On-Demand Learning for Deep Image Restoration (Supplementary Materials)

Ruohan Gao and Kristen Grauman
University of Texas at Austin
{rhgao,grauman}@cs.utexas.edu

The supplementary materials consist of:

A. Pseudocode for our on-demand learning algorithm.

B. Details of our network architecture.

C. Details of the fixated models setup.

D. Fixated models vs. All-rounder on SUN397 and image denoising.

E. Overall performance of our image denoising model.

F. Applications of our image inpainter to real images.

G. Qualitative results for interpolation and denoising.

H. Image denoising qualitative results on DB11.

A. On-Demand Learning Algorithm

We present the pseudocode of our on-demand learning algorithm as follows:

Algorithm 1: On-Demand Learning

N sub-tasks of increasing difficulty: $T_1, T_2, \ldots, T_N$
# of training examples for sub-task T_i per batch: B_i
Batch Size: $\mathbb{B}$
Initialization: $B_i = \mathbb{B}/N$
while *not converge* **do**
  continue training for one epoch and snapshot;
  if *end of epoch* **then**
    $i = 1$;
    for $i \leq N$ **do**
      validate snapshot model on sub-task T_i;
      get mean PSNR P_i;
    end
    update $B_i = \frac{1/P_i}{\sum_{i=1}^{N} 1/P_i} \cdot \mathbb{B}$;
  end
end

B. Deep Learning Network Architecture

Fig. 1 shows the complete network architecture used for all tasks to implement our on-demand learning idea. Our image restoration network is a symmetric encoder-decoder pipeline. The encoder takes a corrupted image of size 64×64 as input and encodes it in the latent feature space. The decoder takes the feature representation and outputs the restored image. Our encoder and decoder are connected through a channel-wise fully-connected layer.

Specifically, for our encoder, we use four convolutional layers. Following similar design choices in DCGAN [12], we put a batch normalization layer [7] after each convolutional layer to accelerate training and stabilize learning. The leaky rectified linear unit (LeakyReLU) activation [10, 15] is used in all layers in the encoder.

The four convolutional layers in the encoder only connect all the feature maps together, but there are no direct connections among different locations within each specific feature map. Fully-connected layers are usually used to handle this information propagation in present successful network architectures [8, 14]. In our network, the latent feature dimension is $4 \times 4 \times 512 = 8192$ for both encoder and decoder. Fully-connecting our encoder and decoder will increase the number of parameters explosively. To more efficiently train our network and demonstrate our concept, we use a channel-wise fully-connected layer to connect the encoder and decoder, as in [11]. The channel-wise fully-connected layer is designed to only propagate information within activations of each feature map. In our case, each 4×4 feature map in the encoder side is fully-connected with each 4×4 feature map in the decoder side. This largely reduces the number of parameters in our network and accelerates training significantly.

The decoder consists of four up-convolutional layers [9, 5, 17], each of which is followed by a rectified linear unit (ReLU) activation except the output layer. We use the Tanh function in the output layer, and the output is of the same size as the input image. The series of up-convolutions and non-linearities conducts a non-linear weighted upsampling

of the feature produced by the encoder and generates a higher resolution image of our target size (64×64).

C. Details of the Fixated Models Setup

We followed the current literature to train deep networks to target a certain degree of corruption for four tasks—image inpainting, pixel interpolation, image deblurring and image denoising—and demonstrate how severe the fixation problem is. We show the qualitative examples of fixated models for pixel interpolation and image denoising tasks in Fig. 2 as a supplement to Fig. 2 in the main paper.

Specifically, **for the image inpainting task**, we follow similar settings in [11, 16] and train a model to inpaint a large central missing block of size 32×32. During testing, the resulting model can inpaint the central block of the same size at the same location very well (first row in Fig. 2-a in the main paper). However, if we remove a block that is slightly shifted away from the central region, or remove a much *smaller* block, the model fails to inpaint satisfactorily (second row in Fig. 2-a in the main paper). Following [11], we replace pixels in removed blocks with the average pixel values in training images (which tend to look grey). We can observe that grey areas are retained in regions outside of the central block in the failure cases, which is a strong indicator that the trained network severely overfits to the central location.

For the pixel interpolation task, we train a model only based on heavily corrupted images (80% of random pixels deleted), following [16]. During testing, if we use the obtained model to restore images of the same corruption level, the images are recovered very well (first row in Fig. 2-a). However, if we test the same model on *lightly* corrupted (easier) images, the model performs very poorly (second row in Fig. 2-a). The trained network either produces common artifacts of deep networks like the checkerboard artifacts, or a much blurrier low-quality restored image.

For the image deblurring task, results are similar. We train a model only based on heavily blurred images ($\sigma_x = \sigma_y = 5$). The trained model can successfully restore very blurry images (same blurry level as training examples), but is unable to restore images that are much less blurry. In the second row of Fig.2-b in the main paper, we can observe some ripple artifacts, which may be similar to the shape of the Gaussian kernel function that the network overfits to.

For the image denoising task, we train a model only based on lightly corrupted images ($\sigma = 10$ for AWG noise). During testing, the model can successfully restore images of the same level of noise (first row in Fig. 2-b). However, it fails catastrophically when we increase the severity of noise on test images (second row in Fig. 2-b).

	CelebA Dataset		SUN397 Dataset	
	L2 Loss	PSNR	L2 Loss	PSNR
Rigid Joint Learning	5.90	26.38 dB	7.56	25.69 dB
Cumulative Curriculum	6.10	26.31 dB	7.73	25.60 dB
Cumulative Anti-Curriculum	5.90	26.35 dB	7.57	25.66 dB
Staged Curriculum	7.10	24.74 dB	9.11	23.87 dB
Staged Anti-Curriculum	53.1	21.19 dB	55.5	19.66 dB
Hard Mining	6.53	25.10 dB	8.52	24.10 dB
On-Demand Learning	**5.79**	**26.48 dB**	**7.49**	**25.80 dB**

Table 1. Summary of the overall performance of all algorithms for image denoising on CelebA and SUN397. This table is a supplement to Table 1 in the main paper, where due to space constraints we could show only the results for three tasks. Overall performance is measured by the mean L2 loss (in ‰, lower is better) and mean PSNR (higher is better) averaged over all sub-tasks.

D. Fixated Models vs. All-Rounder on SUN397 and Image Denoising

We show the complete comparison of our algorithm with fixated models on CelebA and SUN397 for all of the four tasks in Fig. 4, as a supplement to Fig. 4 in the main paper, where due to space constraints we could show only the CelebA results for three tasks. Results on SUN397 and image denoising are similar. Fixated models overfit to a specific corruption level (easy or hard). It succeeds beautifully for images within its specialty, but performs poorly when forced to attempt instances outside its specialty. In contrast, models trained using our algorithm perform well across the whole spectrum of difficulty. For inpainting, the fixated models even perform poorly at the size they specialize in, because they also overfit to the central location, thus cannot inpaint satisfactorily at random locations at test time.

E. Overall Performance of Our Image Denoising Model

In Table 1, we report average L2 loss and PSNR over all test images for the image denoising task, as a supplement to Table 1 in the main paper, where due to space constraints we could show only the results for three tasks. The results for image denoising are similar. Our proposed algorithm consistently outperforms all the well-designed baselines.

F. Applications of Our Image Inpainter

We show some applications of our image inpainter to real world scenarios in this section. Fig. 3 shows some examples of using our image inpainter to do scar removal on human face images, and object removal on natural scene images. For each example, the left image is the target real world image. Our inpainter can successfully remove scars on human faces, and selectively remove objects in photographs.

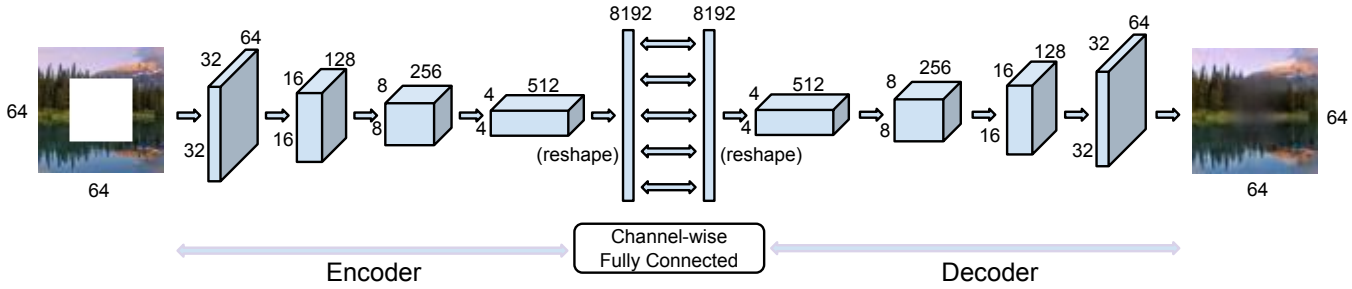


Figure 1. Network architecture for our image restoration framework. Our image restoration framework is an encoder-decoder pipeline with the encoder and decoder connected by a channel-wise fully-connected layer. The illustration is for image inpainting task. The same network architecture also holds for the other three tasks: pixel interpolation, image deblurring, and image denoising.

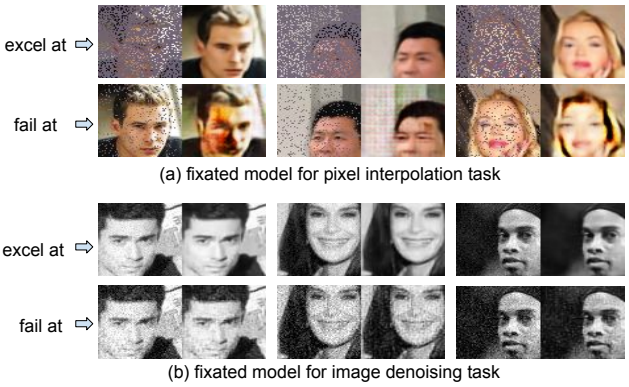


Figure 2. Qualitative examples of fixated models for pixel interpolation and image denoising tasks. The models overfit to a certain degree of corruption. They perform extremely well at that level of corruption, yet fail to produce satisfactory restoration results even for much easier sub-tasks.

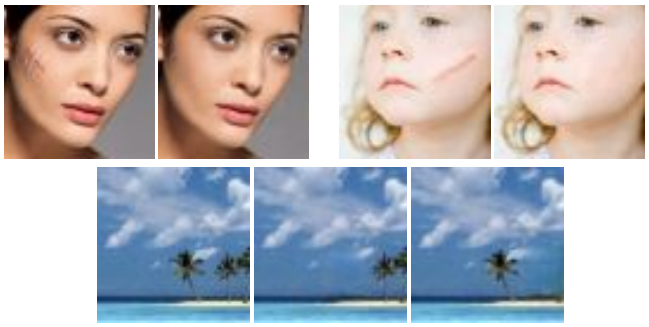


Figure 3. Real applications of our image inpainter. For each example, the left image is the target real image, and the right images are images processed by our image inpainter. Our inpainter can successfully remove scars on human faces, and selectively remove objects (trees in the last example) in photographs.

G. Qualitative results for interpolation and denoising

We show the qualitative examples output by our method for pixel interpolation and image denoising tasks in Fig. 5 as a supplement to Fig. 5 in the main paper. For each task, the first and second rows show test examples from CelebA and SUN397, respectively. For each quintuple, the first column shows the ground-truth image from the original dataset; the second column shows the corrupted image; the third column shows the restored image using the model trained using rigid joint learning; the fourth column shows the restored image using a fixated model; the last column shows the restored image using the all-rounder model trained by our algorithm. The fixated models can only perform well at a particular level of corruption. Models trained using our proposed on-demand approach are all-rounders that perform well on images of different degrees of corruption. With a single model, we restore corrupted images with different percentage of deleted pixels and denoise images of various noise levels.

H. Image Denoising Results on DB11

This section serves as a supplement to Section 6.7 in the main paper, where due to space constraints we could not describe the details of the setup of our image denoising system and present qualitative results.

We first describe the details of our image denoising system. Because the input of our network is of size 64×64, given a larger corrupted image $\mathcal{C}$, we first decompose the image into overlapping patches of size 64×64 and use a sliding-window approach to denoise each patch separately (stride 3 pixels), then average outputs at overlapping pixels.

We then present the qualitative results. Particularly, we first compare the denoising results of image Lena across the spectrum of difficulty in Fig. 6. We show image denoising results at four different corruption levels ($\sigma = 10, 25, 50, 75$). For each column, the first row shows the original real image; the second row shows the image corrupted by AWG noise with the specified sigma value; the third and fourth rows show the restoration results using KSVD [1] and BM3D [4] correspondingly assuming $\sigma = 25$ for the test image; the fifth row shows the denoising re-

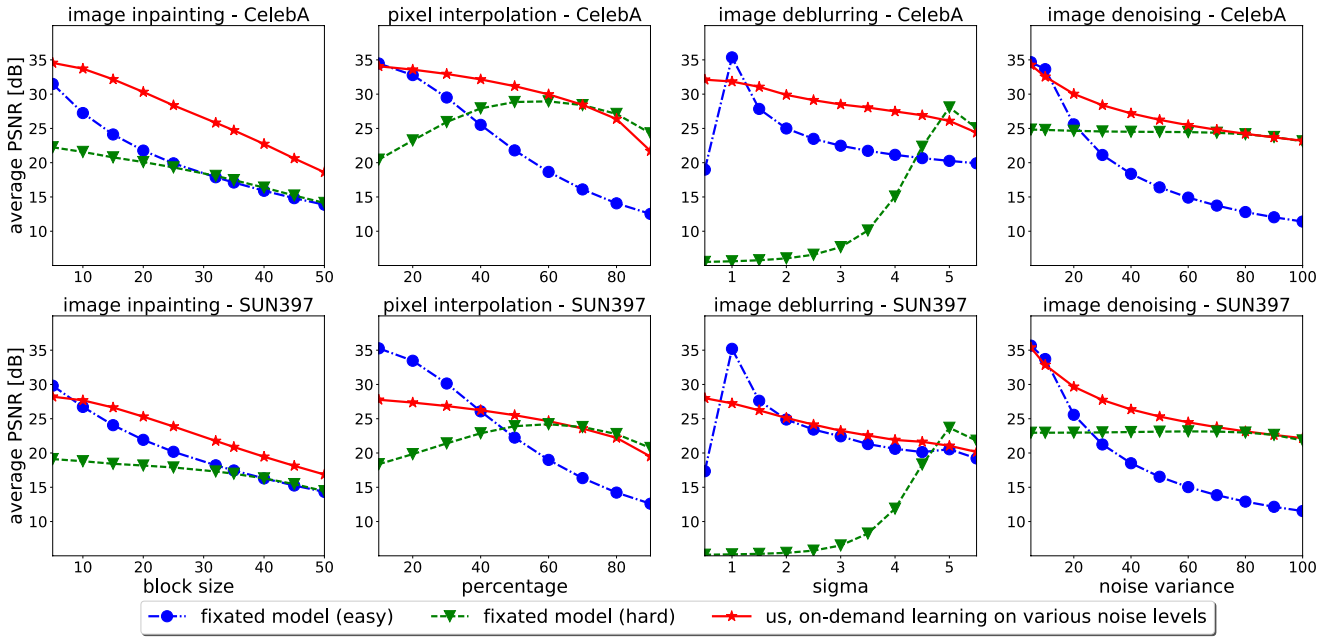


Figure 4. Our on-demand learning algorithm vs. fixated models for all the four tasks on CelebA and SUN397. This figure is a supplement to Fig. 4 in the main paper, where due to space constraints we could show only the results for three task on CelebA. Models trained using our algorithm perform well over the spectrum of difficulty, while fixated models perform well at only a certain level of corruption.

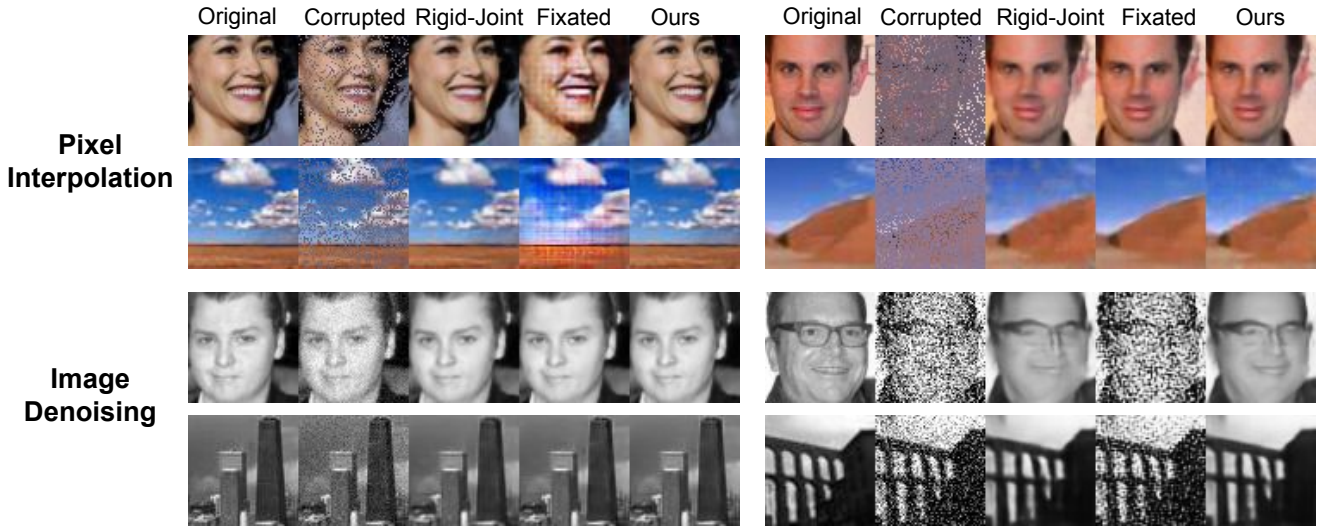


Figure 5. Qualitative examples of pixel interpolation and image denoising. For both tasks, the first row shows testing examples of CelebA dataset, and the second row shows examples of SUN397 dataset. For each quintuple, **Column 1**: Original image from the dataset; **Column 2**: Corrupted image; **Column 3**: Restored image using rigid joint training; **Column 4**: Restored image using a fixated model; **Column 5**: Restored image using our method. Models trained using our method can handle arbitrary levels of distortions, while the fixated models can only perform well at a particular level of corruption.

sult of the MLP [2] model trained for $\sigma = 25$[1]; the sixth row shows the restoration result using WCNN [6] assuming $\sigma = 25$ for the test image; the seventh and eighth rows show the restoration results of the CSF [13] model and the TNRG [3] model trained for $\sigma = 25$[2][3] correspondingly; the last row shows the denoising result of the model trained using our on-demand learning algorithm. K-SVD, BM3D and WCNN only work well when given the correct sigma value at test time, which is impractical because it is difficult to gauge the corruption level in a novel image and decide which sigma value to use. The MLP, CSF, TNRG models trained for $\sigma = 25$ are fixated models that perform well only at that specific level of corruption. However, the model trained using our proposed method performs well on all four corruption levels, and it is a single model without knowing the correct sigma value of corrupted images at test time. Finally, in the end we append the image denoising results using our denoising system of all the 11 images at noisy level $\sigma = 25$.

[1]We use the authors publicly available code (`http://people.tuebingen.mpg.de/burger/neural_denoising/`) in which the system is trained for $\sigma = 25$. The authors also propose a variant of the system trained on various corruption levels with σ given as input to the network, and it requires the σ value to be available at test time. This version is not available in the public code, and it is also unclear how the true σ value would be available for a novel image with unknown distortions.

[2]We use the authors publicly available code (`https://github.com/uschmidt83/shrinkage-fields/`) and use the model trained for $\sigma = 25$.

[3]We use the authors publicly available code (`http://gpu4vision.icg.tugraz.at/index.php?content=downloads.php`) and use the model trained for $\sigma = 25$.

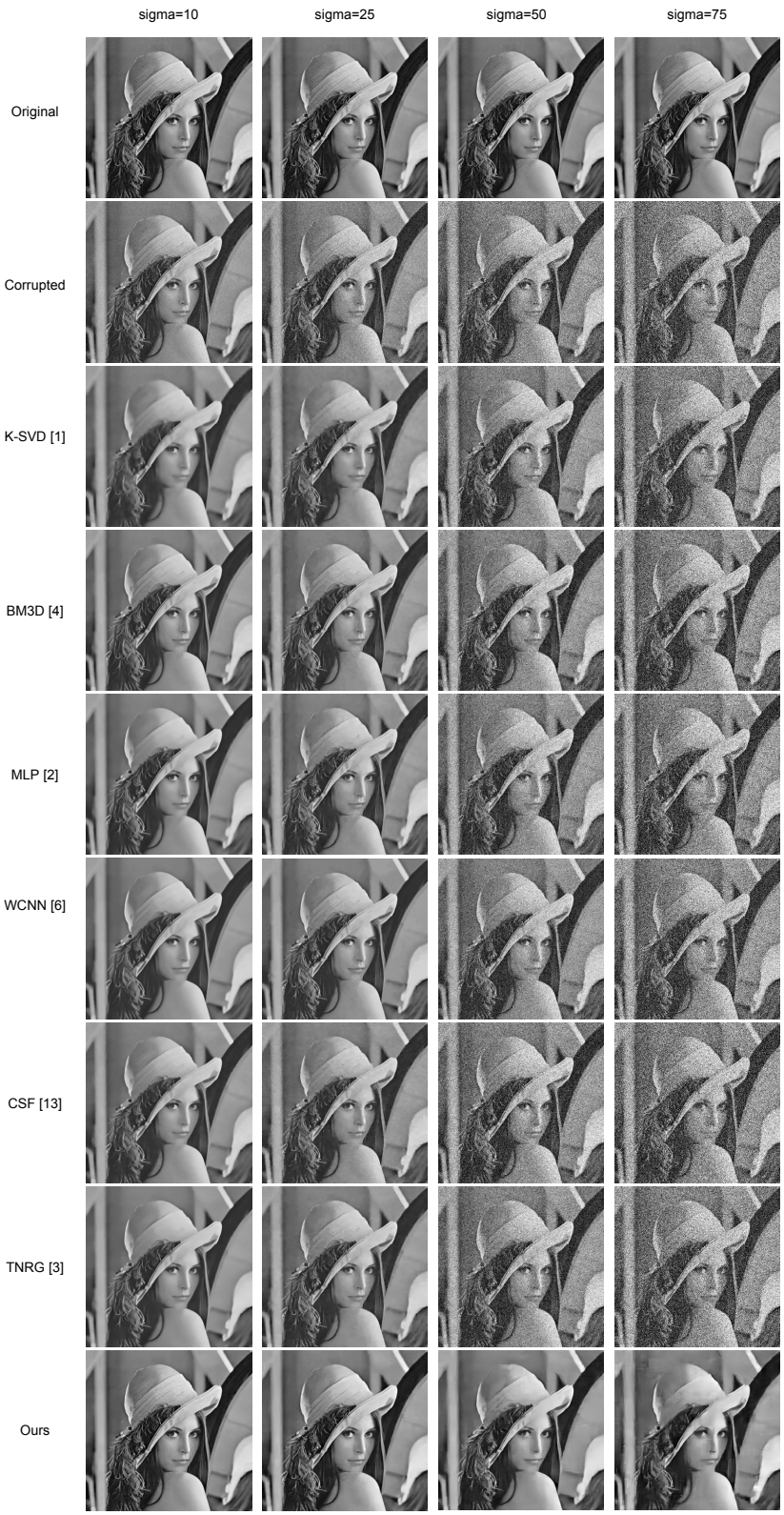
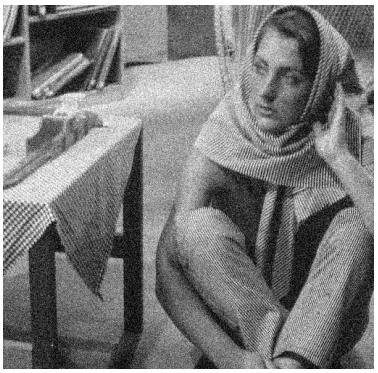


Figure 6. Denoising results of image Lena at various corruption levels. All methods are applied as a single model to all test images. KSVD [1], BM3D [4], MLP [2], WCNN [6], CSF [13] and TNRG [3] perform well only at a particular level of corruption, while the image denoising model trained using our method performs well at all corruption levels.

clean (name: Barbara)



noisy (PSNR: 20.28 dB)

ours (PSNR: 28.92 dB)

clean (name: Boat)

noisy (PSNR: 20.29 dB)

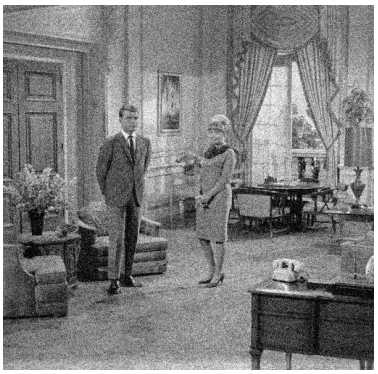
ours (PSNR: 30.11 dB)

clean (name: C.man)

noisy (PSNR: 20.55 dB)

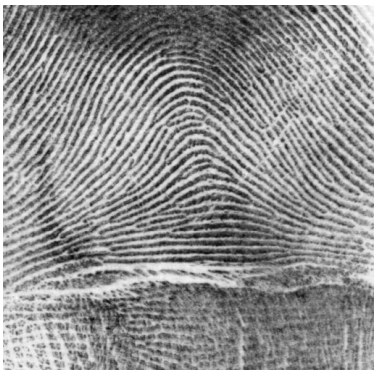
ours (PSNR: 29.41 dB)

clean (name: Couple)

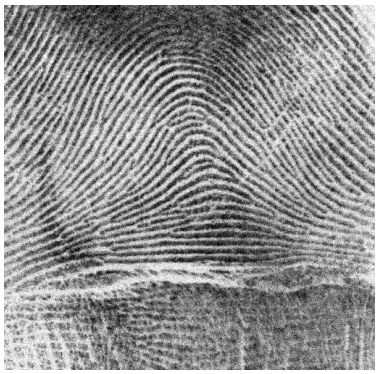


noisy (PSNR: 20.28 dB)

ours (PSNR: 30.04 dB)



clean (name: F.print)



noisy (PSNR: 20.26 dB)

ours (PSNR: 27.81 dB)

clean (name: Hill)

noisy (PSNR: 20.28 dB)

ours (PSNR: 30.03 dB)

clean (name: House)

noisy (PSNR: 20.22 dB)

ours (PSNR: 33.14 dB)

clean (name: Lena)

noisy (PSNR: 20.24 dB)

ours (PSNR: 32.44 dB)

clean (name: Man)

noisy (PSNR: 20.24 dB)

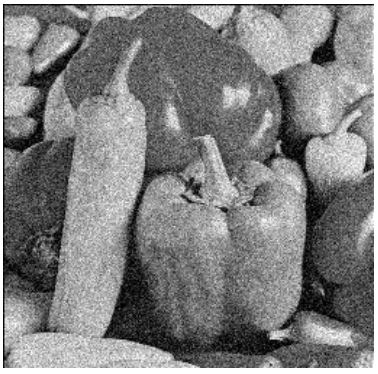
ours (PSNR: 29.92 dB)

clean (name: Montage)

noisy (PSNR: 20.77 dB)

ours (PSNR: 32.34 dB)

clean (name: Peppers)



noisy (PSNR: 20.31 dB)

ours (PSNR: 30.29 dB)

References

[1] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 2006. 3, 6

[2] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *CVPR*, 2012. 5, 6

[3] Y. Chen and T. Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *TPAMI*, 2016. 5, 6

[4] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 2007. 3, 6

[5] A. Dosovitskiy, J. Tobias Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015. 1

[6] S. Gu, L. Zhang, W. Zuo, and X. Feng. Weighted nuclear norm minimization with application to image denoising. In *CVPR*, 2014. 5, 6

[7] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 1

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[9] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

[10] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013. 1

[11] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 1, 2

[12] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016. 1

[13] U. Schmidt and S. Roth. Shrinkage fields for effective image restoration. In *CVPR*, 2014. 5, 6

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1

[15] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015. 1

[16] R. Yeh, C. Chen, T. Y. Lim, M. Hasegawa-Johnson, and M. N. Do. Semantic image inpainting with perceptual and contextual losses. *arXiv preprint arXiv:1607.07539*, 2016. 2

[17] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 1