Image Denoising via CNNs: An Adversarial Approach

Nithish Divakar R. Venkatesh Babu
Video Analytics Lab,
Dept. Computational and Data Sciences
Indian Institute of Science, Bangalore, India

Abstract

Is it possible to recover an image from its noisy version using convolutional neural networks? This is an interesting problem as convolutional layers are generally used as feature detectors for tasks like classification, segmentation and object detection. We present a new CNN architecture for blind image denoising which synergically combines three architecture components, a multi-scale feature extraction layer which helps in reducing the effect of noise on feature maps, an ℓ_p regularizer which helps in selecting only the appropriate feature maps for the task of reconstruction, and finally a three step training approach which leverages adversarial training to give the final performance boost to the model. The proposed model shows competitive denoising performance when compared to the state-of-the-art approaches.

1. Introduction

Image denoising is a fundamental image processing problem whose objective is to remove the noise while preserving the original image structure. Traditional denoising algorithms are given some information about the noise, but the problem of blind image denoising involves computing the denoised image from the noisy one without any knowledge of the noise.

Convolutional Neural Networks(CNNs) have generally been used for classification. They have a set of convolutional layers(convolution followed by a non-linear function) and eventually a few fully connected layers which help in predicting the class.

But these networks have also found multiple other uses as the output of these convolutional layers provide a rich set of features from a seemingly nominal image. But what if these features are not exactly from an actual image, but something very close? Can we reconstruct the clean image from features extracted from a noisy image?

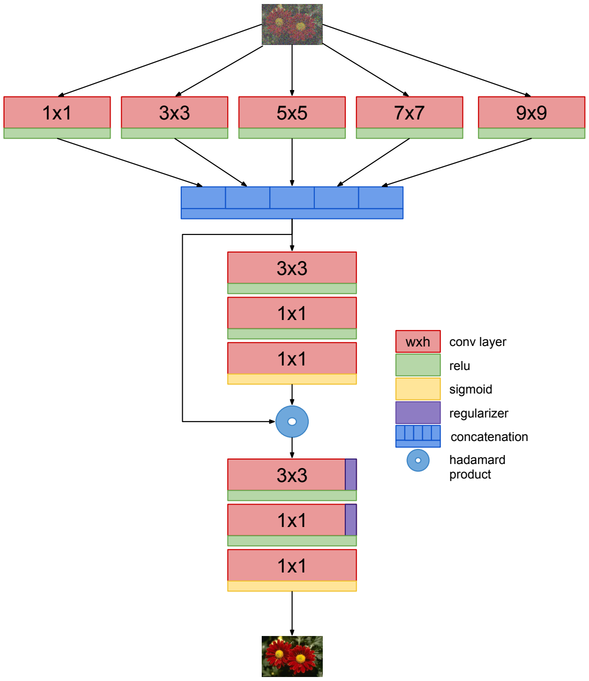

Figure 1. The proposed image denoising model

This paper addresses how CNNs can be used for blind image denoising. The problem does not fit into traditional frameworks as described above since input to the network is not clean images. They are noisy images and require the network to gather enough features from this image so that a noise-free version can be computed from them.

The proposed architecture is shown in Fig. 1. It includes three main components (i) a set of filters that simultaneously extracts features at multiple scales from the image. We call these filters collectively as *multi-scale feature extraction layer* (ii) a combination of filters which allow dampening the features contaminated by noise and (iii) reconstruction layers with filters that do not have any spatial resolution. The architecture is explained in detail in Sec. 2.1.

The following are the major contributions of this paper.

- We propose a multi-scale adaptive CNN architecture which gives a competitive performance to the state-of-the-art image denoising approaches.

- A training regime which exploits clean images as well as noisy images to get good feature maps for reconstruction.

- An adversarial training procedure, which helps to improve the denoiser performance further than the ℓ_2 loss would allow.

2. Proposed approach

The proposed denoising approach contains two main components: (i) an image denoising model and (ii) a three phase training procedure. In this section, we present a detailed overview of both.

2.1. Architecture of the denoiser

Convolutional layers are traditionally used as feature detectors for the classification task. But stacking multiple convolutional layers on top of each other gives the network an inherent feature of abstracting details in deeper layers [7]. This property, although quite useful for classification and other related tasks, is unsuitable for image denoising as the finer details of the image need to be preserved for a good reconstruction.

A naive solution might be to simply use deconvolutions [25]. But this, in turn, imposes more burden on the network to learn to reconstruct details from an abstract representation of the image. Moreover, such a network requires a large number of layers and hence is harder to train.

To circumvent this, we use two techniques.

1. Extract as many features as possible from the image in the first layer itself.

2. Keep all filters of the deeper layers to be 1×1 in size to avoid abstraction and blurring of fine image structures.

To extract all of the necessary features from the image, simply having large number filters of the same size is not enough. Inspired from inception layers of GoogLeNet [23], we employ multiple sets of convolutional filters, each set progressively having larger filter sizes, directly applied on the image. The resulting activations from all these layers are simply stacked together. We call the combination of these filters *multi-scale feature extraction* layer.

The main difference of this layer from inception layer is the absence of initial 1×1 convolutions. Inception layers are usually fed activation of previous layers and hence receive multiple feature maps. In our case, these convolutional layers operate directly over the input image and hence do not require the initial 1×1 convolutions. We can say multi-scale feature extraction layer is more similar to naive inception layer [23].

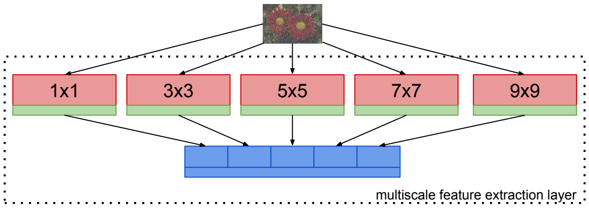

Figure 2. Multi-scale feature extraction layer

Another difference is the number of output channels. Unlike inception layers which have the same number of output channels for each parallel paths, the conv-layers of multi-scale feature extraction layer has a progressive number of output channels since larger filter sizes can extract more information. For our experiments, we have fixed the output channels to 32,40,48,56 and 64.

We avoid learning abstract features in the later layers of our model by limiting expressivity. To achieve this we limit the filter sizes of convolution layers to 1×1.

This results in our model having less number of parameters and also avoids blurring the fine image structures. Hence we are able to use a larger training dataset as opposed to many of the earlier works like [24, 19].

2.2. Three phase training

Simply training the model by feeding noisy images and constraining the output to be close to the clean image can cause the network to quickly converge to averaging out noise. To circumvent this, we make use of the clean images by first teaching the model to simply reconstruct from clean images and then to reconstruct from the noisy image. The training process involves the following:

1. **Clean-to-clean reconstruction** Feed clean images to the model and train it to reconstruct the same image back.

2. **Noisy-to-clean reconstruction** Feed noisy images to the model and train it to reconstruct the corresponding clean image back.

3. **Adversarial training** Train the denoiser model using an adversarial strategy to increase the denoising performance.

Clean-to-clean reconstruction

In the first phase of training, we leverage the availability of clean images to learn useful filters for image reconstruction.

The model is trained to reconstruct the clean image from itself. The intent of this phase is to allow the model to learn good features to reconstruct images. But to prevent the model from simply collapsing to an identity function, we apply a heavy dropout (p = 0.7) immediately after the multi-scale feature extraction layer.

The middle three layers of the architecture in Fig. 1 are provided to dampen the activations of the first layer. The intuition is explained in the next training phase. Since the intent of this phase is to learn features for reconstruction, the skip connection over the middle three layers is short-circuited, resulting in these layers not being part of training. Essentially, we train a model of effective depth of 4 in this phase.

Noisy-to-clean reconstruction The next stage is training the network to reconstruct clean images from noisy images. The dropout added in the previous training phase is removed and the parameters of the *multi-scale feature extraction* layer are frozen. But now, since the images are noisy, the quality of extracted feature maps is adversely affected for those learnt filters which are most sensitive to noise. Feature maps of those filters, which are invariant to noise remains the same. To aid in quick adjustment to these good and bad feature maps (in the context of denoising), we provide a few extra layers that allow to selectively reduce the effect of bad feature maps of the *multi-scale feature extraction* layer.

These layers eventually output a value between 0 and 1 for each pixel position, when fed the activations of the *multi-scale feature extraction* layer. These values are then point-wise multiplied (Hadamard multiplication between tensors) back to the feature maps. The features of the *multi-scale feature extraction* layer as result gets rescaled according to the value. A value close to 0 completely diminishes the feature map while a value of 1 simply allows it to pass unmodified. All the layers of this stage have 240 output channels.

We also impose an l_p regularizer on the 5^{th} and 6^{th} layers (See Fig. 1). These layers have filters of 1×1 and hence imposing a sparsity preserving regularizer will lead to the model selecting only a few connections between the layers. This is an automated way of selecting only a few good activation maps to reconstruct the image. The same idea was implemented in [11] by only allowing a randomly chosen 8 connections to the previous layer. We have found the value of $p = 0.1$ to be satisfactory. Too low a value results in exploding loss function and too high a value results in model simply collapsing to pure averaging. Layers of this stage have 128 output channels except for the last layer which has only 1.

Table 1 shows the denoising performances of the model at the end of this training phase. As can be inferred, the denoising performance is adequate, but far from the state-of-the-art results. Fig. 3 shows some examples of denoised images at this stage of training for various noise levels.

We can see that the model just resorts to averaging all the pixel values in presence of heavy texture and high noise level. This effect can be attributed to the averaging effect of ℓ_2 loss. For a detailed discussion, refer to [15]. To circumvent this effect, we need a better loss function that preserves natural image features.

Table 1. Denoising results after the end of *noisy-to-clean* training phase on test set.

Sigma	10	15	20	25
PSNR	32.37	30.68	29.37	27.93

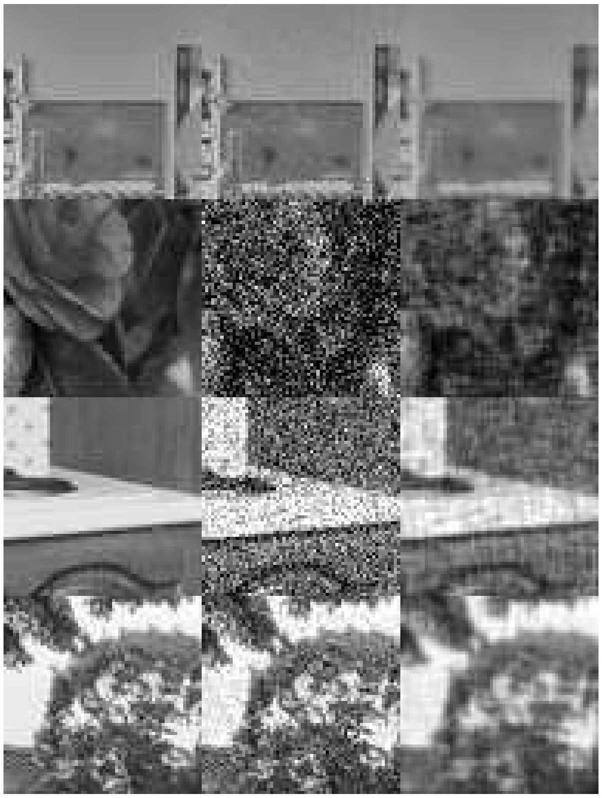

Figure 3. Denoising result after phase 2. The columns respectively show clean images, noisy images and reconstructed images.

Adversarial training Adversarial training of neural networks was introduced by Goodfellow et al. in [8]. We briefly describe it here.

Adversarial training is a method to train a generative network G to generate samples from some real data $x \sim p_{data}$. Generators are fed input noise variables z having distribution p_Z and they are trained to learn the mapping to the data space. The distribution of the generator model is given by

$$p_g \sim G(z; \theta_g) \tag{1}$$

Here, θ_g are the parameters of the generator network. While training the generator, we essentially want to maximize the probability of samples it produces to match the data. Hence we want to maximise $p_{data}(G(z; \theta_g))$.

A discriminator network D on the other hand simply take a data sample x as input and outputs the probability

$D(x, \theta_d)$ of the sample coming from the distribution p_{data} rather than it being generated by the generator. θ_d is the parameter of the discriminator.

Now, the generator wants to generate samples from data distribution. So it must train its parameters so that the generated samples can fool the discriminator. i.e

$$\min_{\theta_g} \mathbb{E}_{z \sim p_Z} \left[\log\left(1 - D(G(z))\right)\right] \tag{2}$$

The discriminator, on the other hand, must learn to tell generated and real samples apart. So it must maximize the probability value assigned to actual data samples and minimize the probability value assigned to generated samples.

$$\max_{\theta_d} \mathbb{E}_{x \sim p_{data}} \left[\log D(x)\right] + \mathbb{E}_{z \sim p_Z} \left[\log\left(1 - D(G(z))\right)\right] \tag{3}$$

Both the generator and the discriminator networks are trained alternatively so that they try to fool each other. The whole process converges when generator eventually learns to generate samples from p_{data}

We use adversarial training in a slightly modified way. Instead of having a generator which maps from input noise to samples to a data distribution, we have a 'generator' that takes a noisy image and 'generates' the corresponding clean image. This network is essentially a denoiser.

Now the discriminator network has to discriminate between clean images and denoised images. The adversarial network is trained such as to find optimum parameters satisfying

$$\theta_g^*, \theta_d^* = \min_{\theta_g} \max_{\theta_d} l_{adv} \tag{4}$$

Where the loss function is given by

$$l_{adv} = \log D(I_c) + \log(1 - D(G(I_n))) \tag{5}$$

Here, I_c is the clean image and I_n is the noisy image

Eq. (4) corresponds to using a binary cross entropy loss on the output of the discriminator that is trained to tell whether the input belongs to one of the two class; true samples or generated samples.

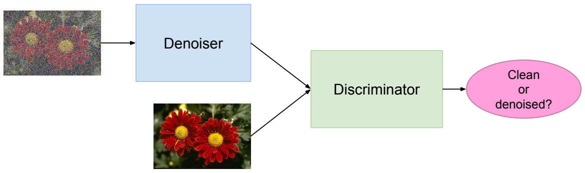

Figure 4. Adversarial training model

But this model allows the generator/denoiser to transform noisy image to any image which the discriminator will classify as a true sample. But for correct denoising, we need the output of denoiser to be very close to the clean image. So we restrict the output of the denoiser to be close to clean image by imposing an extra loss term,

$$l_{deno} = \frac{\|I_d - I_c\|_2^2}{|I_c|} \tag{6}$$

$I_d = G(I_n)$ is the denoised image and $|I_c|$ is the size of the image. This is essentially mean squared loss which penalises any deviation from the original data (here I_c).

Several modifications of adversarial training has been proposed in the literature [18, 20], but the idea to use adversarial training for other tasks other than image generation is not new [4, 15, 13]. But to the best of our knowledge, ours is the first work that uses adversarial training for blind image denoising.

We have used VGG19 [22] model as the discriminator in our experiments. The fully connected layers were replaced by three new layers of size 2048, 1024 and 2 initialized with random weights. Then, these layers are fine-tuned to distinguish between the denoiser output and the clean image.

In VGG19 model, the feature detectors (convolutional layers) are kept unmodified throughout the training and only the fully connected layers are allowed to be trained/modified. The discriminator is pre-trained on the denoiser output and the clean image for 10 epochs which gave a cold start accuracy of about 95%.

After the noisy-to-clean training phase, the denoiser model can already denoise images to some extent. Since adversarial training is very sensitive to the balance of ability of generator and discriminator, the loss function is modified to accommodate this. Essentially, the loss function is the weighted sum of l_{adv} and l_{deno} as follows.

$$loss = l_{deno} + \left(\frac{1 + st}{T}\right) l_{adv} \tag{7}$$

Where $s = 0.99$ is a damping factor , t is the iteration number and T is total number of iterations. This ensures that the adversarial loss is weighted less in the beginning of this phase, but as the training progresses, its contribution to loss increases. This weighing scheme allows the discriminator to slowly learn the difference between denoised image and clean image in the initial iterations. Without this weighing scheme, we have observed that the denoiser model quickly starts to generate images to confuse the discriminator rather than trying to produce noise free images. Essentially, it allows the denoiser to strictly stick to denoising rather than trying prematurely to fool the discriminator.

We have observed that keeping accuracy of discriminator above 95% helps the model learn faster and hence for ensuring this, in each iteration, the discriminator is shown the data twice. We have used Adam optimizer [12] for both networks and set the learning rate of the adversarial network to be 10^{-5} and the discriminator network to be 10^{-6}. The

Algorithm 1 Steps for training the adversarial network. X is a set of clean images in the dataset

```
procedure ADVERSARIAL TRAINING($X$)
    while $t < T$ do
        $x = minibatch(X)$
        $\hat{x} = addnoise(x)$
        $y = G(\hat{x})$
        Train discriminator so that all of $x$ is classified
        as true samples and all of $y$ is classified as false
        samples.
        Train generator/denoiser so that $D(G(\hat{x}))$ al-
        ways evaluates to true.
        Update loss function according to Eq. (7)
```

procedure for adversarial training is enumerated in Algorithm 1.

Connection of adversarial training to patch prior model

Adversarial training is motivated by the fact that the final loss function that our adversarial model minimizes is very similar to the loss function derived from patch prior models [19, 21, 27].

The patch prior model for denoising is given by

$$p(M(I_n)|I_n) = \frac{p(I_n|M(I_n))p(M(I_n))}{Z} \tag{8}$$

where M is the denoiser model and $M(I_n)$ is the output of the model for a noisy image I_n. Z is a normalizing factor.

Assuming Gaussian noise and taking log likelihood, the loss function is given by

$$er[M(I_n), I_n] = \|M(I_n) - I_n\|_2^2 - \frac{1}{C}\log p(M(I_n)) \tag{9}$$

where C is a constant resulting from noise parameters.

In the adversarial model, if we use binary cross-entropy as the loss function for the discriminator and constraint the output of denoiser to be close to the clean image, the model then is optimized over a similar loss function. The only difference being that the output of the network is constrained to be close to the clean image I_c other than I_n. This difference is justified as the patch prior models want the output to be close in structure to the actual image, but it doesn't have the clean patch.

3. Experiments

In this section, we present the observations made during the training and evaluation of our model for denoising.

3.1. Training and testing Data

Training Data: The training data consists of Images from MIT Indoor dataset [17] and Places dataset [26]. These two datasets were chosen because they contain images of two different modalities; indoor scenes and outdoor scenes. Together, these two datasets have provided our model with good examples of most possible textures and patterns available in real world data.

For preparing training data, we have randomly chosen 5000 images from each of these datasets. A random 64×64 crop is extracted from each of the images. Then the pixels are rescaled to the range $[0, 1]$.

During the training process, the noisy images are generated by adding a random level of Gaussian noise to the image. The model is not given any information about the amount of noise added. This has helped our model to be a blind denoiser.

Test Images: The model performances are evaluated on the test set used in [24]. This set of 300 images contains 100 images from BSDS300 [14] and 200 images from PascalVOC [6]. These set of images are a super-set of the test set used in [19, 21, 27] and was first used in [24]. Since the denoiser network is fully convolutional, images need not be re-sized or cropped during testing. They can simply be fed to the model and it will reconstruct the denoised image.

Validation set: We have used the 7 standard images used in [16] as the validation set during training procedures. During training, the model is evaluated for denoising using each of these images for multiple noise levels. All the denoising performances of the model during training has been plotted by the average performance over these images.

3.2. Denoising performance

Peak Signal-to-Noise Ratio (PSNR) is a common measure to gauge denoising performance. PSNR measures dissimilarity between two images and hence to measure denoising performance, we simply measure the PSNR value between the denoised image and the original, noise free image. For a clean image I_c and a denoised image I_d with range of pixel values from 0 to 255, PSNR is computed as

$$PSNR(I_c, I_d) = 10\log_{10}\left(\frac{255^2}{mse(I_c, I_d)}\right) \tag{10}$$

$$mse(I_c, I_d) = \frac{1}{|I_c|}\|I_c - I_d\|_2^2 \tag{11}$$

$$|I_c| \rightarrow \text{ size of the image} \tag{12}$$

During the initial phase of adversarial training, the discriminator accuracy is comparatively lower because the discriminator cannot classify real and denoised images. But as training progresses, the discriminator gets better at this task. The generator (denoiser) now under the influence of adversarial loss, slowly begins to produce natural looking images

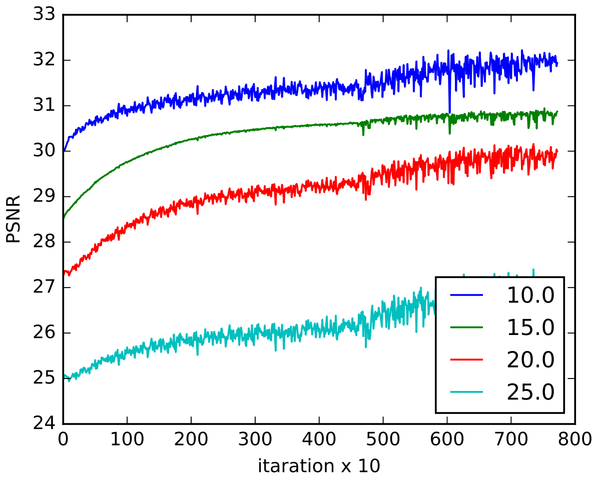

Figure 5. Denoising performance of the denoiser model on validation set during adversarial training. The model performance is evaluated every 10 iteration on each noise level on all 7 images of the validation set. The plotted values are average of all 7 PSNR's

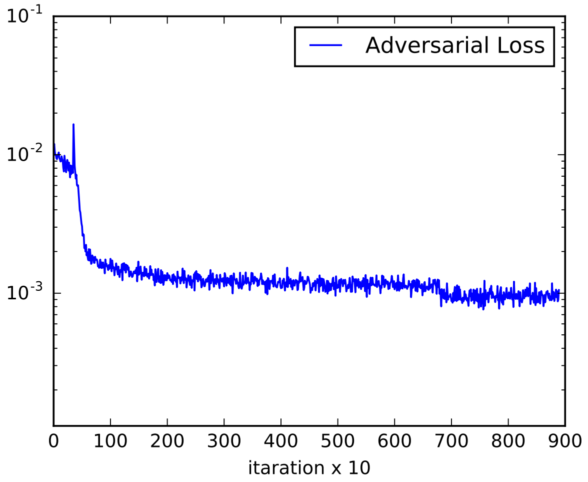

Figure 6. Trend of evolution of Adversarial loss during training iterations. The values are plotted every 10 iterations.

and we see a decrease in the training loss. The adversarial loss value vs iteration number is plotted in Fig 6. Fig 5 shows the average PSNR values over the validation set for each of the noise levels.

Table 2 gives the performance of our model against other denoising algorithms. A point to be noted here is that except [24] and our method, all the other methods are not blind denoising techniques. They are provided standard deviation of the added Gaussian noise and the algorithm adapts to these values accordingly.

Table 2. PSNR values of denoised images on test set introduced by [24]. Only DCGRFN[24] and our method are blind denoising approaches. Other methods are explicitly given standard deviation of the additive gaussian noise.

Sigma	10	15	20	25
BM3D [3]	33.38	31.09	29.53	28.36
WNNM [9]	33.57	31.28	29.7	28.50
EPLL [27]	33.32	31.06	29.52	28.34
CSF [21]	-	-	-	28.43
DCGRFN [24]	33.56	31.35	29.84	28.67
Ours	33.41	31.17	29.59	28.49

4. Related Work

The corrupting process that results in a noisy image can be seen as

$$I_n = I_c + N \tag{13}$$

where N is the noise and I_c is the clean image(patch).

If the corrupting noise is uncorrelated, and we have a large number of corrupted samples of the same patch, averaging them all, would give us a very good approximation to the clean patch. But a naive application of this idea is limited by two constraints.

1. Large number corrupted versions of same patches are not available.

2. We are limited to working with only noisy patches.

But natural images are full of repeating patterns and textures. The second constraint limits identifying the patterns because high similarity might as well be induced by noise or vice-versa. Solutions to solve these problems have given some of the classical works in denoising.

If we ignore the fact that similarity measure might give incorrect results for noisy patches, then the averaging step has to compensate. A simple Euclidean distance in the local neighborhood will give a set of noisy patches that are similar to each other.

Non-local means algorithm [1] modifies the averaging step to be a weighted averaging, where the weights are given by the similarity measure. BM3D [3] uses collaborative filtering of all the similar patches to achieve superior results. Weighted Nuclear Norm Minimization [9] exploits the fact that set of similar patches would be of low rank if they were noise free. Simply solving for a set which gives a lower weighted nuclear norm removes the noise from the data.

Assuming prior on image patches has lead to denoising methods which does not involve finding similar patches at all. K-SVD [5] method applies a sparse dictionary model to noisy patches which essentially remove the noise from

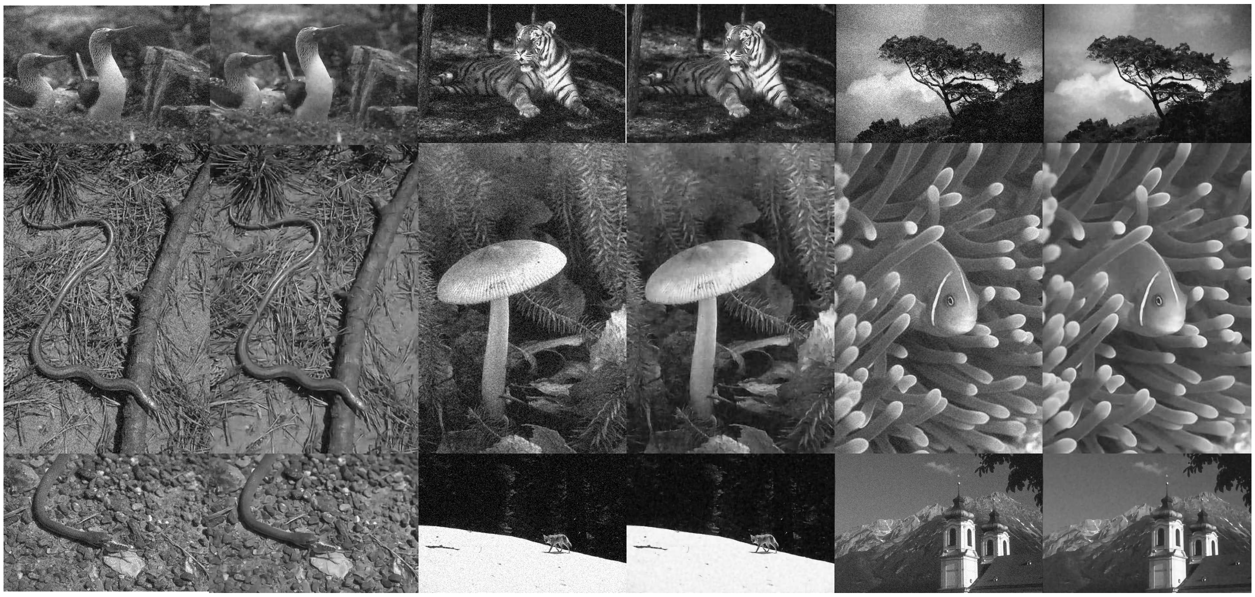

Figure 7. Denoising results of our model. Image in the left of each pair shows the noisy image and the image in the right shows the denoised image.

them. The sparse dictionary used in this method was 'learned' out of the large corpus of natural or clean images.

The first attempt to learn a generic image prior was given by Product-of-Experts [10] which was later extended to image denoising and inpainting by Field-of-Experts [19]. Both methods involve learning a prior from a generic image database and then using the prior for iterating towards a noise free patch. Minimizing the expected Patch Log Likelihood [27] also used a learned Gaussian mixture prior.

But with deep learning techniques, new methods are devised which can learn image prior implicitly as model parameters and simply compute the noise free patch. A network resembling fully convolutional network was used in [11] to get a denoiser model. In [2], a 5 layer fully connected network gave state-of-the-art performance. But both these models require different parameters to be specifically trained for each noise level.

In [24], the authors have used an end-to-end trainable network which uses Gaussian conditional random field. This model uses successive steps of denoising and noise parameter estimation to eventually give a model which can do blind denoising.

In contrast to the existing works, our model is simple and easy to train. It essentially results in a set of convolution and non-linearity and hence using it for denoising is extremely simple. Also, our model is not applied on patches. It takes as input the entire image and simply computes the denoised image. This allows it to be fast in comparison. The model is trained on varying noise levels together and hence it allows our model to be a blind denoiser which is trained end-to-end. There is no parameter estimation and the model is capable of automatically adjusting to the required noise level to give the best output.

5. Conclusion

In this work, we addressed whether Convolutional Neural Networks can solve the problem of image denoising.

We have proposed a simple architecture which gives very competitive denoising results. The architecture contains three unique parts. A multi-scale feature extraction layers, damping layers, and reconstruction layers.

We have also proposed a three stage training procedure to train the model. In the first stage, the multi-scale feature extraction layer is trained to extract features for image reconstruction by using clean images. In the second stage, the damping layers are trained to diminish activations of noise variant filters.

In the final stage, we have successfully adopted adversarial training to this framework with a modified adversarial loss which greatly improves the performance of the denoiser over the limit imposed by ℓ_2 loss. The proposed denoiser, a fully convolutional neural network, is a simple model with fewer parameters. The model denoises the given noisy image in a single pass without any need for patch extraction step and hence is computationally very efficient.

6. Acknowledgement

This work was supported by ISRO-IISc Space Technology Cell, Indian Institute of Science, Bangalore (Project No: ISTC0338).

References

[1] A. Buades, B. Coll, and J.-M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation*, 4(2):490–530, 2005.

[2] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2392–2399, 2012.

[3] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Bm3d image denoising with shape-adaptive principal component analysis. In *Signal Processing with Adaptive Sparse Structured Representations*, 2009.

[4] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[5] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 15(12):3736–3745, 2006.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results.

[7] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[9] 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.

[10] G. E. Hinton. Products of experts. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, volume 1, pages 1–6. IET, 1999.

[11] V. Jain and S. Seung. Natural image denoising with convolutional networks. In *Advances in Neural Information Processing Systems 21*, pages 769–776. Curran Associates, Inc., 2009.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[13] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016.

[14] 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.

[15] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[16] 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.

[17] A. Quattoni and A. Torralba. Eq. indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 413–420. IEEE, 2009.

[18] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[19] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 860–867. IEEE, 2005.

[20] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

[21] U. Schmidt and S. Roth. Shrinkage fields for effective image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2774–2781, 2014.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[24] R. Vemulapalli, O. Tuzel, and M.-Y. Liu. Deep gaussian conditional random field network: A model-based deep network for discriminative denoising. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[25] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, pages 818–833, 2014.

[26] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.

[27] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *2011 International Conference on Computer Vision*, pages 479–486. IEEE, 2011.