# A Survey of Deep Learning Techniques for Image Denoising

**Karan Goel**
2011EE50555
Department of Electrical Engineering
Indian Institute of Technology - Delhi
kgoel93@gmail.com

## Abstract

In this paper, various developments in the field of Deep Learning, specifically for the task of image denoising have been explored. The broad methodology, and the results shown by the various approaches that have developed in this area are outlined, and the the techniques are discussed in some detail. More attention is paid to models called Denoising Auto-Encoders which have shown great promise in learning interesting representations that have a wide range of applicability for image denoising, and beyond. In addition to DAEs, Convolutional Neural Networks are also covered, along with plain Multi-Layered Perceptrons.

## 1 Introduction

The need to remove noise from images is a wide-spread multimedia application that is frequently required in most computer, and mobile technologies. The task of *image denoising* looks at the problem of removing this noise, in a way that the recovered image gives the truest reconstruction of the original, noise-free image.

Several techniques for image denoising have been proposed that work at the cross-section of Multimedia Systems, Machine Learning, and Statistical Signal processing. In this paper, techniques proposed in the Machine Learning community, specifically relating to Deep Learning, have been examined. In particular I look at the evolution of these techniques, starting from plain Neural Networks, to sophisticated deep architectures.

## 2 Multimedia Applications

Image denoising finds a large range of applications in Multimedia Systems. One of the primary uses of denoising is in the process of medical imaging. Due to the importance of minute detail in medical imaging technologies, denoising the image to recover such details, or remove noise that has crept in due to the imaging or compression process is essential.

Another application of denoising lies in cameras; camera technology frequently uses filters, and motion-blur removal. However, natural images are also known to suffer from a problem of noise, that may creep in during the capture process. Image denoisers are frequently integrated into the camera pipeline to improve image quality and PSNR.

Denoising also finds a large scale application in the new age technologies of today; specifically, the Internet of Things framework is a domain in which denoising becomes essential. The nature of small sensing technologies which frequently involve image capture, require an image denoising step to make sensors more effective. If the denoiser can compensate for the quality of the imaging

(sensors are often cheap, and not very high quality), then we can even trade-off a lower quality sensor, with a high quality denoising step.

## 3 Background

In this section, the general process of image de-noising has been laid out, to clarify the task at hand. We start by looking at the various ways in which images may exhibit noise, and what the image denoising process hopes to achieve. Lastly, we look at the metrics that can be used to evaluate the denoised images, and the quality of the denoising.

This paper restricts itself to the denoising task for greyscale images, although the task may be applied to more complex, RGB images, or even hyper-spectral images, by a simple extension.

### 3.1 Noise Models in Images

Let each pixel in a greyscale image $\mathcal{I}$ have intensity $u(i, j)$, where $i, j$ gives the row and column point at which the pixel is located. Denote the true image pixel intensity values by $u_T(i, j)$, and let the noise generated be *additive Gaussian noise*, with value $n(i, j)$. The final pixel intensity value, or the noisy value, will be given by $u_N(i, j)$.

Additive Gaussian noise will be of the form

$$n(i, j) = \mathcal{N}(\mu, \sigma^2) \tag{1}$$

where $\mu = 0$ is commonly taken to be the mean of the Gaussian noise, and $\sigma$ is the standard deviation.

The noisy image, $\mathcal{I}_N$, will have a pixel intensity value given by,

$$u_N(i, j) = u_T(i, j) + n(i, j) \tag{2}$$

where $0 \leq i \leq H$ and $0 \leq j \leq W$, and $H, W$ correspond to the height and width of the greyscale image respectively.

In addition to additive Gaussian noise, it is also possible to model the noise process in other forms.

In particular, *Masking noise* involves choosing (randomly), a fraction $f$ of the true pixel intensities, $u_T(i, j)$ and setting them equal to 0.

$$u_N(i, j) = \begin{cases} 0, & \text{for fraction} f \\ u_T(i, j), & \text{otherwise} \end{cases}$$

*Salt-and-pepper noise* involves choosing (randomly), a fraction $f$ of the true pixel intensities, and setting each of them to their maximum or minimum value, with equal probability.

### 3.2 The Denoising Task

Let $\mathcal{D}$ denote the denoiser, which is taken to be a mapping from the noisy input image, $\mathcal{I}_N$ to a denoised output image, $\mathcal{I}_F$. Thus,

$$\mathcal{D} : \mathcal{I}_N \mapsto \mathcal{I}_F \tag{3}$$

The denoising task constitutes the minimization of some error metric $\mathcal{E}$, between the output image, $\mathcal{I}_F$ and the pre-noise, input image, $\mathcal{I}$ to determine, or train the denoiser $\mathcal{D}$. That is, we minimize the pixel-wise expression

$$\sum_{i,j} \mathcal{E}(u_F(i, j), u_T(i, j)) \tag{4}$$

or some other image wide error metric.

### 3.3 Evaluation Metrics

B The evaluation of the quality of the denoised image is often done using the *Peak Signal to Noise Ratio (PNSR)*, whose expression is given by

$$PNSR = -10 \log_{10}(\epsilon^2) \tag{5}$$

where $\epsilon^2$ is the mean squared error between the original image, $\mathcal{I}$, and the denoised image, $\mathcal{I}_F$.

Another common method of evaluation is simply visual evaluation, since in the case of image denoising, the visual representation of the image is especially important. The image must appear to be clear, and *visually similar* to the original, to the viewer, even if it does not have the highest PSNR.

## 4 Baseline Techniques

The 2 most commonly used state-of-the-art baseline methods in recent papers are the K-SVD algorithm, and the BM3D algorithm. K-SVD utilizes an overcomplete dictionary, and uses an iterative algorithm to learn this dictionary for the noisy image being considered. The noisy patch is thus approximated as a sparse linear combination of atoms belonging to the dictionary. This is done for the entire image, using overlapping patches. The algorithm gives excellent performance for a wide variety of images.

The BM3D algorithm exploits spatial redundancy of noisy patches in an image, and on grouping similar blocks together, performs collaborative filtering on thresholded 3-D transformed co-efficients. The aggregation of the inverse 3-D transformed values are used to estimate the original image (after aggregating overlapping image patches). A complete outline of the algorithm is given in Figure 1. The BM3D turns out to be a much faster algorithm than the K-SVD method.

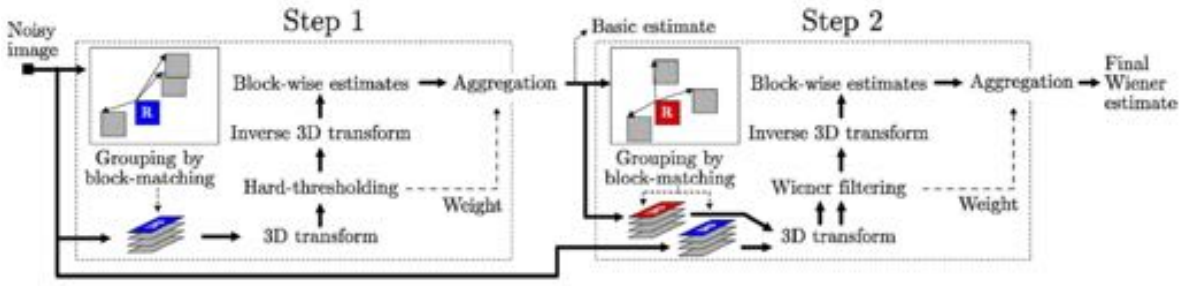



Figure 1: Outline of the BM3D algorithm

Other methods, based on the wavelet transform have also been proposed for the denoising problem including the BLS-GSM algorithm (Bayes Least Square with a Gaussian Scale-Mixture), and methods based on Markov Random Fields. We do not go into further detail for these algorithms, and instead turn our attention to the learning algorithms below.

## 5 Multi-Layered Perceptrons

The work of Burger, et al. looks a the problem of using ordinary neural networks in the task of denoising. The novelty of their work lies in the usage of a pure learning approach, without the explicit incorporation of any priors on the image data. Their work thus has a wide applicability to various kinds of image data, and is not restricted to natural images.

The limitations of their work lie in their utilization of a large image dataset, since they corrupt image patches of the noise-free dataset, using additive white Gaussian noise.

### 5.1 Methodology

The basic idea used by Burger, et al. is to map noisy image patches, to clean de-noised patches. They allow the MLP to learn a function $f : x \mapsto \hat{y}$ from the input image patch $x$ to the output patch

$\hat{y}$. The input image patch $x$ is itself a corrupted version (using additive white Gaussian noise) of the original image patch $y$.

The parameter learning to determine weights for this MLP (and effectively learn $f$) is done using *stochastic gradient descent* with the standard back-propagation algorithm for weight updates. They use the standard squared error formulation,

$$\epsilon = (\hat{y} - y)^2 = (f(x) - y)^2 \tag{6}$$

to minimize the error between the estimated patch, and the original one. They also use simple tricks such as data normalization, and weight initialization to speed up the back-propagation algorithm.

The overall procedure is thus,

- Corruption of a given image sample using an additive white Gaussian noise, to give a noisy image sample.
- Decomposition of the given noisy image into overlapping noisy patches ($x$'s in the MLP equation above), and denoising of each patch separately.
- Composition of a new denoised image by mapping the denoised patches ($\hat{y}$'s) to their original location, and then weighing the overlapping patches using a Gaussian window.

### 5.2 Results

For training, Burger et al. use the Berkeley segmentation dataset, and the LabelMe image data (small and large training sets), while testing on 3 test sets; standard images ("Lena", "Barbara", etc.), the Pascal VOC 2007 dataset, and the McGill dataset. Some of the results obtained by them are reproduced in Figure 2.

In most cases, the PSNR obtained by using the MLP method, is comparable or slightly worse than the BM3D method. Note that the standard deviation of the Gaussian noise is taken to be $\sigma = 25$ in generating the noisy image patches. However, the best model used was after training on 362 million training samples, with four hidden layers of size 2047, and a patch size of 17x17. The training time reported is 1 month, which seems quite large for the task at hand.

### 5.3 Discussion

Burger et al. test their approach in a variety of cases including different values of $\sigma$, and under different formulations of the noise model. Some of the results are shown in Figures 4 and 5. While the MLP tends to do reasonably well on different noise formulations (better than BM3D, which is tailored to Gaussian noise), it is severely limited by the slow training time of the network, and the large amount of data it requires for training. They also find, that increasing the size of the input patches, tends to improve performance, since a larger amount of noise is captured per patch.

In general, the MLP is able to learn various kinds of structures in the input image data, as evidenced by the filters demonstrated in the paper. These include Gabor-like edge detectors, noise patches, and blobs. However, because of the fact that the MLP is a general purpose framework, it does not specifically concentrate attention to any particular type of hidden representation. This is both an advantage, and a disadvantage, since it lends the MLP its flexibility (higher degree of generalization), but severely limits its application.

Overall, the work of Burgar et al. demonstrates the promise of Neural Network style approaches to the task of image denoising.

## 6 Convolutional Neural Networks

The application of Convolutional Neural Networks to the task of image denoising has been carried out by Jain et al. The major motivation for their work is the computational difficulty faced by applying MRF-based methods to image tasks; both because of parameter estimation, and inference in probabilistic models. Applying CNNs to the problem has 2 advantages; the first that since CNNs


Figure 2: Comparison of the MLP method to the BM3D baseline

| image | GSM [18] | KSVD [1] | BM3D [3] | us |
|---|---|---|---|---|
| Barbara | 27.83dB | *29.49dB* | **30.67dB** | 29.21dB |
| Boat | 29.29dB | 29.24dB | *29.86dB* | **29.89dB** |
| C.man | 28.64dB | 28.64dB | **29.40dB** | *29.32dB* |
| Couple | 28.94dB | 28.87dB | *29.68dB* | **29.70dB** |
| F.print | 27.13dB | 27.24dB | **27.72dB** | *27.50dB* |
| Hill | 29.26dB | 29.20dB | *29.81dB* | **29.82dB** |
| House | 31.60dB | 32.08dB | **32.92dB** | *32.50dB* |
| Lena | 31.25dB | 31.30dB | *32.04dB* | **32.12dB** |
| Man | 29.16dB | 29.08dB | *29.58dB* | **29.81dB** |
| Montage | 30.73dB | 30.91dB | **32.24dB** | *31.85dB* |
| Peppers | 29.49dB | 29.69dB | *30.18dB* | **30.25dB** |

Figure 3: PNSR comparison of the best-performing MLP to various baselines

operate under a different non-probabilistic framework, the computational task is easier. The second is that CNNs do not have any specificity to the task of denoising natural images, unless their parameters are specifically tuned to this task.

They restrict themselves to the setting involving an additive white Gaussian noise model, using greyscale images as the input.

### 6.1 Methodology

A CNN contains alternating layers, with linear filtering and nonlinear transformational operations (with an activation function such as the sigmoid) in subsequent layers. The input, in this scenario,

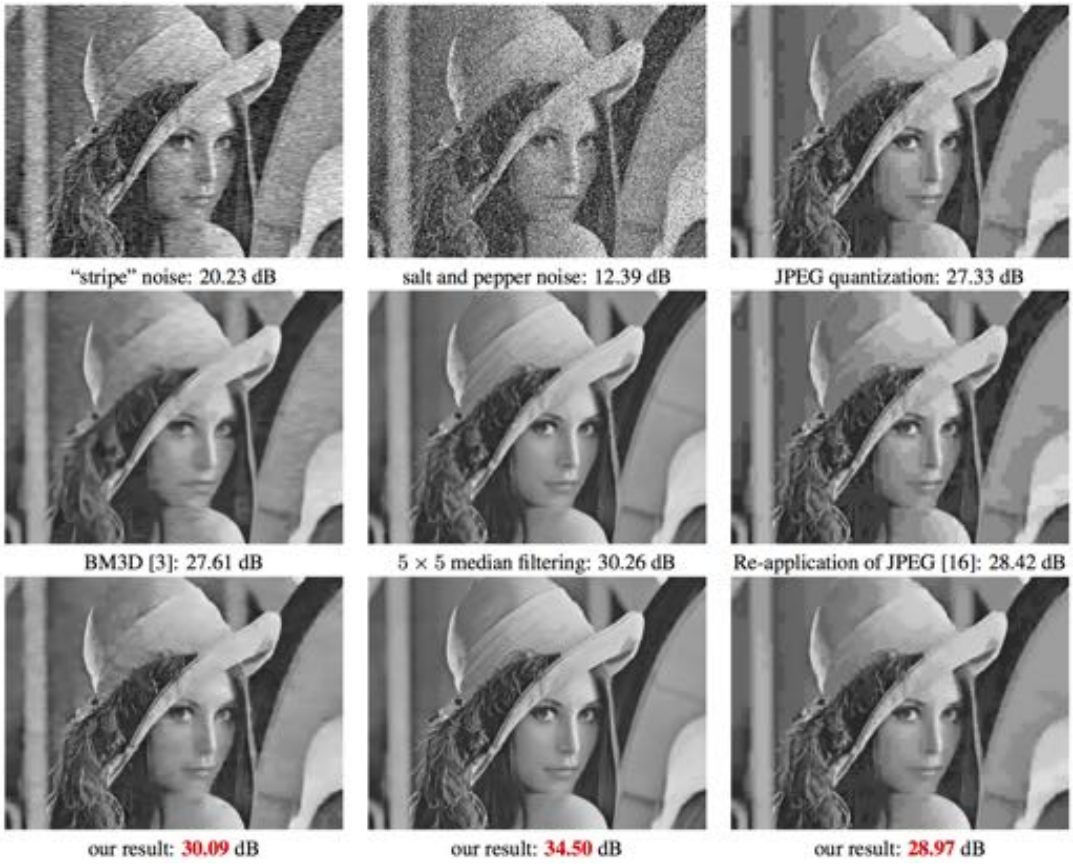



Figure 4: MLP v/s BM3D under different noise-models

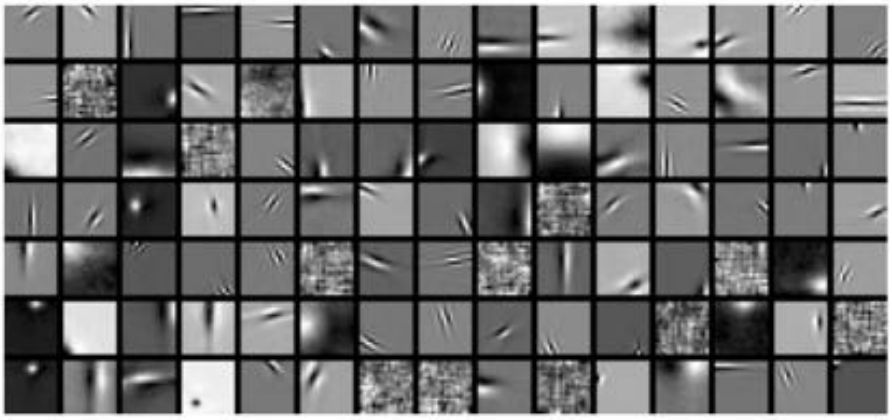


Figure 5: Example of a learned filter in the output layer of the MLP

takes the noisy image, and maps it to the output, denoised image. Intermediate layers, contain feature maps, that have a shared weight, and on which the convolutional operations are performed. The activity of a feature map $a$ in layer $k$ is given by

$$I_{k,a} = f\left(\sum_b w_{k,ab} \otimes I_{k-1,b} - \theta_{k,a}\right) \tag{7}$$

where $I_{k-1,b}$ are feature maps (in the previous layer), that provide input to $I_{k,a}$. $\otimes$ denotes the convolution operator, associated with a $5 \times 5$ filter in the architecture of Figure **??**. In this case, each feature map in the current layer, is connected to 8 random feature maps in the previous layer. Lastly, $f(x) = 1/(1 + e^{-x})$ is the sigmoid function, and $\theta_{k,a}$ is a bias term.

2

3 kinds of CNNs are being learned,

- **CN1:** For CN1, a known noise model is assumed (with a known variance for the underlying Gaussian distribution), and training is done on the same subset of images as used by the Field of Experts (FoE) model (which is taken to be the baseline in this paper).
- **CN2:** For this CNN, the noise model, and the underlying assumptions remain the same. The only difference is that a larger subset of the Berkeley dataset is used in training the model. The learned network (for both CN1 and CN2) is 4 layers deep, as show in Figure **??**, and takes as input a $20 \times 20$ patch.
- **CNBlind:** Lastly, the noise model assumption is done away with, and the variance of the Gaussian generating the noisy images is taken to be unknown. Instead, the amount of noise added to the training data (to train the CNN) is taken by randomly varying $\sigma$ in the range $[0, 100]$. In addition, the CNN is taken to be 5 layers deep instead of 4, as in the previous cases, to compensate for the loss of information about the corruption process. The input patch size is taken to be $24 \times 24$.

The error formulation being minimized is once again the squared error loss, as in the MLP case above. To speed up training, $6 \times 6$ patches of the input image are taken, and the overall gradient descent (stochastic gradient descent) rule, is updated using the combined gradient from 6 separate $6 \times 6$ patches.

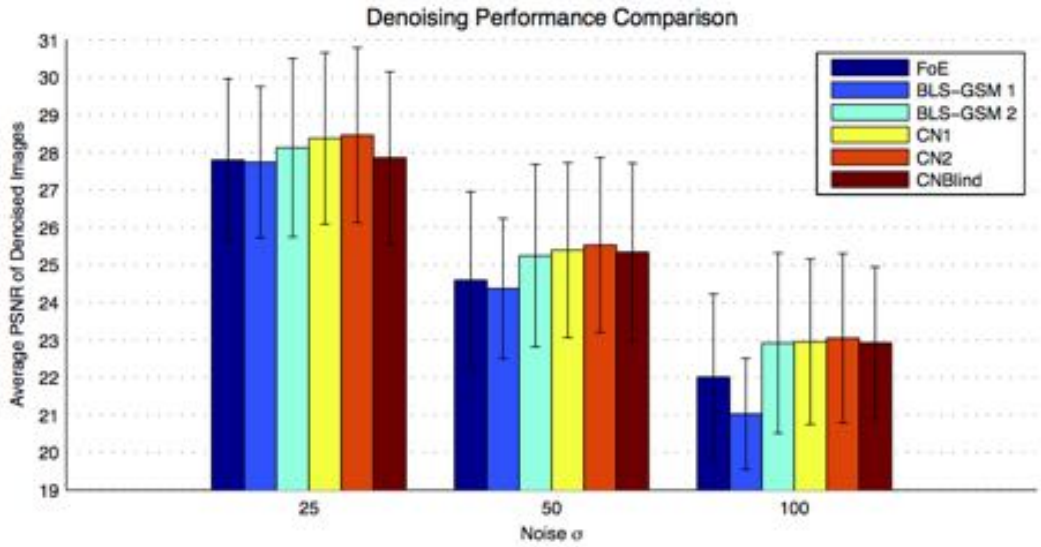



Figure 6: PSNR comparison of the described CNNs v/s baselines

### 6.2 Results and Discussion

Figure 6 shows the PNSR results for the various architectures defined in the previous section. Note how CN2 outperforms the state-of-the-art algorithms, and the CNBlind algorithm also seems to perform quite well. Figure 7 also reveals how CN2 is produces a better denoised image, than either of BLS-GSM, or FoE. The image patch shown suffers less from the problem of artifacts, than the BLS-GSM output, and is also much clearer than the FoE output.

The CNN architecture, while more difficult to interpret than simpler methods of image denoising, shows excellent generalization (much like the MLP of the previous section), to various image types. However, unlike the MLP, the CNN shows much better performance, and while it does suffer from the problem of a long training time (1 week), it does not require the same amount of training data (orders of magnitude less). Wavelet transforms such as the GSM model however, show excellent performance without much parameter tuning, because they are specifically chosen for their expressiveness in representing natural image statistics.

Overall, CNNs seem to perform quite well for the task of image denoising.

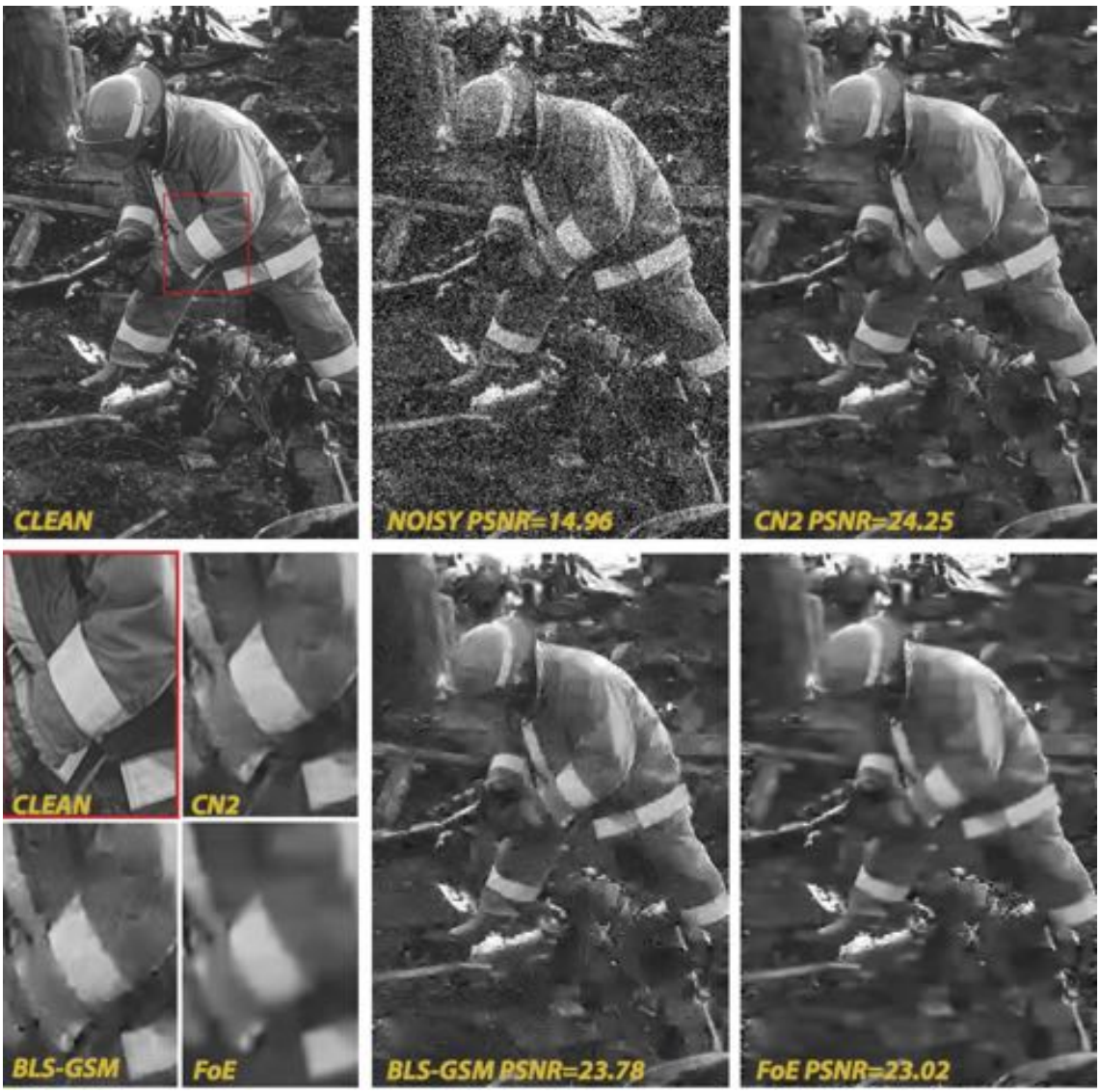



Figure 7: Visual example of the denoising quality of CN2, with comparison to BLS-GSM and FoE

## 7 Denoising Auto-Encoders

In this section, the concept of a Denoising Auto-Encoder (DAE) is first developed, as described by Vincent, et al, and then its performance is examined for the image denoising task, at hand. Further, more complex models, that use the DAE as a building-block, are also examined.

To begin with, the Auto-Encoder model attempts to learn an identity mapping with a hidden layer in between, that corresponds to an interesting representation that we seek to learn. In general, the number of units in the hidden layer may be more, or less than the units in the input layer. A lower number of units in the hidden layer, corresponds to an *under-complete* representation, while a higher number of units, to an *over-complete* representation.

There is an inherent advantage in learning an over-complete representation and enforcing a sparsity criterion, by imposing a penalty on non-sparse representations. The over-completeness allows the representation to display higher or lower levels of sparsity (effectively settling on a dimensionality, that corresponds most 'comfortably' to the input), whereas with a small, fixed number of hidden layer units, the data might be forced to entangle information. In addition, sparse representations are also easier to interpret, understand, and work with.

Taking this further, we can develop the concept of a DAE, which simply learns a mapping from a corrupted, or noisy form of the original input, to a denoised output. The weights corresponding to the DAE are learned by minimization of the error between the output denoised image, and the original image that was passed through the corruption process to yield the DAE input. There are several intuitive advantages to using this approach,

- Intuitively, we would like that the higher level representation that is learned by any Auto-Encoder, should be invariant, or insensitive to noise in the original input. The key advantage of the higher level representation must be that it captures the underlying regularity in, and the important features of, the input image. The addition of noise to the image would not cause the removal of these regularities, in ordinary circumstances.
- By forcing the input to be a noisy version of the actual image, we are compelling the DAE to search for a more meaningful representation. The only effective way in which the DAE will be able to denoise the input image will be if it is able to understand what parts of the noisy image are likely to be part of the true image, and what aspects are likely to correspond to noise.

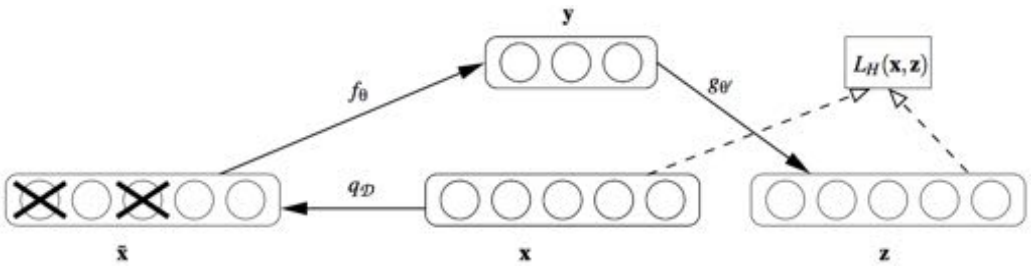



Figure 8: Architecture of the Basic Denoising Auto-Encoder

Figure 8 corresponds to the basic structure of the DAE, that has been laid out above. The original input, $\mathbf{x}$, is passed through a corruption process, with a stochastic mapping $\mathbf{q}_\mathcal{D}$ such that $\tilde{\mathbf{x}} \sim \mathbf{q}_\mathcal{D}(\tilde{\mathbf{x}}|\mathbf{x})$. $\tilde{\mathbf{x}}$ is thus the noisy input to the DAE.

The function $f_\theta$ maps the input to the hidden representation $\mathbf{y}$, using a mapping of the form

$$\mathbf{y} = f_\theta(\tilde{\mathbf{x}}) = s(\mathbf{W}\tilde{\mathbf{x}} + \mathbf{b}) \tag{8}$$

$\mathbf{y}$ is then mapped to the denoised output $\mathbf{z}$, by the function $g_{\theta'}$. The loss $L_H(\mathbf{x}, \mathbf{z})$ to be minimized can be either the cross-entropy loss, where $g_{\theta'}$ would be identical in form to $f_\theta$. It can also be the squared loss, which would remove the sigmoid component of the mapping, leaving only the affine component. From the point of information theory, minimizing the reconstruction error corresponds to maximizing the lower bound on the mutual information between the original input $\mathbf{x}$, and the intermediate representation $\mathbf{y}$.

Multiple units of DAE can be put together in a stacked fashion, to create a Stacked Denoising Auto-Encoder (S-DAE), as shown in Figure 9. To learn an S-DAE, we can proceed in the following manner,

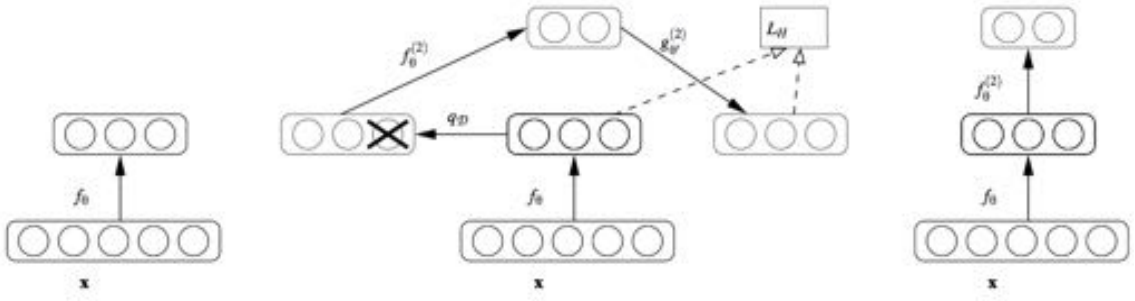


Figure 9: Stacking Denoising Auto-Encoders

- Learn the first mapping, $f_\theta$, by training a single DAE on the original input $\mathbf{x}$ in its noisy form $\tilde{\mathbf{x}}$.
- Using $f_\theta$ directly on the clean input $\mathbf{x}$ (as shown in Figure 9), obtain the higher level representation $\mathbf{y}$. The next level mapping $f_\theta^{(2)}$ will be learned on the representation on the current level, which in this case will be $\mathbf{y}$.
- This procedure can be repeated until the desired number of DAEs has been stacked.

The network obtained in this form can be fine-tuned by using backpropagation with the stochastic gradient descent algorithm. Further discussion on S-DAEs is done in sections below.

### 7.1 Using the Denoising Auto-Encoder

HuiMing Li, utilizes the framework defined above to create a straightforward single/two layer DAE, that can be used to denoise images. The methodology followed is to first train a DAE, using a suitable training set (CIFAR-bw in this case). Unsupervised learning is carried out on $m$ random noisy patches of size $n \times n$ that have been extracted from the noisy versions of the original images. In addition, a squared loss formulation is used, with both a weight-decay term (for $L_2$ regularization on weight magnitudes), and a sparse penalty term, to enforce low average activations on hidden units. Thus, the DAE being learned can be considered to be a Sparse Denoising Auto-Encoder (SDAE, which differs from the stacked S-DAE above).

A standard additive white Gaussian noise with $\sigma = 25$ is used in the experiments carried out by HuiMing Li.

To denoise an image, it is decomposed into overlapping patches; each patch is then denoised separately, and the results are averaged over the entire image (aggregation over the overlapping parts). It is found that the larger the patch size, the better the noise performance turns out to be, in general. However, increasing the patch size also naturally increases the complexity of the DAE, since the size of the sparse representation must now be larger.

The results of this approach (with 2 hidden layers), turn out to be similar to the K-SVD, and BM3D baseline methods (in terms of PSNR). Some qualitative results for image quality, are shown in Figure 10.

KyunHung Cho's work suggests that there is relation between the depth of the DAE, and its ability to compensate for increasing noise levels in the input image. As the amount of injected noise increases in the input image, the presence of a larger number of hidden layers, exhibits increasingly better performance. Due to the greater capacity for learning of the deeper models, they are more robust to sensitivity in the amount of image noise. DAEs are also suitable for the task of *blind* image denoising, where there is no prior information about the amount, or type of noise present in the input image.

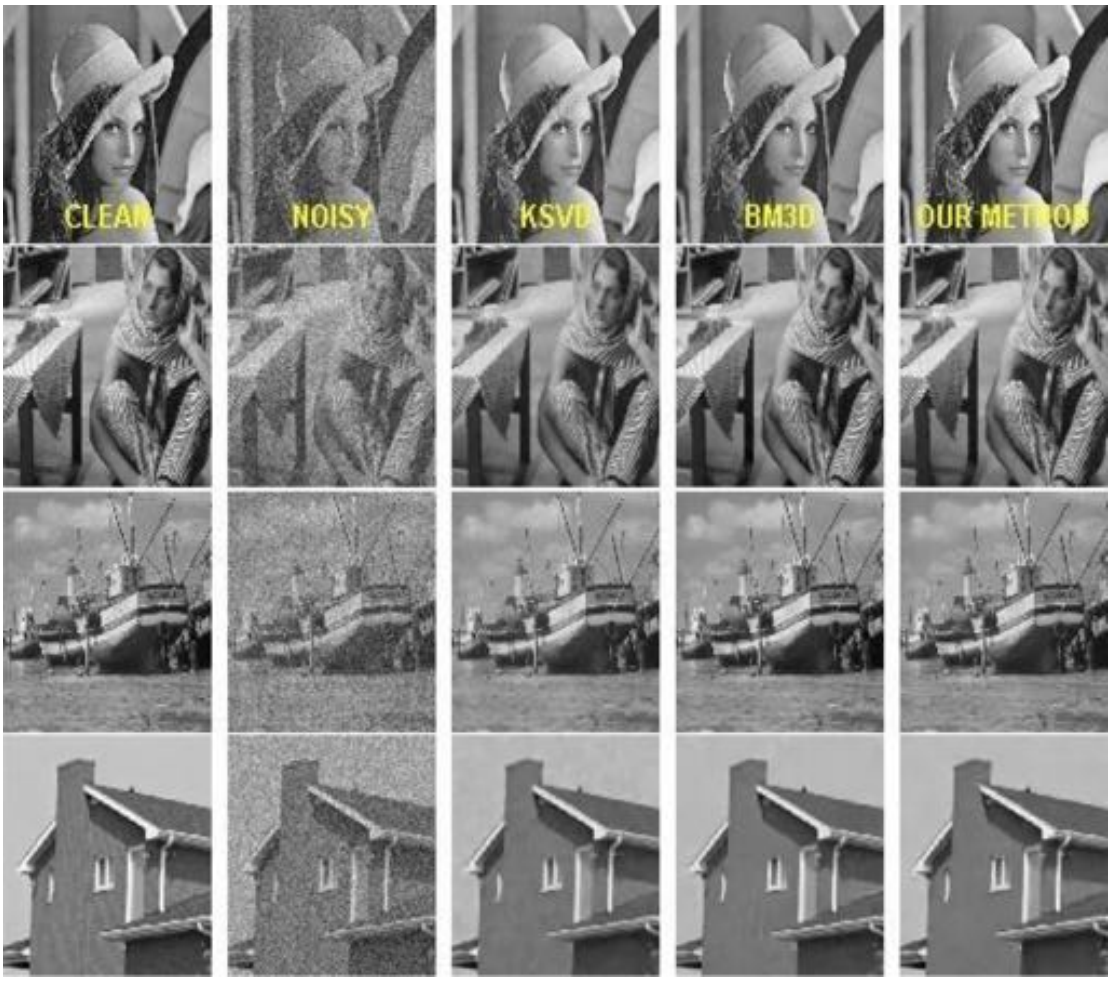



Figure 10: Denoising results for the SDAE against the K-SVD and BM3D baselines

### 7.2 Stacked Sparse Denoising Auto-Encoders

The work of Xie, et al. looks at using DAEs that have been stacked, and upon whom a sparsity constraint has been placed. Like the previous section they look to minimize the following reconstruction loss

$$L_1(\mathbf{x}, \mathbf{y}; \theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \|\mathbf{y_i} - \hat{\mathbf{y}}(\mathbf{x_i})\| + \beta \mathrm{KL}(\hat{\rho} \| \rho) + \frac{\lambda}{2} (\|\mathbf{W}\|_F^2 + \|\mathbf{W}'\|_F^2) \tag{9}$$

where the Kullback-Leibler divergence term is given in its standard form.

The intuition behind this expression is identical to the explanation given in the previous section. The PSNR results for the SSDA, turn out to exactly similar to those from algorithms such as K-SVD, and BLS-GSM. Some qualitative results for this work are shown in Figure 13.

### 7.3 Adaptive Multi-Column Stacked Sparse Denoising Auto-Encoders

The work of Agostinelli, et al. looks at an extension of the SSDA, called the AMC-SSDA. The basic functional unit of their model, is the SSDA developed by Xie, et al. in the previous section. They stack 2 Sparse Denoising Auto-Encoders, within each SSDA block.

To train the AMC-SSDA, first the SSDAs are trained individually (on different noise models), exactly similar to the previous section. Next, the optimal weights (for combination of the column

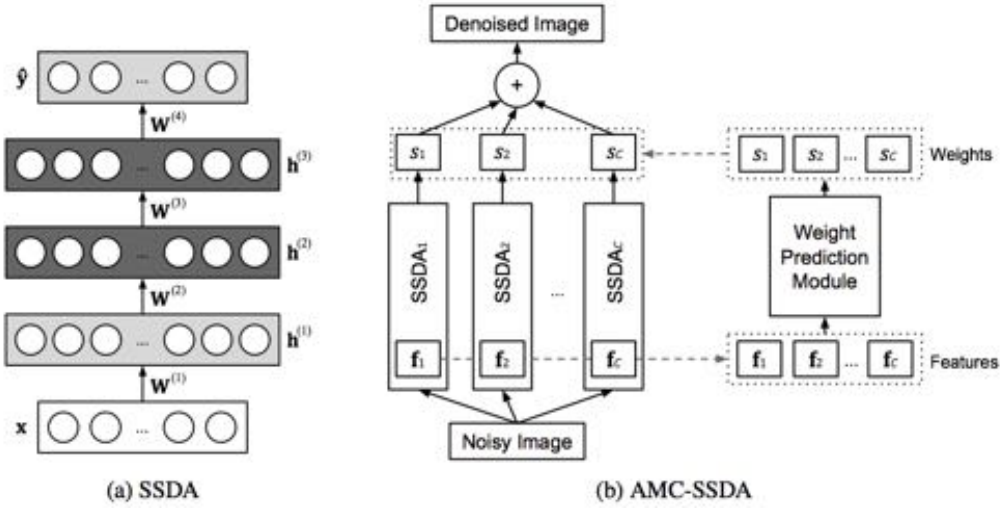



Figure 11: Architecture of the AMC-SSDA

results) for a set of training images is determined, and then the weight prediction module (an RBF network) is trained using these optimal weights as training data.

Essentially, each column $c$, in Figure 11 produces an output $\hat{\mathbf{y}}_c$, for input $\mathbf{x}$, which is the noisy version of the original image $\mathbf{y}$.

To determine the optimal weights for each training example, the following quadratic optimization problem was solved,

$$\text{minimize}_{\{s_c\}} \quad \frac{1}{2}\|\hat{\mathbf{Y}}\mathbf{s} - \mathbf{y}\|^2 \tag{10}$$

$$\text{subject to} \quad 0 \leq s_c \leq 1, \forall c \tag{11}$$

$$1 - \delta \leq \sum_{c=1}^{C} s_c \leq 1 + \delta \tag{12}$$

where the output of each column is collected into $\hat{\mathbf{Y}}$. Using these optimal weights, we train the RBF network.

Finally, on using the AMC-SSDA model, the output generated is given by $\hat{\mathbf{y}} = \hat{\mathbf{Y}}\mathbf{s}*$. Reported results are given below in Figure 12. The model reports statistically significant improvement over the previous baseline SSDA model. With this model, the efficacy of deep learning representations for denoising has been significantly demonstrated.

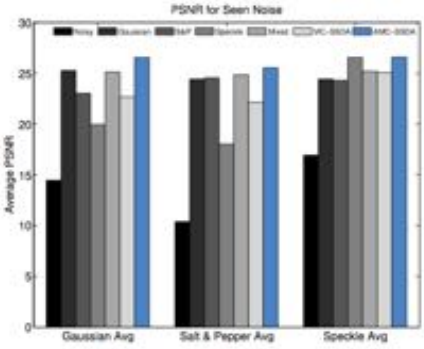
## 8 Conclusion

Research has gone into looking at the application of Deep Learning models to the task of image denoising. Results, from the most recent paper have shown that the field of Deep Learning holds a great deal of promise for the task at hand.

The future challenge is to better be able to understand the interesting representations that are being learned by models such as the Denoising Auto-Encoders, and develop new frameworks (either supervised, or unsupervised), that can better leverage these representations.

| Noise Type | Noisy Image | Gaussian SSDA | S&P SSDA | Speckle SSDA | Mixed SSDA | MC-SSDA | AMC-SSDA |
|---|---|---|---|---|---|---|---|
| G 1 | 22.10 | 26.64 | 26.69 | 26.84 | 27.15 | 27.37 | **29.60** |
| G 2 | 13.92 | 25.83 | 23.07 | 19.76 | 25.52 | 23.34 | **26.85** |
| G 3 | 12.52 | 25.50 | 22.17 | 18.35 | 25.09 | 22.00 | **26.10** |
| G 4 | 9.30 | 23.11 | 20.17 | 14.88 | 22.72 | 17.97 | **23.66** |
| SP 1 | 13.50 | 25.86 | 26.26 | 22.27 | 26.32 | 25.84 | **27.72** |
| SP 2 | 11.76 | 25.40 | 25.77 | 20.07 | 25.77 | 24.54 | **26.77** |
| SP 3 | 8.75 | 23.95 | 23.96 | 15.88 | 24.32 | 20.42 | **24.65** |
| SP 4 | 7.50 | 22.46 | 22.20 | 13.86 | 22.95 | 17.76 | **23.01** |
| S 1 | 19.93 | 26.41 | 26.37 | 28.22 | 26.97 | 27.43 | **28.59** |
| S 2 | 18.22 | 25.92 | 25.80 | **27.75** | 26.44 | 26.71 | 27.68 |
| S 3 | 15.35 | 23.54 | 23.36 | **25.79** | 24.42 | 23.91 | 25.72 |
| S 4 | 14.24 | 21.80 | 21.69 | **24.41** | 22.93 | 22.20 | 24.35 |
| Avg | 13.92 | 24.70 | 23.96 | 21.51 | 25.05 | 23.29 | **26.23** |

(a) PSNRs for previously seen noise, best values in bold.

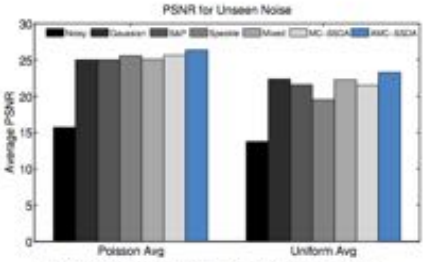



(b) Average PNSRs for specific noise types

Figure 3: Average PSNR values for denoised images of various previously seen noise types (G: Gaussian, S: Speckle, SP: Salt & Pepper).

| Noise Type | Noisy Image | Gaussian SSDA | S&P SSDA | Speckle SSDA | Mixed SSDA | MC-SSDA | AMC-SSDA |
|---|---|---|---|---|---|---|---|
| P 1 | 19.90 | 26.27 | 26.48 | 27.99 | 26.80 | 27.35 | **28.83** |
| P 2 | 16.90 | 25.77 | 25.92 | 26.94 | 26.01 | 26.78 | **27.64** |
| P 3 | 13.89 | 24.61 | 24.54 | 24.65 | 24.43 | 25.11 | **25.50** |
| P 4 | 12.11 | 23.36 | 23.07 | 22.64 | 23.01 | 23.28 | **23.43** |
| U 1 | 17.20 | 23.40 | 23.68 | **25.05** | 23.74 | 24.71 | 24.50 |
| U 2 | 16.04 | 26.21 | 25.86 | 23.21 | 26.28 | 26.13 | **28.06** |
| U 3 | 12.98 | 23.24 | 21.36 | 17.83 | 22.89 | 21.07 | **23.70** |
| U 4 | 8.78 | 16.54 | 15.45 | 12.01 | 16.04 | 14.11 | **16.78** |
| Avg | 14.72 | 23.67 | 23.29 | 22.54 | 23.65 | 23.57 | **24.80** |

(a) PSNR for unseen noise, best values in bold.




(b) Average results for noise types.

Figure 12: Results for the AMC-SSDA

## References


[1] Burger, Harold Christopher, Christian J. Schuler, and Stefan Harmeling."Image denoising: Can plain Neural Networks compete with BM3D?." *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on.* IEEE, 2012.

[2] Jain, Viren, and Sebastian Seung. "Natural image denoising with convolutional networks." *Advances in Neural Information Processing Systems.* 2009.

[3] Vincent, Pascal, et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." *The Journal of Machine Learning Research* 11 (2010): 3371-3408.

[4] Li, HuiMing. "Deep Learning for Image Denoising." (2014). IJSIP 2014.

[5] Cho, Kyunghyun. "Boltzmann machines and denoising autoencoders for image denoising." *arXiv preprint arXiv:1301.3468* (2013).

[6] Xie, Junyuan, Linli Xu, and Enhong Chen. "Image denoising and inpainting with deep neural networks." *Advances in Neural Information Processing Systems.* 2012.

[7] Agostinelli, Forest, Michael R. Anderson, and Honglak Lee. "Adaptive multi-column deep neural networks with application to robust image denoising." *Advances in Neural Information Processing Systems.* 2013.

[8] Elad, Michael, and Michal Aharon. "Image denoising via sparse and redundant representations over learned dictionaries." *Image Processing, IEEE Transactions* on 15.12 (2006): 3736-3745.

[9] Dabov, Kostadin, et al. "Image denoising by sparse 3-D transform-domain collaborative filtering." *Image Processing, IEEE Transactions* on 16.8 (2007): 2080-2095.


Figure 13: Denoising results for the SSDA against the K-SVD and BLS-GSM baselines