# Self-Similarity Block for Deep Image Denoising

Edoardo Peretti[1], Diego Stucchi[1], Diego Carrera[2], and Giacomo Boracchi[1]

[1] Politecnico di Milano, DEIB, Milano, Italy {name.surname}@polimi.it
[2] STMicroelectronics, Agrate Brianza, Italy {name.surname}@st.com

**Abstract.** Non-Local Self-Similarity (NLSS) is a widely exploited prior in image denoising algorithms. The first deep Convolutional Neural Networks (CNNs) for image denoising ignored NLSS and were made of a sequence of convolutional layers trained to suppress noise. The first denoising CNNs leveraging NLSS prior were performing non-learnable operations outside the network. Then, pre-defined similarity measures were introduced and finally learnable, but scalar, similarity scores were adopted inside the network. We propose the Self-Similarity Block (SSB), a novel differentiable building block for CNN denoisers to promote the NLSS prior. The SSB is trained in an end-to-end manner within convolutional layers and learns a multivariate similarity score to improve image denoising by combining similar vectors in an activation map. We test SSB on additive white Gaussian noise suppression, and we show it is particularly beneficial when the noise level is high. Remarkably, SSB is mostly effective in image regions presenting repeated patterns, which most benefit from the NLSS prior.

## 1 Introduction

Digital sensors are affected by photon counting, thermal, and quantization noise, which needs to be suppressed to provide visually-pleasant images. As a matter of fact, denoising is a fundamental step in almost every image processing pipeline [25], since noise can affect subsequent processing. During the last few decades, image denoising algorithms heavily relied on statistical modeling of images and carefully crafted signal processing techniques to promote priors such as sparsity and Non-Local Self-Similarity (NLSS) [3,1]. The NLSS of natural images suggests that most small patches contain patterns that are repeated across the same image, possibly at non-adjacent locations. Recently, deep-learning-based methods adopted Convolutional Neural Networks (CNNs) to solve the denoising problem by directly learning a sophisticated denoising function from a dataset of noise-free images corrupted by artificially added noise. However, while several classic methods exploit the NLSS prior, most CNN denoisers act locally as DnCNN [26], even when they have a large receptive field.

Recent research has focused on designing new building blocks and training procedures enforcing traditional priors in deep denoisers. Traditional non-local operations used to promote NLSS, e.g., block matching, are not differentiable and must be performed outside the CNN [11], yielding sub-optimal performance.

The few differentiable non-local blocks that can be employed inside a CNN rely on scalar hand-crafted similarity measures [17] or are based on the attention mechanism [22], which is defined over the entire feature, while NLSS is typically exploited relatively to a search neighborhood.

We propose the Self-Similarity Block (SSB), a novel differentiable block promoting NLSS in deep denoisers. In practice, the SSB exploits the similarity of activation vectors from different spatial locations in the same activation map to perform denoising. To this purpose, the SSB uses an *entirely learnable multivariate similarity score* in a search neighborhood, computed as a series of $1 \times 1$ convolutions mixing the information from different locations. Our SSB can be easily included after any layer of a CNN, as it does not modify the dimension of the input activation map. In particular, the SSB can be trained with the whole network or fine-tuned in a pre-trained network to boost its denoising performance. In this regard, we design Self-Similarity Networks (SSN), which consist of a sequence of convolutional blocks as in DnCNN [26], interleaved by SSBs.

Our experiments confirm that inserting an SSB in denoising CNNs improves their performance. Moreover, SSN achieves comparable performance with respect to state-of-the-art methods based on comparable convolutional architectures, including deep denoisers exploiting the NLSS prior, while outperforms them on high noise levels. The source code is available at https://github.com/edpere/SSN.

## 2 Related Work

Many image priors are exploited in denoising [4,30,19,18,16], being NLSS one of the most popular. Usually, similarity scores are computed over non-adjacent patches, and then used to average pixel intensities (NLM [1]), in collaborative filtering in transform domain (BM3D [3]), and in low-rank approximation [6].

Here, we focus on deep CNN denoisers trained on clean-noisy image pairs. These comprise RED [14], which is a convolutional encoder-decoder architecture, and DnCNN [26], which is a stack of convolutions, batch normalizations [8] and ReLU activations. FFDNet [27] extended DnCNN by introducing a noise level map as input and a reversible downscaling operator. The first attempts to leverage the NLSS prior in CNN denoisers consisted in enlarging the receptive fields. However, stacking more layers [14] or inserting downscaling operators [27] does not significantly increase the *effective* receptive field [13] of CNN denoisers, which explains why very deep architectures do not yield a significantly better denoising [20]. Therefore, several methods [11,24,10], inspired by BM3D [3], introduced block matching for grouping similar patches as a pre-processing step before CNN filtering. For example, NN3D [2] combines a local CNN denoiser with a traditional non-local filter in an iterative manner. In all this methods the non-local operations are predefined and not learned.

Interestingly, recent works investigate differentiable layers promoting the NLSS within activation maps, to enable end-to-end training alongside the network. $N^3$Net [17] proposes a differentiable relaxation of KNN, and then concatenates similar features in activation maps. One of the most relevant attempt

to leverage NLSS within activation maps are the nonlocal networks [22], which implement the attention mechanism in images, drawing a parallel with NLM. However, this is a global operation which was not originally proposed for image restoration but for visual recognition tasks. This non local-block was later adapted to image denoising both in a recurrent [12] and in a sophisticated feed-forward [28] architecture comprising multiple branches. This approach can be generalized by graph neural networks [21].

All these non-local blocks operate on activation maps using a scalar similarity measure, like the Euclidean one. Remarkably, our proposed SSB learns a *multivariate similarity score* directly on training data, capturing different aspects of the similarity embedded in the activation vectors, and improving convolutional denoisers, as shown in our experiments.

## 3 Problem Formulation

We consider grayscale image denoising where the noisy image $z$ is defined over a finite grid $X \subset \mathbb{Z}^2$ and described as follows:

$$z(x) = y(x) + \eta(x) \qquad \forall x \in X, \tag{1}$$

where $y$ is the clean image, $\eta(\cdot) \sim \mathcal{N}(0, \sigma^2)$ is the *additive white Gaussian noise* (AWGN), and $\sigma$ is the noise standard deviation. An image denoiser $\mathcal{D}_\theta$, depending on the set of parameters $\theta \in \Theta$, is a map that provides an estimate $\widehat{y} = \mathcal{D}_\theta(z)$ of the noise-free image $y$. We consider $\mathcal{D}_\theta$ to be a CNN trained in a supervised manner, from a training set $\{(y_j, z_j)\}_j$ of clean-noisy image pairs. Remarkably, AWGN denoisers can handle different noise models by variance stabilizing transforms [5], and the extension to color images is trivial when operating with CNNs.

## 4 Proposed Method

In this section, we present the Self-Similarity Block (SSB), a differentiable layer for CNN denoisers designed to exploit the NLSS prior. Traditional algorithms exploit the NLSS prior to estimate each reference patch by combining information from similar patches within a search neighborhood. As in [22], our intuition is to leverage NLSS among *activation vectors* of the same activation map returned by a convolutional layer. More specifically, we consider an activation vector $\mathbf{v} \in \mathbb{R}^d$ obtained by stacking all the values along the $d$ channels from an activation map at a given spatial location. Each activation vector can be interpreted as a learned embedding of the corresponding region in the input image.

The SSB estimates the similarity among pairs of activation vectors in a search region of the same activation map through a trainable multivariate similarity score. This similarity is used to guide the filtering of the activation vectors and to compute the weights for the final average. To improve denoising performance, we adopt SSB within a residual mapping. This is custom in CNN denoisers [26], and in practice forces SSB to learn the noise realization to be removed from

each input activation map. Therefore, our SSB can be easily inserted in any pre-trained CNN denoiser since *i)* it takes any 3D tensor as input and outputs one of the same size and *ii)* it can be initialized as the identity map to be fine-tuned.

In Section 4.1, we illustrate the *differentiable multivariate similarity score* adopted by the SSB, and in Section 4.2, we describe the implementation of the proposed end-to-end learnable layer. Finally, in Section 4.3, we present the Self-Similarity Network (SSN), a customizable CNN combining DnCNN and SSBs.

### 4.1 Non-Local Filtering Guided by Vector Similarities

Let $\mathbf{v} \in \mathbb{R}^d$ be a reference activation vector computed from a noisy image $z$ at a specific convolutional layer. We perform denoising by a *learnable residual mapping* $\Phi$ promoting the NLSS as follows

$$\widehat{\mathbf{v}} = \mathbf{v} - \Phi\left(\mathbf{v} \mid \{\mathbf{v}^k\}_{k=1}^M\right), \tag{2}$$

where $\{\mathbf{v}^k\}_k \subset \mathbb{R}^d$ are $M$ vectors in a search neighborhood of $\mathbf{v}$ from the same activation map. Since we adopt residual learning, the goal of $\Phi$ is to estimate the noise realization affecting $\mathbf{v}$, then subtracted to obtain a noise-free estimate $\widehat{\mathbf{v}}$. We define

$$\Phi\left(\mathbf{v} \mid \{\mathbf{v}^k\}_{k=1}^M\right) = A\sum_{k=1}^{M}\left(\alpha_k \mathbf{r}^k\right), \tag{3}$$

where $\{\mathbf{r}^k\}_k \subset \mathbb{R}^d$ are neighbor contributions, each depending on a single activation vector belonging to the search neighborhood, $A \in \mathbb{R}^{d\times d}$ is a learnable weight matrix, and the weights $\{\alpha_k\}_k \subset \mathbb{R}$ define the contribution of each $\mathbf{r}^k$, based on the similarity score between $\mathbf{v}$ and $\mathbf{v}^k$.

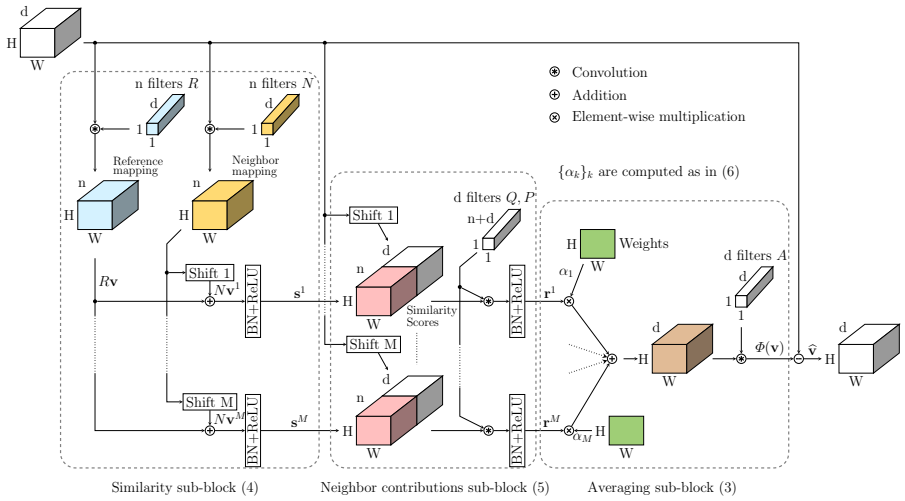
We define the similarity score $\mathbf{s}^k \in \mathbb{R}^n$ between two activation vectors $\mathbf{v}$ and $\mathbf{v}^k$ as a multivariate weighted combination of the two, namely:

$$\mathbf{s}^k = \mathrm{ReLU}\big(R\mathbf{v} + N\mathbf{v}^k\big), \tag{4}$$

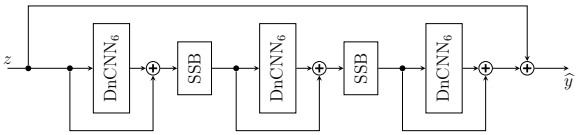
where $R, N \in \mathbb{R}^{n\times d}$ are learnable matrices defining two linear embeddings, that are independent of the reference $\mathbf{v}$ or the neighbor $\mathbf{v}^k$. Since these operations are linear, we introduce non-linearity with a ReLU function. After training, we expect each component of $\mathbf{s}^k$ to capture some form of similarity between $\mathbf{v}$ and $\mathbf{v}^k$, as a deep learning alternative of the patch distance of NLM. Then, we use $\mathbf{s}^k$ to guide the computation of a *neighbor contribution* $\mathbf{r}^k$, which is defined by extracting relevant information from a neighbor $\mathbf{v}^k$ and the similarity $\mathbf{s}^k$. More precisely, we compute $\mathbf{r}^k \in \mathbb{R}^d$ as

$$\mathbf{r}^k = \mathrm{ReLU}\big(P\mathbf{v}^k + Q\mathbf{s}^k\big), \tag{5}$$

where $P \in \mathbb{R}^{d\times d}$ and $Q \in \mathbb{R}^{d\times n}$ are learnable linear embedding matrices, and we introduce non-linearity by the ReLU function. To conclude, each $\mathbf{r}^k$ contributes to the final residual in (3) by a weight $\alpha_k$ that depends on the similarity $\mathbf{s}^k$ between $\mathbf{v}$ and $\mathbf{v}^k$. More precisely, we define the weights $\{\alpha_k\}_k \subset \mathbb{R}$ in (3) as




(a) The proposed Self-Similarity Block (SSB)




(b) Self-Similarity Network (SSN)

Fig. 1: (a) The Self-Similarity Block (SSB) exploits the NLSS prior for denoising by residual computation. It comprises three sub-blocks: *i)* similarity sub-block, that computes the scores $\{\mathbf{s}^k\}_k$, *ii)* neighbor contributions sub-block, which computes $\{\mathbf{r}^k\}_k$, and *iii)* averaging sub-block, aggregating all the neighbor contributions. SSB is employed with a skip connection to perform residual learning. (b) A Self-Similarity Network (SSN) consisting of 6-layer DnCNNs interleaved by SSBs.

the inner product of the similarity scores $\{\mathbf{s}^k\}_k$ against a shared learnable vector $\mathbf{h} \in \mathbb{R}^n$, normalizing the results with the Softmax:

$$\alpha_k = \text{Softmax}\left(\mathbf{h}\cdot\mathbf{s}^1, \mathbf{h}\cdot\mathbf{s}^2, \ldots, \mathbf{h}\cdot\mathbf{s}^M\right)_k, \tag{6}$$

such that weights sum to 1. Finally, the residual computed as in (3) is subtracted from the input vector $\mathbf{v}$ as in (2).

### 4.2 Self-Similarity Block

Here, we illustrate the architecture of the SSB, which implements the denoising procedure described in Section 4.1. In particular, similarly to the search neighborhood adopted by NLM [1], for each reference vector $\mathbf{v}$, we define the neighbors $\{\mathbf{v}^k\}_{k=1}^M$ as the $M$ vectors in a $(2\,\Omega+1)\times(2\,\Omega+1)$ grid centered around

$\mathbf{v}$ separated by a stride $w$, where $\Omega$ is the half-neighborhood size. Moreover, we implement the matrix-by-vector multiplications of the linear embeddings in (3), (4) and (5) as layers of $1 \times 1$ convolutions against the rows of the corresponding matrices $A, R, N, P, Q$. As shown in Figure 1a, SSB is made of three sub-blocks: a similarity sub-block implementing (4), a neighbor contributions sub-block implementing (5) and an averaging sub-block implementing (3).

The *similarity sub-block* (Figure 1a, left) computes the similarity scores $\{\mathbf{s}^k\}_k$ (4) for all the reference/neighbor vector pairs. Noting that all the activation vectors $\mathbf{v}^k$ are, in turn, references and neighbors for other references, we decompose the similarity computation in two branches. One is responsible for the computation of $R\mathbf{v}$ for all $\mathbf{v}$ in the activation map, while the other computes $N\mathbf{v}^k$. We shift the input data to line up each neighbor $\mathbf{v}^k$ with the reference. Therefore, we consider $M$ shift branches and compute all the similarity scores $\{\mathbf{s}^k\}_k$ at once by summing two properly shifted outputs of the branches. After the sum, we apply the Batch Normalization (BN) and the ReLU activation, which inserts non-linearity in an otherwise linear computation.

The *neighbor contributions sub-block* (Figure 1a, center) computes the neighbor contributions $\{\mathbf{r}^k\}_k$ for every reference/neighbor vector pairs. The sum in (5) is computed by concatenating the similarity scores and the shifted neighbors and then performing $d$ $1 \times 1$ convolutions. Again, we apply the BN and the ReLU to the results of this operation.

The *averaging sub-block* (Figure 1a, right) combines the neighbor contributions in a weighted sum. We compute the weights $\{\alpha_k\}_k$ as in (6) and apply them to the contributions through a component-wise multiplication. The results are then summed up and convolved against the $d$ rows of $A$, as in (3) to yield the final residuals $\Phi(\mathbf{v})$ for every $\mathbf{v}$ in the activation map. Finally, the residuals are subtracted from the input of the block via a skip connection.

We remark that inserting an SSB in a CNN does not significantly increase the number of trainable parameters, which are the matrices $R, N, Q \in \mathbb{R}^{n \times d}$, $P \in \mathbb{R}^{d \times d}$ and $A \in \mathbb{R}^{n \times n}$ and the vector $\mathbf{h} \in \mathbb{R}^n$, for a total of $n^2 + d^2 + 3nd + n$ parameters. However, the number of operations increases significantly because of the $M$ shifts performed to compute the similarity scores.

### 4.3 Network Architecture

We design the Self-Similarity Network (SSN) as a customizable denoising CNN architecture based on DnCNN [26] and our SSB. The architecture of SSN is reported in Figure 1b and consists of instances of DnCNN interleaved by SSBs. Remarkably, the SSB takes a tensor of arbitrary size as input and returns an output of the same size. For this reason, it can be seamlessly inserted between layers of any CNN to leverage the NLSS. Moreover, if initialized as the identity transformation by setting $A = 0$, the SSB can be fine-tuned in pre-trained denoising networks to improve the denoising performance.

We denote by $\text{DnCNN}_D$ a DnCNN of depth $D$, which consists of 1 Conv+ReLU followed by $(D-2)$ Conv+BN+ReLU layers and 1 Conv layer. Then, we denote by $m\text{SSN}_D$ a denoising CNN consisting of $m$ instances of $\text{DnCNN}_D$ interleaved

by $m-1$ instances of SSB. Figure 1b represents the $3\text{SSN}_6$ model that we adopt in our experiments, which consists of $3\times\text{DnCNN}_6 + 2\times\text{SSB}$. We point out that large SSN architectures are difficult to train, and a careful hyperparameter selection is needed to obtain satisfactory results. Moreover, the running time for an inference with $3\text{SSN}_6$ is about 7.5 times that of the baseline $\text{DnCNN}_{18}$.

### 4.4 Differences with Non-Local Neural Networks

The proposed solution shares the same rationale underpinning the Non-Local Block (NLB) in [22], since they are both inspired by NLM [1] principles. This section therefore discusses the main differences between the two, while the experimental comparison is in Section 5.3. NLB is defined as

$$\widehat{\mathbf{v}}_i = \frac{1}{C}\sum_j f(\mathbf{v}_i, \mathbf{v}_j) g(\mathbf{v}_j), \tag{7}$$

where $f$ computes a scalar similarity between two activation vectors, and $g$ is a linear embedding. In practice, NLB estimates each latent vector $\widehat{\mathbf{v}}_i$ by a weighted average of the embedded features $g(\mathbf{v}_j)$ from the whole input, where the weights are the scalar similarities $f(\mathbf{v}_i, \mathbf{v}_j)$ representing the attention. The primary difference with SSB is that this computes a multivariate similarity score $\mathbf{s}^k$ (represented in pink in Figure 1a, center). Moreover, the use of scores is different as $\mathbf{s}^k$ is mixed with the corresponding latent feature vector $\mathbf{v}^k$ and then averaged with learned weights $\alpha_k$ in the averaging sub-block (Figure 1a, right). Our experiments in Section 5.2 shows that a multivariate similarity scores improves the denoising performance. Another difference is that SSB operates on a search neighbor, considering all the activation vectors belonging to a spatial neighborhood using a stride. Instead, the NLB considers all the activation vectors in the activation maps. This is a viable approach only for small maps (used in high-level tasks addressed in [22]), but it is computationally intractable in dense regression tasks like image restoration. In our experiments, we test a modified version of NLB that compares references only with activation vectors within a fixed-size search neighborhood, as in [12]. This version, which we denote as NLBr, considers all the activation vectors in the neighborhood without a stride. We additionally include a strided alternative, denoted NLBr+. Our experiments demonstrate that stride, which we also employ in SSB, is beneficial and that SSB outperforms both the improved variants of NLB at high noise levels.

## 5 Experiments

We analyze the effectiveness of the SSB in grayscale image denoising. First, we present the datasets, figures of merit, and competing methods employed in our experiments (Section 5.1). Then, we investigate how the configuration of SSN influences the denoising performance (Section 5.2) and compare a selected SSN against state-of-the-art CNN denoisers (Section 5.3).

Table 1: Denoising performance of $2SSN_9$ for different values of half-neighborhood size $\Omega$ and stride $w$. These parameters determine the number of activation vectors $M$ used by the SSB.

| ($\Omega$, $w$) | (2,2) | (4,2) | (6,2) | (5,5) | (10,5) | (15,5) |
|---|---|---|---|---|---|---|
| **PSNR** | 30.15 | 30.19 | 30.25 | 30.18 | 30.30 | 30.38 |
| **SSIM** | 0.887 | 0.888 | 0.888 | 0.888 | 0.891 | 0.894 |
| $M$ | 9 | 25 | 49 | 9 | 25 | 49 |

Table 2: Denoising performance of $2SSN_9$ for different dimensions $n$ of the similarity score. Adopting a multivariate similarity boosts the denoising performance.

| $n$ | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 |
|---|---|---|---|---|---|---|---|---|
| **PSNR** | 30.131 | 30.134 | 30.197 | 30.285 | 30.358 | 30.368 | 30.375 | 30.374 |
| **SSIM** | 0.886 | 0.886 | 0.889 | 0.891 | 0.893 | 0.893 | 0.894 | 0.893 |

### 5.1 Experimental Settings

Our experiments tackle the denoising problem with $\sigma \in \{25, 50, 70\}$ by training a distinct model for each noise level. We train all models on $80 \times 80$ patches randomly cropped from BSD400 [15], corrupted with Gaussian noise with fixed variance $\sigma^2$. As test sets, we adopt Set12, BSD68 [15], and Urban100 [7]. Training and testing images are strictly disjoint. We assess the denoising performance by the Peak Signal-to-Noise Ratio (PSNR) and the Structural Similarity (SSIM) index [23]. We consider the following methods in our experiments.

**SSN:** The architecture of an SSN consists of instances of DnCNN interleaved by SSBs. Figure 1b reports the architecture of the SSN that we use in the main experiment. We set the number of filters $n$ in the similarity sub-block (Figure 1a) as the number $d$ of channels of the input. We train each SSN for 40 epochs using the Adam optimizer [9] with a mini-batch size of 16 and a weight decay factor of $10^{-6}$ on all convolutional weights except for those of SSB. We use a learning rate of $10^{-4}$ for the first 20 epochs, and then we divide it by 10 every 10 epochs. For a fair comparison, we adopt the $\ell_2$ loss, which most deep denoisers use even though it is not the best choice for image restoration models [29].

**Competing methods:** We compare SSN against a traditional denoising algorithm (BM3D [3]) and the following deep denoisers: DnCNN [26], NN3D [2] and $N^3$Net [17]. Since SSB aims to improve the denoising performance of CNN denoisers, we do not include recurrent networks or complex, branching, and computationally expensive convolutional or graph architectures. Furthermore, we train surrogates of SSN, replacing the proposed SSB with NLB [22]. We consider the setup and the results from [17] for all the methods. All the learning-based methods share the same training set as SSN and adopt comparable training procedures.

Table 3: Denoising performance of SSN for different configurations.

| Model | Description | PSNR | SSIM | # Params |
|---|---|---|---|---|
| $\text{DnCNN}_{18}$ | $1\times\text{DnCNN}_{18}$ | 29.9917 | 0.885 | 594 113 |
| $\text{DnCNN}_{20}$ | $1\times\text{DnCNN}_{20}$ | 30.0051 | 0.884 | 668 225 |
| $\text{2SSN}_9$ | $2\times\text{DnCNN}_9 + 1\times\text{SSB}$ | 30.3750 | 0.894 | 627 202 |
| $\text{3SSN}_6$ | $3\times\text{DnCNN}_6 + 2\times\text{SSB}$ | 30.4238 | 0.895 | 660 291 |
| $\text{4SSN}_5$ | $4\times\text{DnCNN}_5 + 3\times\text{SSB}$ | 30.4120 | 0.893 | 767 492 |

### 5.2 Assessing Different SSN Architectures

Many parameters affect the effectiveness of SSN, like the number of neighbors $M$, their distance to the reference, the dimension $n$ of the similarity scores, and the number of SSBs employed by the network. Here, we compare the performance achieved by different configurations of SSN over Urban100 with $\sigma = 25$.

The first experiment considers the $\text{2SSN}_9$, which contains a single SSB, with different values of half-neighborhood size $\Omega$ and stride $w$. Table 1 reports the number of neighbors $M$, and the PSNR and SSIM achieved by the SSN for various configurations of $\Omega$ and $w$. The results show that increasing $M$ leads to better denoising performance because we have more chances to find similar vectors. Moreover, for a fixed $M$, considering a larger search neighborhood (i.e., a larger stride) is beneficial. In the following, we set $\Omega = 15$ and $w = 5$, which corresponds to a $31 \times 31$ search region containing $M = 49$ activation vectors.

The second experiment assesses the importance of adopting a multivariate similarity score. To this end, we train several instances of $\text{2SSN}_9$ varying the parameter $n$, which defines the number of components of $\mathbf{s}_k$. With $n = 1$, the similarity of SSB is scalar and resembles the attention of NLB [22]. Table 2 shows a steady performance improvement when using multivariate similarity metrics, up to $n = 16$. Further increase of $n$ yields a marginal performance boost.

Finally, we evaluate the denoising performance of SSN when increasing the number of SSB instances. Precisely, we consider SSNs with 1, 2, or 3 SSBs compared with two baselines $\text{DnCNN}_{18}$ and $\text{DnCNN}_{20}$ having the same number of convolutional layers. Table 3 shows that a single SSB already improves the denoising performance with respect to the DnCNN architecture, and a second block grants further improvement. However, when adding a third SSB, the network training becomes particularly difficult and more unstable, leading to a small decrease of the denoising performance. In the following, we adopt the $\text{3SSN}_6$ architecture, depicted in Figure 1b.

### 5.3 Comparison against State-of-the-Art

In this section, we compare the denoising performance of $\text{3SSN}_6$ (Figure 1b) against the competing methods in Section 5.1. To fairly compare our method with NLB, we train similar architectures, where we replace our SSB with a NLBr or NLBr+ (defined in Section 4.4). The strided version NLBr+ adopts a selection

Table 4: Denoising performance (PSNR) achieved by the considered methods on three datasets. The best results are in bold.

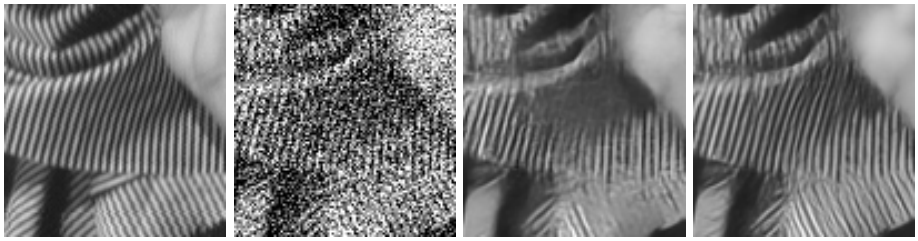
| | $\sigma$ | **BM3D** | **DnCNN** | **NN3D** | **N$^3$Net** | **NLBr** | **NLBr+** | **SSN** (ours) |
|---|---|---|---|---|---|---|---|---|
| **Set12** | 25 | 29.96 | 30.44 | 30.45 | 30.55 | 30.51 | 30.61 | **30.64** |
| | 50 | 26.70 | 27.19 | 27.24 | 27.43 | 27.43 | 27.41 | **27.46** |
| | 70 | 25.21 | 25.56 | 25.61 | 25.90 | 25.71 | 25.82 | **25.92** |
| **BSD68** | 25 | 28.56 | 29.23 | 29.19 | 29.30 | 29.30 | 29.36 | **29.37** |
| | 50 | 25.63 | 26.23 | 26.19 | 26.39 | 26.31 | 26.40 | **26.42** |
| | 70 | 24.46 | 24.85 | 24.89 | 25.14 | 25.04 | 25.08 | **25.16** |
| **Urban100** | 25 | 29.71 | 29.97 | 30.09 | 30.19 | 30.19 | 30.40 | **30.42** |
| | 50 | 25.95 | 26.28 | 26.47 | 26.82 | 26.56 | 26.83 | **26.92** |
| | 70 | 24.27 | 24.36 | 24.53 | 25.15 | 24.73 | 25.02 | **25.24** |

of neighbors similar to SSB. Both NLBr and NLBr+ consider the same number of activation vectors of our $3\text{SSB}_6$ (i.e., $M = 49$).

Table 4 reports the PSNR achieved by the considered methods on the test sets corrupted by noise of standard deviation $\sigma \in \{25, 50, 75\}$. The results for the competing methods are from [17]. SSN outperforms the competitors on all datasets and noise levels. In particular, the improvement is more significant on Urban100, whose images present many repeated patterns and structures, suggesting that our SSB effectively promotes NLSS. Our SSN significantly outperforms the NLBr implemented as in [22]. In contrast, SSN achieves performance equivalent to NLBr+ for low $\sigma$, while we observe a significant improvement for high noise levels. This suggests that relying on a multivariate similarity becomes beneficial when the degradation is strong. More generally, exploiting the NLSS prior is beneficial to image denoising. Indeed SSN, N$^3$Net, and NLB consistently outperform DnCNN, which in turn achieves a higher average PSNR than BM3D, demonstrating the effectiveness of deep learning methods. To gain further insights into the effect of NLSS, Figure 2 reports two denoised versions of a detail from Barbara (Set12), presenting a repeated fabric pattern. This example shows that SSN successfully recovers the fabric pattern also in low-contrast regions where DnCNN fails, demonstrating the effectiveness of our SSB at exploiting the NLSS of this texture.

## 6 Conclusions

We present SSB, a novel building block promoting the NLSS in CNN denoisers. Inspired by NLM, our block aggregates neighbor contributions from a search neighborhood, using weights that depend on the similarities with neighbors. The patch embedding, the multivariate similarity score, and the mixing function are learned during training. We show that SSB improves the performance of a baseline on AWGN suppression, particularly when $\sigma$ is high.



(a) Clean (PSNR/SSIM) | (b) Noisy $\sigma = 70$ (12.26 / 0.315) | (c) DnCNN (20.42 / 0.609) | (d) SSN (ours) (21.77 / 0.759)

Fig. 2: Detail of Barbara denoised by DnCNN and SSN. Our method recovers repeated patterns also in low-contrast regions.

Future work comprises training models for blind denoising, extending SSB to RGB images, which typically require adjustments in the network hyperparameters, as in [27]. Moreover, we will investigate the application of multi-head attention to image restoration, leveraging the connection between self-similarity and attention. Finally, we will address the computational efficiency problem by exploring alternative implementations to reduce the running time of SSB.

*Acknowledgments* We gratefully acknowledge the support of NVIDIA Corporation with the four RTX A6000 GPUs granted through the Applied Research Accelerator Program to Politecnico di Milano.

## References

1. Buades, A., Coll, B., Morel, J.: A non-local algorithm for image denoising. In: Proceedings of CVPR. vol. 2, pp. 60–65. IEEE (2005)
2. Cruz, C., Foi, A., Katkovnik, V., Egiazarian, K.: Nonlocality-reinforced convolutional neural networks for image denoising. IEEE Signal Processing Letters **25**(8), 1216–1220 (2018)
3. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: Image denoising by sparse 3-d transform-domain collaborative filtering. IEEE TIP **16**(8), 2080–2095 (2007)
4. Elad, M., Aharon, M.: Image denoising via sparse and redundant representations over learned dictionaries. IEEE TIP **15**(12), 3736–3745 (2006)
5. Foi, A.: Clipped noisy images: Heteroskedastic modeling and practical denoising. Signal Processing **89**(12), 2609–2629 (2009)
6. Gu, S., Zhang, L., Zuo, W., Feng, X.: Weighted nuclear norm minimization with application to image denoising. In: Proceedings of CVPR. pp. 2862–2869 (2014)
7. Huang, J.B., Singh, A., Ahuja, N.: Single image super-resolution from transformed self-exemplars. In: Proceedings of CVPR. pp. 5197–5206. IEEE (2015)
8. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of ICML. pp. 448–456. PMLR (2015)

9. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
10. Lefkimmiatis, S.: Non-local color image denoising with convolutional neural networks. In: Proceedings of CVPR. pp. 3587–3596. IEEE (2017)
11. Lefkimmiatis, S.: Universal denoising networks: a novel cnn architecture for image denoising. In: Proceedings of the CVPR. pp. 3204–3213. IEEE (2018)
12. Liu, D., Wen, B., Fan, Y., Loy, C.C., Huang, T.S.: Non-local recurrent network for image restoration. Advances in NeurIPS **31** (2018)
13. Luo, W., Li, Y., Urtasun, R., Zemel, R.: Understanding the effective receptive field in deep convolutional neural networks. Advances in NeurIPS **29** (2016)
14. Mao, X., Shen, C., Yang, Y.: Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. Advances in NeurIPS **29** (2016)
15. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proceedings of ICCV. vol. 2, pp. 416–423. IEEE (2001)
16. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE TPAMI **12**(7), 629–639 (1990)
17. Plötz, T., Roth, S.: Neural nearest neighbors networks. Advances in NeurIPS **31** (2018)
18. Portilla, J., Strela, V., Wainwright, M.J., Simoncelli, E.P.: Image denoising using scale mixtures of gaussians in the wavelet domain. IEEE TIP **12**(11), 1338–1351 (2003)
19. Roth, S., Black, M.J.: Fields of experts: A framework for learning image priors. In: Proceedings of CVPR. vol. 2, pp. 860–867. IEEE (2005)
20. Tai, Y., Yang, J., Liu, X., Xu, C.: Memnet: A persistent memory network for image restoration. In: Proceedings of ICCV. pp. 4539–4547. IEEE (2017)
21. Valsesia, D., Fracastoro, G., Magli, E.: Deep graph-convolutional image denoising. IEEE TIP **29**, 8226–8237 (2020)
22. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: Proceedings of CVPR. pp. 7794–7803. IEEE (2018)
23. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE TIP **13**(4), 600–612 (2004)
24. Yang, D., Sun, J.: BM3D-Net: A convolutional neural network for transform-domain collaborative filtering. IEEE Signal Processing Letters **25**(1), 55–59 (2017)
25. Zhang, K., Li, Y., Zuo, W., Zhang, L., Van Gool, L., Timofte, R.: Plug-and-play image restoration with deep denoiser prior. IEEE TPAMI (2021)
26. Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L.: Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. IEEE TIP **26**(7), 3142–3155 (2017)
27. Zhang, K., Zuo, W., Zhang, L.: Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. IEEE TIp **27**(9), 4608–4622 (2018)
28. Zhang, Y., Li, K., Zhong, B., Fu, Y.: Residual non-local attention networks for image restoration. In: International Conference on Learning Representations (2019)
29. Zhao, H., Gallo, O., Frosio, I., Kautz, J.: Loss functions for image restoration with neural networks. IEEE TCI **3**(1), 47–57 (2016)
30. Zoran, D., Weiss, Y.: From learning models of natural image patches to whole image restoration. In: Proceedings of ICCV. pp. 479–486. IEEE (2011)