

# xUnit: Learning a Spatial Activation Function for Efficient Image Restoration

Idan Kligvasser, Tamar Rott Shaham and Tomer Michaeli  
Technion–Israel Institute of Technology, Haifa, Israel

{kl igvasser@campus, stamarot@campus, tomer. m@ee}. technion. ac. il

## Abstract

*In recent years, deep neural networks (DNNs) achieved unprecedented performance in many low-level vision tasks. However, state-of-the-art results are typically achieved by very deep networks, which can reach tens of layers with tens of millions of parameters. To make DNNs implementable on platforms with limited resources, it is necessary to weaken the tradeoff between performance and efficiency. In this paper, we propose a new activation unit, which is particularly suitable for image restoration problems. In contrast to the widespread per-pixel activation units, like ReLUs and sigmoids, our unit implements a learnable non-linear function with spatial connections. This enables the net to capture much more complex features, thus requiring a significantly smaller number of layers in order to reach the same performance. We illustrate the effectiveness of our units through experiments with state-of-the-art nets for denoising, de-raining, and super resolution, which are already considered to be very small. With our approach, we are able to further reduce these models by nearly 50% without incurring any degradation in performance.*

## 1. Introduction

Deep convolutional neural networks (CNNs) have revolutionized computer vision, achieving unprecedented performance in high-level vision tasks such as classification [45, 14, 17], segmentation [38, 1] and face recognition [32, 43], as well as in low-level vision tasks like denoising [20, 48, 3, 35], deblurring [30], super resolution [9, 23, 21] and dehazing [37]. Today, the performance of CNNs is still being constantly improved, mainly by means of increasing the net’s depth. Indeed, identity skip connections [15] and residual learning [14, 48], used within ResNets [14] and DenseNets [17], now overcome some of the difficulties associated with very deep nets, and have even allowed to cross the 1000-layer barrier [15].

The strong link between performance and depth, has major implications on the computational resources and running times required to obtain state-of-the-art results. In parti-

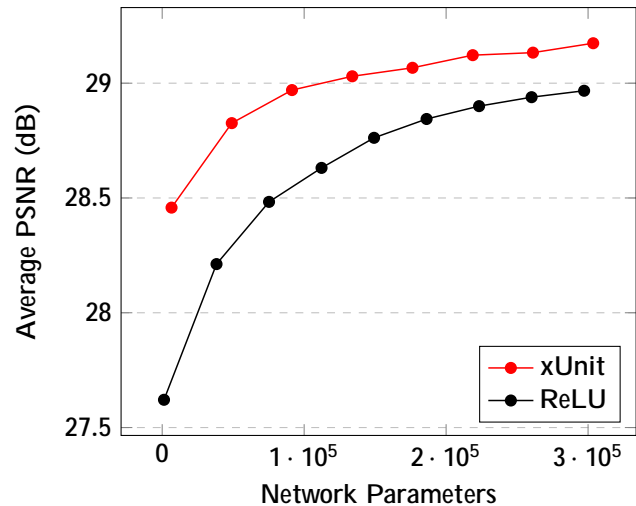


Figure 1. **xUnit activations vs. ReLU activations.** xUnits are nonlinear activations with *learnable spatial connections*. When used in place of the popular per-pixel activations (e.g. ReLUs), they lead to significantly better performance with a smaller number of total net parameters. The graph compares the denoising performance of a conventional ConvNet (Conv+BN+ReLU layers) with our xNet (Conv+xUnit layers) at a noise level of  $\sigma = 25$ . As can be seen, xNet attains a much higher PSNR with the same number of parameters. Alternatively, it can achieve the same PSNR with roughly 1/3 the number of ConvNet parameters.

cular, it implies that applications on real-time, low-power, and limited resource platforms (e.g. mobile devices), cannot currently exploit the full potential of CNNs.

In this paper, we propose a different mechanism for improving CNN performance (see Fig. 1). Rather than increasing depth, we focus on making the nonlinear activations more effective. Most popular architectures use per-pixel activation units, e.g. rectified linear units (ReLUs) [11], exponential linear units (ELUs) [5], sigmoids [31], etc. Here, we propose to replace those units by *xUnit*, a layer with *spatial* and *learnable* connections. The xUnit computes a continuous-valued weight map, serving as a soft gate to its input. As we show, although it is more computationally demanding and memory consuming than a per-pixel unit,

the xUnit has a dramatic effect on the net’s performance. Therefore, it allows using far fewer layers to match the performance of a CNN with ReLU activations. Overall, this results in a significantly improved tradeoff between performance and efficiency, as illustrated in Fig. 1.

The xUnit has a set of learnable parameters. Therefore, to conform to a total budget of parameters, xUnits must come at the expense of some of the convolutional layers in the net or some of the channels within those convolutional layers. This raises the question: What is the optimal percentage of parameters to invest in the activation units? Today, most CNN architectures are at one extreme of the spectrum, with 0% of the parameters invested in the activations. Here, we show experimentally that the optimal percentage is much larger than zero. This suggests that the representational power gained by using spatial activations can be far more substantial than that offered by convolutional layers with per-pixel activations.

We illustrate the effectiveness of our approach in several image restoration tasks. Specifically, we take state-of-the-art CNN models for image denoising [48], super-resolution [9, 23], and de-raining [10], which are already considered to be very light-weight, and replace their activations with xUnits. We show that this allows us to further reduce the number of parameters (by discarding layers or channels) without incurring any degradation in performance. In fact, we show that for small models, we can save nearly 50% of the parameters while achieving the same performance or even better. As we show, this often allows to use three orders of magnitude less training examples.

## 2. Related Work

The quest for accurate image enhancement algorithms attracted significant research efforts over the past several decades. Until 2012, the vast majority of algorithms relied on generative image models, usually through maximum a-posteriori (MAP) estimation. Models were typically either hand-crafted or learned from training images, and included *e.g.* priors on derivatives [41], wavelet coefficients [33], filter responses [39], image patches [7, 50], etc. In recent years, generative approaches are gradually being pushed aside by discriminative methods, mostly based on CNNs. These architectures typically directly learn a mapping from a degraded image to a restored one, and were shown to exhibit excellent performance in many restoration tasks, including *e.g.* denoising [3, 48, 20], deblurring [30], super-resolution [8, 9, 23, 21], dehazing [4, 25], and de-raining [10].

A popular strategy for improving the performance of CNN models, is by increasing their depth. Various works suggested ways to overcome some of the difficulties in training very deep nets. These opened the door to a line of algorithms using ever larger nets. Specifically, the residual net (ResNet) architecture [14], was demonstrated to achieve

exceptional classification performance compared to a plain network. Dense convolutional networks (DenseNets) [17] took the “skip-connections” idea one step further, by connecting each layer to every other layer in a feed-forward fashion. This allowed to achieve excellent performance in very deep nets.

These ideas were also adopted by the low-level vision community. In the context of denoising, Zhang *et al.* [48] were the first to train a very deep CNN for denoising, yielding state-of-the-art results. To train their net, which has 0.5M parameters, they utilized residual learning and batch normalization [48], alleviating the vanishing gradients problem. In [20], a twice larger model was proposed, which is based on formatting the residual image to contain structured information instead of learning the difference between clean and noisy images. Similar ideas were also proposed in [34, 36, 49], leading to models with large numbers of parameters. Recently, a very deep network based on residual learning was proposed in [2], which contains more than 60 layers, and 17M parameters.

In the context of super-resolution, the progress was similar. In the near past, state-of-the-art methods used only a few tens of thousands of parameter. For example, the SR-CNN model [9] contains only three convolution layers, with only 57K parameters. The very deep super-resolution model (VDSR) [21] already used 20 layers with 660K parameters. Nowadays, much more complex models are in use. For example, the well-known SRResNet [23] uses more than 1.5M parameters, and the EDSR network [27] (winner of the NTIRE2017 super resolution challenge [46]), has 43M parameters.

The trend of making CNNs as deep as possible, poses significant challenges in terms of running those models on platforms with low-power and limited computation and memory resources. One approach towards diminishing memory consumption and access times, is to use binarized neural networks [6]. These architectures, which were shown beneficial in classification tasks, constrain the weights and activations to be binary. Another approach is to replace the multi-channel convolutional layers by depth-wise convolutions [16]. This offers a significant reduction in size and latency, while allowing reasonable classification accuracy. In [44], it was suggested to reduce network complexity and memory consumption for super resolution, by introducing a sub-pixel convolutional layer that learns upscaling filters. In [24], an architecture which exploits non-local self-similarities in images, was shown to yield good results with reduced models. Finally, learning the optimal slope of leaky ReLU type activations has also shown to lead to more efficient models [13].

Figure 2. **The xUnit activation layer.** (a) The popular ReLU activation can be interpreted as performing an element-wise product between its input  $z_k$  and a weight map  $g_k$ , which is a binarized version of  $z_k$ . (b) The xUnit constructs a continuous weight map taking values in  $[0, 1]$ , by performing a nonlinear *learnable spatial function* on  $z_k$ .

### 3. xUnit

Although a variety of CNN architectures exist, their building blocks are quite similar, mainly comprising of convolutional layers and per-pixel activation units. Mathematically, the features  $x_{k+1}$  at layer  $k + 1$  are commonly calculated as

$$\begin{aligned} z_k &= W_k x_k + b_k, \\ x_{k+1} &= f(z_k), \end{aligned} \quad (1)$$

where  $x_0$  is the input to the net,  $W_k$  performs a convolution operation,  $b_k$  is a bias term,  $z_k$  is the output of the convolutional layer, and  $f(\cdot)$  is some nonlinear activation function which operates element-wise on its argument. Popular activation functions include the ReLU [11], leaky ReLU [28], ELU [5], tanh and sigmoid functions.

Note that there is a clear dichotomy in (1): The convolutional layers are responsible for the *spatial processing*, and the activation units for the *nonlinearities*. One may wonder if this is the most efficient way to realize the complex functions needed in low-level vision. In particular, is there any reason not to allow spatial processing also within the activation functions?

Element-wise activations can be thought of as nonlinear gating functions. Specifically, assuming that  $f(0) = 0$ , as is the case for all popular activations, (1) can be written as

$$x_{k+1} = z_k \odot g_k, \quad (2)$$

where  $\odot$  denotes the (element-wise) Hadamard product, and  $g_k$  is a (multi-channel) weight map that depends on  $z_k$  element-wise, as

$$[g_k]_i = \frac{[f(z_k)]_i}{[z_k]_i}. \quad (3)$$

Here  $0/0$  should be interpreted as 0. For example, the weight map  $g_k$  associated with the ReLU function  $f(\cdot)$ , is a

binary map which is a thresholded version of  $z_k$ ,

$$[g_k]_i = \begin{cases} 1 & [z_k]_i > 0, \\ 0 & [z_k]_i \leq 0. \end{cases} \quad (4)$$

This interpretation is visualized in Fig. 2(a) (bias not shown).

Since the nonlinear activations are what grants CNNs their ability to implement complex functions, here we propose to use *learnable spatial activations*. That is, instead of the element-wise relation (3), we propose to allow each element in  $g_k$  to depend also on the spatial neighborhood of the corresponding element in  $z_k$ . Specifically, we introduce *xUnit*, in which

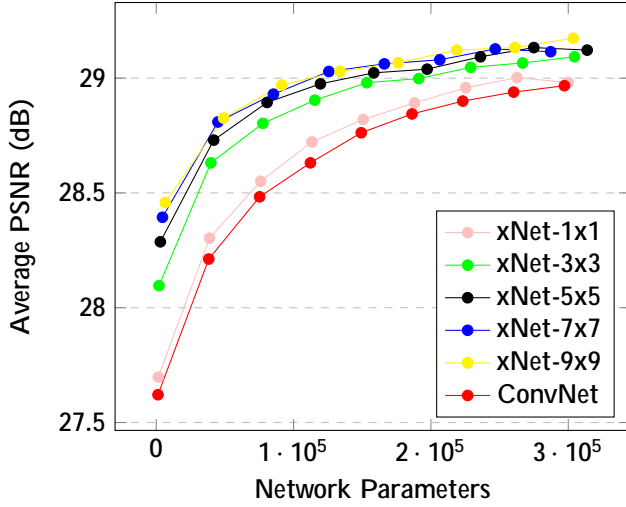
$$[g_k]_i = \exp\{-[d_k]_i^2\}, \quad (5)$$

and

$$d_k = H_k \text{ReLU}(z_k), \quad (6)$$

with  $H_k$  denoting depth-wise convolution [16]. The idea is to introduce (i) nonlinearity (ReLU), (ii) spatial processing (depth-wise convolution), and (iii) construction of gating maps in the range  $[0, 1]$  (Gaussian). The depth-wise convolution applies a single filter to each input channel, and is significantly more efficient in terms of memory and computations than the multi-channel convolution popularly used in CNNs. Note that the filters  $H_k$  have to be learned during training. To make the training stable, we also add batch-normalization layers [19] before the ReLU and before the exponentiation. This is illustrated in Fig. 2(b).

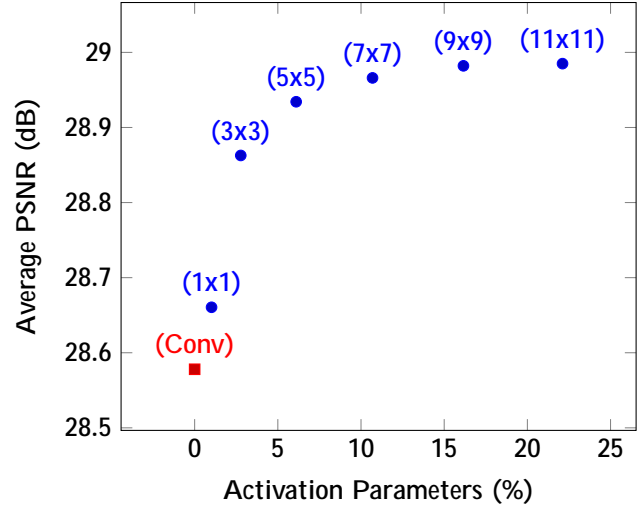
Merely replacing ReLUs with xUnits clearly increases memory consumption and running times at test stage. This is mostly due to their convolutional operations (the exponent can be implemented using a look-up table). Specifically, an xUnit with a  $d$ -channel input and a  $d$ -channel output involving  $r \times r$  filters, introduces an overhead of



**Figure 3. Denoising performance vs. number of parameters.** We compare a ConvNet composed of feed-forward Conv+BN+ReLU layers, with our xNet which comprises a sequence of Conv+xUnit layers. We gradually increase the number of layers for both nets and record the average PSNR obtained in denoising the BSD68 dataset with noise level = 25, as a function of the total number of parameters in the net. Training configurations are the same for both networks. Our xNet attains a much higher PSNR with the same number of parameters. Alternatively, it can achieve the same PSNR with roughly one third the number of ConvNet parameters.

$(r^2 + 4)d$  parameters ( $r \times r \times d$  for the depth-wise filters, and  $2 \times d$  for each batch-normalization layer). However, first, note that this overhead is relatively mild compared to the  $r^2 d^2$  parameters of each  $r \times r \times d \times d$  convolutional layer. Second, in return to that overhead, xUnits provide a performance boost. This means that the same performance can be attained with less layers or with less channels per layer. Therefore, the important question is whether xUnits improve the overall *tradeoff* between performance and number of parameters.

Figure 3 shows the effect of using xUnits in a denoising task. Here, we trained two simple net architectures to remove additive Gaussian noise of standard deviation = 25 from noisy images, using a varying number of layers. The first net is a traditional ConvNet architecture comprising a sequence of Conv+BN+ReLU layers. The second net, which we coin xNet, comprises a sequence of Conv+xUnit layers. In both nets, the regular convolutional layers (not the ones within the xUnits) comprise 64 channel  $3 \times 3$  filters. For the xUnits, we varied the size of the depth-wise filters from  $1 \times 1$  to  $9 \times 9$ . We trained both nets on 400 images from the BSD dataset [29] using residual learning (*i.e.* learning to predict the noise and subtracting the noise estimate from the noisy image at test time). This has been shown to be advantageous for denoising in [48]. As can be

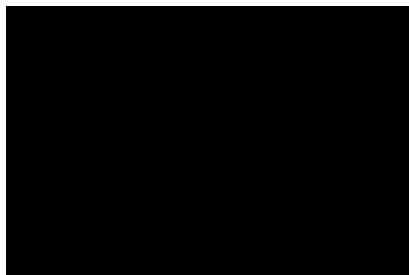


**Figure 4. Denoising performance vs. percentage of activation parameters.** Varying the supports of the xUnit filters improves performance, but also increases the overall number of parameters. Here, we show the average denoising PSNR as a function of the *percentage of the overall parameters* invested in the xUnit activations, when the total number of parameters is constrained to 99,136. This corresponds to a vertical cross section of the graph in Fig. 3. As can be seen, while conventional ConvNets invest 0% of the parameters in the activations, this is clearly sub-optimal. A significant improvement in performance is obtained when investing about 20% of the total parameters in the activations.

seen in the figure, when the xUnit filters are  $1 \times 1$ , the peak signal to noise ratio (PSNR) attained by xNet exceeds that of ConvNet by only a minor gap. In this case, the xUnits are not spatial. However, as the xUnits' filters become larger, xNet's performance begins to improve, for any given total number of parameters. Note, for example, that a 3 layer xNet with  $9 \times 9$  activations outperforms a 9 layer ConvNet, although having less than 1/3 the number of parameters.

To further understand the performance-computation tradeoff when using spatial activations, Fig. 4 shows a vertical cross section of the graph in Fig. 3 at an overall of 99,136 parameters. Here, the PSNR is plotted against the percentage of parameters invested in the xUnit activations. In a traditional ConvNet, 0% of the parameters are invested in the activations. As can be seen in the graph, this is clearly sub-optimal. In particular, the optimal percentage can be seen to be at least 22%, where the performance of xNet reaches a plateau. In fact, after around 15% (corresponding to  $9 \times 9$  activation filters), the benefit in further increasing the filters' supports becomes relatively minor.

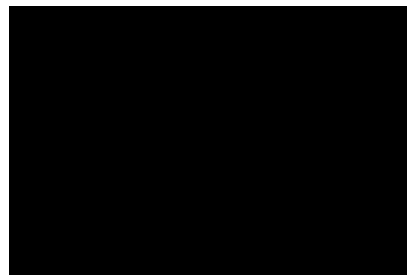
To gain intuition into the mechanism that allows xNet to achieve better results with less parameters, we depict in Fig. 5 the layer 4 feature maps  $z_4$ , weight (activation) maps  $g_4$ , and their products  $x_5$ , for a ConvNet and an xNet operating on the same noisy input image. Interestingly, we



(a) xNet-4: layer-4 features.



(b) xNet-4: layer-4 activation maps.



(c) xNet-4: layer-4 multiplication results.

(d) ConvNet-4: layer-4 features.

(e) ConvNet-4: layer-4 activation maps.

(f) ConvNet-4: layer-4 multiplication results.

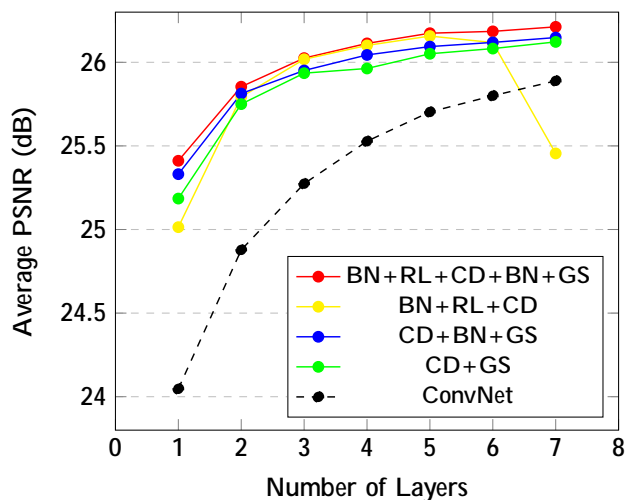
**Figure 5. Visualization of xNet activations.** A 4-layer xNet and a 4-layer ConvNet (with ReLU activations) were trained to denoise images with noise level  $\sigma = 25$  using direct learning. The 64 feature maps, activation maps, and their products, at layer 4 are shown for both nets, when operating on the same input image. In xNet, each activation map is a spatial function of the corresponding feature map, whereas in ConvNet, it is a per-pixel function. As can be seen, the ConvNet’s activations are very sparse, while the xNet’s activations are quite dense. Thus, it seems that in xNet, more feature maps participate in the denoising effort.

see that many more xNet activations are close to 1 (white) than ConvNet activations. Thus, it seems that in xNet, more channels take part in the denoising effort. Moreover, it can be seen that the xNet weight maps are quite complex functions of the features, as opposed to the simple binarization function of the ReLUs in ConvNet.

Figure 6 compares several alternative xUnit designs. The suggested design, which contains Batch Norm. (BN), ReLU (RL), Conv. Depth-wise (CD) and Gaussian (GS), achieves the best results. However, note that all designs perform significantly better than a conventional ConvNet, indicating that the spatial processing (CD, which appears in all designs) contributes the most. Interestingly, the BN+RL+CD combination, which allows the weight maps to contain negative values, performs quite similarly to our suggested design when the number of layers is small. Nonetheless, unlike the other designs, for a larger number of layers we experience gradients exploding during training. This highlights the importance of the Gaussian, which regulates training by keeping weights in the range  $[0, 1]$ .

## 4. Experiments and Applications

Our goal is to show that many small-scale and medium-scale state-of-the-art CNNs can be made almost 50% smaller with xUnits, without incurring any degradation in performance.



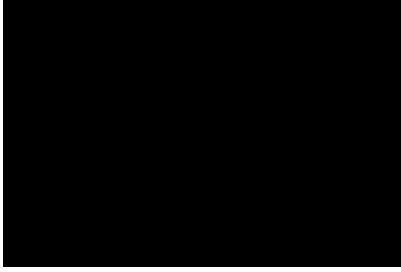
**Figure 6. xUnit design comparison.** We compare various xUnit designs with a traditional ConvNet. We gradually increase the number of layers for all nets and record the average PSNR obtained in denoising the BSD68 dataset with noise level  $\sigma = 50$ . Training configurations are the same for all nets. The suggested version, BN+RL+CD+BN+GS, attains the highest PSNR.

We implemented the proposed architecture in Pytorch. We ran all experiments on a desktop computer with an Intel i5-6500 CPU and an Nvidia 1080Ti GPU. We used the



Methods	BM3D	WNNM	EPLL	MLP	DnCNN-S	xDnCNN
# of parameters	-	-	-	-	555K	<b>303K</b>
$\sigma = 25$	<b>28.56</b>	<b>28.82</b>	<b>28.68</b>	<b>28.95</b>	<b>29.22</b>	<b>29.21</b>
$\sigma = 50$	<b>25.62</b>	<b>25.87</b>	<b>25.67</b>	<b>26.01</b>	<b>26.23</b>	<b>26.26</b>

Table 1. **Denoising performance.** The average PSNR in [dB] attained by several state of the art denoising algorithms on the BSD68 dataset. As can be seen, our xDnCNN outperforms all non-CNN methods and achieves results that are on par with DnCNN, although having roughly 1/2 the number of parameters.



(a) Noisy : 14.79dB

(b) EPLL: 29.34dB

(c) BM3D: 29.76dB

(d) MLP : 30.14dB

(e) DnCNN: 30.22dB

(f) xDnCNN (Our) : **30.43dB**

Figure 7. **Image denoising result.** Comparison of the denoised images produced by EPLL, BM3D, MLP, DnCNN and our xDnCNN for a noise level of  $\sigma = 50$ . In contrast to the competing methods, our xDnCNN manages to restore more of the image details, and introduces no distracting artifacts. This is despite the fact that it has nearly half the number of parameters as DnCNN.

Adam [22] optimizer with its default settings for training the nets. We initialized the learning rate to  $10^{-3}$  and gradually decreased it to  $10^{-4}$  during training. We kept the mini-batch size fixed at 64. In all applications, we used  $9 \times 9$  depth-wise convolutions in the xUnits, and minimized the mean square error (MSE) over the training set.

#### 4.1. Image Denoising

We begin by illustrating the effectiveness of xUnits in image denoising. As a baseline architecture, we take the state-of-the-art DnCNN denoising network [48]. We replace all ReLU layers with xUnit layers and reduce the number of convolutional layers from 17 to 9. We keep all convolutional layers with 64 channel  $3 \times 3$  filters, as in the original architecture. Our net, which we coin *xDnCNN*, has only 54% the number of parameters of DnCNN (303K for xDnCNN and 555K for DnCNN).

As in [48], we train our net on 400 images. We use images from the Berkeley segmentation dataset (BSD) [29], en-

riched by random flipping and random cropping ( $80 \times 80$ ). The noisy images are generated by adding Gaussian noise to the training images (different realization to each image). We examine the performance of our net at noise levels  $\sigma = 25, 50$ . Table 1 compares the average PSNR attained by our xDnCNN to that attained by the original DnCNN (variant ‘S’), as well as to the state-of-the-art non-CNN denoising methods BM3D [7], WNNM [12], EPLL [50], and MLP [3]. The evaluation is performed on the BSD68 dataset [40], a subset of 68 images from the BSD dataset, which is not included in the training set. As can be seen, our xDnCNN outperforms all non-CNN denoising methods and achieves results that are on par with DnCNN. This is despite the fact that xDnCNN is nearly half the size of DnCNN in terms of number of parameters. The superiority of our method becomes more significant as the noise level increases. At a noise level of  $\sigma = 50$ , our method achieves the highest PSNR values on 57 out of the 68 images in the dataset.

(a) Real rain

(b) DerainNet

(c) DnCNN

(d) xDnCNN

Figure 8. **De-raining a real rain image.** Our xDnCNN deraining network manages to remove rain streaks from a real rainy image, significantly better than DerainNet. This is despite the fact that our net has only 40% the number of parameters of DerainNet.

Figure 7 shows an example denoising result obtained with xDnCNN, compared with BM3D, EPLL, MLP and DnCNN-s, for a noise level of  $\sigma = 50$ . As can be seen, our xDnCNN best reconstructs the fine details and barely introduces any distracting artifacts. In contrast, all the other methods (including DnCNN), introduce unpleasing distortions.

## 4.2. Single image rain removal

Next, we use the same architecture in the task of removing rain streaks from a single image. We only introduce one modification to our denoising xDnCNN, which is to work on three channel (RGB) input images and to output three channel images. This results in a network with 306K parameters. We compare our results to a 3-channel input 3-channel output DnCNN version and to DerainNet [10], a network with 753K parameters, which comprises three convolutional layers:  $16 \times 16 \times 512$ ,  $1 \times 1 \times 512$  and  $8 \times 8 \times 3$ , respectively. Similarly to denoising, we learn the residual mapping between a rainy image and a clean image. Training is performed on the dataset of DerainNet [10], which contains 4900 pairs of clean and synthetically generated rainy images. However, we evaluate our net on the Rain12 dataset [26], which contains 12 artificially generated images. Although the training data is quite different from the test data, our xDnCNN performs significantly better than DnCNN and DerainNet, as shown in Table 2. This behavior is also seen when de-raining real images. As can be seen in Fig. 8, xDnCNN perform significantly better in cleaning actual rain streaks. We thus conclude that xDnCNN is far more robust to different rain appearances, while maintaining its efficiency. Pay attention that our xDnCNN deraining net has only 40% the number of parameters of DerainNet and only 55% the number of parameters of DnCNN.

## 4.3. Single image super resolution

Our xUnit activations can be also applied in single image super resolution. We illustrate this with the state-of-the-art SRResNet architecture [23] and with the very small SRCNN [9] model. For SRResNet, we replace the PReLU

Methods	De-rainNet	DnCNN	xDnCNN
# of parameters	753K	558K	<b>306K</b>
PSNR [dB]	28.94	30.90	<b>31.17</b>

Table 2. **De-raining performance on the Rain12 dataset.** Our xDnCNN attains a significantly higher PSNR than DnCNN and De-rainNet, with significantly less parameters.

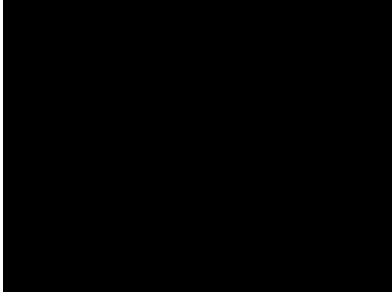
activations in the residual blocks by xUnits, and reduce the number of residual blocks from 16 to 10. This variant, which we coin xSRResNet, has only 75% the number of parameters of SRResNet (1.546M for SRResNet and 1.155M for xSRResNet). The SRCNN architecture contains three convolutional layers:  $9 \times 9 \times 64$ ,  $5 \times 5 \times 32$  and  $5 \times 5 \times 1$ . Here, we study two different modifications to SRCNN, where we replace the two ReLU layers with xUnit layers. In the first modification, we reduce the size of the filters in the middle layer from  $5 \times 5 \times 32$  to  $3 \times 3 \times 32$ . This variant, which we coin xSRCNNf, has only 56% the number of parameters of SRCNN (32K for xSRCNNf and 57K for SRCNN). In the second modification, we reduce the number of channels in the second layer from 64 to 42. This variant, which we coin xSRCNNc, has only 77% the number of parameters of SRCNN (44K for xSRCNNc and 57K for SRCNN).

The original SRCNN and SRResNet models were trained on about 400,000 images from ImageNet [42]. Here, we train our models on much smaller datasets. Specifically, for xSRCNN we use only 91 images from [47] and 400 images from BSD. For xSRResNet, we use 25,000 images from the Mirflickr25k [18] dataset. We augment the data by random flipping and random cropping.

Table 3 reports the results attained by all the models on BSD100 dataset. As can be seen, our models attain results that are on par with the original SRCNN and SRResNet, although being much smaller and trained on a significantly smaller number of images. Note that our xSRCNNf has less parameters than xSRCNNc. This may suggest that a better way to discard parameters in xNets is by reducing filter sizes, rather than reducing channels. A possible explana-

Methods	SRCNN	xSRCNNc	xSRCNNf	SRResNet	xSRResNet
# of parameters	57K	44K	<b>32K</b>	1.546M	<b>1.155M</b>
3×	28.41	<b>28.54</b>	28.53	-	-
4×	26.90	27.04	<b>27.06</b>	27.58	<b>27.61</b>

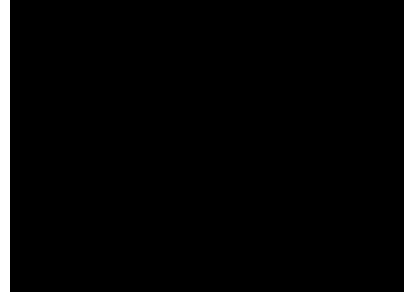
**Table 3. Super-resolution performance.** The average PSNR in [dB] attained in the task of 3× and 4× SR on BSD100 dataset. SRCNN was trained on  $4 \times 10^5$  training examples, whereas our xSRCNN models were trained on only 491 images. SRResNet was trained on  $3.5 \times 10^5$  training examples, whereas our xSRResNet was trained on  $2.5 \times 10^4$  examples.



(a) xSRCNNf: layer-2 features.



(b) xSRCNNf: layer-2 activation maps.



(c) xSRCNNf: layer-2 multiplication results.

(d) SRCNN: layer-2 features.

(e) SRCNN: layer-2 activation maps.

(f) SRCNN: layer-2 multiplication results.

**Figure 9. Visualization of SRCNN and xSRCNNf activations.** The 32 features maps, activation maps, and their products at layer 2 are shown for both nets, when operating on the same input image for magnification 3×. As can be seen, the activations of SRCNN are sparse, while those of xSRCNNf are dense. Thus, more feature maps participate in the reconstruction. This provides an explanation for the superiority of xSRCNNf over SRCNN in terms of PSNR.

tion is that the  $9 \times 9$  filters within the xUnits can partially compensate for the small support of the filters in the convolutional layers. However, the fact that discarding channels can also provide a significant reduction in parameters at the same performance, indicates that the channels in an xNet are more effective than those in ConvNets with per-pixel activations.

Figure 9 shows the layer 2 feature maps, activation maps, and their products for both SRCNN and our xSRCNNf. As in the case of denoising, we can see that in xSRCNN, many more feature maps participate in the reconstruction effort compared to SRCNN. This provides a possible explanation to its ability to perform well with smaller filters (or with less channels).

## 5. Conclusion

Popular CNN architectures use simple nonlinear activation units (*e.g.* ReLUs), which operate pixel-wise on the feature maps. In this paper, we demonstrated that CNNs can greatly benefit from incorporating learnable spatial connections within the activation units. While these spatial connections introduce additional parameters to the net, they significantly improve its performance. Overall, the trade-off between performance and number of parameters, is substantially improved. We illustrated how our approach can reduce the size of several state-of-the-art CNN models for denoising, de-raining and super-resolution, which are already considered to be very small, by almost 50%. This is without incurring any degradation in performance.

**Acknowledgements** This research was supported in part by an Alon Fellowship and by the Ollendorf Foundation.



## References

- [1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015. **1**
- [2] W. Bae, J. Yoo, and J. C. Ye. Beyond deep residual learning for image restoration: Persistent homology-guided manifold simplification. *arXiv preprint arXiv:1611.06345*, 2016. **2**
- [3] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2392–2399. IEEE, 2012. **1, 2, 6**
- [4] B. Cai, X. Xu, K. Jia, C. Qing, and D. Tao. Dehazenet: An end-to-end system for single image haze removal. *IEEE Transactions on Image Processing*, 25(11):5187–5198, 2016. **2**
- [5] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. **1, 3**
- [6] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016. **2**
- [7] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007. **2, 6**
- [8] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *European Conference on Computer Vision*, pages 184–199. Springer, 2014. **2**
- [9] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2016. **1, 2, 7**
- [10] X. Fu, J. Huang, X. Ding, Y. Liao, and J. Paisley. Clearing the skies: A deep network architecture for single-image rain removal. *IEEE Transactions on Image Processing*, 26(6):2944–2956, 2017. **2, 7**
- [11] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011. **1, 3**
- [12] S. Gu, L. Zhang, W. Zuo, and X. Feng. Weighted nuclear norm minimization with application to image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2862–2869, 2014. **6**
- [13] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. **2**
- [14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. **1, 2**
- [15] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016. **1**
- [16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. **2, 3**
- [17] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016. **1, 2**
- [18] M. J. Huiskes and M. S. Lew. The mir flickr retrieval evaluation. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 39–43. ACM, 2008. **7**
- [19] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015. **3**
- [20] J. Jiao, W.-C. Tu, S. He, and R. W. Lau. Formresnet: Formatted residual learning for image restoration. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 1034–1042. IEEE, 2017. **1, 2**
- [21] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1646–1654, 2016. **1, 2**
- [22] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. **6**
- [23] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016. **1, 2, 7**
- [24] S. Lefkimmiatis. Non-local color image denoising with convolutional neural networks. *arXiv preprint arXiv:1611.06757*, 2016. **2**
- [25] B. Li, X. Peng, Z. Wang, J. Xu, and D. Feng. An all-in-one network for dehazing and beyond. *arXiv preprint arXiv:1707.06543*, 2017. **2**
- [26] Y. Li, R. T. Tan, X. Guo, J. Lu, and M. S. Brown. Rain streak removal using layer priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2736–2744, 2016. **7**
- [27] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017. **2**
- [28] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013. **3**
- [29] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001. **4, 6**
- [30] M. Noroozi, P. Chandramouli, and P. Favaro. Motion deblurring in the wild. *arXiv preprint arXiv:1701.01486*, 2017. **1, 2**
- [31] G. B. Orr and K.-R. Müller. *Neural networks: tricks of the trade*. Springer, 2003. **1**

- [32] O. M. Parkhi, A. Vedaldi, A. Zisserman, et al. Deep face recognition. In *BMVC*, volume 1, page 6, 2015. **1**
- [33] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Transactions on Image processing*, 12(11):1338–1351, 2003. **2**
- [34] T. Remez, O. Litany, R. Giryes, and A. M. Bronstein. Deep class aware denoising. *arXiv preprint arXiv:1701.01698*, 2017. **2**
- [35] T. Remez, O. Litany, R. Giryes, and A. M. Bronstein. Deep class-aware image denoising. In *Sampling Theory and Applications (SampTA), 2017 International Conference on*, pages 138–142. IEEE, 2017. **1**
- [36] T. Remez, O. Litany, R. Giryes, and A. M. Bronstein. Deep convolutional denoising of low-light images. *arXiv preprint arXiv:1701.01687*, 2017. **2**
- [37] W. Ren, S. Liu, H. Zhang, J. Pan, X. Cao, and M.-H. Yang. Single image dehazing via multi-scale convolutional neural networks. In *European Conference on Computer Vision*, pages 154–169. Springer, 2016. **1**
- [38] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015. **1**
- [39] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 860–867. IEEE, 2005. **2**
- [40] S. Roth and M. J. Black. Fields of experts. *International Journal of Computer Vision*, 82(2):205–229, 2009. **6**
- [41] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992. **2**
- [42] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. **7**
- [43] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015. **1**
- [44] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, 2016. **2**
- [45] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. **1**
- [46] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, L. Zhang, B. Lim, S. Son, H. Kim, S. Nah, K. M. Lee, et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 1110–1121. IEEE, 2017. **2**
- [47] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE transactions on image processing*, 19(11):2861–2873, 2010. **7**
- [48] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 2017. **1, 2, 4, 6**
- [49] K. Zhang, W. Zuo, S. Gu, and L. Zhang. Learning deep cnn denoiser prior for image restoration. *arXiv preprint arXiv:1704.03264*, 2017. **2**
- [50] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 479–486. IEEE, 2011. **2, 6**