Term Paper Presentation
on
Deep Learning Techniques for Image Denoising

Karan Goel

2011EE50555

23rd April 2015

Image Denoising Task

Objective

Remove noise from a corrupted image

- Natural images typically contain additive white Gaussian noise

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

- Masking noise - drop a fraction of pixels

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

- Salt-and-pepper noise - set fraction of pixels to max/min value

2 types of denoising tasks

- Non-Blind Denoising: Complete knowledge of noise model in test data
- Blind Denoising: No knowledge of noise model

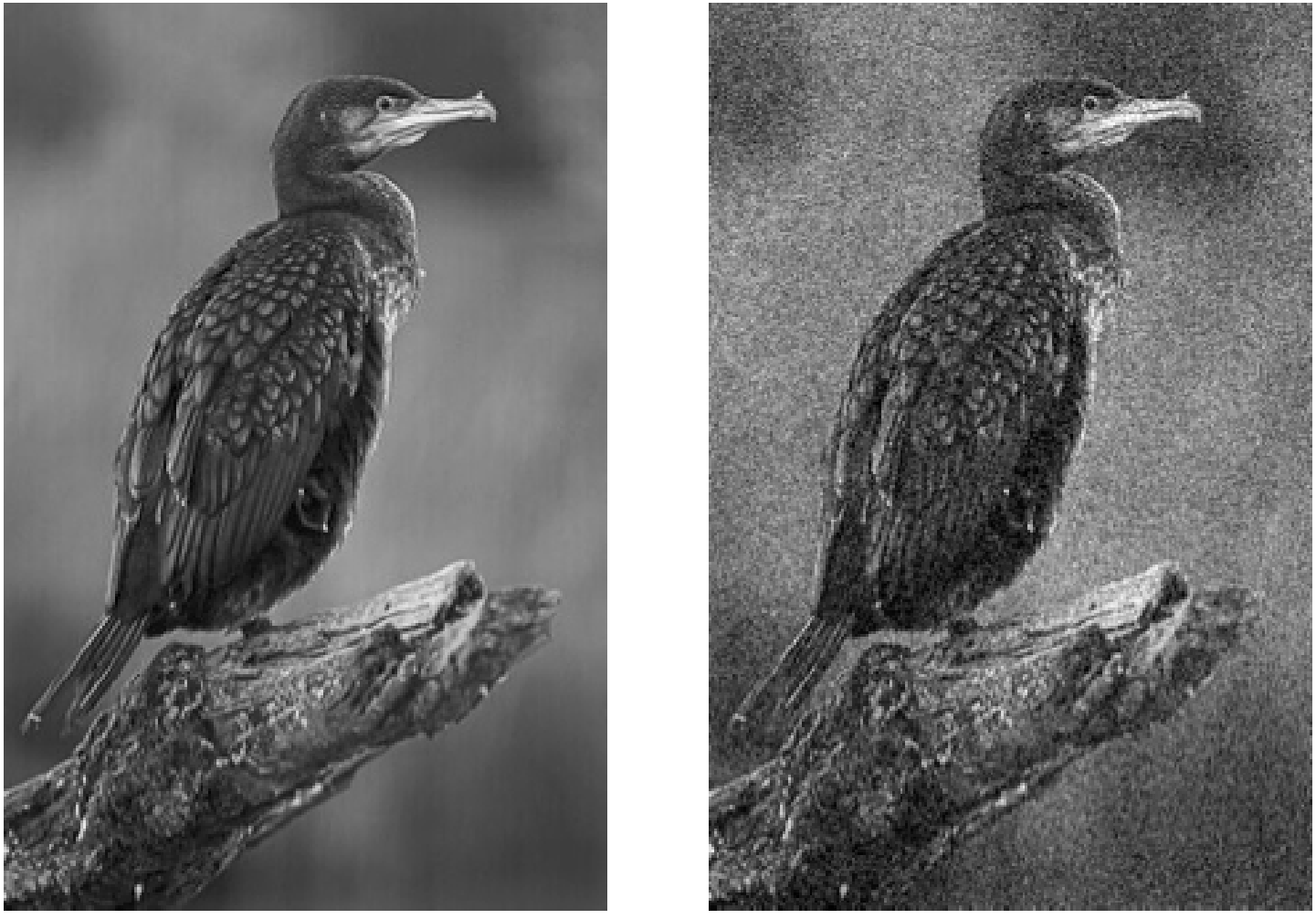


Figure 1: Addition of Gaussian noise ($\sigma = 25$) to an image

The Denoising Task

Let $\mathcal{D}$ be the denoiser, mapping a noisy input image, $\mathcal{I}_N$ to a denoised output image, $\mathcal{I}_F$

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

Denoising involves minimization of some error metric $\mathcal{E}$, between the output image, $\mathcal{I}_F$ and the pre-noise input image, $\mathcal{I}$ for training. Minimize

$$\mathcal{E}(\mathcal{I}_F, \mathcal{I})$$

$\mathcal{E}$ is typically

- Squared Loss
- Cross-Entropy Loss

Evaluation of denoising quality

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

Techniques

- Multi-Layered Perceptrons [Burger, et al]
- Convolutional Neural Networks [Jain et al]
- Denoising Auto-Encoders [Vincent, Pascal et al; Li, Huiming]
- Sparse Denoising Auto-Encoders [Cho, Kyunghyun]
- Stacked Sparse Denoising Auto-Encoders [Xie et al]
- Adaptive Multi-Column Stacked Sparse Denoising Auto-Encoders [Agostinelli et al]

Baselines

- K-SVD: Over-complete dictionary based denoising [Elad et al]
- BM3D: Collaborative filtering with spatial redundancy of noisy patches [Dabov et al]
- BLS-GSM, Markov Random Fields (FoE), etc.

General Procedure

- Take clean training set of input images x
- Corrupt them using noise model N, to give new noisy training set $\tilde{x}$
- Train model on $\tilde{x}$, minimizing error of output y w.r.t. x
- Training typically done on noisy patches, rather than whole image; patch size varies by method
- Denoise by decomposing original image into patches, denoising individual patches, then aggregate result
- Improve training by varying noise model across training set

Multi-Layered Perceptron

- Standard MLP with 4 hidden layers (2047 hidden units) (Backpropogation, stochastic gradient descent)
- 17×17 patch size
- 362 million training examples(!)
- Training time - 1 month
- Performance comparable to BM3D baseline for Gaussian noise; outperforms on quantization, stripe, and salt-and-pepper noise
- Starting Point: Interesting filters observed in the MLP
- Works better image wide, than on local regular patterns

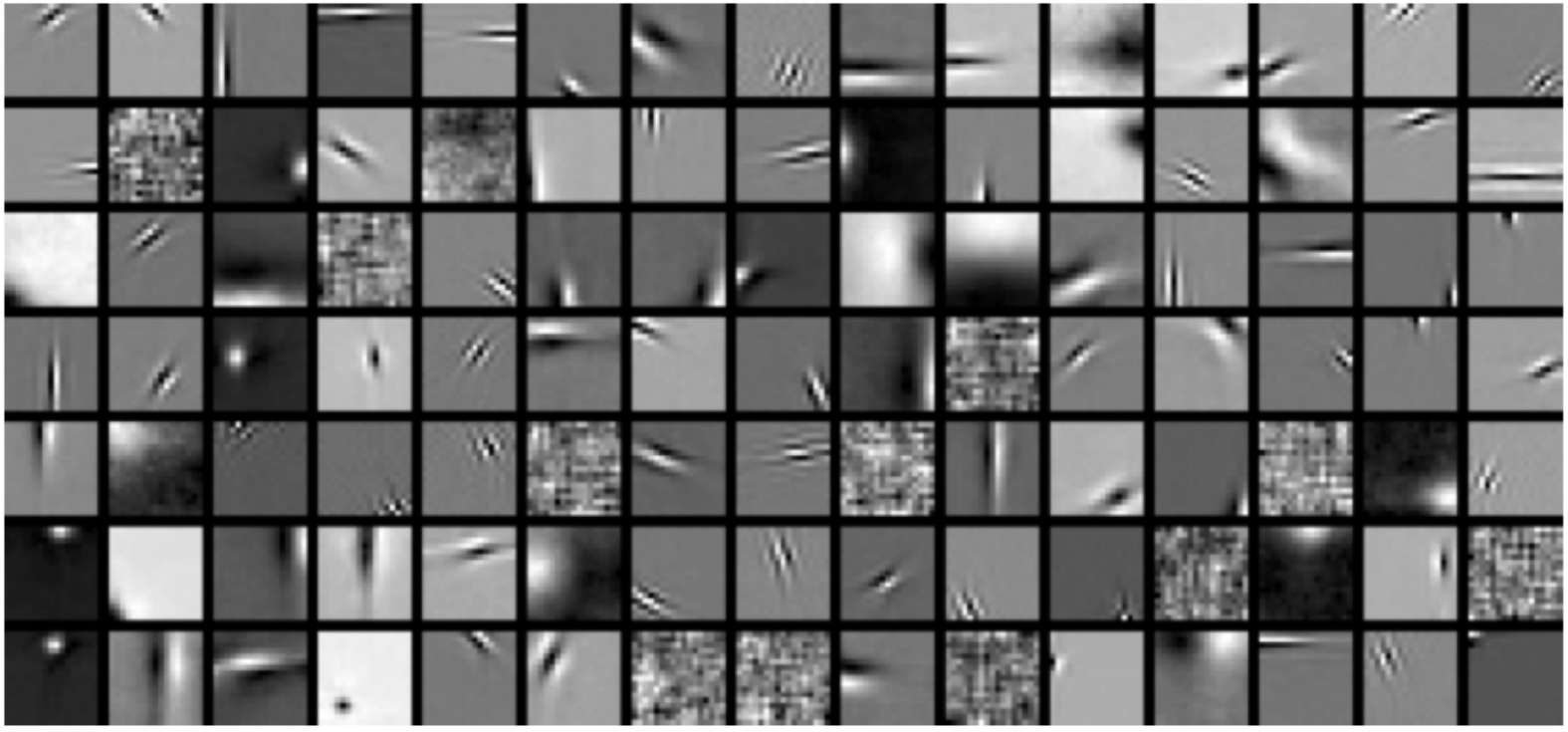


Figure 2: Example of a learned filter in the output layer of the MLP

Figure 3: Comparison of the MLP method to the BM3D baseline

Convolutional Neural Network

- 4 hidden layers, 24 feature maps/layer
- Feature map in current layer connected to 8 random maps in previous layer; patch size 20×20
- Filter size of 5×5 in each layer
- Blind version: 5 hidden layers, 24×24 patch size

The activity of a feature map a in layer k is given by

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

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 the filter. $f(x) = 1/(1+e^{-x})$ is the sigmoid function, and $\theta_{k,a}$ is a bias term.

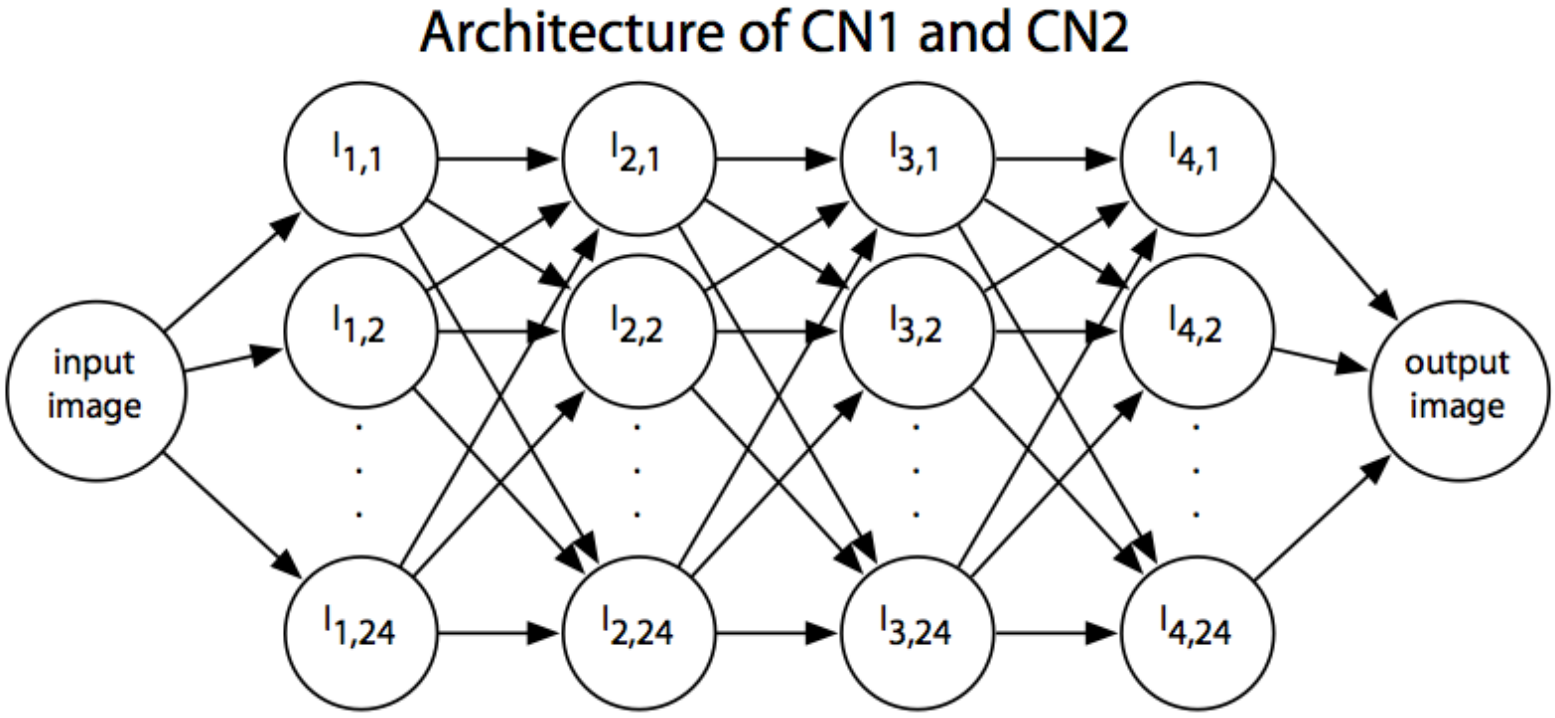


Figure 4: Architecture of the CNNs used (4 hidden layers, 24 feature maps/layer)

- CNN gives excellent performance compared to baselines. CNBlind more robust to higher noise.

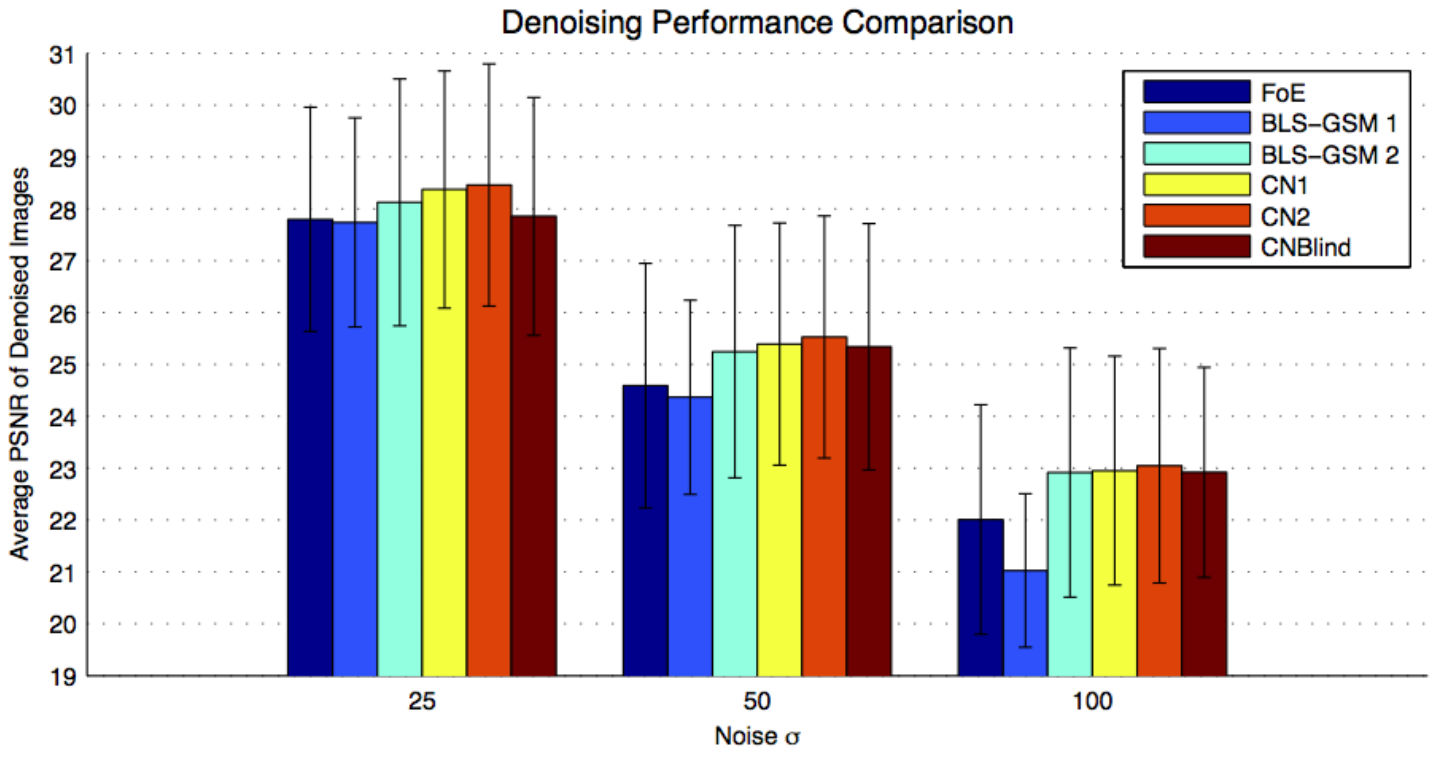


Figure 5: PSNR comparison of the described CNNs v/s baselines

Denosing Auto-Encoders

Motivation: Learn higher level representation that is insensitive to amount of noise in image

- DAE learns Auto-Encoder like mapping from noisy input image, to clean image
- By forcing input to be noisy, compelling DAE to learn more robust representation

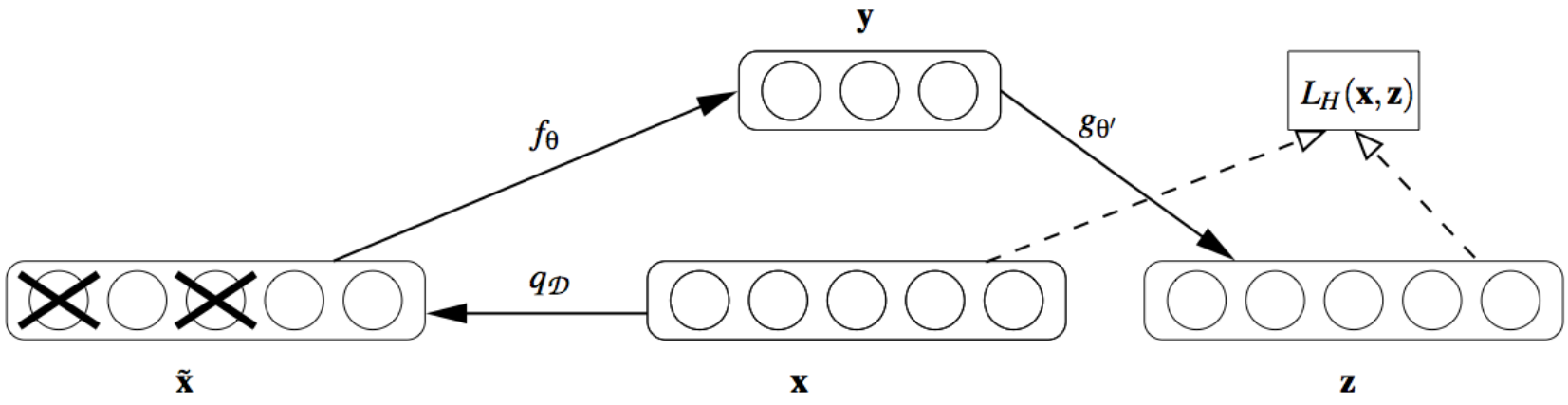


Figure 6: Architecture of the Basic Denoising Auto-Encoder

f_θ maps input to the hidden representation $\mathbf{y}$, using a mapping

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

$\mathbf{y}$ is then mapped to denoised output $\mathbf{z}$, by $g_{\theta'}$.

- If loss $L_H(\mathbf{x}, \mathbf{z})$ to be minimized is cross-entropy loss, then $g_{\theta'}$ identical to f_θ. If squared loss, only the affine component remains.
- Minimizing reconstruction error corresponds to maximizing lower bound on mutual information between original input $\mathbf{x}$, and intermediate representation $\mathbf{y}$
- Regularization term to keep weights small (prevent overfitting)

Sparse DAE Set size of hidden layer $>$ size of input layer (overcomplete representation), and enforce sparsity constraint (KL Divergence term)

Stacking Denoising Auto-Encoders

- **Advantage:** Deeper representation, more robust to larger amounts of Gaussian noise (larger σ)
- **Stacking:** Learn 1st layer; feed hidden representation of 1st layer as input to 2nd layer
- After iteratively stacking all layers, perform global optimization of weights

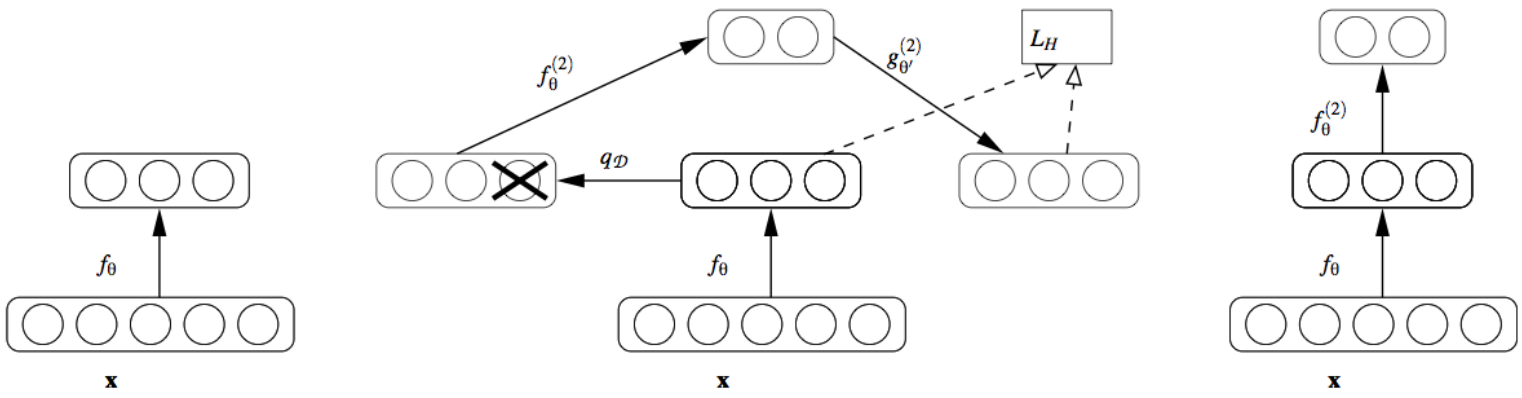


Figure 7: Stacking Denoising Auto-Encoders

Figure 8: Denoising results for the Stacked DAE (SSDA)

Method	Standard deviation σ		
	25/PSNR=20.17	50/PSNR=14.16	100/PSNR=8.13
SSDA	30.52 ± 1.02	27.37 ± 1.10	24.18 ± 1.39
BLS-GSM	30.49 ± 1.17	27.28 ± 1.44	24.37 ± 1.36
KSVD	30.96 ± 0.77	27.34 ± 1.11	23.50 ± 1.15

Figure 9: Denoising results for the Stacked DAE (SSDA)

Adaptive Multi-Column Stacked Sparse DAEs

- State of the art: uses Stacked Sparse DAE as basic unit
- Multiple columns of SSDAs; each SSDA trained on different noise model, for same training data
- Aggregate column results using learned weights

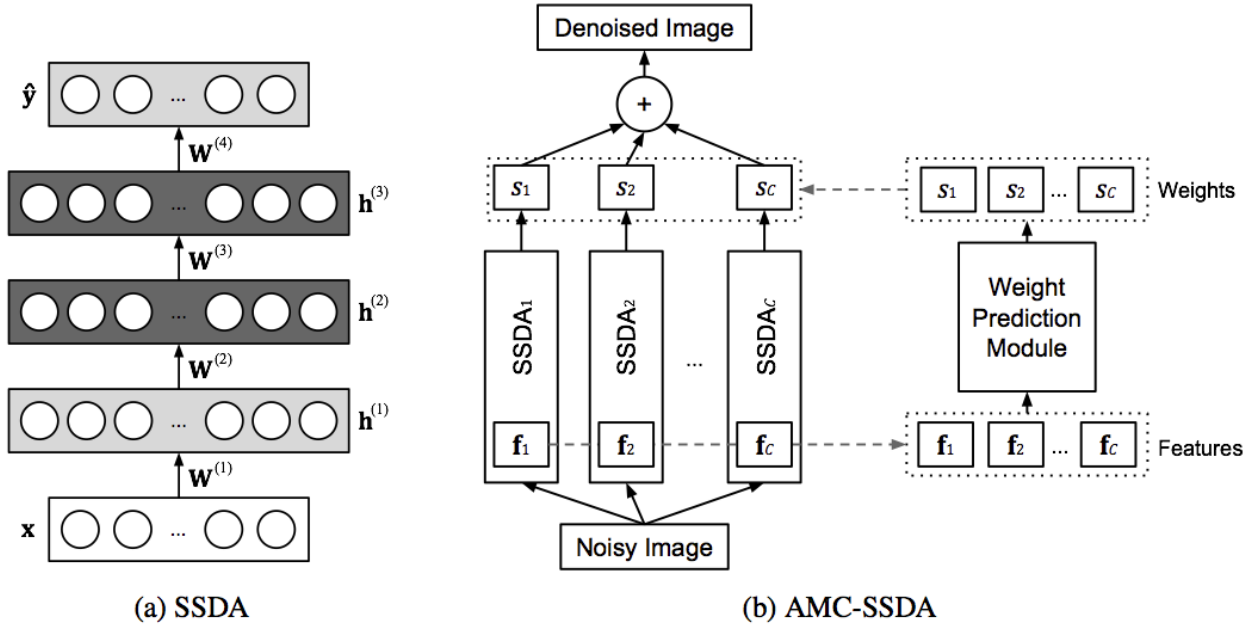


Figure 10: Architecture of the AMC-SSDA

Training the AMC-SSDA

- Individual training of SSDAs
- Optimal weights for training images determined
- RBF network to predict weights trained

Each column c produces output $\hat{\mathbf{y}}_c$ for input $\mathbf{x}$ (noisy version of original image $\mathbf{y}$)
To determine optimal weights solve a quadratic optimization problem

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

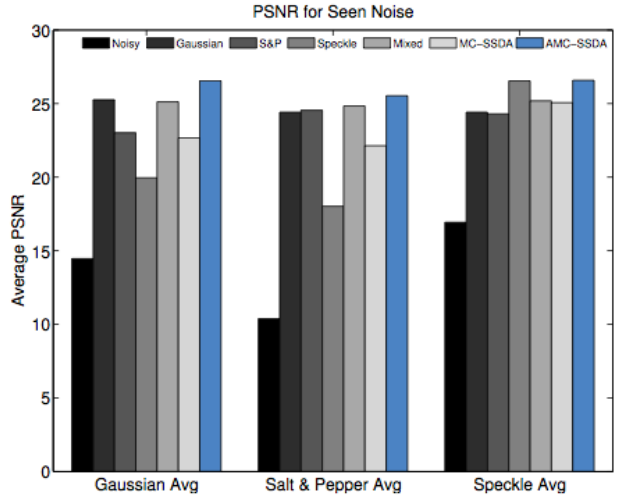
$$\text{subject to} \quad 0 \leq s_c \leq 1, \forall c \tag{2}$$

$$1 - \delta \leq \sum_{c=1}^{C} s_c \leq 1 + \delta \tag{3}$$

where output of each column is collected into $\hat{\mathbf{Y}}$. Train RBF network using these weights. **Output:** $\hat{\mathbf{y}} = \hat{\mathbf{Y}}\mathbf{s}^*$

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.

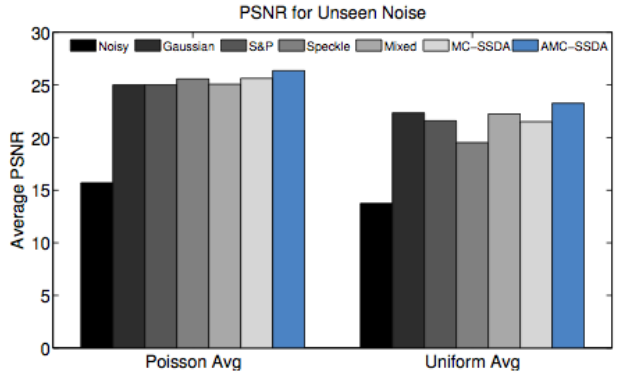


(b) Average PNSRs for specific noise types

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 11: Results for the AMC-SSDA

Conclusion

- Improvement over baseline methods
- Can be learned for wide range of noise models, and image types
- High generalization ability
- Learn interesting representation to remove noise, so applicable to a variety of tasks across Multimedia Systems, Computer Vision, etc.
- Slow to train, and require extensive parameter tuning

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).

[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.