Image Denoising Based on A CNN Model

Zhe Liu, Wei Qi Yan, and Mee Loong Yang
Auckland University of Technology, Auckland 1010 New Zealand
e-mail: 349872994@qq.com

***Abstract*—In digital image processing, filtering noise to reconstruct a high quality image is an important work for further image processing such as object segmentation, detection, recognition and tracking, etc. In this paper, we will use a CNN model in deep learning for image denoising. Compared with traditional image denoising methods such as average filtering, Wiener filtering and median filtering, the advantage of using this CNN model is that the parameters of this model can be optimized through network training; whereas in traditional image denoising, the parameters of these algorithms are fixed and cannot be adjusted during the filtering, namely, lack of adaptivity. In this paper, we design and implement the denoising method based on a linear CNN model. Our experimental results show that the proposed CNN model can effectively remove Gaussian noise and improve the performance of traditional image filtering methods significantly.**

Keywords-image denoising; average filtering; Wiener filtering; median filtering; linear CNN model

I. INTRODUCTION

In digital image processing, whether image noise can be effectively filtered out or not will directly affect the subsequent processing such as object segmentation, edge detection, feature extraction and so on. Therefore, filtering the image noise is thought as a meaningful work. For the 2D visual signals, we usually have two forms of filtering which are based on spatial domain and frequency domain. Image filtering in spatial domain closely relates to the pixel intensity and its neighbors; meanwhile, image filtering in frequency domain utilizes the coefficients of multiple frequencies after visual signal decomposition.

For these two domains, the corresponding denoising methods can be categorized into spatial domain-based or frequency domain-based methods. The spatial domain-based methods directly tackle the intensity of each pixel of an image. The frequency domain-based methods handle the issue of image denosing by adjusting corresponding coefficients of multiple frequencies after image decomposition in frequency domain. The inverse transformation is usually accomplished to reconstruct the image using those modified coefficients and achieve the purpose of image denoising [1].

Artificial neural networks were employed as a classifier for pattern classification in digital image processing, e.g., optical characters recognition (OCR), face detection and recognition, image restoration and reconstruction, image enhancement, etc. With the further study of neural networks, the merits of neural networks in digital image processing have been fully investigated. We are going to discuss how neural networks could be applied to image denoising [2] in this paper.

In this paper, we use convolutional layers of a linear CNN model to implement image filtering. The advantage of using the CNN model is that it continuously optimizes the weights of convolution kernel during network training. We also design the linear CNN model to compare with the traditional linear and nonlinear filtering methods. After network training, we get two models which correspond to filter Gaussian noise and salt-and-pepper noise. From the effectiveness of filtering operations, the denoising through the proposed linear-CNN model has shown its superior performance compared with those traditional ones.

We will introduce the background of this paper in Section II. Our method is depicted in Section III. The results and analysis are presented in Section IV as well as the conclusion and future work are stated in Section V.

II. BACKGROUND

A. Traditional Denoising Apparoches

In this paper, we are going to investigate the denoising problem based on image filtering by using a CNN mode [3]. A slew of denoising approaches from deep learning associated with pattern classification as well as probability and mathematical statistics are also investigated and compared.

Average filtering is one of the typical linear filters [4] for image denosing. In 2007, combined with wavelet transform and average filtering, an efficient denoising method was proposed [5]. The denoised image was reconstructed by composing together three high-frequency components of the image with average filtering and low-frequency approximation. The method has better denoising result than that of the wavelet thresholding method or average filtering method.

Wiener filtering was a typical image denoising method, especially for motion blur removal. In 2007, a new algorithm was proposed where the directionalet coefficients of an image with Gaussian noise are subjected to a normal distribution [6]. By using Wiener filter with the directionalet coefficients, the image can be denoised effectively; meanwhile, the multidirectional framework has shown its super power in image denoising.

Median filtering can enhance the noisy images through blurring operations. In 2011, based on traditional median filtering, an improved fast algorithm of median filtering has

978-1-5386-6338-7/18/$31.00 ©2018 IEEE

been designed [7] which could achieve better results of noise removal.

However, because these traditional denosing methods based on image filtering could not tune the parameters during filtering, in this paper, we introduce a linear CNN for image noise removal.

B. Image Denoising Based on Neural Networks

Neural networks [8] have been widely used in image denoising because the networks adapt the nonlinear operations for digital image processing; furthermore, a nonlinear model of image denoising can be constructed without prior knowledge. Meanwhile, the parallel processing ability of neural networks makes it possible for image denoising and speed up image denoising process [9]. The pulse coupled neural network models, convolutional neural network models and fuzzy neural network models are effectively applied to image denoising.

In 2017, a PCNN (pulse coupling neural network)-based image noise reduction has been proposed [10]. In this method, the basic PCNN model has been simplified and the weights were adjusted adaptively; furthermore, the algorithm's accuracy for identifying noise was improved by posting noise points according to the difference of firing times between a neuron and its surrounding ones. Through using this algorithm, distinct results for image denoising have been achieved.

Different from the existing methods, in this paper, we will design and implement our image denoising method based on a linear CNN model. The model can greatly improve the performance of image noise filters.

III. Our Method

In this paper, we will use CNN for image denoising, we design a linear CNN model and use the convolutional layer in this model to simulate average filtering. Let the initial size of the convolution kernel be 11×11, during the training process, if a smaller kernel can get better results, then the CNN model will optimize the convolution kernel by letting the external elements of convolution kernel be 0; afterward, a new convolution kernel can be obtained which is equivalent to the smaller kernel.

Input layer. The input image is represented as a matrix, and the intensity of each pixel in the matrices is an integer in [0, 255]. In order to make the computing easier, we normalize the pixel values as a floating number in [0, 1] of Layer 1.

Convolutional layer. This layer is the core part in the CNN model. The parameter optimization and filtering operations are conducted in this layer. The initial size of this convolution kernel is 11×11 which is self-adaptive for better image denoising based on network training.

Compared with the traditional average filter, the proposed CNN model uses 9 convolution kernels and each output color component is composed with the convolutional outputs of all input color components and multiple convolution kernels. The essence of this filtering process is to use available images as the training dataset to reconstruct the denoised image.

Predict layer. The pixel values of output image from the convolutional layer are not necessarily in the interval [0, 1]; thus, in the predict layer, we will normalize the output image. For the intensities of those pixels $I(x, y)$ which are less than 0, we make them equal to 0; for those pixel values are greater than 1.0, we take 1.0.

Validation layer. In this layer, the MSE results between the normalized output images and the normalized labeled images will be calculated.

During the training process, we calculate the MSE between the output image before normalization and its normalized labeled image. The reason for this operation is to ensure that the optimization can be continued. We will use an optimizer to optimize the parameters of this CNN model so as to decrease the MSE. We use the stochastic gradient descent (SGD) for the optimization, which is represented as eq. (1).

$$W^{(i+1)} = W^{(i)} - \lambda \cdot \partial L/\partial \mathrm{W}, i=1,2,\ldots \quad (1)$$

where $W^{(i)} = (w_1^{(i)}, w_2^{(i)}, \cdots, w_n^{(i)})$ is the parameters of i-th iteration of our linear CNN model, λ is the learning rate and L is the loss function. For the output results from the convolutional layers, if we choose the outputs, then we use an optimizer to optimize the parameter $W^{(i)}$ rather than select $W^{(0)}$ and calculate the MSE directly. For this reason, let's consider that x_{ij} is the pixel x on the input image, (i, j) is its location, y_{ij} is the expected output of the pixel x. After the model training, the real output of x_{ij} is $f(W_{ij}; x_{ij})$. The corresponding loss function is represented by using eq.(2).

$$L(W; x, y) = h(g(f(W_x, x_{ij})), y_{ij})$$

$$L(W) = h(g(f(W_x))) \quad (2)$$

where g is the normalized function, h is the function for calculating MSE. We use eq.(3) to represent the normalized function.

$$g(x) = \begin{cases} 1 & x > 1 \\ 0 & x < 0 \\ x & 0 \le x \le 1 \end{cases} \quad (3)$$

With regard to pixel x, if the actual output $f(W_x; x_{ij}) < 0$, then we get $g(f(Wx))=0$. We use eq.(4) to represent $\partial L/\partial \mathrm{W}$ in eq. (1).

$$\partial L/\partial \mathrm{W}_x = \partial L/\partial h \cdot \partial h/\partial g \cdot \partial g/\partial f \cdot \partial f/\partial Wx \quad (4)$$

In eq. (4), we see if $g(f(W_x))=0$, then $\partial g/\partial f = 0$; thus, $\partial L/\partial \mathrm{W}_x = 0$, which means the SGD algorithm cannot optimize W_x. In this case, we calculate the MSE one more time during the training process.

IV. Results and Discussion

In this section, we present the datasets and the results of our experiments. We will demonstrate the results by using traditional denoising methods and the proposed linear CNN model. We evaluate the performance of these specific denosing methods by using the relevant metrics.

A. *Method-I: Using Traditional Filtering Algorithms*

In order to get the better denoising results, we need to figure out which traditional method has the best performance for denoising Gaussian noise and salt-and-pepper noise. We use four samples as shown in Fig.1. During the experiments, we first added Gaussian noise and salt-and-pepper noise to the test images. For each test image, three traditional denoising methods were used to remove two kinds of image noise, respectively; eventually, the MSE will be calculated. We found that the results are quite similar; therefore, we choose the image Lena to demonstrate the denoising results as shown in Fig.2.

Figure 1. Four images with the size of 512×512 in the dataset.

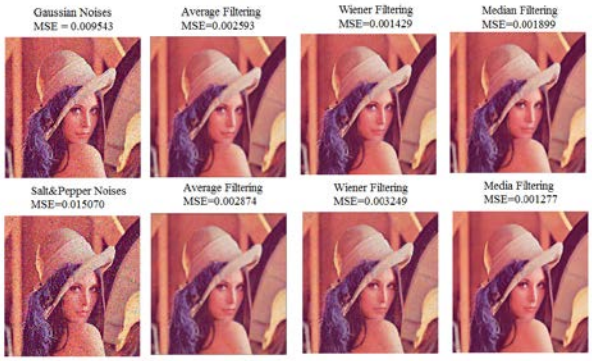

Figure 2. Three filtering methods for Gaussian noise and salt-and-pepper noise.

In Fig.2, we see that Wiener filter for removing Gaussian noise has the lowest MSE. The denoised image is the most similar one to the original image; for salt-and-pepper noise, median filter has the best performance [11]. In image denoising, we apply Wiener filter to remove Gaussian noise and median filter to salt-and-pepper noise.

Figure 3. Noisy images without filtering.

We selected the results with regard to image Lena as the example which is illustrated in Fig.3 and the corresponding metrics are shown in Table I.

TABLE I. SPECIFIC METRICS

	MSE	PSNR	SSIM	NCC	NPCR
no noise	0.0000	N.A.	1.0000	1.0000	0.0000
Gaussian noise	0.0310	15.0862	0.7457	0.9540	0.9816
salt & pepper noise	0.0634	11.9762	0.6419	0.9144	0.9869
average filter 7x7	0.0012	29.1142	0.9754	0.9981	0.9250
crop 100x100 pixels	0.0007	31.6703	0.9923	0.9990	0.9441
rotate 5 degree	0.0025	26.0736	0.9664	0.9968	0.9681

TABLE II. SPECIFIC IMAGE METRICS USING TRADITIONAL FILTERS

	MSE	PSNR	SSIM	NCC	NPCR
no noise	0.0000	N.A.	1.0000	1.0000	0.0000
Gaussian noise	0.0007	31.8067	0.9874	0.9990	0.9265
salt & pepper noise	0.0005	32.7235	0.9889	0.9992	0.8788
average filter 7x7	0.0012	29.1142	0.9754	0.9981	0.9250
crop 100x100 pixels	0.0007	31.6703	0.9923	0.9990	0.9441
rotate 5 degree	0.0025	26.0736	0.9664	0.9968	0.9681

Thus, we apply filtering operations to the noisy images, the corresponding results are shown in Fig.4 and the relevant metrics are shown in Table II. From Table II, we see that for the metrics NPCR and NCC regarding to two types of noise, the optimization effects of image denoising are not obvious; but for the other three metrics, the effects have been improved greatly. SSIM has been raised from less than 0.8 to 0.95, PSNR has been increased more than twice of the original values and the MSE has been dropped obliviously.

Figure 4. Results after traditional filtering.

B. *Method-II: Using Linear CNN Model*

We used 10 color images to create the dataset for training the linear CNN model in Fig.5. We labeled each original image; for each image, we generated 10 images which include Gaussian noise and another 10 images which contain salt-and-pepper noise for use as the training set. We used each original image as the labeled image; for each image, we also generated 2 images with Gaussian noise and 2 images

with salt-and-pepper noise as the validation sets. During the network training, we have two training sets, each set includes 100 images corresponding to one type of noise; we have two validation sets, each set includes 20 images.

Figure 5. 10 color images are used in the network training.

For the training set with Gaussian noise, we train the linear CNN model for 5000 epochs of 100 training images; the number of total training steps is 10000. The result is illustrated in Fig.6. The *x*-axis shows the number of training steps and the *y*-axis indicates MSE. In our experiment, the batch size is 50, which means we trained 50 images for each step, and we calculated the MSE values between 50 trained images and their corresponding original images, and obtained the mean of 50 MSE values in Fig.6. In Fig.6, we see that the MSE values have converged within 1000 training steps and all MSE values are lower than 1.23×10^{-3}, which means the linear CNN model has satisfactory performance for filtering Gaussian noise. For all training steps, we also calculated the mean of MSE for both training set and validation set, the results show that the MSE of training set is very close to the MSE of the validation set; in other words, the linear CNN model has high feasibility and reliability.

In order to compare the results with traditional filtering methods, for each original image, we add Gaussian noise to each image; then, we apply three traditional methods and the linear CNN model to filter the noise; we use MSE to measure the results as shown in Table III. From Table III, we clearly see that our linear CNN model shows its lowest MSE for all of the images; in other words, it has better performance for filtering Gaussian noise compared to the three traditional ones.

TABLE III. RESULTS COMPARISONS FOR GAUSSIAN NOISE

'Fig\Method'	'original'	'gaussian'	'average filter'	'Wiener filter'	'median filter'	'LSCNN'	'Best'
'Fig.1'	[0]	[0.0093]	[0.0022]	[0.0023]	[0.0026]	[0.0012]	'LSCNN'
'Fig.2'	[0]	[0.0094]	[0.0035]	[0.0028]	[0.0040]	[0.0020]	'LSCNN'
'Fig.3'	[0]	[0.0090]	[0.0029]	[0.0026]	[0.0033]	[0.0014]	'LSCNN'
'Fig.4'	[0]	[0.0086]	[0.0021]	[0.0023]	[0.0026]	[0.0013]	'LSCNN'
'Fig.5'	[0]	[0.0087]	[0.0015]	[0.0018]	[0.0021]	[9.8307e-04]	'LSCNN'
'Fig.6'	[0]	[0.0082]	[0.0018]	[0.0020]	[0.0019]	[0.0011]	'LSCNN'
'Fig.7'	[0]	[0.0095]	[0.0028]	[0.0025]	[0.0032]	[0.0016]	'LSCNN'
'Fig.8'	[0]	[0.0096]	[0.0020]	[0.0019]	[0.0023]	[0.0011]	'LSCNN'
'Fig.9'	[0]	[0.0095]	[0.0017]	[0.0019]	[0.0021]	[0.0010]	'LSCNN'
'Fig.10'	[0]	[0.0089]	[0.0013]	[0.0018]	[0.0017]	[8.4823e-04]	'LSCNN'

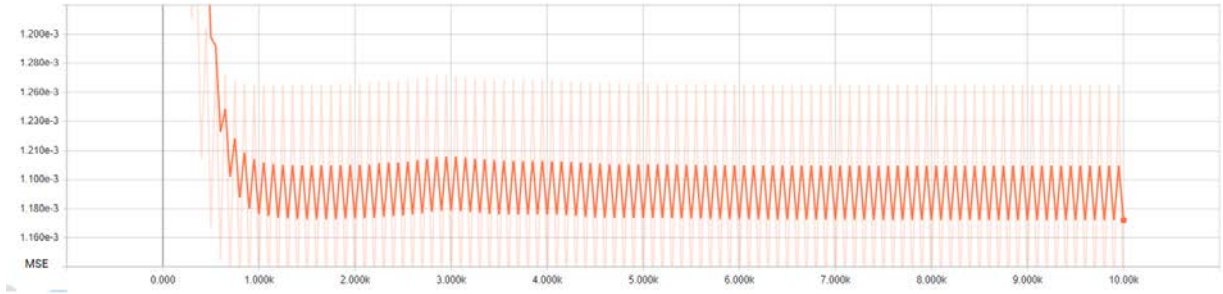

Figure 6. Results of network training for removing Gaussian noise.

TABLE IV. RESULTS COMPARISON FOR SALT-AND-PEPPER NOISE

'Fig\Method'	'original'	'salt & pepper'	'average filter'	'Wiener filter'	'median filter'	'LSCNN'	'Best'
'Fig.1'	[0]	[0.0157]	[0.0034]	[0.0042]	[0.0025]	[0.0017]	'LSCNN'
'Fig.2'	[0]	[0.0158]	[0.0060]	[0.0055]	[0.0050]	[0.0028]	'LSCNN'
'Fig.3'	[0]	[0.0166]	[0.0051]	[0.0052]	[0.0037]	[0.0021]	'LSCNN'
'Fig.4'	[0]	[0.0176]	[0.0036]	[0.0048]	[0.0023]	[0.0019]	'LSCNN'
'Fig.5'	[0]	[0.0182]	[0.0020]	[0.0040]	[0.0012]	[0.0016]	'median filter'
'Fig.6'	[0]	[0.0181]	[0.0024]	[0.0039]	[0.0011]	[0.0016]	'median filter'
'Fig.7'	[0]	[0.0149]	[0.0051]	[0.0044]	[0.0038]	[0.0022]	'LSCNN'
'Fig.8'	[0]	[0.0155]	[0.0027]	[0.0031]	[0.0011]	[0.0015]	'median filter'
'Fig.9'	[0]	[0.0150]	[0.0029]	[0.0032]	[0.0013]	[0.0013]	'median filter'
'Fig.10'	[0]	[0.0168]	[0.0012]	[0.0033]	[1.1400e-04]	[0.0011]	'median filter'

For the training set with salt-and-pepper noise, we also train the linear CNN model for 5000 epochs of 100 training images, the results are shown in Fig.7. We see that the MSE values have converged within 1000 training steps and all

MSE values are lower than 1.71×10^{-3} which reflect that the linear CNN model has satisfactory performance for filtering Gaussian noise as well in order to compare the results with traditional filtering methods. For each original image, we also generate one image with salt-and-pepper noise; then, we apply three traditional methods and the linear CNN model to filter the noise. We use MSE to measure the results shown in Table IV.

From Table IV, we see that the linear CNN model shows its better performance compared with average filter and Wiener filter with regard to salt-and-pepper noise. When compared with median filter, the linear CNN model works as same as this traditional method.

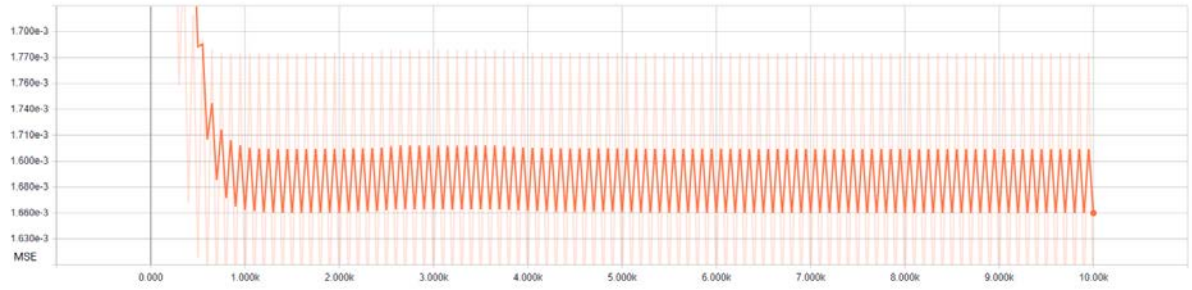

Figure 7. Results of training process for salt-and-pepper noise.

We applied the linear CNN model to image filtering, the corresponding results are shown in Fig.8. The image metrics are shown in Table V. From Table V, we see that the results using the linear CNN model are better than the ones by using traditional methods, compared with Table II.

Figure 8. Results of training process for salt-and-pepper noise.

TABLE V. SPECIFIC METRICS BY USING THE LINEAR CNN MODEL

	MSE	PSNR	SSIM	NCC	NPCR
no noise	0.0000	N.A.	1.0000	1.0000	0.0000
Gaussian noise	0.0004	34.0393	0.9911	0.9994	0.8816
salt & pepper noise	0.0001	39.1221	0.9975	0.9998	0.8429
average filter 7x7	0.0012	29.1142	0.9754	0.9981	0.9250
crop 100x100 pixels	0.0007	31.6703	0.9923	0.9990	0.9441
rotate 5 degree	0.0025	26.0736	0.9664	0.9968	0.9681

V. CONCLUSION AND FUTURE WORK

In this paper, we briefly introduced image denoising approaches based on a linear CNN model. From the experiments, we found that the filtering method based on the linear CNN model shows the best performance for removing Gaussian noise; for salt-and-pepper noise, the linear CNN model shows its better performance than two of traditional filters as well as the same performance to median filtering. The linear CNN model can clearly improve the performance of traditional image filters. Our future work will be on improving the CNN model to get better filtering results [12][13].

REFERENCES

[1] P. Milanfar, "A tour of modern image filtering: New insights and methods,both practical and theoretical," *IEEE Signal Processing (Magazine)*,vol. 30, no. 1, pp. 106–128, 2013.

[2] M. R. Banham and A. K. Katsaggelos, "Digital image restoration," *IEEE Signal Processing (Magazine)*, vol. 14, no. 2, pp. 24–41, 1997.

[3] I. Pitas and A. N. Venetsanopoulos, *Nonlinear Digital Filters: Principles and Applications*. Springer Science & Business Media, 2013.

[4] S. He, X. L. Pan, and Y. M. Li, "Optimization algorithmfor average filtering," *Information Technology*, vol. 3, p. 41, 2012.

[5] G. Changlai, "Image-denoising method based on wavelet transform and mean filtering," *Opto-Electronic Engineering*, vol. 1, p. 19, 2007.

[6] Z. J. Wang, C. W. Qv, and L. Cui, "Images denoising with wiener filter in directionalet domain," *Electronics Optics & Control*, vol. 6, p. 8, 2007.

[7] Y. Ming and S. Li-hua, "The application of an improved fast algorithm of median filter on removing image noise," *Engineering of Surveying and Mapping*, vol. 20, no. 3, pp. 65–69, 2011.

[8] F. Xu, J. G. Lu, and Y. X. Sun, "Application of neural network in image processing," *Information and Control*, vol. 32, no. 4, pp. 344–351, 2003.

[9] P. Y. Chen, C. Y. Lien, and H. M. Chuang, "A low-cost VLSI implementation for efficient removal of impulse noise," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 3, pp. 473–481, 2010.

[10] H. P. Yan and Y. H. Wu, "Filtering image impulse noise by using a PCNN image noise reduction technique," *CAAI Transactions on Intelligent Systems*, vol. 12, no. 2, pp. 272–278, 2017.

[11] T. Huang, G. Yang, and G. Tang, "A fast two-dimensional median filtering algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 1, pp. 13–18, 1979.

[12] Z. Liu, *Comparative evaluations of image encryption algorithms* (Masters Thesis), Auckland University of Technology, New Zealand, 2018.

[13] D. Shen, C. Xin, M. Nguyen, W. Yan, "Flame detection using deep learning," *IEEE ICCAR 2018*.