# Lightweight Multi-Scale Network for Stylized and Controlled Image Restoration

Thibault Durand, Julien Rabin, David Tschumperlé

# Lightweight Multi-Scale Network for Stylized and Controlled Image Restoration*

Thibault Durand[1], Julien Rabin[1], and David Tschumperlé[1]

[1]UNICAEN, ENSICAEN, CNRS, GREYC, Normandie Univ., Caen, France

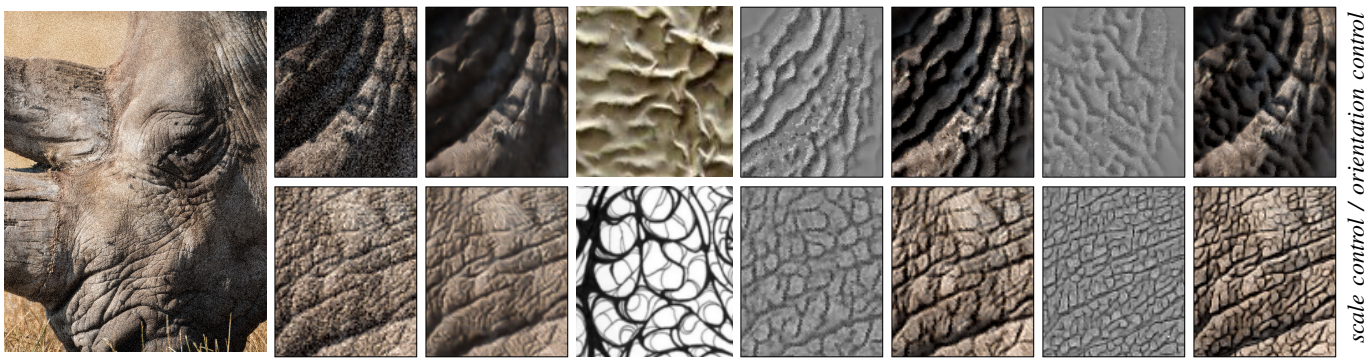



Fig. 1: Illustration of the proposed stylized and user-guided image restoration process. A degraded image (a) is processed in two parallel steps: a lightweight *Restoration network* (200K parameters) which focuses on restoring the main image structures (b), combined with plug and play *Style networks* (50K parameters) which synthesize details in additive layers (c) while controlling its intensity, localisation, orientation and scale (first and second row).

***Abstract*—Image restoration has come a long way since the early age of image processing. Deep learning methods nowadays give outstanding results, yet very few are actually used in digital illustration and photo retouching software due to large memory storage, massive computational requirements, but also the lack of user control and customization. This paper introduces a new lightweight framework for stylized and controlled image restoration using multi-scale networks built with independent parallel branches. The approach -based on two independent and complementary tasks- aims at: *i.* designing a *lightweight* network based on image processing techniques making it usable on light hardware architectures (low memory/computational costs); *ii.* providing a versatile, controllable and *customizable* network to *stylize* results in a plug-and-play manner. For various image restoration tasks (super-resolution, denoising, sharpening and inpainting), we demonstrate that the proposed method offers significant advantages over state-of-the-art reference-based approaches regarding these aspects.**

***Index terms*—** Super-resolution; Denoising; Sharpening; Image restoration; Style Transfer; Lightweight and Shallow Neural Network; Texture Synthesis; Interactive Computation

***Acknowledgement*—** This work has been supported by Région Normandie under PhD grant RIN.

***Note*—** A preliminary version of this work has been published in [6].

## I. Introduction

This work focuses on image restoration problems, such as super-resolution, deblurring, denoising and inpainting. In these applications, one wants to recover from degraded observations an image that is as faithful as possible to the ground-truth image. The task itself may be modeled as an ill-posed inverse problem whose solutions greatly depend on the prior models being used. As a result, for a given observation, very different solutions may yield visually satisfying results, for instance restoring the same structures while having different textures as illustrated in Figure 1.

For all the aforementioned restoration problems considered in this work, the degradation process mainly cuts out high frequencies and deteriorates medium frequencies as well. Following the conventional cartoon-texture image decomposition, we therefore distinguish between structures (such as contours) that are easily recoverable, and textures (patterns, high frequency details) that may be completely lost.

Image processing techniques have been long relying on local models -*i.e.* based on a very few parameters- which proved to be efficient for small image degradations by recovering image structures. These methods -based on local geometric features- are robust, often light in terms of computation and hardware requirements, and can be easily combined to create a specific processing pipeline; this explains why they are still popular in computational photography for instance. When considering more impactful degradation processes, the task of recovering the lost information gets harder and more solutions may be satisfying. Besides recovering the main structures, it shifts to the problem of synthesizing plausible details, such as textures and patterns. This task can be achieved with more complex models based on a large number of parameters. Such models cannot be tuned by hand and require learning techniques combined with large datasets to outperform handcrafted methods.

In this work we investigate a new paradigm for image restoration trying to take the best of both approaches, making use of light trainable generative models that can be combined and manipulated by the end-user to get the desired outcome [1].

To that end, we introduce a model that has two parts. First, a simple *restoration network* is trained on various restoration tasks. This base model aims at delivering a clean image, focusing on recovering the main structures of the image (sharp edges, flat regions and shades). Then, the resulting image is enhanced by dedicated models synthesizing high frequency patterns that have been lost or damaged beyond repair in the degraded image, as illustrated in Figure 1. These generative *style networks* are selected by the end-user and processed in parallel with the *restoration network*, allowing to control its characteristics (such as type, location, orientation, scale and intensity of new details being synthesized). The proposed approach contrasts with the usual end-to-end deep network which has to automatically infer the type of texture to generate. As a consequence, the proposed strategy enables the user to obtain the desired result by manipulating very *lightweight* networks, *i.e.* requiring very few parameters and computational resources. Overall, the proposed pipeline is inspired by computer graphics artists who edit image at different scales using different layers. To that end, we investigate the use of stylizing techniques, combined with image processing techniques to ensure that the combined models are complementary. Our contributions are twofold, corresponding to the two independent parts of our network. First, we present a lightweight *restoration network* architecture to tackle various image restoration tasks (illustrated by Figure 1-b for the denoising task). The resulting model is inspired from linear scale-space analysis where image is decomposed into different frequency bandwidths. Then, we extend the aforementioned *restoration network* with *style networks*, each one dedicated to a specific style (see Figure 1-c) and encoded in nearly as much parameters as required to store the example image. These additional networks are composed of light parallel branches synthesizing missing high frequency details that are compatible with the restored image.

## II. Related Work

### A. Training networks for image restoration and synthesis

Deep-learning based methods have improved image restoration performance over the last decade. First approaches, such as [4] for super resolution and [38] for denoising, have been based on end-to-end convolutional neural networks. Using natural images from large datasets, these networks are trained on artificially degraded images to minimize a pixel-wise objective function such as the MSE loss. Since then, numerous methods using deeper and wider networks have been proposed, still focusing in improving the reconstruction with pixel-wise evaluation metrics such as the PSNR or SSIM. For instance, regarding super-resolution, networks have been steadily gaining complexity to achieve such goal, either processing directly the low resolution input [31], [5] or its bicubic / bilinear interpolation [15], [16], [22]. Similarly, pixel-wise reconstruction performance of denoising networks have been improved by the use of more complex networks, see *e.g.* [42], [21], to succeed even beyond simple white noise degradation, such as hybrid noises [43].

For extreme degradation processes, restoration task is more about data generation than restoration, and texture synthesis turns out to be the most challenging part when restoring an image. As a result, pixel-wise losses inspired from signal processing (such as PSNR) are notoriously limited for image synthesis, and not faithful to human perception [14], [29], [20]. For this reason, two training losses inspired from image generation have been introduced in image restoration literature in order to synthesize visually plausible details.

The first one is directly inspired from Generative Adversarial Networks (GANs [11]). It boils down to training an auxiliary network to assess the quality of the synthesized images. This discriminative network is a binary classifier trained in a non-supervised fashion using cross entropy. In practice, this training is performed simultaneously with an auto-encoder by linearly combining such an adversarial loss with a pixel-wise loss (*e.g.* $\ell_1$ norm in [40] for image inpainting, and in [44] for super-resolution).

Another popular technique makes use of a *perceptual loss*. Such a loss uses a pre-trained network to extract deep features, as inspired from the seminal work of [8] in texture synthesis and [10] in style transfer. In most cases, the VGG classification network [17] trained on ImageNet is used to capture high-level and semantic information relevant to the task [24]. As in [10], the loss is composed of two terms : a content loss that directly compares feature maps from two images at a given layer of VGG, and a texture loss that compares features' Gram matrices, at different layers. Johnson *et al.* [14] have been the first to demonstrate the benefit of training an auto-encoder with a perceptual loss for super-resolution. Since then, various other applications have been shown to benefit from this framework, such as denoising in [24].

Finally, some methods such as SRGAN [20] and [29] combine the adversarial and perceptual losses to take advantage from both supervised and non-supervised representations.

### B. User control in image restoration literature

As illustrated in Figure 1, several solutions to an image restoration problem may be visually satisfying, especially when considering large degradations where details are lost. Those solutions share the same structures (low/middle frequencies) but have different textural details (high frequencies). That is especially the case for super-resolution, which is an ill-posed inverse problem where different high resolution images have the same PSNR, and maximizing this criterion yields smooth images. Even though perceptual losses have been widely used in various methods to avoid this issue and generate missing details, to the best of our knowledge, none of them specifically aims at controlling the type of texture being synthesized.

A proxy task in the literature is Reference-based image super-resolution. It aims at transferring the desired high res-

[1]The practical interest of such an approach have been already investigated for super-resolution in [7].

olution details from a reference image to a low resolution image. In [45], [39], VGG features from the restored image and a reference image are matched to achieve this goal. It yields noticeable improvements when making use of a relevant reference picture (*e.g.* same scene under a similar viewpoint). However, those models do not allow the user to enforce a specific texture, as demonstrated later in the experimental section. Only the very recent 'text to image' models such as DALL-E [27] allow some control on the synthesized textures (by adapting the input text prompt or including some reference image), yet at the cost of a huge amount of computational resources (with several billions of parameters), a lack of explicability and some issues about overfitting and memorization [32], [2].

In contrast, although we make use of perceptual features in this work, we specifically train the network to enforce some reference textures selected by the user during processing. To that end, we take inspiration from style transfer approaches that allow the user to control different aspects of the synthesis (orientation, scale, color, *etc*) such as in [9], [13].

### C. Lightweight Network Architectures

Recent networks for image restoration are often deep, wide and complex, including millions or billions of parameters, especially when encoders are used to extract perceptual features (see [26], [35]). Such heavy models exhibit impressive results, as shown in the experimental section. However, it may be hard to understand the behavior of their complex features and to formulate logical rules which may be used to control the output. Moreover, it results in hard trainings depending on rich datasets to avoid overfitting, but also long inference times on CPU, requiring large memory storage and penalizing near real-time processing.

In contrast, lightweight architectures circumvent those limitations, sometimes at the cost of less appealing results. Yet, they may be able to recover main structures of a degraded image [28] through patch-based representation embodying simple geometric characteristics at different scales or orientations. Even high frequency patterns may be recovered through local patch match. For example, [12] enhances and hallucinates details copying locally patches. Samely, Zheng *et al.* [46] takes advantage of patches of the same scene to enhance low-resolution image details during super-resolution process. Note that very lightweight convolutional networks can be used to achieve high quality texture synthesis by making use of multi-scale architecture with large receptive field [37].

## III. Restoration Network and Style Network Architectures and Training

Our global architecture is composed of two independent yet complementary neural networks trained separately. The *restoration network* (providing *restored image* of Figure 1) aims at reconstructing image structure and the *style network* (providing *stylized images* of Figure 1) attempts at synthesizing plausible textures. As already mentioned, the idea of distinguishing structures (e.g. contours, flat and smooth regions) from high frequency details (e.g. textures) is not new [1]. Note that lightweight and shallow architectures can not learn semantics or abstract features. We make a strength out of this weakness, building an architecture which does not depend on deep semantic features to decide which texture to add, but which relies on the user decision to add specific style with very local features encoded in a minimalist network. This allows to define a *style network* ($\simeq$ 50K parameters) trained on the top of the *restoration network* ($\simeq$ 200K parameters) enabling the user to locally enforce specific parameterized style high frequency characteristics.

### A. Restoration network architecture and training

In this paragraph, we introduce our *restoration network* dedicated to the restoration of the degraded image structures. The aim of this generic model is three-fold: to be able to deal with various restoration problems, to be simple enough to allow for light computation and memory requirements, and to provide clean images that can be simply edited by the user by selecting more specific models generating the missing high frequency patterns as shown in Sec. IV. To achieve theses objectives, our model is inspired from multi-scale image decomposition techniques. Those have been proven to be efficient for various tasks, including image restoration with wavelets [41] or image detection [25]. An overview of our proposed architecture is shown in Figure 2. The network processes the degraded image in a multi-scale fashion, and is composed of $n = 6$ parallel *restoration branches*, which outputs are linearly combined. Each branch focuses on a specific frequency band, as exposed hereafter.

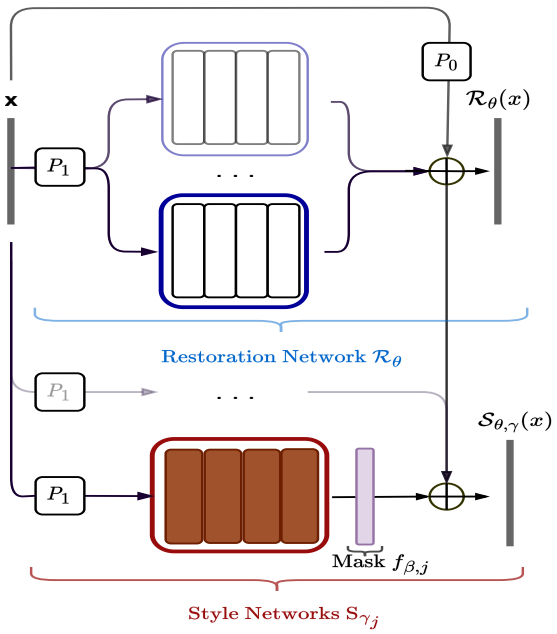



Fig. 2: Overview of the proposed global architecture composed of the *restoration network* for which *restoration branches* are described in section 3, and the *style network* which synthesizes a layer added on the top of the restored output. The output is synthesized using multi-scale additive reconstruction based on difference of gaussian filters. The whole architecture is generic for any degradation process, only $P_0$ and $P_1$ are preprocessing modules adapted to the task.

*1) Considered degradations and corresponding preprocessing modules $P_0$ and $P_1$:* We deal with four different tasks for which the whole model is generic except for the preprocessing modules $P_0$ and $P_1$ illustrated in Figure 2. The first task is super-resolution which aims at restoring an image for which degradation consists in blurring (standard deviation $\sigma_{blur} = 1.5$) and $\times 4$ sub-sampling. For such, $P_0$ and $P_1$ both correspond to the $\times 4$ bicubic interpolation of the low-resolution images. The second task we consider is sharpening. Unlike the super-resolution task, such blurry images (standard deviation $\sigma_{blur} = 1.5$) are not subsampled but corrupted by additive white noise. No preprocessing modules are used. The third task considered is denoising, for which the degradation consists in adding a noise drawn from white gaussian distribution ($\sigma_b$ is specified for each example). For denoising, $P_0$ simply consists in blurring the input data. Thus, the network aims at recovering the border lost through the blur knowing the noise, making the convergence faster. However, $P_1$ is an identity module as one wants the network to process the noisy data. Finally, we consider masked inpainting which intends to restore data for which parts of the pixels have been removed. Then, for generating degraded inputs, a random binary mask $M$ is created for each patch, masking pixel with 75% probability. The input is simply the pixel-wise multiplication between ground truth and such mask for all the patches. For such data, $P_1$ and $P_0$ are the same module and correspond to a heat diffusion model spreading the information and resulting into an image without off pixels.

*2) Definition of one restoration branch:* The detail of a *restoration branch* is displayed in Figure 3. Degraded image is fed into each *restoration branch* of the network, each branch being composed of 4 modules. $H_{\theta,i,k}$ refers to the $k^{th}$ residual module from the $i^{th}$ branch. Each of such module starts with a $3 \times 3$ convolution, followed by a batch normalization, and ends up with a `relu` activation function. Note that the $H_{\theta,i,1}$ module is not residual. We noticed that residual modules tend to stabilize the training in a multi-branch context.

*3) Multi-branch architecture for multi-scale reconstruction:* Lindeberg [23] identifies the Gaussian function as the unique linear scale-space kernel. More precisely, he shows that scale invariant detection can be achieved by normalized differential operators in a Gaussian pyramid. Here, we make use of Difference of Gaussian (DoG) filters which correspond to first order approximation of the multi-scale normalized Laplacian filter, as exploited for corner detection in SIFT [25]. As shown in Figure 3, those DoG operators are used to the filter the restored image rather than the input. Then, we benefit from giving all the information to each branch of the network, and stabilize training through energy balanced branches.

Denoting $\text{DoG}_i$ the $i^th$ DoG convolutional filter and $G_{\sigma_i}$ a gaussian filter with $\sigma_i$ standard deviation, we define: $DoG_i = G_{\sigma_i} - G_{\sigma_{i-1}}$ where $\sigma_i$ follows a geometrical evolution as performed in [25]: $\sigma_i = \sigma_0.p^i, i \in 0, ..., 5, \sigma_0 = 1.0, p = 1.3$ Such filters are passband filters. The highpassband filter is built with a dirac $\delta$ as follows: $DoG_{ST} = \delta - G_{\sigma_0}$. As the mean of a $DoG_i$ is 0, the mean of the output from $DoG_i$ is a centered residual. By making use of $DoG_i$ convolutional filters and $tanh$ activation function at the end of branch $i$, the proposed parallel network performs a multi-scale reconstruction, specializing each branch $i$ on a specific frequency bandwidth. Then, branch outputs are linearly combined in order to reconstruct the global residual output, as shown in Figure 2, each branch $i$ being specialized in a specific frequency band. Note that Laplacian pyramid networks for image restoration already exist in the literature [33], [18] for sequential image restoration instead of multi-scale parallel image reconstruction introduced here.

*4) Formulation:* from now on, $\mathbf{X}$ (resp. $\mathbf{x}$) $\in \mathbb{R}^{K \times N \times N \times 3}$ refers to a collection of $K$ ground-truth (resp. degraded) input color images of size $N \times N$, both used during training and evaluation. The $k$-th degraded image from the collection, noted $X_k \in \mathbb{R}^{N \times N \times 3}$, is encoded using the *YCbCr* color system. Denoting $\theta$ the 200K trainable parameters of the *restoration* model, the 1-channel output of branch is $(\mathrm{R}_\theta)_i(P_1(x))$. With $n = 6$ branches, $R_i$ may be written as follows:

$$(\mathrm{R}_\theta)_i(y) = [tanh \circ DoG_i \circ H_{\theta_{i,4}} \circ H_{\theta_{i,3}} \circ H_{\theta_{i,2}} \circ H_{\theta_{i,1}}](y).$$

With 35 filters per convolutional layers, the number of parameters for encoding kernels and bias is less than 35k parameters per branch. Finally, the *YCbCr* color output of the *restoration network* is the sum of outputs from parallel branches:

$$\mathcal{R}_\theta(x) = P_0(x) + \sum_{i=1}^{n} (\mathrm{R}_\theta)_i(P_1(x)) \qquad (1)$$

where preprocessing modules $P_0$ and $P_1$ depend on the task on which the neural network is trained. The *restoration network*

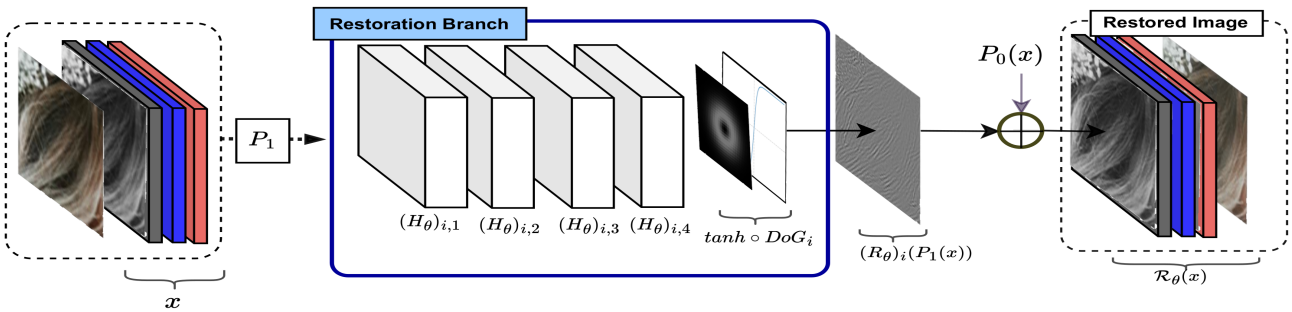



Fig. 3: Overview of the $i^{th}$ parallel *restoration branch* architecture composing the *restoration network* shown in Figure 2.

has 200K parameters and reconstructs the luminance channel only, preprocessed chrominance being concatenated with restored luminance. Yet, the model may be tuned for chrominance reconstruction for chrominance degradations such as color noise. Such parallel independent branches architecture enables easy branches parallelization on CPU cores. In such context, the number of multiply-adds (MAdd [30]) operations in order to evaluate one restored pixel of a given degraded image is reduced to $\simeq$ 70K. This number corresponds to the number of computations in one branch. Note that such approximation may slightly vary with the task, depending among others on modules $P_0$, $P_1$ and on DoG implementation.

*5) Restoration network training:* As previously mentioned in the introduction, the global loss to train the *Restoration network* combines a *mean squared error* with a *perceptual loss*, as it is widely known that optimizing only MSE favors texture-less reconstruction in image restoration tasks [29]. Finally, we solve $\min_\theta \mathcal{L}_\mathcal{R}(X, \mathcal{R}_\theta(\mathbf{x}))$ with the loss:

$$\mathcal{L}_\mathcal{R}(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{K} \frac{1}{K} \|x_k - y_k\|^2 + \lambda_\mathcal{R} \mathcal{L}_{\text{Perc}}(x_k, y_k) \quad (2)$$

where $\|.\|$ stands for the Frobenius norm, $\mathcal{L}_{\text{Perc}}(x, y) = \sum_{\ell \in L_{\text{Perc}}} \|\phi_\ell(x) - \phi_\ell(y)\|^2$ is the perceptual loss, and $\phi_\ell(.)$ corresponds to the normalized feature maps at the $\ell$-th layer of VGG-16 [17]. We consider features from different layers of the VGG in order to capture different scale features at $\texttt{relu}_{\{1\_2\}}$, $\texttt{relu}_{\{2\_2\}}$, $\texttt{relu}_{\{3\_2\}}$ layers *i.e* $L_{\text{Perc}} = \{2, 5, 9\}$.

### B. Style network for texture editing

In this paragraph, we introduce the *style network* dedicated to synthesize high frequency patterns on the top of an image without interfering with its main structures and perceptual characteristics. Note that such branch are trained independently. During evaluation, they may be used separately for image enhancement, or plugged in parallel of the *restoration network* in the context of stylized image restoration, allowing the user to control the generated patterns.

*1) Architecture:* For the sake of simplicity, only one *style network* is represented in Figure 2. Yet, $m$ style branches are trained in adding coherent details on restored image for a given task. Then, they are used in addition to the output of such pre-trained and frozen parameters *restoration network*. Note that *style networks* may also be trained independently in adding coherent details from $m$ different styles on high resolution images, as exposed in paragraph IV-C2b of experimental section. The architecture illustrated in Figure 4 is composed of 4 modules of convolutions. $T_{\gamma_{j,k}}$ refers to one of these residual modules (number $k$ for branch $j$), consisting in two $3 \times 3$ convolutional layers, as proposed in [14]. As for the *restoration network*, note that the first $T_{\gamma_{j,1}}$ module is not residual. For the same amount of parameters, it is better to have deep architectures than wide ones, as one wants long-range dependencies and features in the stylisation context. That is why two convolutional layers are used in each convolutional module. The $DoG_{ST}$ high-passband filter enforces high frequency and centered output, editing the image with a 0-mean residual layer. Additionnaly, we enforce during training the standard deviation of the layer output to be close to half of the standard deviation of the *restoration network* output, as exposed hereafter.

*2) Formulation:* From now on, $\gamma_j$ stands for the trainable parameters of the *style network* $j$. With the previous notations, the 1-channel output of the $j$-th *style network* can be written as follows:

$$S_{\gamma_j}(x) = [tanh \circ DoG_{ST} \circ T_{\gamma_{j,4}} \circ T_{\gamma_{j,3}} \circ T_{\gamma_{j,2}} \circ T_{\gamma_{j,1}}](x).$$

As mentioned previously, a normalization module $f_{\beta,j}$ is used in order to control the variance *style network* output. It aims at enforcing the first and second order statistics ($\overline{y}$ and $\sigma_y$) of each image of the batch output close to respectively 0 and $\sigma_x/2$. During evaluation, the intensity of the synthesized residual patterns can be tuned by the parameter $\beta_j$ which is set to $\frac{1}{2}$ during training. More specifically, the user may adapt the intensity of each pixel through a user-defined pixel map modifying locally each pixel around its 1 default value in a plug-and-play manner.

$$f_{\beta,j}(y) = \beta_j \odot \frac{\sigma_x}{\sigma_y}(y - \overline{y}) \quad (3)$$

where $\odot$ indicates pixel-wise multiplication. Finally, considering $m$ *style networks i.e* $j \in [1, ..., m]$, the output of the whole model (*restoration network* and $m$ *style networks*) is given by

$$\mathcal{S}_{\theta,\gamma}(x) = \mathcal{R}_\theta(x) + \left[\sum_{j=1}^{m} f_{\beta,j}(S_{\gamma_j}((P_1(x)); \mathbf{0}; \mathbf{0}\right] \quad (4)$$

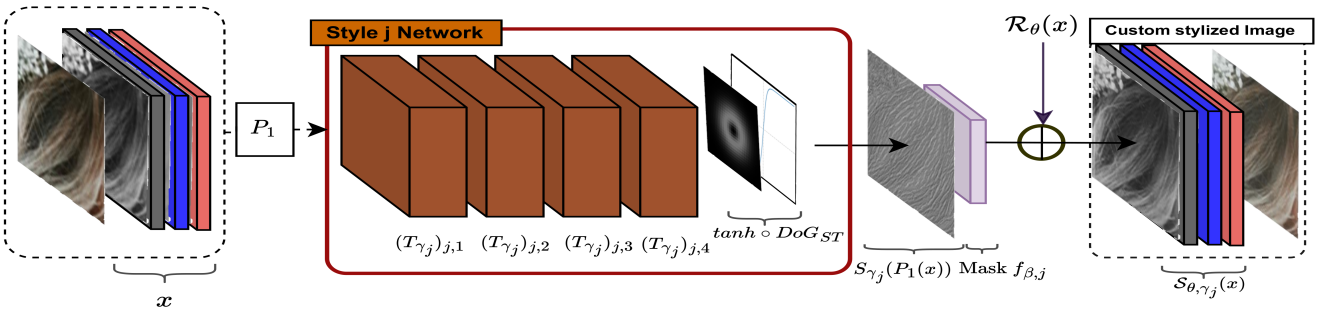



Fig. 4: Overview of a *style network* plugged on the top of the trained *restoration network* shown in 2. *Style network* output is filtered by a high passband filter to focus on generating high frequency patterns.

Note that *style networks* process only the luminance channel of the restored image $\mathcal{R}_\theta(x)$, not the chromaticity channels (set as $\mathbf{0}$ in the above formula). Inspired by artists who edit images through additive layers, we edit only the details of the luminance channel. Such combinations allow us to use multiple branches simultaneously even if they are trained separately. Finally, each layer is composed of 26 convolutions resulting in *style networks* with less than 50K parameters. Yet, *style network* is deeper than *restoration network* to favor longer features. Thus, the number of multiply-adds (MAdds [30]) operations in order to evaluate one stylized pixel of a given restored image is $\simeq$ 100K. Note that in specific experimental context (see Sec. IV-C2b), such *style networks* computations may also be parallelized on CPU cores, likewise the *restoration branches*.

*3) Style network training:* We denote $Y$ as the styles tensor, and $Y_j$ the $j^{th}$ associated style image. Note that luminance from each $Y_j$ style image is normalized with the same standard deviation in order to have a single training procedure no matter the style dynamics. Such normalization does not misrepresent the characteristics in a residual learning context. For training the $j^{th}$ network separately, $\theta$ *restoration network* parameters trained for a specific task are frozen. Such *style network* learns high frequency details from a given normalized reference style image $Y_j$ and transfers it on the restored image $\mathcal{R}_\theta(x)$. During training, $\beta_j = 1$. Thus, the following objective function $\mathcal{L}_S$, for a given reference style image $Y_j$ considering only one reference style image $Y_j$ at a time, may be written:

$$\mathcal{L}_{\mathcal{S}}(\mathbf{X}, Y_j, \mathbf{Z}) = \frac{1}{K}\sum_{k=1}^{K} \lambda_{\mathcal{S}} \mathcal{L}_{\text{Perc}}(Z_k, X_k) + \mathcal{L}_{\text{Tex}}(Z_k, Y_j) \tag{5}$$

where the texture function $\mathcal{L}_{\text{Tex}}$ defined with normalized Gram matrix $G$, accordingly to [8] follows:

$$\mathcal{L}_{\text{Tex}}(x, y) = \sum_{\ell \in L_{\text{Tex}}} \|G(\phi_\ell(x)) - G(\phi_\ell(y))\|^2. \tag{6}$$

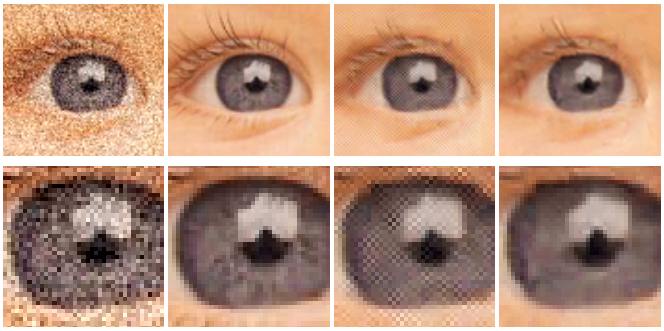
In order to have the same training process for all styles, the same fidelity parameter $\lambda_{\mathcal{S}}$ is set to 100K in all the experiments. We favor small scale details synthesis from the reference image by considering the following layers for gram matrix computation: $\texttt{relu}_{\{1\_2\}}$ and $\texttt{relu}_{\{2\_2\}}$ layers (*i.e* $\text{L}_{\text{Tex}} = \{2, 5\}$) and use for perceptual loss only the layer $\texttt{relu}_{\{3\_2\}}$ (*i.e* $\text{L}_{\text{Perc}} = \{9\}$) to preserve large scale information from the restored output. Finally, such branch is optimized solving:

$$\min_{\gamma_j} \mathcal{L}_{\mathcal{S}}\left(\mathbf{X}, Y_j, \mathcal{S}_{\theta,\gamma_i}(\mathbf{X})\right), \forall\, 1 \leq j \leq m.$$

Training the *style networks* independently as above, rather than task-dependant, that is using the pre-trained restoration network output $\mathbf{X} = \mathcal{R}_\theta(\mathbf{x})$, did not give noticeable difference, as shown later in the experiments.

### C. Checkerboard artifacts filtering

As reported in [29], [14], [37], the perceptual loss based on VGG features induces checkerboard artifacts. Substituting *max-poolings* for *average-pooling* may reduce the amount of artifacts [8] but does not solve completely the problem. Thanks to our multi-scale reconstruction, those easily identifiable artifacts appear only after the high-passband filters, that is in the last branch of the *restoration network* ($DoG_5$), and the *style networks* ($DoG_{ST}$). Thus, a $2\times2$ median filter is applied at the end of these branches during inference, removing VGG artifacts as illustrated in Figure 5. Note that the median filter is only used during inference to prevent the second highest frequency band branch from learning how to compensate and reintroduce those artifacts.



Noisy Image | Ground Truth | No filter / With median filter)

Fig. 5: Illustration of the median filter role applied on the last branch of the trained *restoration network* with perceptual loss.

## IV. Experiments

In this section, we first introduce the experimental settings and the degradation processes associated to specific restoration tasks in Sec. IV-A. Then, we illustrate the interest of the proposed method for stylized restoration on various problems, in Sec. IV-B. Finally, we evaluate the *restoration network* and the *style networks* separately as independent architectures through ablation studies, in sections Sec. IV-C1 and Sec. IV-C2.

### A. Experimental Settings

We use the *DIV2K* dataset [36] intended for the $\times4$ super-resolution challenge. Note that such dataset only provides low and high resolution images pairs for training and validation datasets. As a result, the last 150 images from the training set were hold out to build a testing dataset with ground-truth images. During training, square patches (254x254 degraded patches) are extracted from training images indexed from `001` to `650` and fed through the network. Approximately 20K and 6K patches -with a minimum variance requirement- are used for training and for cross-validation. *Restoration networks* (resp. *style networks*) are trained with Adam algorithm with the loss defined in (2) (resp. (5) ) using a learning rate of $1.10^{-3}$ (resp. $1.5.10^{-3}$), reduced by $10\%$ at each epoch. For such data, we consider the degradation processes and preprocessing modules $P_0$ and $P_1$ described before.

### B. Experiments on stylized restoration

Here, we focus on qualitative evaluation of the the proposed method combining the *restoration network* with specific *style networks* for image restoration. We illustrate that clean images provided by the *restoration network* can be enhanced by appropriate texture synthesis from *style networks*. We also discuss how the user has control over the stylized image, through style choice, but also through affine transformation. Finally, we compare our method against reference-based methods in the context of stylized super-resolution.

*1) Stylized super-resolution task:* Various methods [4], [22], [20], [44] (evaluated later in Table I) are here showcased in the second row of Figure 6 and compared to our approach for the super-resolution task. Observe (line 1, columns 3 & 4) that, similarly to EDSR [22] which is trained with pixel-wise loss, our *restoration network* allows a good reconstruction of simple structures (such as edges and lines) even with a lightweight network. As already mentionned, using only MSE (*i.e* $\lambda_{\mathcal{R}} = 0$) gives better PSNR but slightly less visually pleasing results than when using perceptual loss as reported in [14].

However and as expected, the proposed *restoration network* is not able to generate missing textures, contrary to very-deep adversarial methods such as RDN [44] which has approximately 18 times more parameters. Instead, the proposed approach lets the user choose the desired type of generated details among chosen appropriate reference images (*style networks*, line 1, column 5).

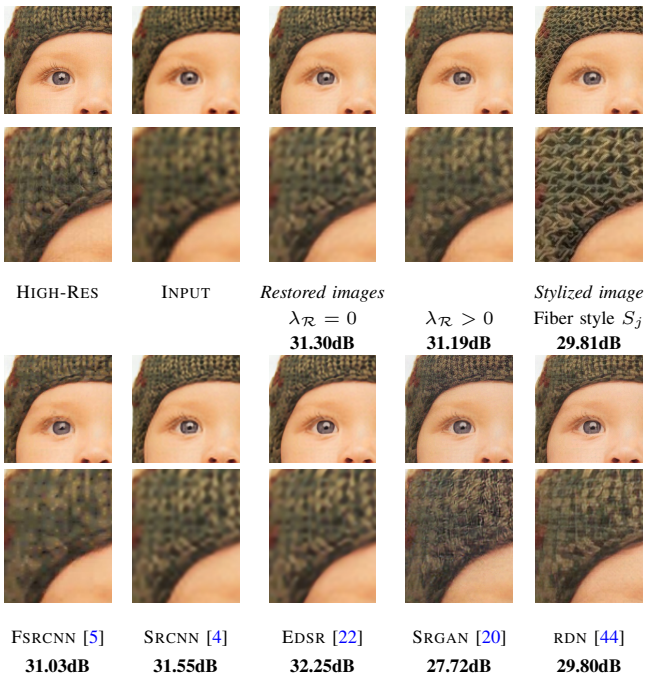



Fig. 6: Comparison of $4\times$ super-resolution results (cropped `baby` image from Set5 dataset) between our networks (first row) and other super-resolution neural networks. In our framework, the restoration of simple structures is performed by the *restoration network* (first row, column 3,4) and enhanced by the texture synthesized by the *style network* (first row, last column) applied on the baby's hat with a fiber style.

*2) Stylized denoising:* In this paragraph, we evaluate on two examples our *restoration networks* for denoising RGB noises against FFDNet [34] which has slightly more parameters than our network (850K against 200K) and BM3D [19], [3]. Note that FFDNet [34] and our networks are trained for a specific standard deviation noise. For both examples, observe how textures (monkey hairs and statue surface) are barely recovered after restoration, even if FFDNet [34] may slightly reconstruct part of them (visually and quantitatively according to PSNR). In contrast, our approach does not intend to recover such textures out of the *restoration network* - allowing it to embody very few parameters - but rather lets the user enforce specific texture locally. The *style networks* are fast and light enough to allow for the user to choose a coherent *style network* through trial and error (associated to hairs or stain styles here).

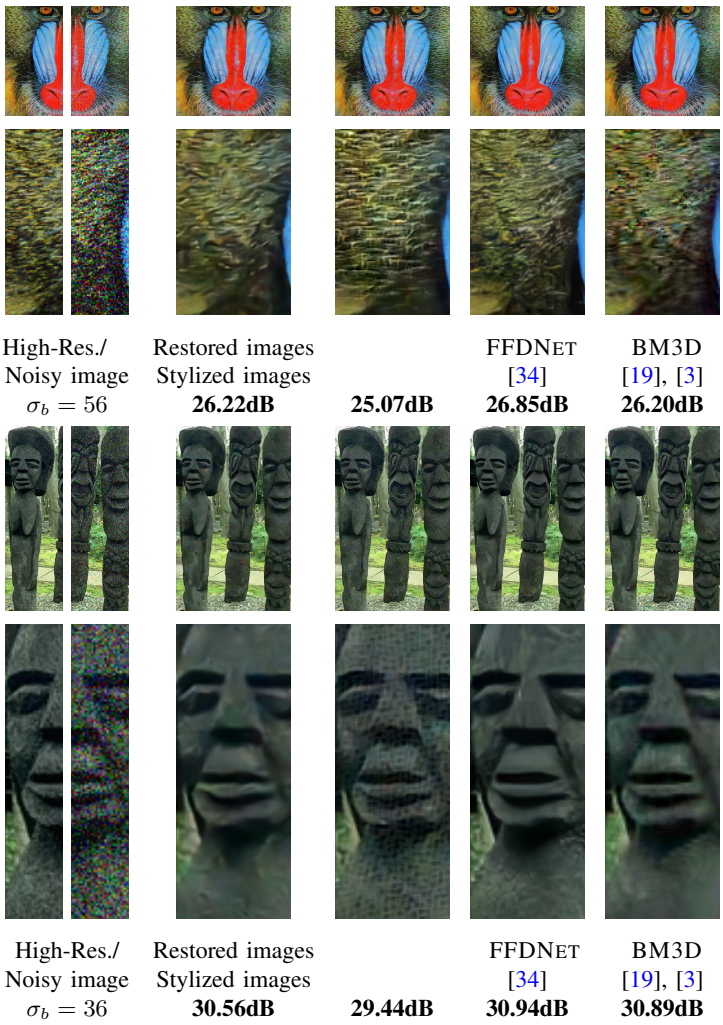
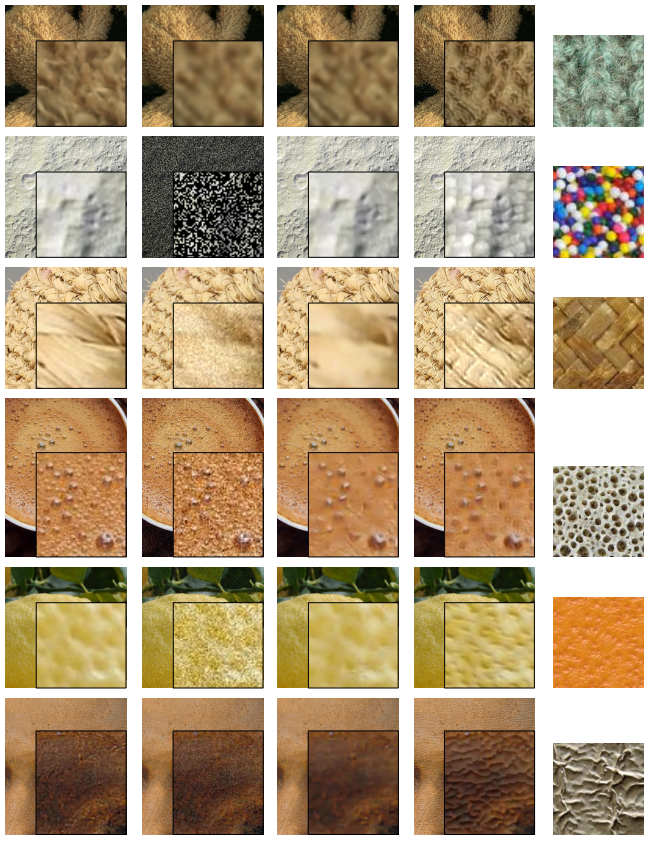



Fig. 7: Comparison of denoising results on images (from Set14 and BSD100 datasets) between our networks and *FFDNet* [34], BM3D [19], [3], with specific zoom on lost textures. Stylized denoising is performed in two steps: structure restoration performed by the *restoration network* and stylisation performed by the chosen *style networks* .

*3) Controlling stylization:* The control by the user of the generated image is achieved through different mechanisms. First, masks $f_{\beta_j}$ can be specified depending on (a) *localisation* (as shown in Fig. 7 where styles are applied on the statue body), *(b) intensity* (as illustrated in Fig. 1 with additive layers with different amplitude) and (c) *blending i.e. combination of different layers*. Second, these generated layers can be geometrically transformed to produce the desired effects, such as a scale or orientation change (without retraining the corresponding *style network*). Figure 1 shows different examples of such transformations for two different styles: with two different orientations and amplitude ($90^o$ and $135^o$ in first row) and two scales ($\times 1$ and $\times 2$ in second row). Notice that some artifacts may be noticeable for extreme affine transformations (e.g. $\times 3$ scale change).

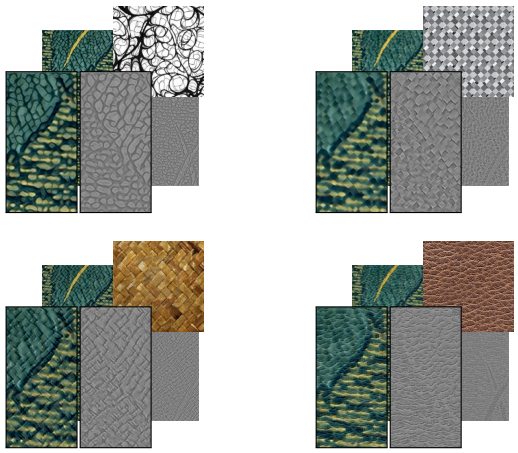
*4) Stylized image restoration:* Figure 8 exhibits different stylized restoration results for which adapted textures have been chosen depending on the missing details in the restored images. Note that *style networks* are combined with the *restoration networks* trained with the perceptual loss (*i.e.* setting $\lambda_{\mathcal{R}} = 1$ in (2)). Observe that arbitrary styles chosen at proper scales may help synthesizing plausible details in the restored images, even if such details are not found in the ground-truth image.



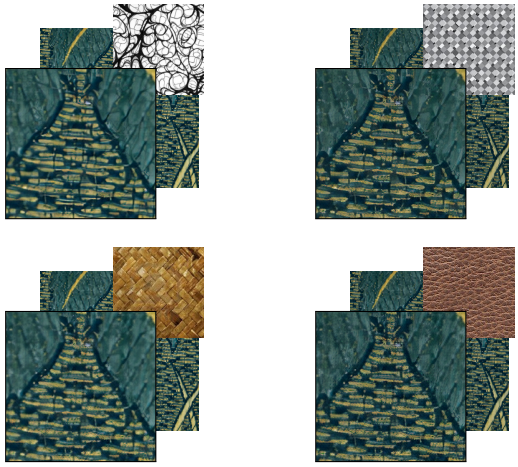
Ground Truth Degraded Restored Stylized image. & style

Fig. 8: *Style network* results for different *restoration tasks*. For each example are shown the ground-truth/degraded/restored/stylized images and the associated style

*5) Comparison of stylized super-resolution with Reference-Based methods:* In this paragraph, we compare our approach for the super-resolution task against TTSR [39] and illustrate visual results in Figure 9. TTSR is used as state-of-the-art baseline for Reference-Based Image Super Resolution. This method lets the user choose a reference image to enhance the result. However, it requires a reference image that is related to the input image (same scene with a different viewpoint). In our experimental setting, it is not able to enforce specific textures from style example. As illustrated in Fig. 9, using a style example barely changes the outcome of TTSR [39] synthesis. Moreover, the former network has more than 9 million parameters, being much deeper, wider than our *restoration* and *style networks*. Not only the inference is longer, but it turns out to be hard to interpret features and understand why some reference image characteristics may be found out or not in the enhanced output image. Inversely, our proposed method may apply and enforce specific patterns on the *restoration network* output. It is up to the user to try textures in a plug-and-play manner, which is eased by the lightness of the networks.



Outputs $\mathcal{S}_{\theta,\gamma_j}(x)$ / Style Nets $\mathrm{S}_{\gamma_j}(x)$ with associated style $Y_j$



TTSR [39] Outputs with the associated reference images $Y_j$

Fig. 9: Comparison of our approach (line 1) with TTSR [39] for stylized super resolution (line 2). The outputs of the networks (along with the additive layer in our case are displayed with the reference image $Y_j$.

### C. Ablation study

*1) Restoration network:*

*a) Qualitative evaluation for the super-resolution task:* We consider now the *restoration network* trained with the *perceptual loss* (*i.e.* setting $\lambda_{\mathcal{R}} = 1$ in (2)). In Figure 10 are shown four different examples of restoration tasks. Note that our architecture is fully convolutional, which allows us to process arbitrarily large images. On each example are displayed the original image $X$ on the first column, the degraded images $x$ (low-resolution, noisy, blurry or subsampled) on the second column, and the restored image $\mathcal{R}(x)$ on the last column. Here, all degradations concern the 3-channels except noise which may be added on luminance channel only (the bottom example of the first row). For such case, the *restoration* network restores the luminance only. Observe how image structures are well reconstructed with the same architecture trained on different tasks. Indeed, remark how noisy jewels structures are well restored and distinguishable. Samely, note the sharpness of the inpainted castle walls or the zebra stripes.

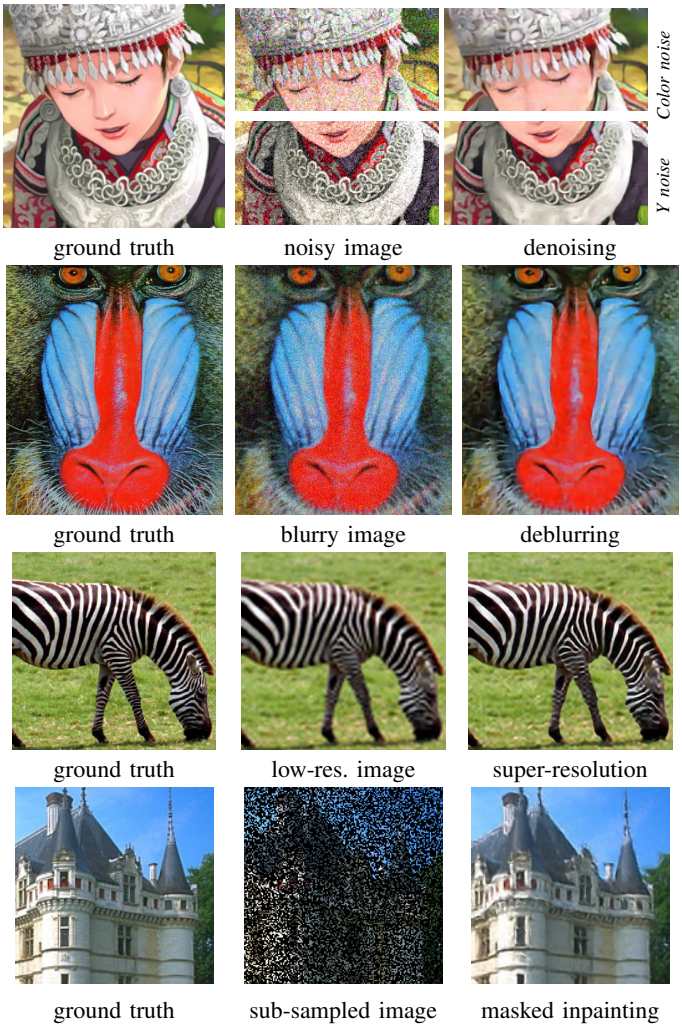



Fig. 10: Examples of different results for the *restoration networks* trained on 4 different image restoration tasks. The same architecture is used for the different tasks, only modules preprocessing input degraded images differ.

*b) Quantitative evaluation for the super-resolution task:* Evaluations are conducted on three standard benchmark datasets (*Set5*, *Set14*, and *BSD100*). We now evaluate our model with the MSE loss (*i.e* $\lambda_{\mathcal{R}} = 0$ in (2)). The restoration of the structures performed by *restoration network* is evaluated using PSNR and SSIM, two metrics inspired from classic signal processing approaches. Even if PSNR is somewhat a imperfect metric (as illustrated in [20]), it remains a standard metric for benchmarks to investigate the quality of our proposed *restoration network*.

| Model | #Parameters | Set5 | Set14 | Bsd100 |
|---|---|---|---|---|
| | #MAdd [30] | Δ PSNR | | |
| | | $\Delta SSIM$ | | |
| $\mathcal{R}$ | **200**K | **2.22** | **1.36** | **0.76** |
| | 70K | 0.083 | 0.085 | 0.051 |
| SRCNN [4] | **440**K | **1.95** | **0.81** | **0.54** |
| | 880K | 0.044 | 0.036 | 0.032 |
| EDSR [22] | **1517**K | **4.71** | **1.86** | **1.24** |
| | 3030K | 0.102 | 0.061 | 0.053 |
| SRGAN [20] | **1554**K | **2.00** | **-0.19** | **-0.48** |
| | 3110K | 0.072 | -0.005 | -0.007 |
| RDN [44] | **2205**K | **4.76** | **1.83** | **1.29** |
| | - | 0.059 | 0.036 | 0.059 |

TABLE I: Comparison of average PSNR / SSIM gains from bicubic interpolation on different datasets for $4\times$ super-resolution. Methods are ranked based on the number of parameters. $\mathcal{R}$ corresponds to the *restoration network* restricted to the MSE criteria in loss for luminance reconstruction. Note that #MAdd [30] corresponds to the number of computations for synthesizing one restored pixel, the associated number for $\mathcal{R}$ model being when considering branches parallelization. Also #Parameters corresponds to the number of parameters needed to store the whole model.

Table I compares the quantitative results for our *restoration network* on super-resolution task $\times 4$ with other models [4], [22], [20], [44] showcased in Figure 6. First, the table shows the number of parameters (*#Params* in Table I) and the number of multiply-adds computations for restoring one pixel (FLOPs per pixel, *#MAdd* in Table I), rounded up to thousandth. Note that *#MAdd* comparison is valid when considering fully convolutional models, as the number of FLOPs depends linearly on the image size. Second, the average PSNR/SSIM gains compared to bicubic upsampling are also displayed. While having a number of parameters/Flops in the bottom bracket, the proposed lightweight network $\mathcal{R}$ for image super-resolution still achieves interesting performance, reconstructing fastly main structures of the images, as expected for the stylisation which comes afterwards. However, notice that such pixel-wise metric is limited to quantify performance. Indeed, the generative adversarial method SRGAN [20] which is visually very satisfying (see Fig. 6) returns lower PSNR than our method. This shows that the lightweight restoration network is more faithful in recovering the main structures of the image, while GANs (and the proposed stylization networks) are able to generate plausible details that are penalized by such non-perceptual metrics.

*c) Hyper-parameters tuning:* Here, we specify the choice regarding hyper-parameters and more specifically the parallel network design. The usage of parallel and shallow branches not only allow for fast inference, but may as well improve performances. Indeed, for such lightweight architecture, it is better to distribute the parameters over multiple branches and corresponding frequency bands as shown hereafter. Recall that the proposed model with 6 branches has 200K parameters. The frequency cutoff of the last branch is defined from the $DoG_5$ filter. In Table II, we compare the performance of the network by varying the number of branch while keeping constant the number of parameters. To that end, we adapt the number of filters and the $DoG$ filter definition ($\sigma_0$ and $p$) for each architecture. The average PSNR of each model (trained with MSE only) is reported for datasets *Set5* and *Set14* for color denoising and super-resolution.

Interestingly, we observed that there exists an optimal number of branches for the proposed architecture given a budget of trainable parameters. Indeed, if the parameters are distributed over too few branches, the additive reconstruction is limited by the number of branches and their depth. Inversely, if the parameters of the network are distributed over too many branches, the branch get limited by the small number of filters. For simplicity, the number of branches has been set to 6 for all tasks in this experimental section, even if it may not the optimal setup for the denoising task for instance.

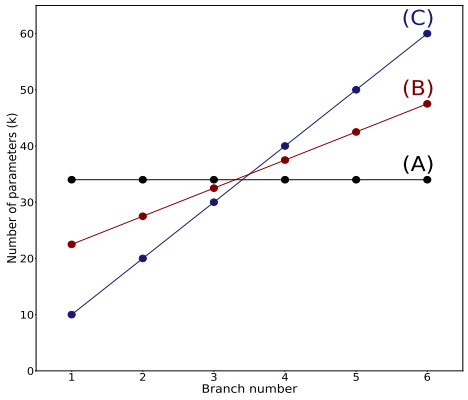
| | | | | Super-Res. | | Denoising | |
|---|---|---|---|---|---|---|---|
| #branchs/#filters/$\sigma_0$/p | | | | Set5 | Set14 | Set5 | Set14 |
| 1 | 85 | 1.00 | - | 33.37 | 30.32 | 35.33 | 34.20 |
| 2 | 60 | 3.71 | 3.710 | 33.47 | 30.32 | 35.50 | 34.35 |
| 4 | 42 | 1.00 | 1.550 | 33.54 | 30.41 | **35.71** | **34.50** |
| 6 | 35 | 1.00 | 1.300 | **33.57** | **30.43** | 35.65 | 34.47 |
| 8 | 29 | 1.00 | 1.205 | 33.46 | 30.36 | 35.68 | 34.44 |

TABLE II: Comparison of average PSNR on Set5 and Set14 datasets between different $\mathcal{R}$ network hyper-parameters tunings. Each value corresponds to a *restoration network* trained with MSE criteria only for a specific task (super-resolution and color denoising). Associated parameters are specified in the right (number of branches, filters and the geometrical progression for passband filters).

Note that parameters have been uniformly distributed over every branch in Table II. We analyze in Figure 11 the impact of other parameters distributions showing that allowing more parameters for high frequencies may improve overall performance. More precisely, we tuned the parameters distributions (through number of filters) among branches favoring branches synthesizing high-frequency features.

*2) Ablation studies for the style networks:*

*a) Hyper-parameters setting of the* style networks*:* We challenge here the fact that the number of parameters has been arbitrarily set to 50K for every style networks, which aim at capturing high-frequency patterns. In Figure 12, the same style branch $j$ is trained with different hyper-parameters (depth, width and number of parameters). For this style example, the high frequency pattern can be actually reproduced by a smaller




| | PSNR | |
|---|---|---|
| | Set5 | Set14 |
| (A) | 35.65 | 34.47 |
| (B) | **35.71** | **34.52** |
| (C) | 35.63 | 34.44 |

Fig. 11: Performance comparison of different parameters distributions for the *restoration network*.

number of parameters. In any case, the generated details preserve the main structures of the input image. However, as expected, larger models tends to produce more visually appealing results.

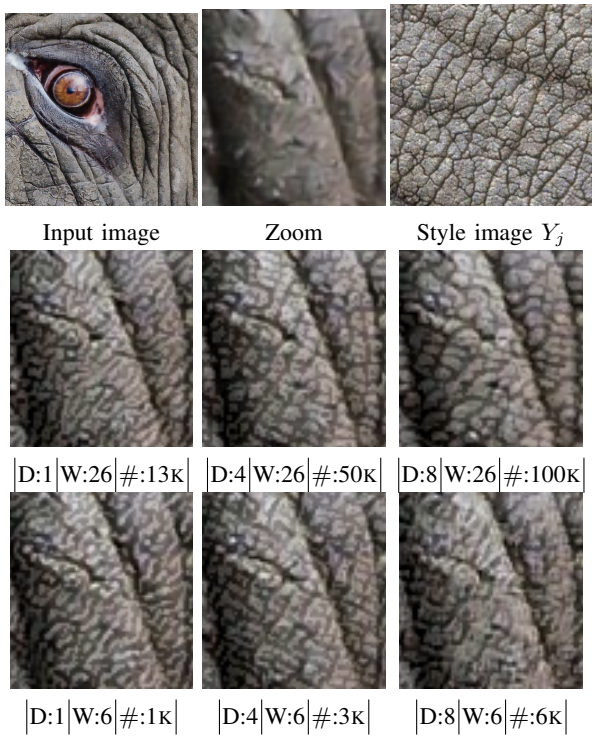
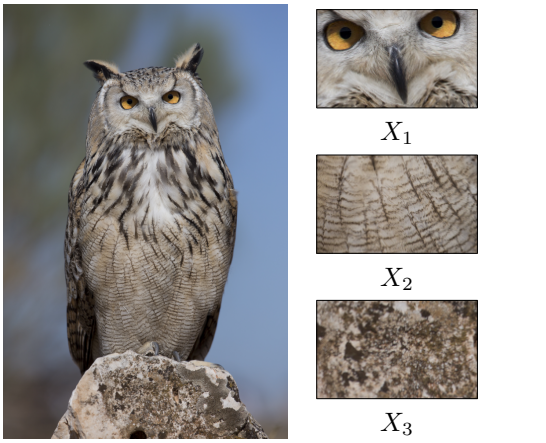
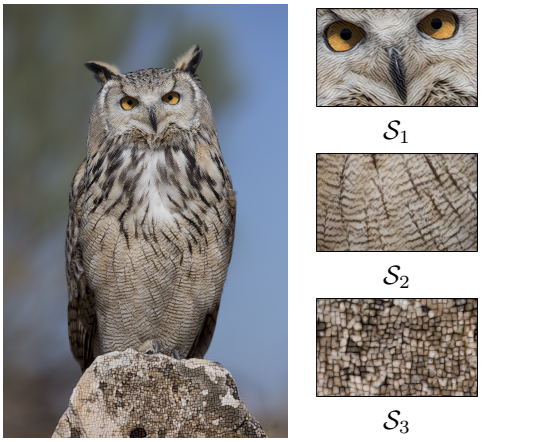


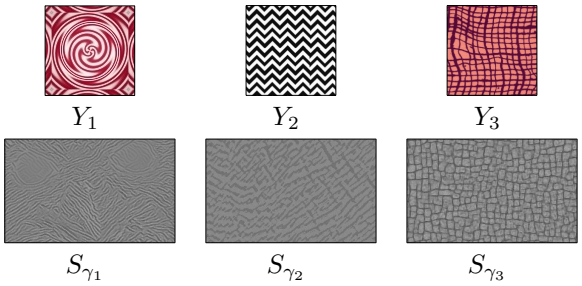

Fig. 12: Different combinations of hyper-parameters for the style network architecture trained for the same style image $Y_j$. Depth ($D$) refers to the number of modules $T_{\gamma_j}$, Width ($W$) to the number of features per module, and # to the number of parameters.

*b)* Style networks *for image enhancement:* In our restoration model *Style networks* are defined independently from the main network. We consider here their use when combined with another model or when directly applied to high-resolution images. In such a case, the style network are used for image enhancement, as illustrated in Figure 13, to create sharper images with additional high-frequency details.




Original image and custom enhanced output

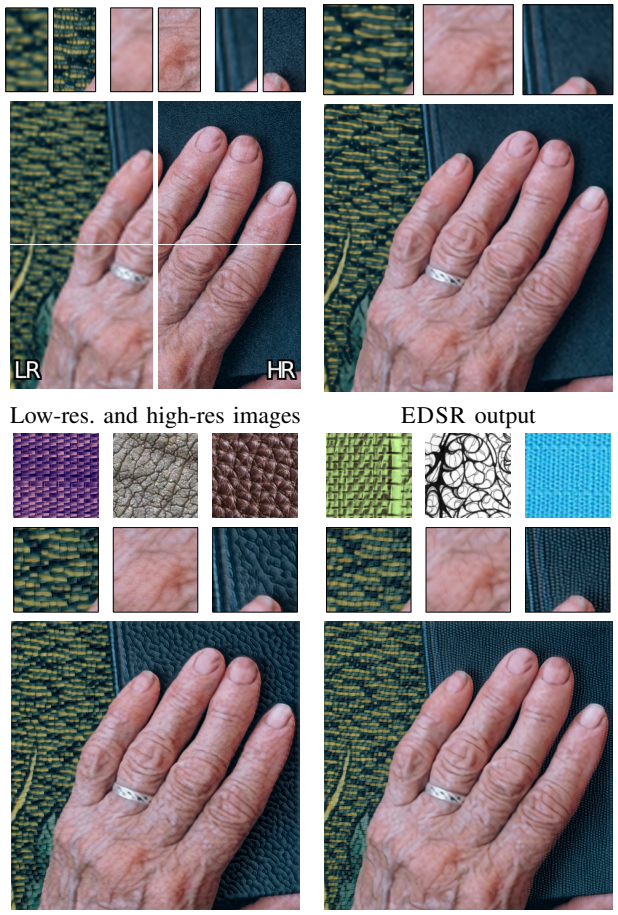



Zoom on additive enhancement layers and associated styles $Y_j$

Fig. 13: Illustration of *style networks* trained and infered for enhancing high-resolution images.

*c) Training* style network *with different* restoration networks*:* More generally, *Restoration network* introduced in Sec. III may be replaced with any other model. *Style networks* are trained here on the top of the EDSR [22] network (performance already measured in Table I and displayed in Figure 6) for the super-resolution $\times 4$. As previously done on the top of our *restoration network*, EDSR parameters are frozen and only the *style networks* parameters are trainable. Figure 14 shows results from *style networks* trained and infered with EDSR [22] on a *Div2K* [36] image. EDSR by itself gives already satisfying results (*second image*). Yet, observe how our style branches, very lightweight, improve qualitatively the image with some control, even on larger and deeper models.




Low-res. and high-res images EDSR output

*Style networks* results and local associated styles $Y_j$

Fig. 14: Different results for *style networks* trained and infered on the top of the EDSR [22] super resolution network.

## V. Conclusion

We have proposed a generic lightweight and shallow architecture composed of a multi-scale *restoration network*, combined with high resolution *style networks*. These modules can be trained independently and used in a complementary way. The proposed method allow to restore the main structures of images affected by different types of degradation and let the user generate the desired missing details.

The *restoration network* consists in a multi-scale convolutional neural network with only 200K parameters and 70K parallel multiply-adds operations per pixel. Thanks to a multi-scale decomposition, this shallow model focuses into restoring medium frequencies.

The complementary and very light *style networks* (with only 50K parameters and 100K multiply-adds computations per pixel) are trained to generate the missing high-frequency details that cannot be recovered from the degraded input image, based on style transfer techniques. Thanks to the synthesis by additive layers, the user can interact with the style models to adjust the localisation, intensity, orientation and scale of the synthesized details. While all these modules are operating at the same scale, a natural extension would be to consider multi-scale style transfer as proposed recently [7].

# REFERENCES


[1] J.-F. Aujol, G. Gilboa, T. Chan, and S. Osher. Structure-texture image decomposition—modeling, algorithms, and parameter selection. *International journal of computer vision*, 67(1):111–136, 2006.

[2] N. Carlini, J. Hayes, M. Nasr, M. Jagielski, V. Sehwag, F. Tramèr, B. Balle, D. Ippolito, and E. Wallace. Extracting training data from diffusion models. *arXiv preprint arXiv:2301.13188*, 2023.

[3] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 16:2080–95, 09 2007.

[4] C. Dong, C. Loy, K. He, and X. Tang. Image Super-Resolution Using Deep Convolutional Networks. *IEEE TPAMI*, 38, Feb. 2016.

[5] C. Dong, C. C. Loy, and X. Tang. Accelerating the Super-Resolution Convolutional Neural Network. In *Proceedings of ECCV*, Aug. 2016.

[6] T. Durand, J. Rabin, and D. Tschumperlé. Shallow multi-scale network for stylized super-resolution. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2758–2762, 2021.

[7] T. Durand, J. Rabin, and D. Tschumperlé. Modular and lightweight networks for bi-scale style transfer. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 1871–1875, 2022.

[8] L. Gatys, A. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. *Advances in neural information processing systems*, 28:262–270, 2015.

[9] L. Gatys, A. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. Controlling perceptual factors in neural style transfer, 07 2017.

[10] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *NIPS*, 27:2672–2680, 2014.

[12] Y. HaCohen, R. Fattal, and D. Lischinski. Image upsampling via texture hallucination. In *2010 IEEE International Conference on Computational Photography (ICCP)*, pages 1–8. IEEE, 2010.

[13] Y. Jing, Y. Liu, Y. Yang, Z. Feng, Y. Yu, and M. Song. Stroke controllable fast style transfer with adaptive receptive fields, 2018.

[14] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *Proceedings of ECCV*, volume 9906, pages 694–711. 2016.

[15] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate Image Super-Resolution Using Very Deep Convolutional Networks. In *Proceedings of CVPR*, pages 1646–1654, 2016.

[16] J. Kim, J. K. Lee, and K. M. Lee. Deeply-Recursive Convolutional Network for Image Super-Resolution. In *Proceedings of CVPR*, pages 1637–1645, Las Vegas, NV, USA, June 2016. IEEE.

[17] K.Simonyan and A.Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[18] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5835–5843, Honolulu, HI, July 2017. IEEE.

[19] M. Lebrun. An Analysis and Implementation of the BM3D Image Denoising Method. *Image Processing On Line*, 2:175–213, 2012. https://doi.org/10.5201/ipol.2012.l-bm3d.

[20] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, 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*, pages 4681–4690, 2017.

[21] S. Lefkimmiatis. Non-local color image denoising with convolutional neural networks. pages 5882–5891, 07 2017.

[22] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced Deep Residual Networks for Single Image Super-Resolution. In *Proceedings of CVPR Workshops*, July 2017.

[23] T. Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, 21:224–270, 09 1994.

[24] D. Liu, B. Wen, X. Liu, Z. Wang, and T. S. Huang. When image denoising meets high-level vision tasks: A deep learning approach. In *IJCAI*, 2018.

[25] D. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 2004.

[26] A. Lugmayr, M. Danelljan, and R. Timofte. Ntire 2021 learning the super-resolution space challenge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 596–612, June 2021.

[27] A. Radford, J. Kim, C. Hallacy, A. Ramesh, G. Goh, and S. A. et al. Learning transferable visual models from natural language supervision. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763, 18–24 Jul 2021.

[28] Y. Romano, J. Isidoro, and P. Milanfar. Raisr: Rapid and accurate image super resolution. *IEEE Transactions on Computational Imaging*, PP, 06 2016.

[29] M. Sajjadi, B. Scholkopf, and M. Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *Proceedings of ICCV*, pages 4491–4500, 2017.

[30] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[31] 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 *Proceedings of CVPR*, pages 1874–1883, Las Vegas, NV, USA, June 2016. IEEE.

[32] G. Somepalli, V. Singla, M. Goldblum, J. Geiping, and T. Goldstein. Diffusion art or digital forgery? investigating data replication in diffusion models. *arXiv preprint arXiv:2212.03860*, 2022.

[33] X. Tao, H. Gao, X. Shen, J. Wang, and J. Jia. Scale-recurrent network for deep image deblurring. pages 8174–8182, 06 2018.

[34] M. Tassano, J. Delon, and T. Veit. An Analysis and Implementation of the FFDNet Image Denoising Method. *Image Processing On Line*, 9:1–25, 2019. https://doi.org/10.5201/ipol.2019.231.

[35] C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo, and C.-W. Lin. Deep learning on image denoising: An overview. *Neural Networks*, 131:251–275, 2020.

[36] R. Timofte, S. Gu, J. Wu, L. Van Gool, L. Zhang, M.-H. Yang, M. Haris, et al. Ntire 2018 challenge on single image super-resolution: Methods and results. In *Proceedings of CVPR Workshops*, June 2018.

[37] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, volume 1, page 4, 2016.

[38] Q. Xu, C. Zhang, and l. Zhang. Denoising convolutional neural network. pages 1184–1187, 08 2015.

[39] F. Yang, H. Yang, J. Fu, H. Lu, and B. Guo. Learning Texture Transformer Network for Image Super-Resolution. In *Proceedings of CVPR*, pages 5790–5799, Seattle, WA, USA, June 2020. IEEE.

[40] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5505–5514, 2018.

[41] B. Zhang, J. M. Fadili, and J.-L. Starck. Wavelets, ridgelets, and curvelets for poisson noise removal. *IEEE Transactions on Image Processing*, 17(7):1093–1108, 2008.

[42] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[43] K. Zhang, W. Zuo, and L. Zhang. Ffdnet: Toward a fast and flexible solution for CNN based image denoising. *IEEE Transactions on Image Processing*, 2018.

[44] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu. Residual dense network for image super-resolution. In *Proceedings of CVPR*, pages 2472–2481, 2018.

[45] Z. Zhang, Z. Wang, Z. Lin, and H. Qi. Image super-resolution by neural texture transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[46] H. Zheng, M. Ji, L. Han, Z. Xu, H. Wang, Y. Liu, and L. Fang. Learning Cross-scale Correspondence and Patch-based Synthesis for Reference-based Super-Resolution. In *Procedings of BMCV*, page 138, London, UK, 2017.