (bottom) the same architecture was trained using MSE. Although both of them seem to be generalizing well, the use of L1 as the cost function generalizes faster.

L1 L2

Fig. 19. Images obtained using the cost functions for pixel loss, it is noticeable that the results with L1 show a reduction in the output blurring.

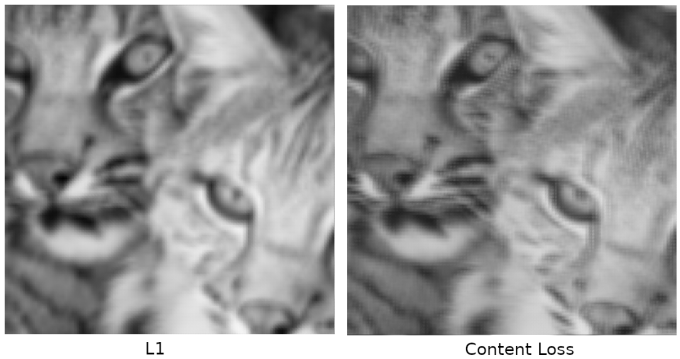
pixels are close. All the interdependencies are useful when representing objects in an image. Performing an optimization on these image parameters should provide better results; however, these results are visually less pleasant (see Fig. 20). The general structure of the images is kept, although more artifacts have appeared, significantly affecting the quality of the image. There is no clear explanation of these results, though we consider these results are a consequence of the lack of capacity of the architecture, which a deeper network might correct, taking advantage in a better way of the benefits of this function.



L1 SSIM

Fig. 20. Resulting images optimizing L1 and SSIM. Although SSIM is a more complex cost function, the obtained results when using it do not surpass the already obtained when using L1.

- PSNR: Although it is not a totally perceptive measure, it generates acceptable results, although most of them do not surpass the ones obtained with L1. Everything indicates that using PSNR we can obtain better results than when using SSIM. In [27], they mention that PSNR usually leads to bad performance in the representation of the reconstruction quality in real-life scenes, where human perception tends to be more important.
  Some alternatives point out at the use of PSNR with pixel loss L1 instead of L2. Experiments were performed using PSNR-L1 and PSNR-L2; results were similar to the ones with direct optimization of L1 or L2. The pixel-level difference, measured by L2 (PSNR) leads to blurred results; with L1, however, results show a reduction in blur.
- Content Loss: Coste basd on contents is obtained by means of the distance that exists between the high-level representations of images I_y and $\hat{I}_y$. According to Wang et al. [27], this kind of cost transfers knowledge from hierarchical features, previously learned by the classification network ϕ. Cost functions based on contents provide images that are perceptually more similar to the objective image, instead of forcing pixels to be strictly similar.
  We consider that the use of this kind of cost function lead to noticeably different results (see Fig. 22). Nonetheless, the scores obtained using the proposed measures did not improve drastically after the analysis of the generalization curves; we believe it can be explained by a lack of capability from the architecture to take whole advantage of the potential of content-based cost functions (see Fig. 23).
  Although we tried to increase the capacity of the architecture, the available hardware was not enough when pre-trained networks were used at the same time as CAEs.



L1 Content Loss

Fig. 22. Results when the L1 function is optimized (left), or when it is a content-level cost function (right). We attribute the variation in the structure of the output image in the right side to a lack of capacity from the architecture.

- Texture Loss: For this kind of cost function, we follow the same intuition as in the content-based functions. We con-

L1 PSNR

Fig. 21. Results whuen using PSNR as the cost function, comparing it vs L1. We can see there are more deficiencies at a structure level than when L1 is used. As it can be seen in the images, blur appears in the resulting image.

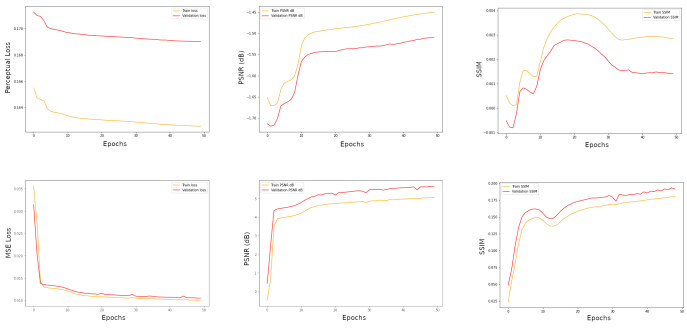


Fig. 23. Performance of two variations of SRCAE during training (50 epochs): (top) the architecture was trained with a perceptual cost function (content loss); (bottom) the same architecture was trained with a pixelwise cost function (MSE). It is clear that the version trained with Content Loss lacks good generalization, which is indicative of a need for increasing the capacity of the architecture.

sidered this kind of cost function based on the assumption that the blurred outputs were caused by a problem with the texture, and, given that texture-based cost functions seek to recreate style (such as colors, textures or contrast) in the input images, we proceeded to develop variations that optimized error based on texture-based functions. Similar to content-based functions, texture-based ones use high-level representations extracted from a pre-trained network, used to capture meaningful features in the high-resolution images [27].

Although the results looked different, the outputs were not necessarily better (neither visually nor perceptually) than those generated by architectures with pixelwise cost functions (see Fig. 24). Because the texture-based cost functions implied too high a memory consumption, as compared to pixelwise cost functions, we discarded them as an option to solve the problem.

Fig. 24. Results when L1 function is optimized (left), or when it is a texture-based cost function (right). We attribute the lack of visual quality to two aspects: the nature of the cost function, and the lack of capacity from the architecture.

E. About the adjustments to the topology

Throughout experimentation, we learned that when kernel size is reduced, the resulting image tends to content textures similar to a chessboard; on the other hand, if the kernel size increased, the resulting images showed an increase in texture blurring. The adjustment to the kernel size in the encoder always seeked balance between both textures (by trial and error), always intending to eliminate the chessboard effect in the outputs, i.e., that the kernel size was divisible by the stride size.

In the case of the encoder depth, to add or remove layers we followed a single principle: if we wanted the architecture to find more abstract representations of the inputs, then the encoder depth was to be increased; however, as we previously showed, the encoder depth modified drastically the visual quality of the outputs. Based on the analysis of previous results, we looked for a balance between encoder depth and the visual quality of the outputs. Commonly, the smaller encoder depth, the outputs were visually more natural.

For the decoder, we achieved a balance in the decoder layers by analyzing the generalization curves, also evaluating the resulting images; concluding that the best results are obtained using between 2 and 4 transposed convolution layers.

VIII. Conclusions

The experimental work we carried out indicated in a subtle way that the AE might not be the best option for solving the SR task. Traditional algorithms perform better in most of the environments, being lighter in terms of computations and memory used. However, it is noteworthy that AEs do not provide an ideal tool to encode images in a latent space.

We have shown experiments with RGB images (three channels), as well as using grayscale images (1 channel), reducing problem complexity and accelerating training time. A big difference appears in the amount of epochs the algorithms required to learn and color images: the RGB version had to train, in average, 10 additional epochs to start adding colors to the scaled images.

Although completely convolutional networks had no problem in processing different-dimension inputs, the result obtained by the same architecture varied considerably when the input image size changed and the results showed that the more pixels in the input image, the better the results obtained.

The architectures were capable of generating images that varied in intensity, contrast, illumination, and definition; however, they all presented some kind of blurring, and this blurring did not disappear. We must highlight these results surpassed the Nearest Neighbors method using the three evaluation measures proposed.

Solving the SR problem is a complex challenge, which by definition is an ill-posed problem, complicating the search for optimal solutions and making it impossible finding unique solutions. Even though the SR problem has been attacked following different approaches, several methods and techniques are still emerging and are capable of improving the existing results. Unfortunately, these approaches require, mostly, a huge amount of resources, which are not always available. The difficulty in the access to this kind of hardware tends to complicate the applicability of certain models in real-life and

derives in the search for computationally lighter alternatives, such as the ones described in this paper.

Each variation proposed was considered a potential solution, exploring advantages and disadvantages in each topology with diverse configurations of the hyperparameters. During the exploration of the design space we carried out several analysis both qualitative and quantitative, which lead to informed decisions on the possible working paths. Although results improved with each iteration, the general evolution of the output images is far from being optimal; however, some alternatives are more promising than others and might offer good indicators about alternative solutions to the SR problem using CAEs. This work represents, as can be seen, a thorough and in-depth exploration of the design space and can be useful for the reader who tries to enrich his/her intuition about the behavior of CAEs in the SR task, as well as for compatible tasks, such as image processing and the like.

Experimental results showed that a component, which is almost as relevant as the architecture is the cost function, which might seem counterintuitive. In this sense, the simplest cost functions reached better results than the most complex ones. The best quality results were obtained using L1 norm as the cost function, with clearer and more detailed results. It has been shown in literature that, commonly, using complex cost functions (such as the perceptual ones) results in more quality reconstructions, which are more pleasant to humans as well; however, our experiments indicate that these functions might require the use of considerably more complex and deeper architectures which, when used in low-capacity topologies, tend to produce unsatisfactory results.

The exploration carried out in this work opens the door to different application possibilities for CAEs in data processing, specially when the problem requires models capable of increasing dimensionality while obtaining more compact and complex representations of the inputs. In this work we focused in data processing using grids; however, convolution is applicable to other kinds of data (e.g., text or signals), hence, the processes hereby described can be used as a guidance for similar design processes.

A. Future work

This research can open the doors to a variety of interesting possibilities. In the future, it can help to explain the design space of AEs for SR, by using more powerful hardware, in a way that deeper and more complex architectures can be designed, without the current memory limitations, which represent a challenge for the experimental development.

Another potential venue to develop consists in using our proposed AEs to find specific representations of SR tasks, for later using the codes generated by the encoder as the input for adversarial generative networks to analyze the quality of the generated outputs. This is similar to a training process in our AEs using and adversarial cost function.

We hope to be working with architectures with variable scaling factors, i.e., not limited to a $6\times$ factor such as in the architectures presented in this work. For this purpose, we suggest the analysis of the design space for architectures with scaling factors of $2\times$, $3\times$, and $4\times$.

ACKNOWLEDGMENT

C.A.H. and I.A.B. are commissioned to their respective institutions through the National Council of Science and Technology (CONACYT) Research Fellows program, by means of projects 278 and 882, respectively (Web page: https://www.conacyt.gob.mx/).

REFERENCES

[1] AGUSTSSON, E., AND TIMOFTE, R. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2017), pp. 126–135.

[2] BIGDELI, S. A., AND ZWICKER, M. Image restoration using autoencoding priors. *arXiv preprint arXiv:1703.09964* (2017).

[3] CHEN, R., QU, Y., LI, C., ZENG, K., XIE, Y., AND LI, C. Single-image super-resolution via joint statistic models-guided deep auto-encoder network. *Neural Computing and Applications 32*, 9 (2020), 4885–4896.

[4] DONG, C., LOY, C. C., HE, K., AND TANG, X. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision* (2014), Springer, pp. 184–199.

[5] DONG, C., LOY, C. C., HE, K., AND TANG, X. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence 38*, 2 (2015), 295–307.

[6] DONG, C., LOY, C. C., AND TANG, X. Accelerating the super-resolution convolutional neural network. In *European conference on computer vision* (2016), Springer, pp. 391–407.

[7] GOODFELLOW, I., BENGIO, Y., COURVILLE, A., AND BENGIO, Y. *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[8] GREENSPAN, H. Super-resolution in medical imaging. *The computer journal 52*, 1 (2009), 43–63.

[9] GUO, X., LIU, X., ZHU, E., AND YIN, J. Deep clustering with convolutional autoencoders. In *International conference on neural information processing* (2017), Springer, pp. 373–382.

[10] HARIS, M., SHAKHNAROVICH, G., AND UKITA, N. Deep back-projection networks for super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 1664–1673.

[11] HUANG, Y., SHAO, L., AND FRANGI, A. F. Simultaneous super-resolution and cross-modality synthesis of 3d medical images using weakly-supervised joint convolutional sparse coding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 6070–6079.

[12] IRANI, M., AND PELEG, S. Improving resolution by image registration. *CVGIP: Graphical models and image processing 53*, 3 (1991), 231–239.

[13] ISAAC, J. S., AND KULKARNI, R. Super resolution techniques for medical image processing. In *2015 International Conference on Technologies for Sustainable Development (ICTSD)* (2015), IEEE, pp. 1–6.

[14] JOHNSON, J., ALAHI, A., AND FEI-FEI, L. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision* (2016), Springer, pp. 694–711.

[15] LAI, W.-S., HUANG, J.-B., AHUJA, N., AND YANG, M.-H. Deep laplacian pyramid networks for fast and accurate super-resolution. In *IEEE Conferene on Computer Vision and Pattern Recognition* (2017).

[16] LEDIG, C., THEIS, L., HUSZÁR, F., CABALLERO, J., CUNNINGHAM, A., ACOSTA, A., AITKEN, A., TEJANI, A., TOTZ, J., WANG, Z., ET AL. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 4681–4690.

[17] LI, Z., YANG, J., LIU, Z., YANG, X., JEON, G., AND WU, W. Feedback network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 3867–3876.

[18] MAO, X.-J., SHEN, C., AND YANG, Y.-B. Image restoration using convolutional auto-encoders with symmetric skip connections. *arXiv preprint arXiv:1606.08921* (2016).

[19] ODENA, A., DUMOULIN, V., AND OLAH, C. Deconvolution and checkerboard artifacts. *Distill* (2016).

[20] RASTI, P., UIBOUPIN, T., ESCALERA, S., AND ANBARJAFARI, G. Convolutional neural network super resolution for face recognition in surveillance monitoring. In *International conference on articulated motion and deformable objects* (2016), Springer, pp. 175–184.

[21] SHAO, Z., WANG, L., WANG, Z., AND DENG, J. Remote sensing image super-resolution using sparse representation and coupled sparse autoencoder. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 12*, 8 (2019), 2663–2674.

[22] SHI, W., CABALLERO, J., HUSZÁR, F., TOTZ, J., AITKEN, A. P., BISHOP, R., RUECKERT, D., AND WANG, Z. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 1874–1883.

[23] WANG, W., REN, C., HE, X., CHEN, H., AND QING, L. Video super-resolution via residual learning. *IEEE Access 6* (2018), 23767–23777.

[24] WANG, X., YU, K., WU, S., GU, J., LIU, Y., DONG, C., QIAO, Y., AND CHANGE LOY, C. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 0–0.

[25] WANG, Y., LIU, Q., ZHOU, H., AND WANG, Y. Learning multi-denoising autoencoding priors for image super-resolution. *Journal of Visual Communication and Image Representation 57* (2018), 152–162.

[26] WANG, Z., BOVIK, A., SHEIKH, H., AND SIMONCELLI, E. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing 13*, 4 (2004), 600–612.

[27] WANG, Z., CHEN, J., AND HOI, S. C. Deep learning for image super-resolution: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[28] YIFAN, W., PERAZZI, F., MCWILLIAMS, B., SORKINE-HORNUNG, A., SORKINE-HORNUNG, O., AND SCHROERS, C. A fully progressive approach to single-image super-resolution. In *CVPR Workshops* (June 2018).

[29] ZENG, K., YU, J., WANG, R., LI, C., AND TAO, D. Coupled deep autoencoder for single image super-resolution. *IEEE transactions on cybernetics 47*, 1 (2015), 27–37.

[30] ZHANG, L., ZHANG, H., SHEN, H., AND LI, P. A super-resolution reconstruction algorithm for surveillance images. *Signal Processing 90*, 3 (2010), 848–859.

[31] ZHANG, M., LIU, Z., AND YU, L. Crnet: Image super-resolution using a convolutional sparse coding inspired network. *arXiv preprint arXiv:1908.01166* (2019).