Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 2096 – 2109

doi:10.3906/elk-1911-138

Image denoising using deep convolutional autoencoder with feature pyramids

Ekrem ÇETİNKAYA[1,2*], **M. Furkan KIRAÇ**[1]

[1]Department of Engineering, Faculty of Computer Science, Özyeğin University, İstanbul, Turkey
[2]ITEC-Institue of Information Technology, Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria

Received: 23.11.2019 • **Accepted/Published Online:** 11.04.2020 • **Final Version:** 29.07.2020

Abstract: Image denoising is 1 of the fundamental problems in the image processing field since it is the preliminary step for many computer vision applications. Various approaches have been used for image denoising throughout the years from spatial filtering to model-based approaches. Having outperformed all traditional methods, neural-network-based discriminative methods have gained popularity in recent years. However, most of these methods still struggle to achieve flexibility against various noise levels and types. In this paper, a deep convolutional autoencoder combined with a variant of feature pyramid network is proposed for image denoising. Simulated data generated by Blender software along with corrupted natural images are used during training to improve robustness against various noise levels. Experimental results show that the proposed method can achieve competitive performance in blind Gaussian denoising with significantly less training time required compared to state of the art methods. Extensive experiments showed the proposed method gives promising performance in a wide range of noise levels with a single network.

Key words: Image denoising, convolutional autoencoder, feature pyramid, image processing

1. Introduction

Image denoising is 1 of the fundamental problems in the image processing field due to being an essential step in many computer vision applications such as medical imaging. For example, medical images tend to corrupt more if the radiation level is decreased [1, 2]. Therefore, denoising techniques are important to shift the balance towards less radiation exposure for patients in radiation level and image quality trade-off without sacrificing the image quality.

The aim of image denoising is to obtain clean image x from corrupted version y that can be modeled as $x = y + n$ where n is the noise of specific type. Most of the methods in the literature [3–7] focus on specific type for n, namely additive white Gaussian noise (AWGN) since natural images are assumed to have additive random noise which can be modeled with AWGN.

Numerous methods have been proposed for image denoising throughout the years. Traditional model-based methods rely on using image priors and exploiting nonlocal self-similarity of the images. They can achieve high denoising performance, especially popular state of the art methods such as BM3D [3] and WNNM [7]. However, they have several common drawbacks. First, these methods work based on solving a complex optimization problem, hence making the inference process resource-consuming. Second, these models require manually chosen image priors that require high domain knowledge and may not be able to characterize complex image structures. Finally, these methods usually cannot be used to remove spatially varying noise.

*Correspondence: ekrem@itec.aau.at

Discriminative learning methods overcome the aforementioned problems by implicitly learning image priors during the training phase using a set of noisy and clean input pairs. A nonlinear diffusion model, trainable nonlinear reaction-diffusion (TNRD), is proposed in which parameters are learned through a loss-based approach that can be used for image denoising in [8]. However, with [9] showing that it is possible to achieve state of the art results with a plain multilayer perceptron (MLP); there has been a shift towards using neural networks for image denoising recently [4–6, 10–12].

Most of the existing neural network-based methods follow a convolutional neural network (CNN) based approach. The first utilization of CNNs for image denoising can be found in [10], but the first CNN based method that can achieve state of the art performance is DnCNN [4]. Recent advancements in network training methods for CNNs such as residual learning [13] and batch normalization [14] are utilized to achieve state of the art performance. After the release of DnCNN, several approaches are proposed to further improve CNN based denoisers. FFDNet [5] introduces a tunable noise level map as input to improve flexibility against noise levels and types. It was shown in [6] that dilated convolution can be used to speed up the network without sacrificing performance. A more realistic noise model than AWGN is proposed in [15] to achieve better real-world image denoising performance.

In this paper, a deep convolutional autoencoder combined with a variant of feature pyramid network [16] component is proposed as a different approach for image denoising. Benefiting from both convolutional autoencoder and feature pyramid structures, the proposed network can achieve competitive results with state of the art methods such as BM3D [3], DnCNN [4] and FFDNet [5] in blind Gaussian image denoising in grayscale images. Extensive experiments showed that it can also be used in color image denoising with promising performance. Moreover, the performance of the proposed network in denoising Blender renders is noteworthy. It should also be noted that the proposed network does not require any prior information about the noise level during the testing phase which makes it possible to use a single network for denoising images with various noise levels.

The rest of the paper is organized as follows. Section 2 provides a summary of existing image denoising methods in the literature. Section 3 presents the proposed method and discusses its features. In Section 4, experimental setup is explained and extensive evaluations are given. Results are presented in Section 5 and paper is concluded by discussing findings in Section 6.

2. Related work

Different approaches have been applied in the image denoising problem. Traditional methods such as BM3D [3] or WNNM [7] uses image priors and exploit nonlocal self-similarity in the images. BM3D stands out with its performance in model-based methods. It focuses on obtaining an enhanced sparse representation of the image in the transform domain and uses that representation for denoising. However, the inference phase of BM3D and WNNM are time-consuming and they require manually defined image priors.

In [8] a nonlinear diffusion model, trainable nonlinear reaction-diffusion (TNRD), is proposed in which all parameters are learned from training data through a loss-based approach and it was shown that it can be used for image denoising. Authors in [9] showed that using plain multilayer perceptron and training it with noisy and clean image pairs is sufficient to achieve state of the art denoising performance.

The usage of CNNs in image denoising can be traced back to [10]. DnCNN [4] was the first CNN based method that could achieve state of the art denoising performance. Residual learning [13] and batch normalization

[14] methods are used in DnCNN and it outperformed traditional methods such as BM3D [3] in blind Gaussian denoising. Following the success of DnCNN, several more approaches are proposed to improve the performance of CNN based denoisers. FFDNet [5] introduces a tunable noise level map as input to the network. This enables a single network to effectively handle a wide range of noise levels and even spatially variant noises by using a nonuniform noise level map. In [6], usage of dilated convolution is proposed to speed up the network without sacrificing performance. To improve the performance of CNN denoisers in real-world images, a more realistic noise model than AWGN that considers both heterogeneous Gaussian noise and in-camera processing pipeline is proposed in [15].

Stacked denoising autoencoders (SDA) are introduced in [17] as a pretraining tool that can extract high-level representation of data. Sparse SDAs (SSDA) are used in [11] to remove noise from images. Instead of discarding hidden layer data of noisy input as in [17], authors use activated values of both noisy and clean data to produce training data for the next layer. However, SSDA is not flexible against noise levels and types. In [12], relatively simple convolutional autoencoder is used to remove noise from grayscale medical images. SSIM [18] instead of PSNR was used as an evaluation metric to obtain a better-correlated report about image denoising performance with human perception.

Generative adversarial network (GAN) [19] is used in [20] for image denoising. This method focuses on projecting noisy images onto the range of GAN by recovering corresponding latent vector for a given image and it can achieve favorable results if the uncorrupted version of the image is seen during GAN training.

Feature pyramids were commonly used in traditional object recognition in which the pyramids are manually engineered to achieve scale-invariant methods. However, deep learning based methods avoided using feature pyramids because of 2 reasons; they are expensive to compute and provided scale-invariance by using convolutional networks were enough for many applications. Despite their drawbacks, feature pyramids are required to achieve the most accurate results and a feature pyramid network (FPN) that can provide benefits of traditional feature pyramids without sacrificing performance is proposed in [16]. The proposed architecture can extract features in various frequency bands and combine them using lateral connections.

3. Proposed method

In this paper, a deep fully convolutional autoencoder with feature pyramid network component (AEFPNC) is proposed as an alternative approach for image denoising. Proposed method benefits from rectifier linear unit (ReLU) [21] and batch normalization [14] layers in the architecture.

3.1. Network architecture

The network takes a single image as input with size MxN. Then it is passed through *early encoder* which consists of 3 convolution layers. Each convolution layer is followed by ReLU and batch normalization operations. Zero padding is applied to keep the input size constant (MxN). 3×3 convolution is used throughout the network. Also, batch normalization is used in the input layer to improve the performance. *Early encoder* part increases the number of feature maps to 64 and prepares the input for feature pyramid subnetwork.

The output of the *early encoder*, C_1, is then downsampled using a convolution layer with *stride = 2* to obtain 3 more inputs for feature pyramid subnetworks. After this operation, there are 4 outputs $\{C_1, C_2, C_3, C_4\}$ with sizes $D = \{M \times N, (M/2) \times (N/2), (M/4) \times (N/4), (M/8) \times (N/8)\}$, respectively. Sizes are saved to be used during the upsampling operation to prevent ambiguity if any of the resulting image sizes are odd. There are 4

feature pyramid branches in total in the network and each branch consists of 4 $Conv + ReLU + BN$ layers with each layer reducing the number of feature maps. Each branch gives a set of 16 feature maps $\{F_1, F_2, F_3, F_4\}$ with the same sizes in D respectively. Before the branches are merged, each output except F_1 is upsampled to size MxN using bilinear upsampling operation. Then, the feature maps are concatenated and passed to 3 *smoothing* convolution layers before passing to the decoder as in [16] to reduce the aliasing effect of upsampling operation. The first 2 of those layers also reduce the number of feature maps to force the network to encode information before passing the output to the decoder. The resulting latent vector in the *encoder* has size MxN and 16 feature maps.

Final part of the proposed network is the *decoder*. Symmetric encoder-decoder architecture is used in the network. The decoder first increases the feature maps in the first half, then reduces in the final half and produces the final output. Sigmoid function in the final layer is used to limit the output values of the network between $0 - 1$.

Proposed network consists of 31 layers and $444,347$ trainable parameters in total. Network architecture is shown in Figure 1. ADAM [22] optimizer with learning rate 10^{-4} is used. Mean squared error (MSE) is used as loss function during training.

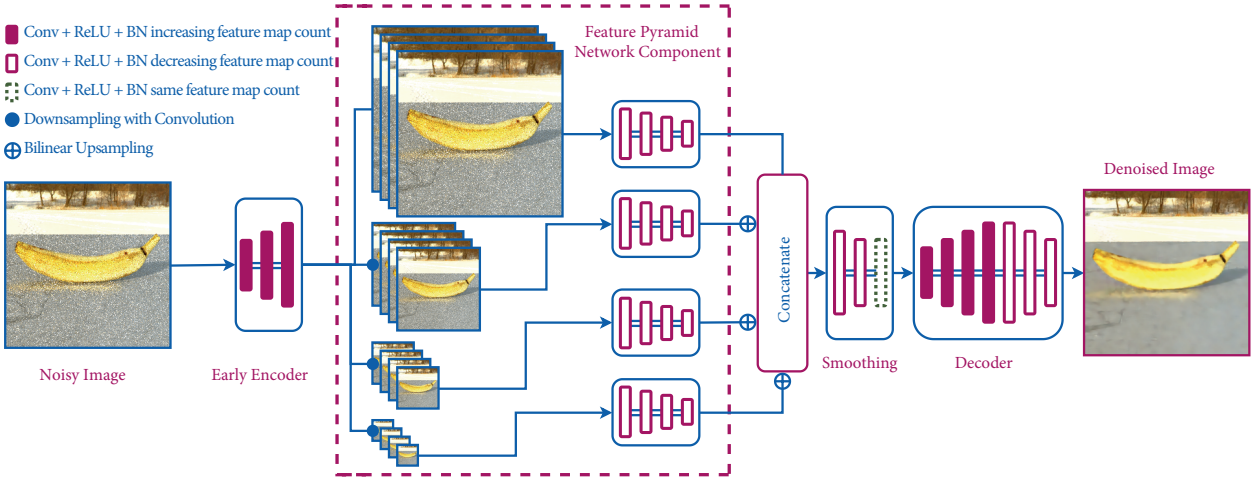


Figure 1. Proposed network architecture

3.2. Training dataset

The dataset for color image denoising contains both simulated and natural images to achieve a robust network against various noise levels. Blender software version 2.80 beta [1] is used to obtain simulated images with various noise levels by altering the sampling count during rendering. Sampling count in Blender defines how many different light paths will be traced during rendering process. The more sampling count means less noisy render will be obtained.

Furthermore, 432 images from Berkeley segmentation dataset (BSDS500) [23] are used and images are corrupted by adding AWGN with noise levels $\sigma \in [15, 25, 35, 50, 75]$. Images are corrupted with Poisson noise additionally and 256×256 patches are cropped from them. For corrupting images, `random_noise` function of scikit-image library [24] in Python is used. This process is done for both color and grayscale versions of

[1]Blender - a 3D modelling and rendering package (2018) [online]. Website http://www.blender.org [accessed 10 July 2019].

BSDS500 [23] dataset, the latter one is used in training network for denoising grayscale images. In the end, there are 5328 images of size 256×256 with 3 channels for training the color network and 2592 images of size 256×256 with a single channel for training the grayscale network.

Training dataset is further separated into 2 subsets to measure the effect of using simulated data in image denoising. Blender Training Set consists of 2736 images generated in Blender and BSDS Training Set consists of 2592 images obtained by corrupting 432 color images in BSDS500 [23] with mentioned noise levels above.

4. Experiments

This paper focuses on removing noise from both grayscale and color images. To demonstrate the effectiveness of the proposed method, Pytorch [25] is used to train and evaluate proposed models. All the experiments are carried out in Python 3.6.8 environment running on a PC with Intel(R) Core(TM) i7-7700K CPU 4.20 GHz and Nvidia GTX 1080 GPU. The training of a single model can be done in about 7.5 h for the grayscale model and 10 h for the color model. Training setups for different models are given in Table 1. Source code for this paper is made available[2].

Table 1. Training hardware and times for different models. Reported training setups in [4] and [5] are used for DnCNN and FFDNet values.

Method	Hardware	Training time
DnCNN	Intel(R) Core(TM) i7-5820K CPU @ 3.3GHz 32 GB of RAM NVIDIA Titan X Pascal GPU	3 days
FFDNet	Intel(R) Core(TM) i7-5820K CPU @ 3.3GHz 32 GB of RAM NVIDIA Titan X Pascal GPU	2 days
AEFPNC	Intel(R) Core(TM) i7-7700K CPU @ 4.2GHz 32 GB of RAM NVIDIA GTX 1080 GPU	10 h

4.1. Test datasets

Three test sets are used for evaluating color image denoising performance. Blender test set, CBSDS68 and Kodak24[3]. Blender test set consists of 52 different renders with 5, 10 and 20 sample counts. CBSDS68 consists of 68 images from validation set of BSDS500 [23] that are also used in evaluation of [4–6]. Images in CBSDS68 are corrupted by adding AWGN with noise levels $\sigma \in [15, 25, 35, 50, 75]$. Kodak24 contains 24 natural images and corrupted with the same settings as well.

For evaluating grayscale image denoising performance, 2 datasets are used, namely BSDS68 and Set12. BSDS68 contains grayscale version of images in CBSD68 set. Images are first converted to grayscale then the same corruption process is applied to obtain noisy samples. Set12 consists of commonly used 12 images in the literature for evaluating different image processing methods with same corruption process applied as above.

4.2. Effect of downsampling images

To test the effect of downsampling, 2 separate networks are trained with and without downsampling operation. Changing image size during encoding or decoding significantly declines denoising performance and causes

[2] https://github.com/ekremcet/AEFPNC
[3] Kodak Lossless True Color Image Suite (1999) [online]. Website http://r0k.us/graphics/kodak [accessed 02 August 2019].

artifacts in the resulting image. These artifacts are produced because network loses spatial location information during downsampling and it cannot be recovered for decoding. To avoid that, zero padding is used throughout the layers in encoder and decoder as many methods in literature follow the same approach (*e.g.* ,[4–6]).

4.3. Incorporating simulated data

To test the effect of using simulated data, network is trained with 2 different datasets. The first training was done using only BSDS Training set and second training was done by using Blender dataset only. Then, network is once again trained using both datasets. Including Blender dataset resulted in ≈ 1.5 dB increase in average PSNR values in natural image datasets BSDS68 and Kodak 24. Including Blender dataset also significantly improves Blender render denoising performance since AWGN fails to model noise in Blender renders.

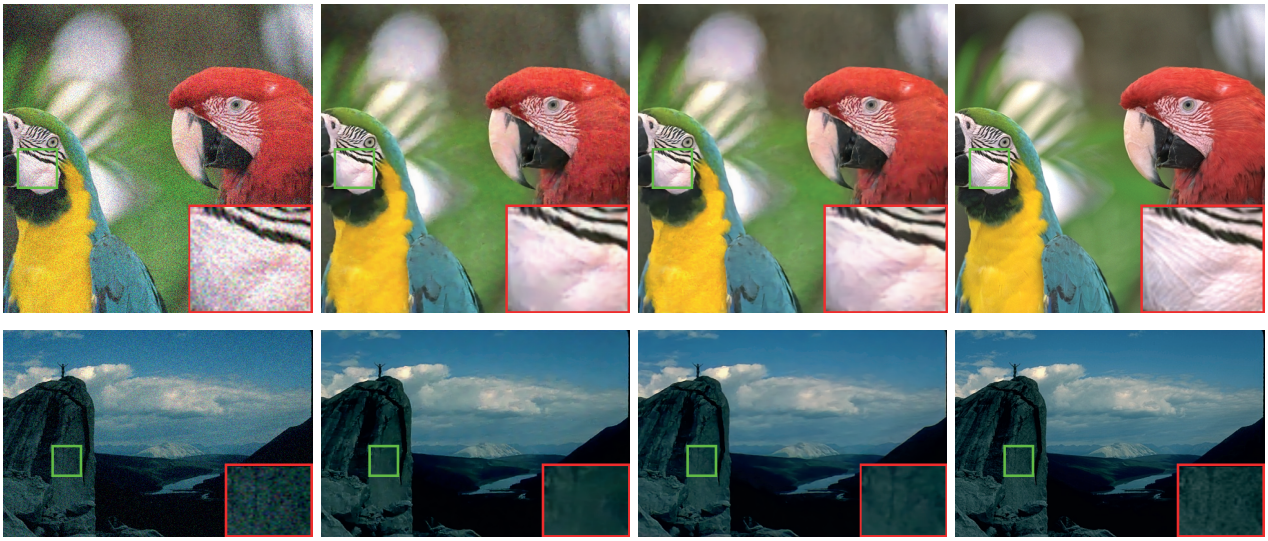
To better understand how simulated data in Blender boosts the performance, one can examine the noise generated in Blender. Noise generated in Blender rendering engine is dependent on the context and it can be seen in Figure 2 that the noise is concentrated around strong features of the image such as edges and corners. The noise is also dependent on the materials in the scene. Materials that affect the path of light traces such as glass have higher noise density compared to opaque materials. Moreover, the noise follows a similar pattern to AWGN in the rest of the image.

(a) Noise (b) Noisy Image (c) Ground Truth

Figure 2. Noise images of Blender renders.

4.4. Including feature pyramid network-based component

The main reason behind including a feature pyramid network-based component (FPNC) is to benefit from strong features in different frequency bands. Feature pyramid networks have already proven to be useful in image classification [16]. To test how they perform in image denoising, different network architecture is used. The network is a deep convolutional autoencoder with the same encoder and decoder structures as the proposed model. This is what the network would be if outputs of FPNC branches are ignored and the latent vector that is obtained from input with size $d \times d$ is passed to decoder in the proposed network. Using FPNC in the proposed network increases average PSNR by ≈ 1 dB and also helps preserving details which can be seen in Figure 3.



(a) Noisy σ = 25 (b) Without FPNC (c) With FPNC (d) Ground Truth

Figure 3. Denoised images with and without FPNC in autoencoder. Notice how FPNC helps preserving details in the image and produces smoother images.

4.5. Using wavelet transformation

Inspired by [26], the effect of wavelet transformation is also tested in this study. To achieve that, the early encoder component of our network is modified by including a discrete wavelet transformation (DWT) at the beginning with Haar wavelet and applying inverse wavelet transform (IWT) at the end. Then, the output is passed to FPNC. The resulting network increased the average PSNR by ≈ 0.15 dB for natural images and ≈ 0.30 dB for Blender images. However, training time is also increased by 35%. The usage of wavelet seemed promising, especially for Blender, but it is not included in the final network due to prolonged training time.

4.6. Effect of smoothing layers

In [16], authors use 3 convolutional layers after adding upsampled feature maps together to smooth out the aliasing effect of upsampling operation. Those layers are called *smoothing layers* in the proposed network. To test the effect of *smoothing layers* in the network, a separate network is trained without using them. It is seen that aliasing effect is observable if smoothing layers are removed. Figure 4 shows denoised images with noise level $\sigma = 25$ in Kodak 24 dataset by both networks.

(a) Without Smoothing (b) With Smoothing

Figure 4. Denoised images with and without *smoothing* layers. Notice how smoothing reduces aliasing effect in the images.

5. Results

This section presents results of the proposed network in blind AWGN removing in both grayscale and color images both quantitatively and qualitatively. The results are compared with three state of the art methods, BM3D [3], DnCNN [4] and FFDNet [5]. Same datasets are used for testing and images are corrupted with the same settings, hence reported results of those works are referred to while doing the comparison. Moreover,

qualitative results in Blender render denoising are given. More denoised image samples are available on a demo website[4].

5.1. Grayscale image denoising

The proposed network can achieve competitive performance with state of the art methods and even surpasses BM3D [3] in BSDS68 dataset. It can also achieve competitive denoising performance in Set12 dataset. It falls behind BM3D in this dataset and the possible main cause behind this is the repetitive structures of the images help BM3D to exploit nonlocal self-similarity. Table 2 shows the average PSNR values of different methods in BSDS68 and Set12 datasets. Figure 5 displays denoised versions of image 102061 in BSDS68 dataset with different methods.

Table 2. Average PSNR values of different methods in BSDS68 and Set12 datasets. Reported results in [5] are used for BM3D, DnCNN and FFDNet values.

Dataset	Methods	$\sigma = 15$	$\sigma = 25$	$\sigma = 35$	$\sigma = 50$
BSDS68	BM3D	31.07	28.57	27.08	25.62
	DnCNN	31.72	29.23	27.69	26.23
	FFDNet	31.63	29.19	27.73	26.29
	AEFPNC	31.14	28.84	27.38	25.92
Set12	BM3D	32.37	29.97	28.40	26.72
	DnCNN	32.86	30.43	28.82	27.18
	FFDNet	32.75	30.43	28.92	27.32
	AEFPNC	32.04	29.85	28.37	26.73

5.2. Color image denoising

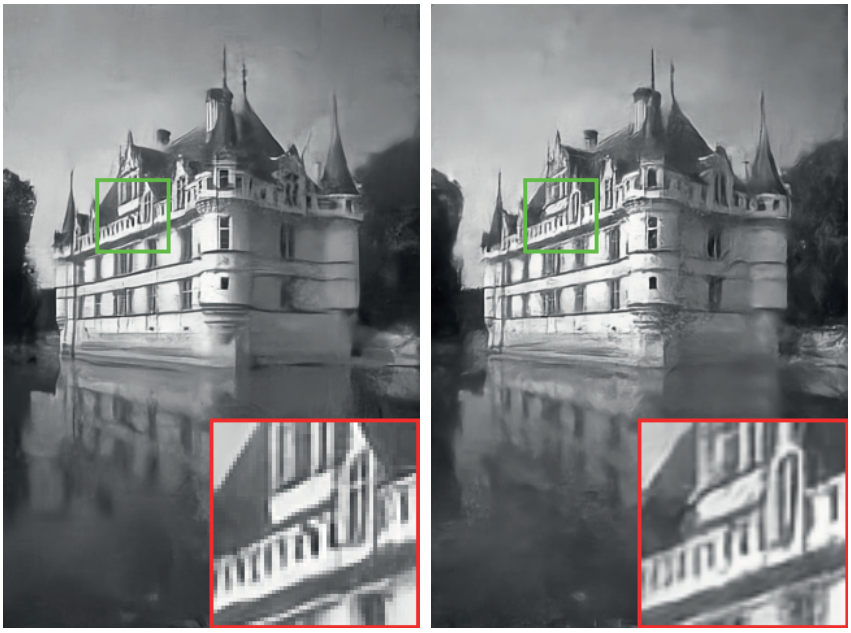
The proposed method falls behind other methods in color image denoising with quite a little margin. It can still be observed that the proposed method produces smoother images as in grayscale denoising and it is also better at preserving details while denoising images. Average PSNR values obtained by different methods in CBSDS68 and Kodak 24 datasets are given in Table 3 and an example denoising result is shown in Figure 6.

Table 3. Average PSNR values of different methods in CBSDS68 and Kodak24 datasets. Reported results in [5] are used for CBM3D, CDnCNN and FFDNet values.

Dataset	Methods	$\sigma = 15$	$\sigma = 25$	$\sigma = 35$	$\sigma = 50$
CBSDS68	CBM3D	33.52	30.71	28.89	27.38
	CDnCNN	33.89	31.23	29.58	27.92
	FFDNet	33.87	31.21	29.58	27.96
	AEFPNC	32.35	30.24	28.79	27.26
Kodak24	CBM3D	34.28	31.68	29.90	28.46
	CDnCNN	34.48	32.03	30.46	28.85
	FFDNet	34.63	32.13	30.57	28.98
	AEFPNC	33.24	31.13	29.69	28.16

[4] https://aefpncdemo.github.io/

(a) Ground Truth (b) Noisy σ = 50 (c) BM3D (26.21 dB)



(d) DnCNN (26.89 dB) (e) AEFPNC (26.50 dB)

Figure 5. Grayscale image denoising results of image 102061 in BSDS68 dataset with noise level $\sigma = 50$. BM3D and DnCNN results are taken from [5].

5.3. Blender render denoising

One possible use case of the proposed method is denoising Blender renders. Obtaining noise-free renders in Blender is a time-consuming process since it requires numerous calculations depending on the scene complexity. To reduce the required number of operations, a denoising method is applied after rendering the scene with

fewer samples in Blender. Performance of the proposed method is evaluated for render denoising. *Pavillion, Classroom, Agent 327 Barbershop* and *Car demo* demo files from Blender website are used. Figure 7 shows the results.

(a) Ground Truth (b) Noisy σ = 35 (c) CBM3D (25.93 dB)

(d) CDnCNN (26.58 dB) (e) AEFPNC (26.01 dB)

Figure 6. Color image denoising results of image 101085 in BSDS68 dataset with noise level $\sigma = 35$. CBM3D and CDnCNN results are taken from [4].

Using the proposed method for taking renders can dramatically speed up the process. To demonstrate it, two renders of the *Pavillion* scene are taken with 50 and 1000 sample counts. Then 50 sampled renders are

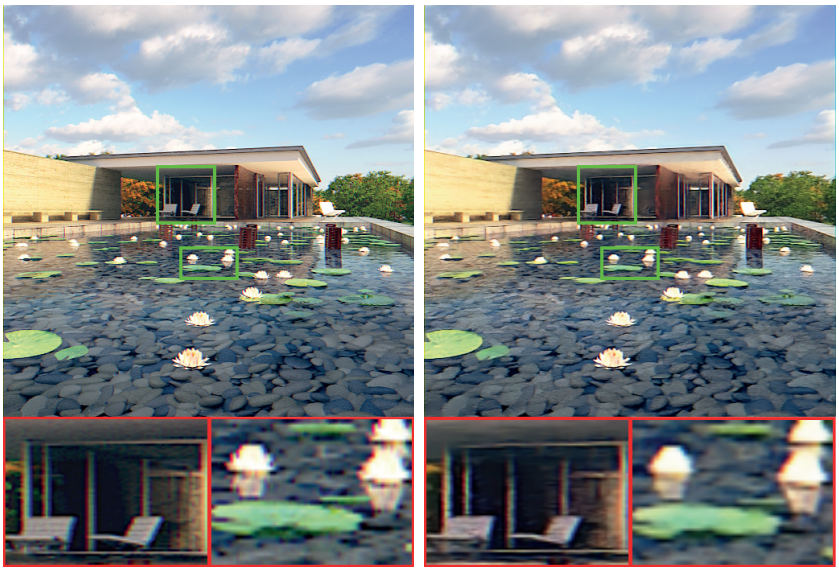
(a) Pavillion (b) Classroom (c) Barbershop (d) BMW

Figure 7. Blender render denoising results with four different scenes. Top row is noisy renders and bottom row is denoised renders by proposed model. Notice that when an object takes up relatively large portion of the scene, as pool in *Pavillion*, proposed method achieves better denoising performance.

denoised with the proposed method and resulting images are compared in Figure 8. Rendering the scene with 1000 sample counts takes 107.32 s while denoising approach takes 7.08 s in total.

6. Conclusions

In this paper, a convolutional autoencoder combined with a variant of feature pyramid network is proposed as an alternative approach to image denoising problem. Blender software is used to generate simulated noisy data to improve the robustness of the proposed network against noise types. Results show that the proposed network can achieve competitive results with state of the art methods in blind Gaussian image denoising for both color and grayscale images with significantly less training time. Moreover, it can achieve favorable denoising performance in removing corruption from Blender renders and it can significantly speed up rendering process in Blender. The proposed framework has the potential to achieve improved denoising performance with improvements, especially in color image denoising. As a future study, Wavelet transformation can be better utilized since it was proved to be useful in the experiments. Also, the proposed method can be extended to handle other image restoration tasks such as single image super-resolution (SISR) and JPEG image deblocking.



(a) 1000 Samples (b) 50 Samples + Denoised

Figure 8. Blender render denoising results in *Pavillion* scene with two different approaches. Image on the left is rendered with 1000 samples and it takes 107.32 seconds to complete. Image on the right is first rendered with 50 samples and then denoised by the proposed network which takes 7.08 seconds to complete in total.

References

[1] Goldman LW. Principles of CT: radiation dose and image quality. Journal of Nuclear Medicine Technology 2007; 1; 35 (4): 213-225.

[2] Huda W. Dose and image quality in CT. Pediatric Radiology 2002; 1; 32 (10): 709.

[3] Dabov K, Foi A, Katkovnik V, Egiazarian K. Image denoising by sparse 3-D transform-domain collaborative filtering. IEEE Transactions on Image Processing 2007; 16; 16 (8): 2080-2095.

[4] Zhang K, Zuo W, Chen Y, Meng D, Zhang L. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. IEEE Transactions on Image Processing 2017; 1; 26 (7): 3142-3155.

[5] Zhang K, Zuo W, Zhang L. FFDNet: Toward a fast and flexible solution for CNN-based image denoising. IEEE Transactions on Image Processing 2018; 25; 27 (9): 4608-4622.

[6] Wang T, Sun M, Hu K. Dilated deep residual network for image denoising. In: 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI); Boston, MA, USA; 2017. pp. 1272-1279.

[7] Gu S, Zhang L, Zuo W, Feng X. Weighted nuclear norm minimization with application to image denoising. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); Columbus, OH, USA; 2014. pp. 2862-2869.

[8] Chen Y, Pock T. Trainable nonlinear reaction diffusion: a flexible framework for fast and effective image restoration. IEEE Transactions on Pattern Analysis and Machine Intelligence 2016; 39 (6): 1256-1272.

[9] Burger HC, Schuler CJ, Harmeling S. Image denoising: can plain neural networks compete with BM3D? In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); Providence, RI, USA; 2012. pp. 2392-2399.

[10] Jain V, Seung S. Natural image denoising with convolutional networks. In: Advances in Neural Information Processing systems (NIPS); Vancouver, B.C., Canada; 2009. pp. 769-776.

[11] Xie J, Xu L, Chen E. Image denoising and inpainting with deep neural networks. In: Advances in Neural Information Processing Systems (NIPS); Lake Tahoe, NV, USA; 2012. pp. 341-349.

[12] Gondara L. Medical image denoising using convolutional denoising autoencoders. In: 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW); Barcelona, Spain; 2016. pp. 241-246.

[13] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); Las Vegas, NV, USA; 2016. pp. 770-778.

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

[15] Guo S, Yan Z, Zhang K, Zuo W, Zhang L. Toward convolutional blind denoising of real photographs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); Long Beach, CA, USA; 2019. pp. 1712-1722.

[16] Lin TY, Dollár P, Girshick R, He K, Hariharan B et al. Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); Honolulu, HI, USA; 2017. pp. 2117-2125.

[17] Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol PA. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research 2010; 11: 3371-3408.

[18] Wang Z, Bovik AC, Sheikh HR, Simoncelli EP. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing 2004; 13 (4): 600-12.

[19] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D et al. Generative adversarial nets. In: Advances in Neural Information Processing Systems (NIPS); Montreal, QC, Canada; 2014. pp. 2672-2680.

[20] Tripathi S, Lipton ZC, Nguyen TQ. Correction by projection: denoising images with generative adversarial networks. arXiv preprint arXiv:1803.04477. 2018.

[21] Nair V, Hinton GE. Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML); Haifa, Israel; 2010. pp. 807-814.

[22] Kingma DP, Ba J. Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980. 2014.

[23] Arbelaez P, Maire M, Fowlkes C, Malik J. Contour detection and hierarchical image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 2010; 33 (5): 898-916.

[24] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B et al. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 2011; 12: 2825-2830.

[25] Paszke A, Gross S, Chintala S, Chanan G, Yang E et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Advances in Neural Information Processing Systems (NeurIPS); Vancouver, VN, Canada; 2019. pp. 8024-8035.

[26] Liu P, Zhang H, Zhang K, Lin L, Zuo W. Multi-level wavelet-CNN for image restoration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW); Salt Lake City, UT, USA; 2018. pp. 773-782.