Loss Functions for Image Restoration With Neural Networks

Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz

***Abstract*—Neural networks are becoming central in several areas of computer vision and image processing and different architectures have been proposed to solve specific problems. The impact of the loss layer of neural networks, however, has not received much attention in the context of image processing: the default and virtually only choice is ℓ_2. In this paper, we bring attention to alternative choices for image restoration. In particular, we show the importance of perceptually-motivated losses when the resulting image is to be evaluated by a human observer. We compare the performance of several losses, and propose a novel, differentiable error function. We show that the quality of the results improves significantly with better loss functions, even when the network architecture is left unchanged.**

***Index Terms*—Image processing, image restoration, neural networks, loss functions.**

I. Introduction

FOR decades, neural networks have shown various degrees of success in several fields, ranging from robotics, to regression analysis, to pattern recognition. Despite the promising results already produced in the 1980s on handwritten digit recognition [1], the popularity of neural networks in the field of computer vision has grown exponentially only recently, when deep learning boosted their performance in image recognition [2].

In the span of just a couple of years, neural networks have been employed for virtually every computer vision and image processing task known to the research community. Much research has focused on the definition of new architectures that are better suited to a specific problem [3], [4]. A large effort was also made to understand the inner mechanisms of neural networks, and what their intrinsic limitations are, for instance through the development of deconvolutional networks [5], or trying to fool networks with specific inputs [6]. Other advances were made on the techniques to improve the network's convergence [7].

The loss layer, despite being the effective driver of the network's learning, has attracted little attention within the image processing research community: the choice of the cost function generally defaults to the squared ℓ_2 norm of the error [3], [8]–[10]. This is understandable, given the many desirable properties this norm possesses. There is also a less well-founded, but just as relevant reason for the continued popularity of ℓ_2: standard neural networks packages, such as Caffe [11], only offer the implementation for this metric.

However, ℓ_2 suffers from well-known limitations. For instance, when the task at hand involves image quality, ℓ_2 correlates poorly with image quality as perceived by a human observer [12]. This is because of a number of assumptions implicitly made when using ℓ_2. First and foremost, the use of ℓ_2 assumes that the impact of noise is independent of the local characteristics of the image. On the contrary, the sensitivity of the Human Visual System (HVS) to noise depends on local luminance, contrast, and structure [13]. The ℓ_2 loss also works under the assumption of white Gaussian noise, which is not valid in general.

We focus on the use of neural networks for image restoration tasks, and we study the effect of different metrics for the network's loss layer. We compare ℓ_2 against four error metrics on representative tasks: image super-resolution, JPEG artifacts removal, and joint denoising plus demosaicking. First, we test whether a different local metric such as ℓ_1 can produce better results. We then evaluate the impact of perceptually-motivated metrics. We use two state-of-the-art metrics for image quality: the structural similarity index (SSIM [13]) and the multi-scale structural similarity index (MS-SSIM [14]). We choose these among the plethora of existing indexes, because they are established measures, and because they are differentiable—a requirement for the backpropagation stage. As expected, on the use cases we consider, the perceptual metrics outperform ℓ_2. However, and perhaps surprisingly, this is also true for ℓ_1, see Fig. 1. Inspired by this observation, we propose a novel loss function and show its superior performance in terms of all the metrics we consider.

We offer several contributions. First we bring attention to the importance of the error metric used to train neural networks for image processing: despite the widely-known limitations of ℓ_2, this loss is still the *de facto* standard. We investigate the use of three alternative error metrics (ℓ_1, SSIM, and MS-SSIM), and

Manuscript received July 22, 2016; revised November 4, 2016 and December 11, 2016; accepted December 12, 2016. Date of publication December 23, 2016; date of current version February 8, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Brendt Wohlberg.

H. Zhao is with NVIDIA Corporation, Santa Clara, CA 95050 USA, and also with the MIT Media Lab, Cambridge, MA 02139 USA (e-mail: hangzhao@mit.edu).

O. Gallo, I. Frosio, and J. Kautz are with the NVIDIA Corporation, Santa Clara, CA 95050 USA (e-mail: ogallo@nvidia.com; ifrosio@nvidia.com; jkautz@nvidia.com).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. The material includes additional numerical and visual comparisons, and mathematical derivations. Contact ogallo@nvidia.com for further questions about this work.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCI.2016.2644865

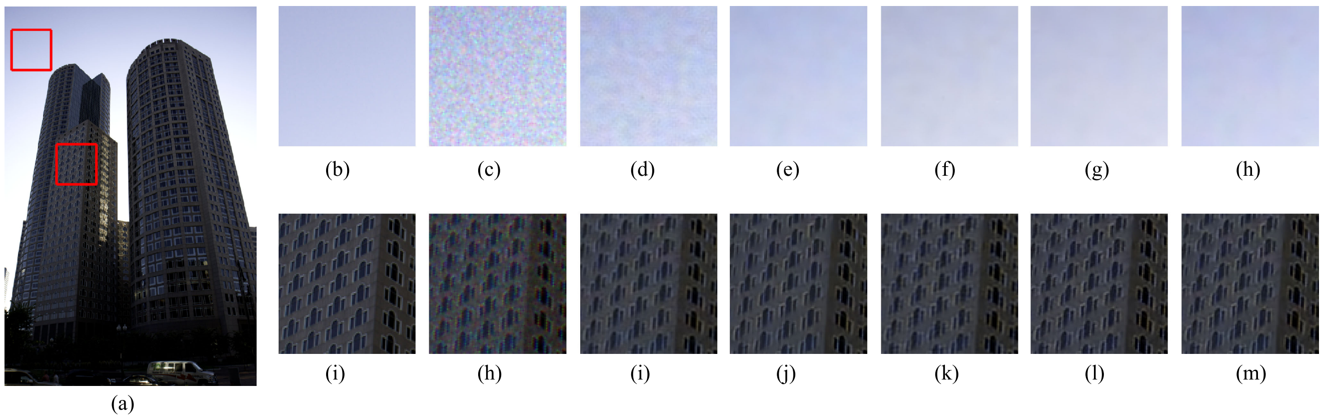


Fig. 1. Comparisons of the results of joint denoising and demosaicking performed by networks trained on different loss functions (best viewed in the electronic version by zooming in). ℓ_2, the standard loss function for neural networks for image processing, produces splotchy artifacts in flat regions (d). (a) Clean image. (b) Clean. (c) Noisy. (d) ℓ_2. (e) ℓ_1. (f) SSIM. (g) MS-SSIM. (h) Mix. (i) Clean. (j) Noisy. (k) ℓ_2. (l) ℓ_1. (m) SSIM. (n) MS-SSIM. (o) Mix.

define a new metric that combines the advantages of ℓ_1 and MS-SSIM (Section III). We also perform a thorough analysis of their performance in terms of several image quality indexes (Section IV). Finally, we discuss their convergence properties. We empirically show that the poor performance of some losses is related to local minima of the loss functions (Section V-A), and we explain the reasons why SSIM and MS-SSIM alone do not produce the expected quality (Section V-B). For each of the metrics we analyze, we implement a loss layer for Caffe, which we make available to the research community.[1]

II. Related Work

In this paper, we target neural networks for image restoration, which, in our context, is the set of all the image processing algorithms whose goal is to output an image that is appealing to a human observer. To this end, we use the problems of super-resolution, JPEG artifacts removal, and joint demosaicking plus denoising as benchmark tests. Specifically, we show how established error measures can be adapted to work within the loss layer of a neural network, and how this can positively influence the results. Here we briefly review the existing literature on the subject of neural networks for image processing, and on the subject of measures of image quality.

A. Neural Networks for Image Restoration

Following their success in several computer vision tasks [2], [15], neural networks have received considerable attention in the context of image restoration. Neural networks have been used for denoising [3], [8], deblurring [4], demosaicking [10], and super-resolution [9] among others. To the best of our knowledge, however, the work on this subject has focused on tuning the architecture of the network for the specific application; the loss layer, which effectively drives the learning of the network to produce the desired output quality, is based on ℓ_2 for all of the approaches above.

We show that a better choice for the error measure has a strong impact on the quality of the results. Moreover, we show that even when ℓ_2 is the appropriate loss, alternating the training loss function with a related loss, such as ℓ_1, can lead to finding a better solution for ℓ_2.

B. Evaluating Image Quality

The mean squared error, ℓ_2, is arguably the dominant error measure across very diverse fields, from regression problems, to pattern recognition, to signal and image processing. Among the main reasons for its popularity is the fact that it is convex and differentiable—very convenient properties for optimization problems. Other interesting properties range from the fact that ℓ_2 provides the maximum likelihood estimate in case of independent and identically distributed Gaussian noise, to the fact that it is additive for independent noise sources. There is longer list of reasons for which we refer the reader to the work of Wang and Bovik [16].

These properties paved the way for ℓ_2's widespread adoption, which was further fueled by the fact that standard software packages tend to include tools to use ℓ_2, but not many other error functions for regression. In the context of image processing, Caffe [11] actually offers *only* ℓ_2 as a loss layer,[2] thus discouraging researchers from testing other error measures.

However, it is widely accepted that ℓ_2, and consequently the Peak Signal-to-Noise Ratio, PSNR, do not correlate well with human's perception of image quality [12]: ℓ_2 simply does not capture the intricate characteristics of the human visual system (HVS).

There exists a rich literature of error measures, both reference-based and non reference-based, that attempt to address the limitations of the simple ℓ_2 error function. For our purposes, we focus on reference-based measures. A popular reference-based index is the structural similarity index (SSIM [13]). SSIM evaluates images accounting for the fact that the HVS is sensitive to changes in local structure. Wang *et al.* [14] extend SSIM observing that the scale at which local structure should be analyzed is a function of factors such as image-to-observer distance. To

[1]https://github.com/NVlabs/PL4NN

[2]Caffe indeed offers other types of loss layers, but they are only useful for classification tasks.

account for these factors, they propose MS-SSIM, a multi-scale version of SSIM that weighs SSIM computed at different scales according to the sensitivity of the HVS. Experimental results have shown the superiority of SSIM-based indexes over ℓ_2. As a consequence, SSIM has been widely employed as a metric to evaluate image processing algorithms. Moreover, given that it can be used as a differentiable cost function, SSIM has also been used in iterative algorithms designed for image compression [16], image reconstruction [17], denoising and super-resolution [18], and even downscaling [19]. To the best of our knowledge, however, SSIM-based indexes have never been adopted to train neural networks.

Recently, novel image quality indexes based on the properties of the HVS showed improved performance when compared to SSIM and MS-SSIM [12]. One of these is the Information Weigthed SSIM (IW-SSIM), a modification of MS-SSIM that also includes a weighting scheme proportional to the local image information [20]. Another is the Visual Information Fidelity (VIF), which is based on the amount of shared information between the reference and distorted image [21]. The Gradient Magnitude Similarity Deviation (GMSD) is characterized by simplified math and performance similar to that of SSIM, but it requires computing the standard deviation over the whole image [22]. Finally, the Feature Similarity Index (FSIM), leverages the perceptual importance of phase congruency, and measures the dissimilarity between two images based on local phase congruency and gradient magnitude [23]. FSIM has also been extended to FSIM$_c$, which can be used with color images. Despite the fact that they offer an improved accuracy in terms of image quality, the mathematical formulation of these indexes is generally more complex than SSIM and MS-SSIM, and possibly not differentiable, making their adoption for optimization procedures not immediate.

III. Loss Layers for Image Restoration

The loss layer of a neural network compares the output of the network with the ground truth, i.e., processed and reference patches, respectively, for the case of image processing.

In our work, we investigate the impact of different loss function layers for image processing. Consider the case of a network that performs denoising and demosaicking jointly. The insets in Fig. 1 show a zoom-in of different patches for the image in Fig. 1(a) as processed by a network trained with different loss functions (see Section IV for the network's description). A simple visual inspection is sufficient to appreciate the practical implications of the discussion on ℓ_2 (Section II).

Specifically, Fig. 1(d) shows that in flat regions the network strongly attenuates the noise, but it produces visible splotchy artifacts. This is because ℓ_2 penalizes larger errors, but is more tolerant to small errors, regardless of the underlying structure in the image; the HVS, on the other hand, is more sensitive to luminance and color variations in texture-less regions [24]. A few splotchy artifacts are still visible, though arguably less apparent, in textured regions, see Fig. 1(k). The sharpness of edges, however, is well-preserved by ℓ_2, as blurring them would result in a large error. Note that these splotchy artifacts have been systematically observed before in the context of image processing with neural networks [3], but they have not been attributed to the loss function. In Section V-A we show that the quality achieved by using ℓ_2 is also dependent on its convergence properties.

In this section we propose the use of different error functions. We provide a motivation for the different loss functions and we show how to compute their derivatives, which are necessary for the backpropagation step. We also share our implementation of the different layers that can be readily used within Caffe.

For an error function $\mathcal{E}$, the loss for a patch P can be written as

$$\mathcal{L}^{\mathcal{E}}(P) = \frac{1}{N}\sum_{p\in P}\mathcal{E}(p), \tag{1}$$

where N is the number of pixels p in the patch.

A. *The ℓ_1 Error*

As a first attempt to reduce the artifacts introduced by the ℓ_2 loss function, we want to train the exact same network using ℓ_1 instead of ℓ_2. The two losses weigh errors differently—ℓ_1 does not over-penalize larger errors—and, consequently, they may have different convergence properties.

Equation (1) for ℓ_1 is simply:

$$\mathcal{L}^{\ell_1}(P) = \frac{1}{N}\sum_{p\in P}\left|x(p) - y(p)\right|, \tag{2}$$

where p is the index of the pixel and P is the patch; $x(p)$ and $y(p)$ are the values of the pixels in the processed patch and the ground truth respectively. The derivatives for the backpropagation are also simple, since $\partial\mathcal{L}^{\ell_1}(p)/\partial q = 0, \forall q \neq p$. Therefore, for each pixel p in the patch,

$$\partial\mathcal{L}^{\ell_1}(P)/\partial x(p) = \text{sign}\left(x(p) - y(p)\right). \tag{3}$$

The derivative of $\mathcal{L}^{\ell_1}$ is not defined at 0. However, if the error is 0, we do not need to update the weights, so we use the convention that $\text{sign}(0) = 0$. Note that, although $\mathcal{L}^{\ell_1}(P)$ is a function of the patch as a whole, the derivatives are back-propagated for each pixel in the patch. Somewhat unexpectedly, the network trained with ℓ_1 provides a significant improvement for several of the issues discussed above, see Fig. 1(e) where the splotchy artifacts in the sky are removed. Section V-A analyzes the reasons behind this behavior. Although ℓ_1 shows improved performance over ℓ_2, however, its results are still sub-optimal [see for instance the artifacts in the sky and at the boundary of the building in Fig. 3(c) and (g)].

B. *SSIM*

If the goal is for the network to learn to produce visually pleasing images, it stands to reason that the error function should be perceptually motivated, as is the case with SSIM.

SSIM for pixel p is defined as

$$\begin{aligned}\text{SSIM}(p) &= \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}\cdot\frac{2\sigma_{xy} + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} && (4)\\ &= l(p)\cdot cs(p) && (5)\end{aligned}$$

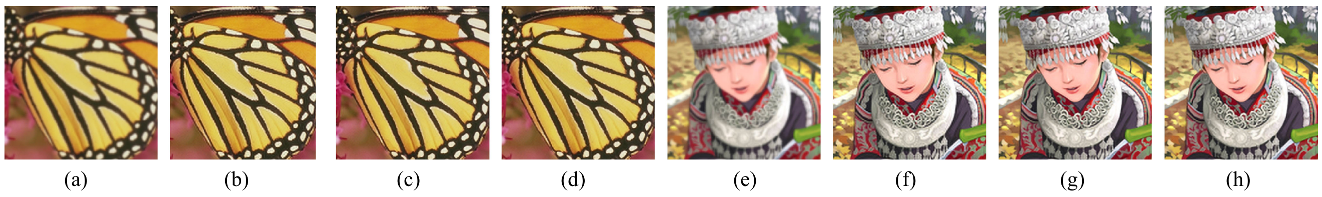


Fig. 2. Results for super-resolution. Notice the grating artifacts on the black stripes of the wing and around the face of the girl produced by ℓ_2. (a) LR. (b) ℓ_2. (c) ℓ_1. (d) Mix. (e) LR. (f) ℓ_2. (g) ℓ_1. (h) Mix.

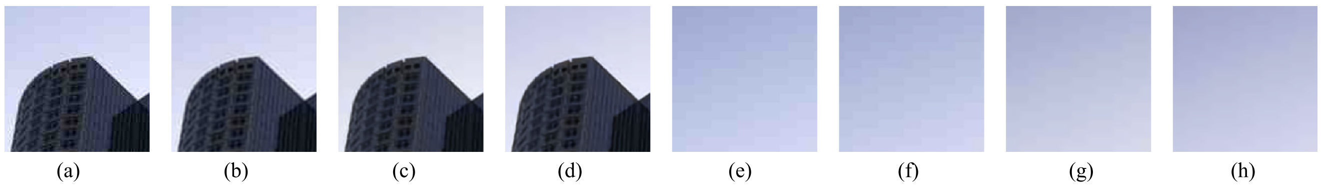


Fig. 3. Results for JPEG de-blocking. The insets are taken from the image in Fig. 1. Notice the artifacts around the edges (a)-(c) and how Mix (d) removes them better than either ℓ_1 or ℓ_2. Mix also outperforms the other metrics in the relatively flat regions, where the blocketization is more apparent, e.g., (e)-(h). (a) JPEG. (b) ℓ_2. (c) ℓ_1. (d) Mix. (e) JPEG. (f) ℓ_2. (g) ℓ_1. (h) Mix.

where we omitted the dependence of means and standard deviations on pixel p. Means and standard deviations are computed with a Gaussian filter with standard deviation σ_G, G_{σ_G}. The loss function for SSIM can be then written setting $\mathcal{E}(p) = 1 - \text{SSIM}(p)$:

$$\mathcal{L}^{\text{SSIM}}(P) = \frac{1}{N}\sum_{p\in P} 1 - \text{SSIM}(p). \tag{6}$$

Equation (4) highlights the fact that the computation of SSIM(p) requires looking at a neighborhood of pixel p as large as the support of G_{σ_G}. This means that $\mathcal{L}^{\text{SSIM}}(P)$, as well as its derivatives, cannot be calculated in some boundary region of P. This is not true for ℓ_1 or ℓ_2, which only need the value of the processed and reference patch at pixel p.

However, the convolutional nature of the network allows us to write the loss as

$$\mathcal{L}^{\text{SSIM}}(P) = 1 - \text{SSIM}(\tilde{p}), \tag{7}$$

where $\tilde{p}$ is the center pixel of patch P. Again, this is because, even though the network learns the weights maximizing SSIM for the central pixel, the learned kernels are then applied to all the pixels in the image. Note that the error can still be back-propagated to all the pixels within the support of G_{σ_G} as they contribute to the computation of (7).

Recall that we have to compute the derivatives at $\tilde{p}$ with respect to any other pixel q in patch P. More formally:

$$\begin{aligned}\frac{\partial \mathcal{L}^{\text{SSIM}}(P)}{\partial x(q)} &= -\frac{\partial}{\partial x(q)}\text{SSIM}(\tilde{p}) \\ &= -\left(\frac{\partial l(\tilde{p})}{\partial x(q)}\cdot cs(\tilde{p}) + l(\tilde{p})\cdot\frac{\partial cs(\tilde{p})}{\partial x(q)}\right),\end{aligned} \tag{8}$$

where $l(\tilde{p})$ and $cs(\tilde{p})$ are the first and second term of SSIM (5) and their derivatives are

$$\frac{\partial l(\tilde{p})}{\partial x(q)} = 2\cdot G_{\sigma_G}(q-\tilde{p})\cdot\left(\frac{\mu_y - \mu_x\cdot l(\tilde{p})}{\mu_x^2+\mu_y^2+C_1}\right) \tag{9}$$

and

$$\begin{aligned}\frac{\partial cs(\tilde{p})}{\partial x(q)} = &\frac{2}{\sigma_x^2+\sigma_y^2+C_2}\cdot G_{\sigma_G}(q-\tilde{p})\cdot\big[(y(q)-\mu_y) \\ &- cs(\tilde{p})\cdot(x(q)-\mu_x)\big],\end{aligned} \tag{10}$$

where $G_{\sigma_G}(q-\tilde{p})$ is the Gaussian coefficient associated with pixel q. We refer the reader to the additional material for the full derivation.

C. MS-SSIM

The choice of σ_G has an impact on the quality of the processed results of a network that is trained with SSIM, as can be seen from the derivatives in the previous section. Specifically, for smaller values of σ_G the network loses the ability to preserve the local structure and the splotchy artifacts are reintroduced in flat regions, see Fig. 9(e). For large values of σ_G, we observe that the network tends to preserve noise in the proximity of edges, Fig. 9(c). See Section V-B for more details.

Rather than fine-tuning the σ_G, we propose to use the multi-scale version of SSIM, MS-SSIM. Given a dyadic pyramid of M levels, MS-SSIM is defined as

$$\text{MS-SSIM}(p) = l_M^{\alpha}(p)\cdot\prod_{j=1}^{M} cs_j^{\beta_j}(p) \tag{11}$$

where l_M and cs_j are the terms we defined in Section III-B at scale M and j, respectively. For convenience, we set $\alpha = \beta_j = 1$, for $j = \{1, ..., M\}$. Similarly to (7), we can approximate the loss for patch P with the loss computed at its center pixel $\tilde{p}$:

$$\mathcal{L}^{\text{MS-SSIM}}(P) = 1 - \text{MS-SSIM}(\tilde{p}). \tag{12}$$

Because we set all the exponents of (11) to one, the derivatives of the loss function based on MS-SSIM can be written as

$$\frac{\partial \mathcal{L}^{\text{MS-SSIM}}(P)}{\partial x(q)} = \left(\frac{\partial l_M(\tilde{p})}{\partial x(q)} + l_M(\tilde{p}) \cdot \sum_{i=0}^{M} \frac{1}{cs_i(\tilde{p})} \frac{\partial cs_i(\tilde{p})}{\partial x(q)} \right) \cdot \prod_{j=1}^{M} cs_j(\tilde{p}), \tag{13}$$

where the derivatives of l_j and cs_j are the same as in Section III-B. For the full derivation we refer the reader to the supplementary material.

Using $\mathcal{L}^{\text{MS-SSIM}}$ to train the network, (11) requires that we compute a pyramid of M levels of patch P, which is a computationally expensive operation given that it needs to be performed at each iteration. To avoid this, we propose to approximate and replace the construction of the pyramid: instead of computing M levels of the pyramid, we use M different values for σ_G, each one being half of the previous, on the full-resolution patch. Specifically, we use $\sigma_G^i = \{0.5, 1, 2, 4, 8\}$ and define $cs_i \triangleq G_{\sigma_G^i} \cdot cs_0(\tilde{p})$ and $\partial cs_i(\tilde{p})/\partial x(q) \triangleq G_{\sigma_G^i} \cdot \partial cs_0(\tilde{p})/\partial x(q)$, where the Gaussian filters are centered at pixel $\tilde{p}$, and "$\cdot$" is a point-wise multiplication. The terms depending on l_M can be defined in a similar way. We use this trick to speed up the training in all of our experiments.

D. *The Best of Both Worlds: MS-SSIM + ℓ_1*

By design, both MS-SSIM and SSIM are not particularly sensitive to uniform biases (see Section V-B). This can cause changes of brightness or shifts of colors, which typically become more dull. However, MS-SSIM preserves the contrast in high-frequency regions better than the other loss functions we experimented with. On the other hand, ℓ_1 preserves colors and luminance—an error is weighed equally regardless of the local structure—but does not produce quite the same contrast as MS-SSIM.

To capture the best characteristics of both error functions, we propose to combine them:

$$\mathcal{L}^{\text{Mix}} = \alpha \cdot \mathcal{L}^{\text{MS-SSIM}} + (1-\alpha) \cdot G_{\sigma_G^M} \cdot \mathcal{L}^{\ell_1}, \tag{14}$$

where we omitted the dependence on patch P for all loss functions, and we empirically set $\alpha = 0.84$.[3] The derivatives of (14) are simply the weighed sum of the derivatives of its two terms, which we compute in the previous sections. Note that we add a point-wise multiplication between $G_{\sigma_G^M}$ and $\mathcal{L}^{\ell_1}$: this is because MS-SSIM propagates the error at pixel q based on its contribution to MS-SSIM of the central pixel $\tilde{p}$, as determined by the Gaussian weights, see (9) and (10).

IV. Results

For our analysis of the different loss functions we focus on joint demosaicking plus denoising, a fundamental problem in image processing. We also confirm our findings by testing the different loss functions on the problems of super-resolution and JPEG artifacts removal.

[3]We chose the value for this parameter so that the contribution of the two losses would be roughly balanced. While we tested a few different values, we did not perform a thorough investigation. We did, however, notice that the results were not significantly sensitive to small variations of α.

A. *Joint Denoising and Demosaicking*

We define a fully convolutional neural network (CNN) that takes a 31 × 31 × 3 input. The first layer is a 64 × 9 × 9 × 3 convolutional layer, where the first term indicates the number of filters and the remaining terms indicate their dimensions. The second convolutional layer is 64 × 5 × 5 × 64, and the output layer is a 3 × 5 × 5 × 64 convolutional layer. We apply parametric rectified linear unit (PReLU) layers to the output of the inner convolutional layers [15]. The input to our network is obtained by bilinear interpolation of a 31 × 31 Bayer patch, which results in a 31 × 31 × 3 RGB patch; in this sense the network is really doing joint denoising + demosaicking and super-resolution. We trained the network considering different cost functions (ℓ_1, ℓ_2, SSIM$_5$, SSSIM$_9$, MS-SSIM and MS-SSIM+ℓ_1)[4] on a training set of 700 RGB images taken from the MIT-Adobe FiveK Dataset [25], resized to a size of 999 × 666. To simulate a realistic image acquisition process, we corrupted each image with a mix of photon shot and zero-mean Gaussian noise, and introduced clipping due to the sensor zero level and saturation. We used the model proposed by Foi *et al.* [26] for this task, with parameters $a = 0.005$ and $b = 0.0001$. The average PSNR for the testing images after adding noise was 28.24 dB, as reported in Table I. Fig. 1(c) shows a typical patch corrupted by noise. We used 40 images from the same dataset for testing (the network did not see this subset during training).

Beyond considering different cost functions for training, we also compare the output of our network with the images obtained by a state-of-the-art denoising method, BM3D, operating directly in the Bayer domain [27], followed by a standard demosaicking algorithm [28]. Since BM3D is designed to deal with Gaussian noise, rather than the more realistic noise model we use, we apply a Variance Stabilizing Transform [26] to the image data in Bayer domain before applying BM3D, and its inverse after denoising. This is exactly the strategy suggested by the authors of BM3D for RAW data [27], the paper we compare against for BM3D.

Figs. 1 and 5 show several results and comparisons between the different networks. Note the splotchy artifacts for the ℓ_2 network on flat regions, and the noise around the edges for the SSIM$_5$ and SSIM$_9$ networks. The network trained with MS-SSIM addresses these problems but tends to render the colors more dull, see Section V-B for an explanation. The network trained with MS-SSIM+ℓ_1 generates the best results. Some differences are difficult to see in side-by-side comparisons, please refer to the additional material, which allows to flip between images.

We also perform a quantitative analysis of the results. We evaluate several image quality metrics on the output of the CNNs trained with the different cost functions and with BM3D [27], which is the state of the art in denoising. The image quality indexes, range from the traditional ℓ_2 metric and PSNR, to the

[4]SSIM$_k$ means SSIM computed with $\sigma_G = k$.

TABLE I
AVERAGE VALUE OF DIFFERENT IMAGE QUALITY METRICS ON THE TESTING DATASET FOR THE DIFFERENT COST FUNCTIONS. FOR SSIM, MS-SSIM, IW-SSIM, GMSD AND FSIM THE VALUE REPORTED HERE HAS BEEN OBTAINED AS AN AVERAGE OF THE THREE COLOR CHANNELS. BEST RESULTS ARE SHOWN IN BOLD. (LOWER IS BETTER FOR ℓ_1, ℓ_2, AND GMSD, HIGHER IS BETTER FOR THE OTHERS.)

Denoising + demosaicking			Training cost function					
Image quality metric	Noisy	$BM3D$	ℓ_2	ℓ_1	SSIM$_5$	SSIM$_9$	MS-SSIM	Mix
$1000 \cdot \ell_2$	1.65	0.45	0.56	0.43	0.58	0.61	0.55	**0.41**
PSNR	28.24	34.05	33.18	34.42	33.15	32.98	33.29	**34.61**
$1000 \cdot \ell_1$	27.36	14.14	15.90	13.47	15.90	16.33	15.99	**13.19**
SSIM	0.8075	0.9479	0.9346	0.9535	0.9500	0.9495	0.9536	**0.9564**
MS-SSIM	0.8965	0.9719	0.9636	0.9745	0.9721	0.9718	0.9741	**0.9757**
IW-SSIM	0.8673	0.9597	0.9473	0.9619	0.9587	0.9582	0.9617	**0.9636**
GMSD	0.1229	0.0441	0.0490	0.0434	0.0452	0.0467	0.0437	**0.0401**
FSIM	0.9439	0.9744	0.9716	0.9775	0.9764	0.9759	0.9782	**0.9795**
FSIM$_c$	0.9381	0.9737	0.9706	0.9767	0.9752	0.9746	0.9769	**0.9788**

Super-resolution		Training cost function			
Image quality metric	Bilinear	ℓ_2	ℓ_1	MS-SSIM	Mix
$1000 \cdot \ell_2$	2.5697	1.2407	1.1062	1.3223	**1.0990**
PSNR	27.16	30.66	31.26	30.11	**31.34**
$1000 \cdot \ell_1$	28.7764	20.4730	19.0643	22.3968	**18.8983**
SSIM	0.8632	0.9274	0.9322	0.9290	**0.9334**
MS-SSIM	0.9603	0.9816	0.9826	0.9817	**0.9829**
IW-SSIM	0.9532	0.9868	0.9879	0.9866	**0.9881**
GMSD	0.0714	0.0298	0.0259	0.0316	**0.0255**
FSIM	0.9070	0.9600	0.9671	0.9601	**0.9680**
FSIM$_c$	0.9064	0.9596	0.9667	0.9597	**0.9677**

JPEG de-blocking		Training cost function			
Image quality metric	Original JPEG	ℓ_2	ℓ_1	MS-SSIM	Mix
$1000 \cdot \ell_2$	0.6463	0.6511	0.6027	1.9262	**0.5580**
PSNR	32.60	32.73	32.96	27.66	**33.25**
$1000 \cdot \ell_1$	16.5129	16.2633	16.0687	33.6134	**15.5489**
SSIM	0.9410	0.9427	0.9467	0.9364	**0.9501**
MS-SSIM	0.9672	0.9692	0.9714	0.9674	**0.9734**
IW-SSIM	0.9527	0.9562	0.9591	0.9550	**0.9625**
GMSD	0.0467	0.0427	0.0413	0.0468	**0.0402**
FSIM	0.9805	0.9803	0.9825	0.9789	**0.9830**
FSIM$_c$	0.9791	0.9790	0.9809	0.9705	**0.9815**

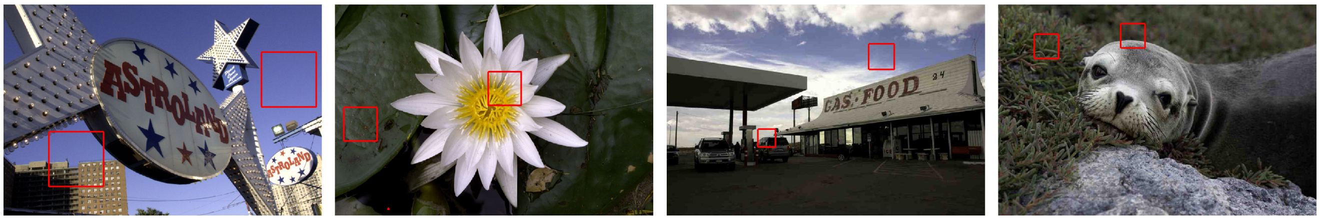


Fig. 4. The reference images for the details shown in Figs. 5 and 6.

most refined, perceptually inspired metrics, like FSIM [23]. The average values of these metrics on the testing dataset are reported in Table I. When the network is trained using ℓ_1 as a cost function, instead of the traditional ℓ_2, the average quality of the output images is superior for all the quality metrics considered. It is quite remarkable to notice that, even when the ℓ_2 or PSNR metrics are used to evaluate image quality, the network trained with the ℓ_1 loss outperforms the one trained with the ℓ_2 loss. We offer an explanation of this in Section V-A. On the other hand, we note that the network trained with SSIM performs either at par or slightly worse than the one trained with ℓ_1, both on traditional metrics and on perceptually-inspired losses. The network trained on MS-SSIM performs better than the one based on SSIM, but still does not clearly outperforms ℓ_1. This is due to the color shifts mentioned above and discussed in Section V-B. However, the network that combines MS-SSIM and ℓ_1 achieves the best results on all of the image quality metrics we consider.

For this image restoration task, we also evaluated the combination of ℓ_2 and MS-SSIM to test whether the use of ℓ_1 is critical to the performance of Mix. The results on all the indexes considered in Table I show that Mix is still superior. As expected, however, ℓ_2 + MS-SSIM outperforms ℓ_2 alone on the perceptual metrics (see supplementary material).

B. Super-Resolution

We also verify the outcome of our analysis on the network for super-resolution proposed by Dong *et al.* [9]. We start from the network architecture they propose, and make a few minor but important changes to their approach. First we use PReLU, instead of ReLU, layers. Second we use bilinear instead of bicubic interpolation for initialization (we do this for both training

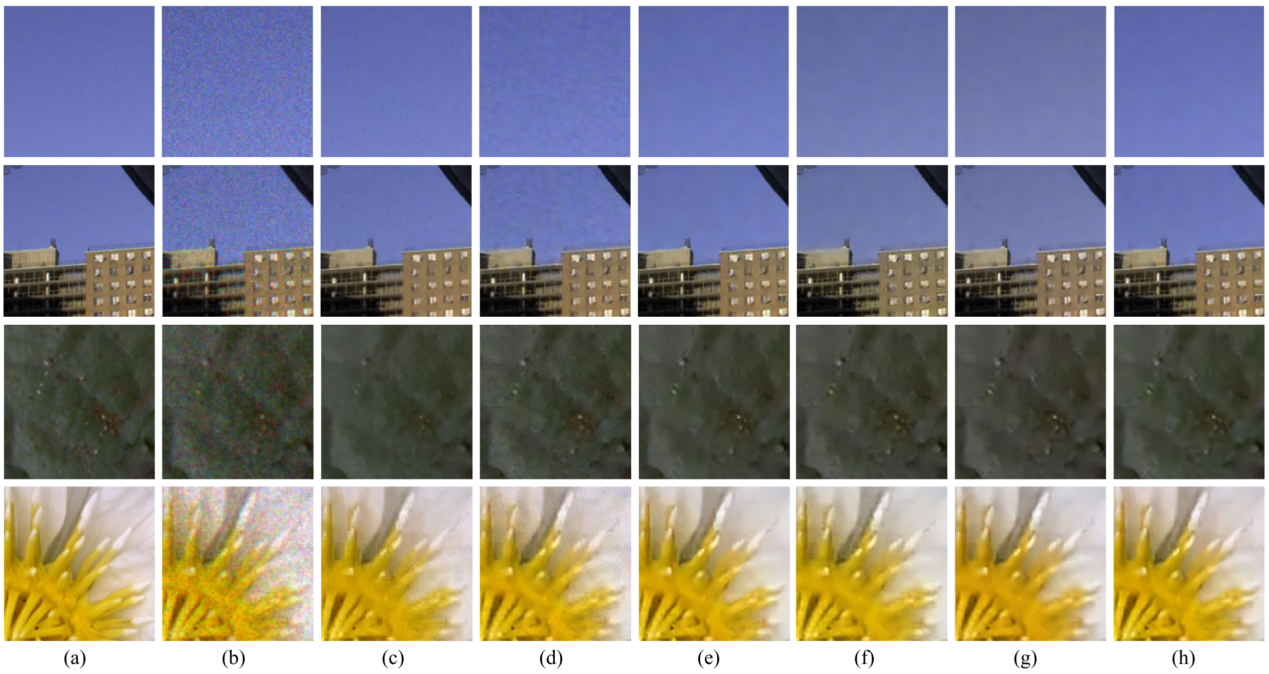


Fig. 5. Results for denosing+demosaicking for different approaches. The noisy patches are obtained by simple bilinear interpolation. Note the splotchy artifacts ℓ_2 produces in flat regions. Also note the change in colors for SSIM-based losses. The proposed metric, MS-SSIM+ℓ_1 referred to as Mix, addresses the former issues. Reference images are shown in Fig. 4. (a) Clean. (b) Noisy. (c) BM3D. (d) ℓ_2. (e) ℓ_1. (f) SSIM. (g) MS-SSIM. (h) Mix.

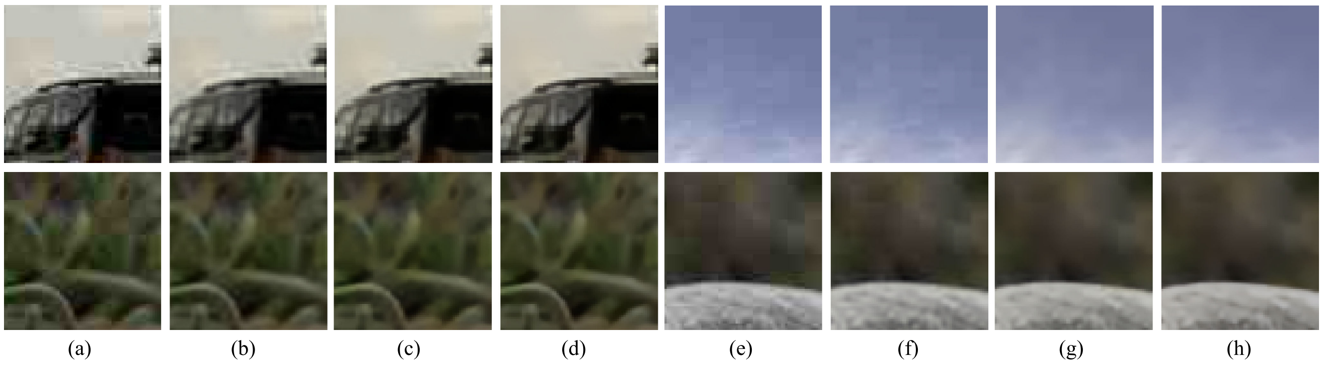


Fig. 6. Results for JPEG de-blocking for different approaches. Note that Mix outperforms both ℓ_2 and ℓ_1 on uniform regions as well as on the ringing artifacts at the in high-gradient regions. Reference images are shown in Fig. 4. (a) JPEG. (b) ℓ_2. (c) ℓ_1. (d) Mix. (e) JPEG. (f) ℓ_2. (g) ℓ_1. (h) Mix.

and testing). The latter introduces high-frequency artifacts that hurt the learning process. Finally we train directly on the RGB data. We made these changes for all the loss functions we test, including ℓ_2, which also serves as a comparison with the work by Dong *et al.* We use this modified network in place of their proposed architecture to isolate the contribution of the loss layer to the results—and because it produces better or equal results than the one produced by the original architecture. Fig. 2 shows some sample results. Given the results of the previous section, we only compared the results of ℓ_1, ℓ_2, MS-SSIM, and Mix, see Table I. An analysis of the table brings similar considerations as for the case of joint denoising and demosaicking.

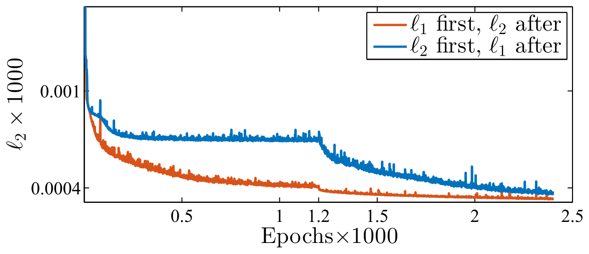


Fig. 7. ℓ_2 loss on the testing set for the two networks that switch loss functions during training.

C. JPEG Artifacts Removal

Our final benchmark is JPEG artifact removal. We use the same network architecture and ground truth data as for joint denoising and demosaicking, but we create the corrupted data by means of aggressive JPEG compression. For this purpose we use the Matlab's function *imwrite* with a quality setting of 25. Note that the corruption induced by JPEG compression is spatially variant and has a very different statistics than the noise introduced in Section IV-A. We generate the input patches with two different strides. First, we use a stride of 8, which causes

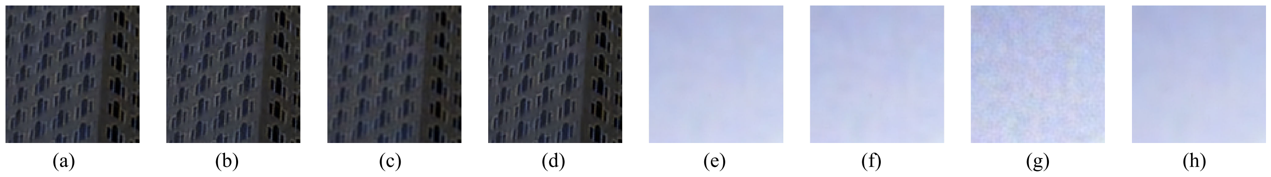


Fig. 8. Visual results of the networks trained alternating loss functions. The insets here correspond the regions marked in Fig. 1. (a) ℓ_1. (b) $\ell_1 \to \ell_2$. (c) ℓ_2. (d) $\ell_2 \to \ell_1$. (e) ℓ_1. (f) $\ell_1 \to \ell_2$. (g) ℓ_2. (h) $\ell_2 \to \ell_1$. Note that $\ell_i \to \ell_j$ indicates a network trained first with ℓ_i and then with ℓ_j.

the boundaries of the 8×8 DCT blocks to be aligned in each 31×31 patch. We also use a stride of 7, which causes the grid of blocks to be in different locations in each patch. We found that the latter strategy better removes the JPEG artifacts while producing sharper images, which is why we ran all of our experiments with that configuration. Again, we only compared the results of ℓ_1, ℓ_2, MS-SSIM, and Mix, see Table I. Fig. 3 shows that our loss function, Mix, outperforms ℓ_1 and ℓ_2 on uniform regions and that it attenuates the ringing artifacts around the edge of the building better. More results are shown in Fig. 6.

More results and comparisons, both numerical and visual, can be found in the supplementary material.

V. Discussion

In this section we delve into a deeper analysis of the results. Among other considerations, we look into the convergence properties of the different losses, and we offer an interpretation of the reasons some losses perform better than others.

A. Convergence of the Loss Functions

Table I highlights an unexpected result: even after convergence, CNNs trained on one loss function can outperform another network even based on the very loss with which the second was trained. Consider, for instance, the two networks trained with ℓ_1 and ℓ_2 respectively for joint denoising and demosaicking: the table shows that the network trained with ℓ_1 achieves a lower ℓ_2 loss than then network trained with ℓ_2. Note that we ran the training multiple times with different initializations. We hypothesize that this result may be related to the smoothness and the local convexity properties of the two measures: ℓ_2 gets stuck more easily in a local minimum, while for ℓ_1 it may be easier to reach better minimum, both in terms of ℓ_1 and ℓ_2—the "good" minima of the two should be related, after all. To test this hypothesis, we ran an experiment in which we take two networks trained with ℓ_1 and ℓ_2 respectively, and train them again until they converge using the other loss. Fig. 7 shows the ℓ_2 loss computed on the testing set at different training iterations for either network. The network trained with ℓ_1 only (before epoch 1200 in the plot) achieves a better ℓ_2 loss than the one trained with ℓ_2. However, after switching the training loss functions, both networks yield a lower ℓ_2 loss, confirming that the ℓ_2 network was previously stuck in a local minimum. While the two networks achieve a similar ℓ_2 loss, they converge to different regions of the space of parameters. At visual inspection the network trained with ℓ_2 first and ℓ_1 after produces results similar to those of ℓ_1 alone; the output of the network trained with ℓ_1 first and ℓ_2 after is still affected by splotchy artifacts in flat areas,

TABLE II
Average Value of Different Image Quality Metrics for the Networks Trained for Denoising + Demosaicking With Alternating Loss Functions. Bold Indicates That the Network Achieves a Better Score Than Any of the Networks in Table I, see Section V-A

	Training cost function	
Image quality metric	ℓ_2 first, ℓ_1 after	ℓ_1 first, ℓ_2 after
$1000 \cdot \ell_2$	0.3939	**0.3896**
PSNR	34.76	**34.77**
$1000 \cdot \ell_1$	**12.7932**	12.8919
SSIM	0.9544	0.9540
MS-SSIM	0.9753	0.9748
IW-SSIM	0.9634	0.9624
GMSD	0.0432	0.0405
FSIM	0.9777	0.9781
FSIM$_c$	0.9770	0.9774

though it is better than ℓ_2 alone, see Fig. 8. Specifically, the network trained with ℓ_1 first and ℓ_2 afterwards achieves an ℓ_2 loss of $0.3896 \cdot 10^3$, which is the lowest across all the networks we trained, confirming that the network trained with ℓ_2 alone had convergence issues. However, neither of these two networks outperforms Mix on any of the perceptual metrics we use, confirming that the advantage of the proposed loss goes beyond convergence considerations. Table II shows the complete evaluation of the two networks, where bold indicates a result that is better than any of the results shown in Table I.

B. On the Performance of SSIM and MS-SSIM

Table I also reveals that SSIM and MS-SSIM do not perform as well as ℓ_1. This does not have to do with the convergence properties of these losses. To investigate this, we trained several SSIM networks with different σ_G's and found that smaller values of σ_G produce better results at edges, but worse results in flat regions, while the opposite is true for larger values, see Fig. 9. This can be understood by looking at Fig. 10(a), where the size of the support for the computation of SSIM for the three values of σ_G is shown for a pixel P close to an edge. A larger σ_G is more tolerant to the same amount of noise because it detects the presence of the edge, which is known to have a masking effect for the HVS: in this toy example, SSIM$_9$ yields a higher value for the same pixel because its support spans both sides of the edge. Fig. 10(b) shows SSIM across the edge of the same profile of (a), and particularly at pixel P. Note that SSIM$_9$ estimates a higher quality in a larger region around the step.

Despite of the size of its support, however, SSIM is not particularly sensitive to a uniform bias on a flat region. This is particularly true in bright regions, as shown in Fig. 10(d), which

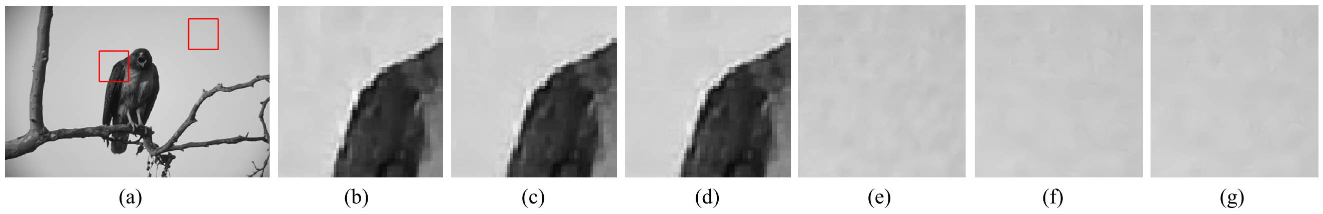


Fig. 9. Comparison of the results of networks trained with SSIM with different sigmas (SSIM$_k$ means $\sigma_G = k$). Insets (b)–(d), show an increasingly large halo of noise around the edge: smaller values of σ help at edges. However, in mostly flat regions, larger values of σ help reducing the splotchy artifacts (e)–(g). Best viewed by zooming in on the electronic copy. (a) Clean image. (b) SSIM$_1$. (c) SSIM$_3$. (d) SSIM$_9$. (e) SSIM$_1$. (f) SSIM$_3$. (g) SSIM$_9$.

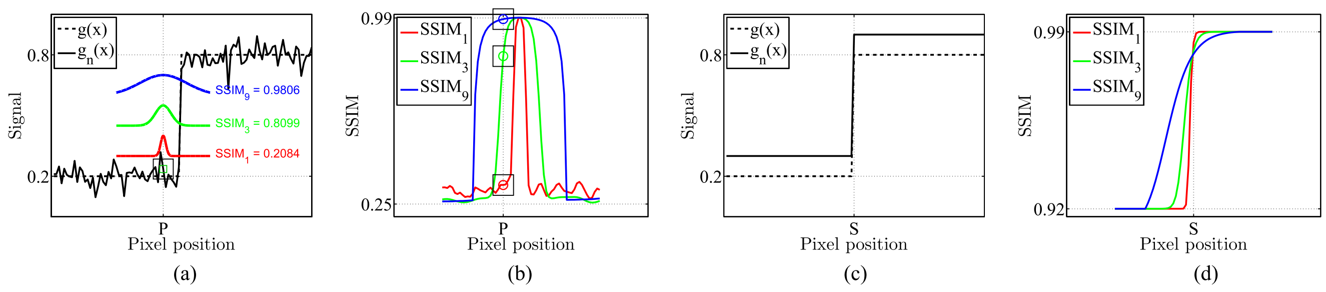


Fig. 10. Panels (a) and (b) show how the size of the support affects SSIM for a pixel P close to an edge, in the presence of zero-mean Gaussian noise. Panels (c) and (d) show that SSIM is less sensitive to a uniform bias in bright regions, such as the one to the right of pixel S (see Section V-B).

plots the value of SSIM for the noisy signal of Fig. 10(c). The same bias on the two sides of pixel S impacts SSIM's quality assessment differently. Because the term $l(p)$ in (5) measures the error in terms of a contrast, it effectively reduces its impact when the background is bright. This is why, thanks to its multi-scale nature, MS-SSIM solves the issue of noise around edges, but does not solve the problem of the change colors in flat areas, in particular when at least one channel is strong, as is the case with the sky.

Note that, while these are important observations because they generalize to any optimization problem that uses SSIM or MS-SSIM, the definition of a novel image quality metric is beyond the scope of this paper. We limit ourselves to the analysis of their performance within the context of neural networks.

Finally, we would like to point out that using SSIM-based losses in the context of color images introduces an approximation, since these metrics were originally designed for grayscale images. A more principled approach would be to base the loss function on a metric defined specifically for color images, such as FSIM$_c$. However, to the best of our knowledge, no such metric has been proposed that is also differentiable. We plan on addressing this problem in future work.

C. *Are we Committing the Inverse Crime?*

To numerically evaluate and compare the results of the different loss functions, we need access to the ground truth. For the applications described in this paper, this is only possible if we generate synthetic data. For instance, for the super-resolution experiments, we generate the low-resolution inputs by low-passing and down-sampling the original images. However, when generating synthetic data, one needs to be careful about the validity of the results. In the context of inverse problems, Colton and Kress introduce the concept of *inverse crime*, which is "committed" when synthetic data is fabricated with a forward model that is related to the inverse solver ([29], page 133), potentially hiding shortcomings of the approach. By this definition, it could be argued that we did commit the inverse crime, as we generate the synthetic data for both training and testing with the same process. However, two considerations support our choice. First, while in general the inverse crime is to be avoided, the forward model should resemble the physical system as closely as possible ([29], page 163). In the super-resolution example described above, for instance, performing down-sampling followed by low-passing mimics what the PSF of the lens and the sensor do to the incoming light. Second, we are studying the effect of the loss layer when all other conditions are equal: if we change the forward model, the quality of the results may degrade, but we expect the relationship between losses to hold. This is exactly what we observed when we tested the networks trained with the different losses on data generated with a varying standard deviation for the Gaussian filter. Changing the standard deviation from 3 to 5 pixels, for instance, causes the average MS-SSIM yielded by the network trained on Mix to drop from 0.9829 to 0.9752. However, the networks trained with ℓ_1 and ℓ_2 experience a similar drop, yielding an average MS-SSIM of 0.9744 and 0.9748 respectively. We observed a similar behavior for the other metrics. These considerations also apply to joint denoising and demosaicking, as well as JPEG de-blocking.

VI. Conclusion

We focus on an aspect of neural networks that is usually overlooked in the context of image restoration: the loss layer. We propose several alternatives to ℓ_2, which is the *de facto* standard, and we also define a novel loss. We use the problems of joint denoising and demosaicking, super-resolution, and JPEG artifacts removal for our tests. We offer a thorough analysis of the results in terms of both traditional and perceptually-motivated metrics, and show that the network trained with the proposed loss outperforms other networks. We also notice that the poor performance of ℓ_2 is partially related to its convergence properties, and this can help improve results of other ℓ_2-based approaches. Because the networks we use are fully convolutional, they are extremely efficient, as they do not require an aggregation step. Nevertheless, thanks to the loss we propose, our joint denoising and demosaicking network outperforms CFA-BM3D, a variant of BM3D tuned for denoising in Bayer domain, which is the state-of-the-art denoising algorithm. Table I shows the superiority of Mix over all the other losses. This can also be appreciated by visual inspection even for networks trained with ℓ_1, the closes competitor, which produce stronger artifacts (see, for instance, Fig. 3). We also make the implementation of the layers described in this paper available to the research community.

REFERENCES

[1] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[3] H. Burger, C. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with BM3D?" in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2392–2399.

[4] L. Xu, J. S. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 1790–1798.

[5] M. Zeiler, D. Krishnan, G. Taylor, and R. Fergus, "Deconvolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 2528–2535.

[6] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015.

[7] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, 2012, arXiv: 1207.0580.

[8] V. Jain and S. Seung, "Natural image denoising with convolutional networks," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. New York, NY, USA: Curran, 2009, pp. 769–776.

[9] C. Dong, C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 184–199.

[10] Y. Q. Wang, "A multilayer neural network for image demosaicking," in *Proc. IEEE Int. Conf. Image Process.*, 2014, pp. 1852–1856.

[11] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 675–678, arXiv:1408.5093.

[12] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "A comprehensive evaluation of full reference image quality assessment algorithms," in *Proc. IEEE Int. Conf. Image Process.*, 2012, pp. 1477–1480.

[13] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[14] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *Proc. 37th Asilomar Conf. Signals, Syst. Comput.*, 2003, vol. 2, pp. 1398–1402.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *CoRR*, 2015, arXiv:1502.01852.

[16] Z. Wang and A. Bovik, "Mean squared error: Love it or leave it? A new look at signal fidelity measures," *IEEE Signal Process. Mag.*, vol. 26, no. 1, pp. 98–117, Jan. 2009.

[17] D. Brunet, E. R. Vrscay, and Z. Wang, "Structural similarity-based approximation of signals and images using orthogonal bases," in *Proc. Int. Conf. Image Anal. Recognit.*, 2010, pp. 11–22.

[18] A. Rehman, M. Rostami, Z. Wang, D. Brunet, and E. Vrscay, "SSIM-inspired image restoration using sparse representation," *EURASIP J. Adv. Signal Process.*, vol. 2012, no. 1, 2012.

[19] A. C. Öztireli and M. Gross, "Perceptually based downscaling of images," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 77:1–77:10, 2015.

[20] Z. Wang, and Q. Li, "Information content weighting for perceptual image quality assessment," *IEEE Trans. Image Process.*, vol. 20, no. 5, pp. 1185–1198, Nov. 2011.

[21] H. Sheikh and A. Bovik, "Image information and visual quality," *IEEE Trans. Image Process.*, vol. 15, no. 2, pp. 430–444, Feb. 2006.

[22] W. Xue, L. Zhang, X. Mou, and A. Bovik, "Gradient magnitude similarity deviation: A highly efficient perceptual image quality index," *IEEE Trans. Image Process.*, vol. 23, no. 2, pp. 684–695, Feb. 2014.

[23] L. Zhang, D. Zhang, X. Mou, and D. Zhang, "FSIM: A feature similarity index for image quality assessment," *IEEE Trans. Image Process.*, vol. 20, no. 8, pp. 2378–2386, Aug. 2011.

[24] S. Winkler and S. Susstrunk, "Visibility of noise in natural images," in *Proc. Electron. Imag. 2004, Int. Soc. Opt. Photon.*, 2004, pp. 121–129.

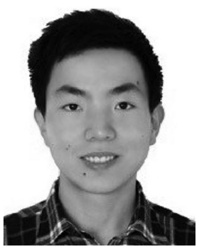
[25] V. Bychkovsky, S. Paris, E. Chan, and F. Durand, "Learning photographic global tonal adjustment with a database of input/output image pairs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 97–104.

[26] A. Foi, "Clipped noisy images: Heteroskedastic modeling and practical denoising," *Signal Process.*, vol. 89, no. 12, pp. 2609–2629, 2009.

[27] A. Danielyan, M. Vehvilainen, A. Foi, V. Katkovnik, and K. Egiazarian, "Cross-color BM3D filtering of noisy raw data," in *Proc. Int. Workshop Local Non-Local Approximation Image Process.*, 2009, pp. 125–129.

[28] D. Zhang and X. Wu, "Color demosaicking via directional linear minimum mean square-error estimation," *IEEE Trans. Image Process.*, vol. 14, no. 12, pp. 2167–2178, Dec. 2005.

[29] D. Colton and R. Kress, *Inverse Acoustic and Electromagnetic Scattering Theory*, vol. 93. New York, NY, USA: Springer, 2012.



Hang Zhao received the B.E. degree in information engineering from Zhejiang University, Hangzhou, China, in 2013. He is currently working toward the Ph.D. degree at the Massachusetts Institute of Technology, Cambridge, MA, USA. His research interests include computer vision, machine learning, and their robotic applications.

Orazio Gallo received the M.S. degree in biomedical engineering from Politecnico di Milano, Milano, Italy, and the Ph.D. degree in computer engineering from the University of California, Santa Cruz, CA, USA, in 2004 and 2011, respectively. From 2004 to 2006, he was a Research Assistant with the Smith-Kettlewell Eye Research Institute, San Francisco, CA, USA. In 2011, he joined NVIDIA, Santa Clara, CA, USA, where he is a Senior Research Scientist. His research interests include computational photography, image processing, applied visual perception, and computer vision. He is also an Associate Editor of the IEEE TRANSACTIONS ON COMPUTATIONAL IMAGING, as well as *Signal Processing: Image Communication*.

Iuri Frosio received the M.S. and Ph.D. degrees in biomedical engineering from the Politecnico di Milano, Milano, Italy, in 2003 and 2006, respectively. He was a Research Fellow in the Department of Computer Science, Universitá degli Studi di Milano, Milano, Italy, in 2003–2006, and an Assistant Professor in the same Department from 2006 to 2013. In the same period, he worked as a Consultant for various companies in Italy and abroad. In 2014, he joined NVIDIA, Santa Clara, CA, USA as a Senior Research Scientist. His research interests include image processing, computer vision, inertial sensors, and machine learning. He is also an Associate Editor of the *Journal of Electronic Imaging*.

Jan Kautz received the B.Sc. degree in computer science from the University of Erlangen-Nürnberg, Erlangen, Germany, the MMath degree from the University of Waterloo, Waterloo, ON, Canada, and the Ph.D. degree from the Max-Planck-Institut für Informatik, Saarbrücken, Germany, in 1999, 1999, and 2003, respectively. He currently leads the Visual Computing Research Team, NVIDIA, Santa Clara, CA, USA, working predominantly on computer vision problems—from low-level vision (denoising, super-resolution, computational photography) and geometric vision (structure from motion, SLAM, optical flow) to high-level vision (detection, recognition, classification), as well as machine learning problems such as deep reinforcement learning and efficient deep learning. Before joining NVIDIA in 2013, he was a tenured Faculty Member with University College London, London, U.K. He worked as a Postdoctoral with the Massachusetts Institute of Technology, Cambridge, MA, USA during 2003–2006.

He was the Program Co-Chair of the Eurographics Symposium on Rendering 2007, the Program Chair of the IEEE Symposium on Interactive Ray-Tracing 2008, the Program Co-Chair for Pacific Graphics 2011, and the Program Chair of CVMP 2012. He co-chaired Eurographics 2014 and was on the Editorial Board of the IEEE TRANSACTIONS ON VISUALIZATION & COMPUTER GRAPHICS as well as the *Visual Computer*.