Deep Learning of Compressed Sensing Operators with Structural Similarity Loss

Y. Zur and A. Adler

***Abstract*—Compressed sensing (CS) is a signal processing framework for efficiently reconstructing a signal from a small number of measurements, obtained by linear projections of the signal. In this paper we present an end-to-end deep learning approach for CS, in which a fully-connected network performs both the linear sensing and non-linear reconstruction stages. During the training phase, the sensing matrix and the non-linear reconstruction operator are *jointly* optimized using *Structural similarity index (SSIM)* as loss rather than the standard Mean Squared Error (MSE) loss. We compare the proposed approach with state-of-the-art in terms of reconstruction quality under both losses, i.e. SSIM score and MSE score.**

***Index Terms*—compressed sensing, neural network, deep learning, structural similarity index (SSIM).**

I. Introduction

Compressed sensing (CS) [1], [2] is a mathematical framework that defines the conditions and tools for the recovery of a signal from a small number of its linear projections (i.e. measurements). In the CS framework, the measurement device acquires the signal in the linear projections domain, and the full signal is reconstructed by convex optimization techniques. CS has diverse applications including image acquisition [3], radar imaging [4], Magnetic Resonance Imaging (MRI) [5], [6], spectrum sensing [7], indoor positioning [8], bio-signals acquisition [9], and sensor networks [10]. In this paper we address the CS problem by using a novel loss function, the SSIM loss. Our approach is based on a deep neural network [11], which simultaneously learns the linear sensing matrix and the non-linear reconstruction operator under the SSIM loss.

The contributions of this paper are two-fold: (1) It presents for the first time, to the best knowledge of the authors, a training with SSIM loss fonction of a deep neural network for the tasks of reconstruction; and (2) During training, the proposed network *jointly* optimizes both the linear sensing matrix and the non-linear reconstruction operator.

This paper is organized as follows: section II reviews compressed sensing concepts. Section III reviews the novel SSIM loss function. Section IV presents the end-to-end deep learning approach. Section V discusses structure and training aspects, while evaluating the performance of the proposed approach for image reconstruction, and comparing it with state-of-the-art alternatives. Section VI concludes the paper and discusses future research directions.

Yochai Zur is with the Department of Computer-Science, Technion Israel Institute of Technology.

Amir Adler is with the McGovern Institute for Brain Research, Massachusetts Institute of Technology.

II. Compressed Sensing Overview

Given a signal $\mathbf{x} \in \mathbf{R}^N$, an $M \times N$ sensing matrix Φ (such that $M \ll N$) and a measurements vector $\mathbf{y} = \Phi\mathbf{x}$, the goal of CS is to recover the signal from its measurements. The sensing rate is defined by $R = M/N$, and since $R \ll 1$ the recovery of $\mathbf{x}$ is not possible in the general case. According to CS theory [1], [2], signals that have a sparse representation in the domain of some linear transform can be exactly recovered with high probability from their measurements: let $\mathbf{x} = \Psi\mathbf{c}$, where Ψ is the inverse transform, and $\mathbf{c}$ is a sparse coefficients vector with only $S \ll N$ non-zeros entries, then the recovered signal is synthesized by $\hat{\mathbf{x}} = \Psi\hat{\mathbf{c}}$, and $\hat{\mathbf{c}}$ is obtained by solving the following convex optimization problem:

$$\hat{\mathbf{c}} = \arg\min_{\mathbf{c}'} \|\mathbf{c}'\|_1 \text{ subject to } \mathbf{y} = \Phi\Psi\mathbf{c}', \tag{1}$$

where $\|\alpha\|_1$ is the l_1-norm, which is a convex relaxation of the l_0 pseudo-norm that counts the number of non-zero entries of α. The exact recovery of $\mathbf{x}$ is guaranteed with high probability if $\mathbf{c}$ is sufficiently sparse and if certain conditions are met by the sensing matrix and the transform [1].

III. Structural similarity index (SSIM)

The Structural SIMilarity (SSIM) index [12] is a method for measuring the similarity between two images. The SSIM index can be viewed as a quality measure of one of the images being compared, provided the other image is regarded as of perfect quality. The difference with respect to other techniques such as Mean Squared Error (MSE) or Peak Signal-to-Noise Ratio (PSNR) is that these approaches estimate absolute errors; on the other hand, SSIM is a perception-based model that considers image degradation as perceived change in structural information, while also incorporating important perceptual phenomena, including both luminance masking and contrast masking terms. Structural information is the idea that the pixels have strong inter-dependencies especially when they are spatially close. These dependencies carry important information about the structure of the objects in the visual scene. Luminance masking is a phenomenon whereby image distortions (in this context) tend to be less visible in bright regions, while contrast masking is a phenomenon whereby distortions become less visible where there is significant activity or "texture" in the image. Given two images $\boldsymbol{X}$ and $\boldsymbol{Y}$, let $\mathbf{x}$ and $\mathbf{y}$ be column vector representations of two image patches (e.g., 8×8 windows) extracted from the same spatial location from $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively. Let $\boldsymbol{\mu_x}$, $\boldsymbol{\sigma_x^2}$ and $\boldsymbol{\sigma_{xy}}$ represent the sample mean of the components of $\mathbf{x}$, the

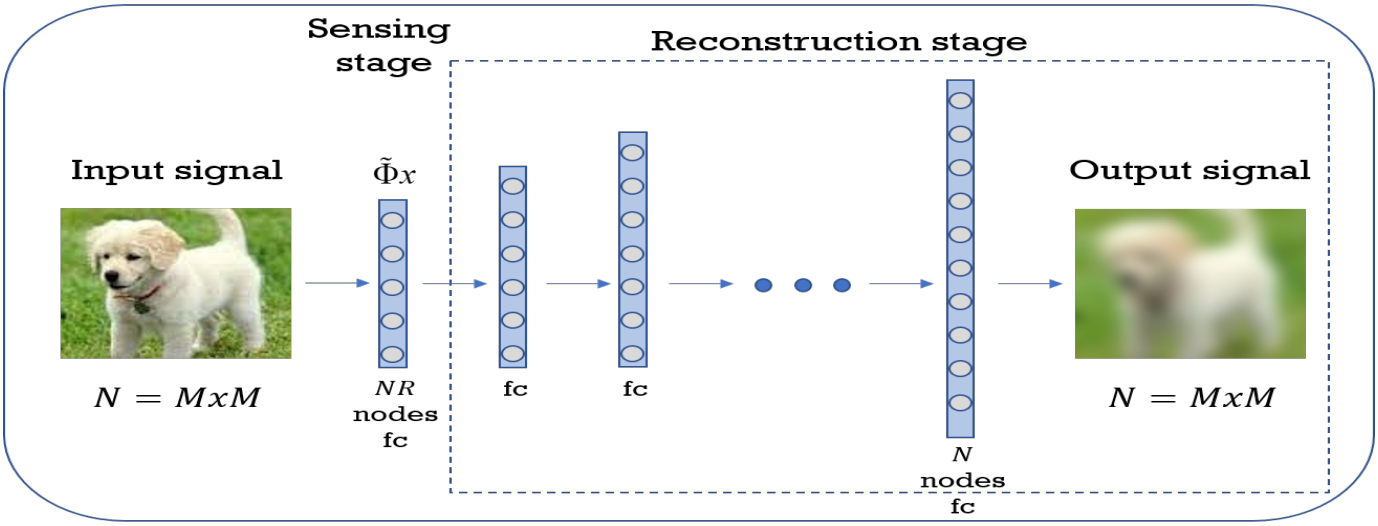

Fig. 1. Scheme of the end-to-end deep neural network for reconstruction architecture, which jointly optimizes the sensing and the non-linear reconstruction operators.

Sensing Rate	No. of Measurements	B	K		Training with SSIM loss $W(\mathbf{x},\mathbf{y})$ is Equation (12)	Training with MSE loss	Training with SSIM loss $W(\mathbf{x},\mathbf{y}) \equiv 1$
0.125	128	2	1	SSIM score	0.885392	0.892905	0.868903
				MSE score	0.005893	0.003517	0.005757
0.0625	64	1	2	SSIM score	0.760333	0.793642	0.719304
				MSE score	0.010316	0.006469	0.011553

TABLE I
SSIM VS. MSE LOSS SCORE OF NN TRAINING WITH SSIM LOSS FUNCTION VS. MSE LOSS FUNCTION FOR DIFFERENT SENSING RATES

sample variance of $\mathbf{x}$ and the sample covariance of $\mathbf{x}$ and $\mathbf{y}$, respectively:

$$\mu_x = \frac{1}{N_P}(\underline{1}^T \cdot \mathbf{x}) \tag{2}$$

$$\sigma_{\mathbf{x}}^2 = \frac{1}{N_P - 1}(\mathbf{x} - \mu_{\mathbf{x}})^T(\mathbf{x} - \mu_{\mathbf{x}}) \tag{3}$$

$$\sigma_{\mathbf{xy}} = \frac{1}{N_P - 1}(\mathbf{x} - \mu_{\mathbf{x}})^T(\mathbf{y} - \mu_{\mathbf{y}}) \tag{4}$$

where N_P is the number of pixels in patch $\mathbf{x}$ and $\mathbf{1}$ is a vector with all entries equaling 1.
For two image patches, $\mathbf{x}$ and $\mathbf{y}$ we define

$$\begin{aligned} A_1 &= 2\mu_{\mathbf{x}}\mu_{\mathbf{y}} + C_1 \qquad A_2 = 2\sigma_{\mathbf{xy}} + C_2 \\ B_1 &= \mu_{\mathbf{x}}^2 + \mu_{\mathbf{y}}^2 + C_1 \quad B_2 = \sigma_{\mathbf{x}}^2 + \sigma_{\mathbf{y}}^2 + C_2 \end{aligned} \tag{5}$$

The SSIM index between $\mathbf{x}$ and $\mathbf{y}$ is defined as

$$S(\mathbf{x},\mathbf{y}) = \frac{A_1 \cdot A_2}{B_1 \cdot B_2} \tag{6}$$

where C_1 and C_2 are small constants. It can be shown that the SSIM index achieves its maximum value of 1 if and only if the two image patches $\mathbf{x}$ and $\mathbf{y}$ being compared are exactly the same.
The SSIM index is computed using a sliding window approach. The window moves pixel by pixel across the whole image. At each step, the SSIM index is calculated within the local window. Finally, we compute a weighted average of the SSIM indexes to yield an overall SSIM index of the whole image:

$$S(X,Y) = \frac{\sum_{i=1}^{N_S} W(\mathbf{x}_i,\mathbf{y}_i) \cdot S(\mathbf{x}_i,\mathbf{y}_i)}{W(\mathbf{x}_i,\mathbf{y}_i)} \tag{7}$$

where $\mathbf{x}_i$ and $\mathbf{y}_i$ are the ith sampling sliding window in images X and Y, respectively. N_S is the total number of sampling windows. $W(\cdot,\cdot)$ is a weighting function where $W(\mathbf{x}_i,\mathbf{y}_i)$ is the ith sampling sliding window weight.
If we use uniform weighting function, i.e. $W(\mathbf{x}_i,\mathbf{y}_i) \equiv 1$, equation (7) becomes:

$$S(X,Y) = \frac{1}{N_S}\sum_{i=1}^{N_S} S(\mathbf{x}_i,\mathbf{y}_i) \tag{8}$$

As developed in appendix B, the gradient of SSIM index between X and Y with respect to Y is:

$$\begin{aligned} \nabla_Y S(X,Y) &= \nabla_Y\left(\frac{\sum_{i=1}^{N_S} W(\mathbf{x}_i,\mathbf{y}_i) \cdot S(\mathbf{x}_i,\mathbf{y}_i)}{\sum_{i=1}^{N_S} W(\mathbf{x}_i,\mathbf{y}_i)}\right) \\ &= \frac{\sum_{i=1}^{N_S} \nabla_Y\left(W(\mathbf{x}_i,\mathbf{y}_i) \cdot S(\mathbf{x}_i,\mathbf{y}_i)\right)}{\sum_{i=1}^{N_S} W(\mathbf{x}_i,\mathbf{y}_i)} \\ &- \frac{\sum_{i=1}^{N_S} W(\mathbf{x}_i,\mathbf{y}_i) \cdot S(\mathbf{x}_i,\mathbf{y}_i)}{\left(\sum_{i=1}^{N_S} W(\mathbf{x}_i,\mathbf{y}_i)\right)^2} \cdot \sum_{i=1}^{N_S}\left(\nabla_Y W(\mathbf{x}_i,\mathbf{y}_i)\right) \end{aligned} \tag{9}$$

If we use uniform weighting function, i.e. $\boldsymbol{W(x_i, y_i) \equiv 1}$, equation (9) becomes:

$$\boldsymbol{\nabla_Y S(X,Y) = \frac{1}{N_S} \sum_{i=1}^{N_S} \nabla_Y S(x_i, y_i)} \qquad (10)$$

See appendix B and Figure (4) for further explanations.

IV. The Proposed Approach

In this paper we propose an end-to-end deep learning solution for CS, which jointly optimizes the sensing matrix $\boldsymbol{\Phi}$ and the non-linear reconstruction operator, which is parameterized by a coefficients matrix $\boldsymbol{W}$. The proposed method provides a solution to the following joint optimization problem:

$$\boldsymbol{\{\widetilde{\Phi}, \widetilde{W}\} = \arg\min_{\Phi, W} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\mathcal{N}_W(\Phi x_i), x_i),} \qquad (11)$$

where $\boldsymbol{\{x_i\}_{i=1}^N}$ is the collection of $\boldsymbol{N}$ signals. The loss function $\boldsymbol{\mathcal{L}(\bullet, \bullet)}$ measures the distance between the input signal and the reconstructed one, provided by the reconstruction operator $\boldsymbol{\mathcal{N}_W(\bullet)}$, whose input is the compressed samples, denoted by $\boldsymbol{\Phi x_i}$. In this paper we compare SSIM vs. MSE as loss function, since MSE is commonly used for learning reconstruction networks. Note that during training the sensing layer (matrix) and the subsequent layers, represented by $\boldsymbol{\mathcal{N}_W(\bullet)}$, are treated as a single deep network. However, once training is complete, the sensing matrix is detached from the subsequent reconstruction layers, and used for performing signal sensing. The input of the reconstruction operator, is therefore the second layer of the end-to-end learned network. Our choice is motivated by the success of deep neural networks for the task of full-image reconstruction [13]. In our approach, the first layer learns the sensing matrix $\boldsymbol{\widetilde{\Phi}}$, and the following fully-connected layers perform the non-linear reconstruction stage. The proposed method was tested on CIFAR10 image recognition database.

V. Performance Evaluation

This section describes the proposed architectures and provides performance evaluation of the results. The CIFAR10 [14] dataset contains 60,000 color images of $\boldsymbol{32 \times 32}$ = 1024 pixels, drawn from 10 different classes. This dataset is divided into training and test sets, containing 50,000 and 10,000 images respectively. Since our proposed network expects grayscale images, for training on CIFAR10 we used only the $\boldsymbol{1^{st}}$ channel for each dataset image. Moreover, we enlarged the training set to 200,000 samples by rotating the original training set images by 90, 180 and 270 degrees. The fully-connected network includes the following layers:

1. An input layer with $\boldsymbol{N}$ nodes.
2. A compressed sensing fully-connected layer with $\boldsymbol{NR}$ nodes, $\boldsymbol{R \ll 1}$ (its weights form the sensing matrix).
3. $\boldsymbol{K \geq 1}$ reconstruction layers with $\boldsymbol{NB}$ nodes in each layer, where $\boldsymbol{B \in \{1, 2\}}$. Each layer is followed by a sigmoid activation unit.
4. An output layer with $\boldsymbol{N}$ nodes.

Figure (1) illustrates the aforementioned flow.

The SSIM loss function was used with $\boldsymbol{8 \times 8}$ window size. We tested 2 different weighting functions for the SSIM loss. The first function is the uniform weighting function, i.e. $\boldsymbol{W(x_i, y_i) \equiv 1}$. The second function was recommended in [15]:

$$\boldsymbol{W(x, y) = log\left[\left(1 + \frac{\sigma_x^2}{C_2}\right)\left(1 + \frac{\sigma_y^2}{C_2}\right)\right]} \qquad (12)$$

The optimization algorithm of choice was Adam [16] with an initial learning rate of $\boldsymbol{5 \cdot 10^{-4}}$. The training stopping criteria was 50 consecutive epochs without reaching a new minima.

Table I compares between the neural network trained with SSIM as loss function and the neural network trained with MSE as loss function. The table shows the average SSIM and MSE loss over 10,000 CIFAR10 test images on these networks. The results show that a neural network trained with MSE loss achieves better reconstruction quality even in SSIM index score on CIFAR10 dataset.

VI. Conclusion

This paper presents SSIM as loss function for training end-to-end deep neural network for compressed sensing and non-linear reconstruction, in which the sensing matrix and the non-linear reconstruction operator are jointly optimized during the training phase.The proposed approach does not outperform state-of-the-art in terms of reconstruction quality by SSIM loss. Since calculating SSIM score is more complicated than calculating MSE score for a given image, as explained in details on appendix A, our approach can be further improved by combining the SSIM as loss function with block-based compressed sensing approach [17]. Learning a deep neural network for blocks reconstruction under the SSIM loss would be significantly faster than learning a network for full-image reconstruction in cases where there are many patches per image. Another possible future work direction is to expand the SSIM loss function to multi-scale SSIM loss function as described on [18].

Appendix A
Efficient training with SSIM loss

As described on section III, using SSIM as a loss function for reconstruction neural network, instead of the commonly used MSE function, sets few computational challenges since the SSIM function is much more complicated computationally compared to the MSE function.

The $\boldsymbol{1^{st}}$ challenge is to divide the image to patches. Extracting all image patches might be time consuming if it is programmed straightforward, i.e. using for-loop for example. We experienced significant acceleration once we replaced the for-loop with another method. Our efficient method for extracting the image patches was to build a convolutional neural-network (CNN). The CNN is built from kernels with the same dimensions as the SSIM window dimensions. Each kernel has 0's all over and a value of 1 in a specific position,

Fig. 2. Reconstruction of 2 test images. SSIM(1) is Equation (12) as weight function. SSIM(2) is a uniform weight function $\boldsymbol{W}(\mathbf{x},\mathbf{y}) \equiv \mathbf{1}$. SR stands for Sensing Rate.

i.e. each kernel has the value of 1 in a different position. When we forward an image through the CNN it outputs kernels where each kernel holds all the possible values from the original image that should be placed in the kernel position as part of some patch. The CNN final layer reshapes the kernels to the patches shape as required. See Figure (3) for further explanations.

Another challenge is the trade-off between computation time and storage size. In each training epoch, we calculate SSIM loss for a batch of samples from the training set. It means we have to calculate SSIM loss for the same batch more than once during training, since we iterate over the training set more than once. On one hand, we can calculate once equations (2),(3) prior to training and store the results in memory. It saves us computations during training. On the other hand, it requires some storage space. We can extend the dilemma even to storage of training set patches. We can trade-off between extracting the patches in each epoch, which costs us in computation time, to extracting all training set patches pre-training and store them in memory, which costs us in significant storage space.

APPENDIX B
SSIM INDEX GRADIENT

Given an SSIM index:

$$\boldsymbol{S}(\boldsymbol{X},\boldsymbol{Y}) = \frac{\sum_{i=1}^{N_S} \boldsymbol{W}(\mathbf{x}_i,\mathbf{y}_i)\cdot \boldsymbol{S}(\mathbf{x}_i,\mathbf{y}_i)}{\sum_{i=1}^{N_S} \boldsymbol{W}(\mathbf{x}_i,\mathbf{y}_i)} \tag{13}$$

We are interested in its gradient with respect to $\boldsymbol{Y}$, i.e. $\boldsymbol{\nabla_Y S}(\boldsymbol{X},\boldsymbol{Y})$.

Let us start with a simple case, where there is single window and a uniform weighting function $\boldsymbol{W}(\mathbf{x}_i,\mathbf{y}_i) \equiv \mathbf{1}$, therefore:

$$\boldsymbol{S}(\boldsymbol{X},\boldsymbol{Y}) = \boldsymbol{S}(\mathbf{x}_1,\mathbf{y}_1) \tag{14}$$

Equations (2),(3),(4) and (5) induce the following equivalence:

$$\boldsymbol{S}(\boldsymbol{X},\boldsymbol{Y}) = \boldsymbol{S}(\mathbf{x}_1,\mathbf{y}_1) = \frac{\boldsymbol{A_1}\cdot\boldsymbol{A_2}}{\boldsymbol{B_1}\cdot\boldsymbol{B_2}} \tag{15}$$

Let us develop the gradient with respect to $\boldsymbol{Y}$:

$$\boldsymbol{\nabla_Y} A_1 = 2\mu_{\mathbf{x}}(\boldsymbol{\nabla_Y}\mu_{\mathbf{y}}) = 2\mu_{\mathbf{x}}\left(\frac{\mathbf{1}}{\boldsymbol{N_P}}\cdot\underline{\mathbf{1}}\right) \tag{16}$$

$$\begin{aligned}\boldsymbol{\nabla_Y} B_1 &= \boldsymbol{\nabla_Y}(\mu_{\mathbf{y}}^2) = 2\mu_{\mathbf{y}}(\boldsymbol{\nabla_Y}\mu_{\mathbf{y}}) = \\ &= 2\mu_{\mathbf{y}}\left(\frac{\mathbf{1}}{\boldsymbol{N_P}}\cdot\underline{\mathbf{1}}\right)\end{aligned} \tag{17}$$

$$\boldsymbol{\nabla_Y} A_2 = 2(\boldsymbol{\nabla_Y}\sigma_{\mathbf{xy}}) = 2\left(\frac{\mathbf{1}}{\boldsymbol{N_P}-\mathbf{1}}\cdot(\mathbf{x}-\mu_{\mathbf{x}})\right) \tag{18}$$

$$\boldsymbol{\nabla_Y} B_2 = \boldsymbol{\nabla_Y}(\sigma_{\mathbf{y}}^2) = \frac{\mathbf{2}}{\boldsymbol{N_P}-\mathbf{1}}(\mathbf{y}-\mu_{\mathbf{y}}) \tag{19}$$

$$\boldsymbol{\nabla_Y}(\boldsymbol{A_1 A_2}) = (\boldsymbol{\nabla_Y} A_1)\boldsymbol{A_2} + \boldsymbol{A_1}(\boldsymbol{\nabla_Y} A_2) \tag{20}$$

$$\boldsymbol{\nabla_Y}(\boldsymbol{B_1 B_2}) = (\boldsymbol{\nabla_Y} B_1)\boldsymbol{B_2} + \boldsymbol{B_1}(\boldsymbol{\nabla_Y} B_2) \tag{21}$$

Therefore, the SSIM index gradient with respect to $\boldsymbol{Y}$:

$$\begin{aligned}&\boldsymbol{\nabla_Y}\boldsymbol{S}(\mathbf{x}_1,\mathbf{y}_1) = \\ &\frac{\boldsymbol{\nabla_Y}(\boldsymbol{A_1 A_2})}{(\boldsymbol{B_1 B_2})} - \frac{(\boldsymbol{A_1 A_2})}{(\boldsymbol{B_1 B_2})^{\mathbf{2}}}\cdot\left(\boldsymbol{\nabla_Y}(\boldsymbol{B_1 B_2})\right)\end{aligned} \tag{22}$$

Let us develop $\boldsymbol{\nabla_Y}\boldsymbol{S}(\boldsymbol{X},\boldsymbol{Y})$ for the general case:

$$\begin{aligned}&\boldsymbol{\nabla_Y}\left(\boldsymbol{W}(\mathbf{x}_i,\mathbf{y}_i)\cdot\boldsymbol{S}(\mathbf{x}_i,\mathbf{y}_i)\right) = \\ &\left(\boldsymbol{\nabla_Y}\boldsymbol{W}(\mathbf{x}_i,\mathbf{y}_i)\right)\cdot\boldsymbol{S}(\mathbf{x}_i,\mathbf{y}_i) + \boldsymbol{W}(\mathbf{x}_i,\mathbf{y}_i)\cdot\left(\boldsymbol{\nabla_Y}\boldsymbol{S}(\mathbf{x}_i,\mathbf{y}_i)\right)\end{aligned} \tag{23}$$

$$\begin{aligned}\boldsymbol{\nabla_Y}\boldsymbol{S}(\boldsymbol{X},\boldsymbol{Y}) &= \boldsymbol{\nabla_Y}\left(\frac{\sum_{i=1}^{N_S}\boldsymbol{W}(\mathbf{x}_i,\mathbf{y}_i)\cdot\boldsymbol{S}(\mathbf{x}_i,\mathbf{y}_i)}{\sum_{i=1}^{N_S}\boldsymbol{W}(\mathbf{x}_i,\mathbf{y}_i)}\right) \\ &= \frac{\sum_{i=1}^{N_S}\boldsymbol{\nabla_Y}\left(\boldsymbol{W}(\mathbf{x}_i,\mathbf{y}_i)\cdot\boldsymbol{S}(\mathbf{x}_i,\mathbf{y}_i)\right)}{\sum_{i=1}^{N_S}\boldsymbol{W}(\mathbf{x}_i,\mathbf{y}_i)} \\ &- \frac{\sum_{i=1}^{N_S}\boldsymbol{W}(\mathbf{x}_i,\mathbf{y}_i)\cdot\boldsymbol{S}(\mathbf{x}_i,\mathbf{y}_i)}{\left(\sum_{i=1}^{N_S}\boldsymbol{W}(\mathbf{x}_i,\mathbf{y}_i)\right)^{\mathbf{2}}}\cdot\sum_{i=1}^{N_S}\left(\boldsymbol{\nabla_Y}\boldsymbol{W}(\mathbf{x}_i,\mathbf{y}_i)\right)\end{aligned} \tag{24}$$

If we use uniform weighting function, i.e. $\boldsymbol{W}(\mathbf{x}_i,\mathbf{y}_i) \equiv \mathbf{1}$, equation (24) becomes:

$$\boldsymbol{\nabla_Y}\boldsymbol{S}(\boldsymbol{X},\boldsymbol{Y}) = \frac{\mathbf{1}}{\boldsymbol{N_S}}\sum_{i=1}^{N_S}\boldsymbol{\nabla_Y}\boldsymbol{S}(\mathbf{x}_i,\mathbf{y}_i) \tag{25}$$

Let us recall that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are images, $\mathbf{x}$ and $\mathbf{y}$ are windows extracted from $\boldsymbol{X}$ and $\boldsymbol{Y}$ respectively, therefore the gradient $\boldsymbol{\nabla_Y}\boldsymbol{S}(\mathbf{x},\mathbf{y})$ has same dimensions as $\mathbf{x}$ and $\mathbf{y}$. Notice the sum in (25) is not a standard sum. We can think about it as we have an image with the same dimensions as $\boldsymbol{X}$ and $\boldsymbol{Y}$ which all its values are 0, then we sum each gradient $\boldsymbol{\nabla_Y}\boldsymbol{S}(\mathbf{x}_i,\mathbf{y}_i)$ to its window location and finally divide the whole image by $\boldsymbol{N_S}$. The same follows for sums in the general gradient equation (24), each sum is not a standard sum, but means we should sum each gradient to its window location. See Figure (4) for further explanations.

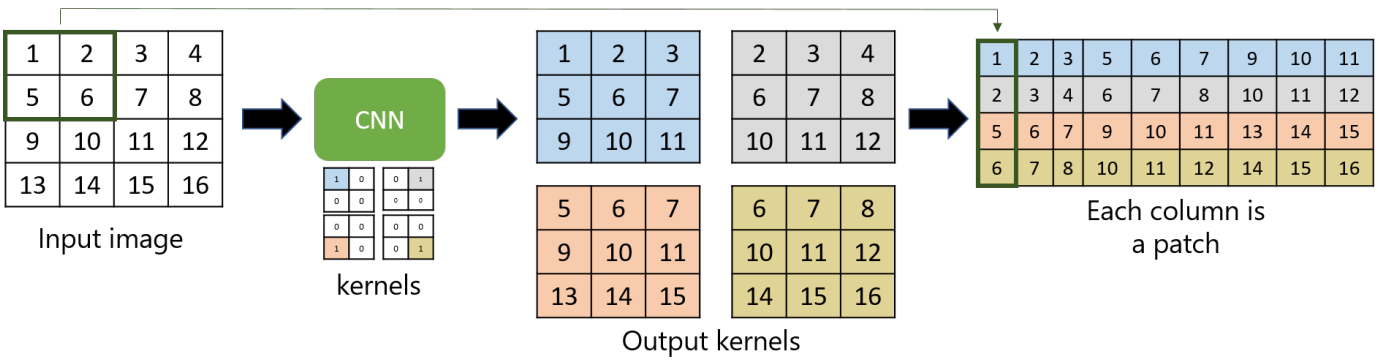

Fig. 3. Scheme of the CNN kernels (as described in appendix A) for extracting 2x2 patches from a 4x4 image. The CNN last layer reshapes the output kernels to 2x2 patches as columns.

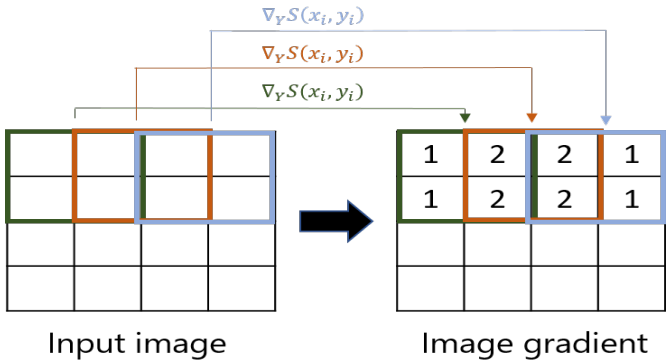

Fig. 4. Scheme of calculating overall SSIM gradient for whole image. The overall SSIM gradient is calculated per pixel. For each pixel we sum all the gradients of windows that include it and divide by the total number of gradients. The numbers inside the image gradient pixels represents the number of gradients we sum for each pixel, i.e. the pixels in the borders sum less gradients than the central pixels.

REFERENCES

[1] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[2] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[3] J. Romberg, "Imaging via compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 14–20, 2008.

[4] L. C. Potter, E. Ertin, J. T. Parker, and M. Cetin, "Sparsity and compressed sensing in radar imaging," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1006–1020, 2010.

[5] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, "Compressed sensing mri," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 72–82, 2008.

[6] M. Murphy, M. Alley, J. Demmel, K. Keutzer, S. Vasanawala, and M. Lustig, "Fast ℓ_1-spirit compressed sensing parallel imaging mri: Scalable parallel implementation and clinically feasible runtime," *IEEE transactions on medical imaging*, vol. 31, no. 6, pp. 1250–1262, 2012.

[7] E. Axell, G. Leus, E. G. Larsson, and H. V. Poor, "Spectrum sensing for cognitive radio: State-of-the-art and recent advances," *IEEE Signal Processing Magazine*, vol. 29, no. 3, pp. 101–116, 2012.

[8] C. Feng, W. S. A. Au, S. Valaee, and Z. Tan, "Received-signal-strength-based indoor positioning using compressive sensing," *IEEE Transactions on Mobile Computing*, vol. 11, no. 12, pp. 1983–1993, 2012.

[9] A. M. Dixon, E. G. Allstot, D. Gangopadhyay, and D. J. Allstot, "Compressed sensing system considerations for ecg and emg wireless biosensors," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 6, no. 2, pp. 156–166, 2012.

[10] S. Li, L. Da Xu, and X. Wang, "Compressed sensing signal and data acquisition in wireless sensor networks and internet of things," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2177–2186, 2013.

[11] Y. Bengio *et al.*, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[12] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[13] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[14] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 dataset," *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

[15] Z. Wang and E. P. Simoncelli, "Maximum differentiation (mad) competition: A methodology for comparing computational models of perceptual quantities," *Journal of Vision*, vol. 8, no. 12, pp. 8–8, 2008.

[16] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] A. Adler, D. Boublil, M. Elad, and M. Zibulevsky, "A deep learning approach to block-based compressed sensing of images," *arXiv preprint arXiv:1606.01519*, 2016.

[18] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 47–57, 2017.