# Non-local mean filtering algorithm based on deep learning

Baozhong LIU[1,2a]and Jianbin LIU[1,2]

[1]Computer School, Beijing Information Science &Technology University ,Beijing 100101 ,China
[2]Software Engineering Research Center, Beijing Information Science &Technology University ,Beijing 100101 ,China

**Abstract.** Aimed at the problem that the traditional image denoising algorithm is not effective in noise reduction, a new image denoising method is proposed. The method combines deep learning and non-local mean filtering algorithms to denoise the noisy image to obtain better noise reduction effect. By comparing with Gaussian filtering algorithm, median filtering algorithm, bilateral filtering algorithm and early non-local mean filtering algorithm, the noise reduction effect of the new algorithm is better than the traditional method and the peak signal to noise ratio is compared with the early non-local mean algorithm. The performance is better.

## 1 Introduction

Non-local mean (NLM) filtering algorithm[1], the basic idea is that the ultrasound image contains a lot of redundant information, as there are many similar image blocks but these similar image blocks are distributed over the whole image. Their positions may be quite different but the grayscale information is similar. The NLM filtering method fully exploits the similarity of the images based on the existence of the above features. Firstly the similarity between these similar blocks and the image block where the current noise is located is calculated, and secondly the weighted average is used to recover the value of the pixel to be restored. Non-local mean filtering is one of the better noise reduction algorithms in recent years. Many researchers have improved it. Although the improved filtering algorithm is better than the previous filtering effect, there is no big change in the improved thinking. Many papers are improved on the attenuation parameter $h$ that controls the degree of filtering. This paper will jump out the idea of improving the weighted average, exponential function and parameter $h$ and use the pixel value and similarity value of all adjacent points of the noisy image as the input of the deep learning network. The original image is trained as an output and the trained model can be directly used for filtering[2].

## 2 Traditional nlm filtering algorithm

The principle of the NLM algorithm is as follows. As shown in Figure 1, the figure contains three pixel points $P$ , $Y1$ and $Y2$ their respective field. It can be seen that the pixel points $P$ and $Y1$ have similar field structures while the field of $P$ and $Y2$ are similar. The property is very small then $Y2$ similarity weight $\omega(P,Y2)$ is larger than $Y1$ similarity weight $\omega(P,Y1)$ hen filtering the $P$ point pixel. The final recovery result of the pixel point $P$ can be obtained by searching all the pixels in the entire image having similar field with the pixel $P$ and then weighting the similarity of the obtained similarities[3].

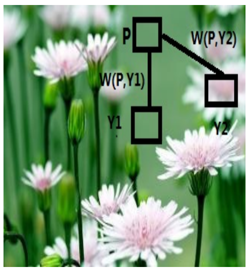

**Figure 1.** NLM filtering algorithm principle

The NLM filtering algorithm performs image denoising to calculate the similarity of pixel points in units of image blocks. A so-called image block is a square field center on a certain pixel point. Let the gray scale value of the contaminated image at pixel $i$ be $v(i)$, and the filtered gray scale estimate be $NL(v)(i)$. For any pixel $i$ , the filtered $NL(v)(i)$ can be obtained by calculating the weighted average of the pixels with similar field in the entire noise image

$$NL(v)(i) = \sum_{j\in I}\omega(i,j)v(j) \qquad (1)$$

Where $I$ is the entire image space, the weight coefficient $\omega(i,j)$ is the degree of influence of pixel $j$ on pixel $i$ , as shown below

$$\omega(i,j) = \frac{1}{C(i)}e^{-\frac{\left\|v(N_i)-v(N_j)\right\|^2_{2,\alpha}}{h^2}} \qquad (2)$$

[a] Corresponding author: 13651176051@163.com

Where $N_i$ is the reference block which also referred to as the image block of the pixel to be restored. $N_j$ is a similar block which is the same size as $N_i$ ; $v(N_i)$ is the gray value of each pixel in the reference block; $v(N_j)$ is similar the gray value of each pixel in the block; $\left\| v(N_i) - v(N_j) \right\|_{2,\alpha}^2$ is the Gaussian weighted euclidean distance; $\alpha$ is the standard deviation of the Gaussian kernel; $h$ is the attenuation parameter for controlling the degree of filtering; $C(i) = \sum_{j \in I} e^{-\frac{\left\| v(N_i) - v(N_j) \right\|_{2,\alpha}^2}{h^2}}$ is the normalized parameter. $\omega(i,j)$ satisfies $\omega(i,j) \in [0,1]$ and $\sum_{j \in I} \omega(i,j) = 1$.

## 3 Deep learning based nlm filtering

### 3.1 Denoising frame

The idea of the NLM filtering algorithm is to fully exploit the information of similar image blocks in the image, use the image block to express the characteristics of the pixel points, estimate the similarity between the pixels by the similarity between similar blocks and then recover the value of the pixel to-be recovered by weighted average. Because the neural network and deep learning models have strong expressive power, they are very suitable for learning the function of picture block to picture block. First, a clean picture block $y$ is randomly selected from the picture data set and then a corresponding noise picture block $x$ is generated by artificially adding Gaussian white noise noise. Then we use the vectorized noise picture block $x$ as the input of the neural network and use the corresponding vectorized clean picture block as the output of the neural network, update the parameters of the neural network by means of backward propagation and then iteratively gradually learn the required model. $x$ represents the observed noise picture, $y$ represents the original clean and noiseless picture then the process of noise pollution can be described as

$$x = \rho(y) \qquad (3)$$

In the formula : $R^{m \times n} \to R^{m \times n}$ is a process of randomly adding noise, $m$ and $n$ are the length and width of the digital image respectively. Then the task of noise reduction is to seek a function $f$ to satisfy

$$f = argmin_f E_y f(x) - y_2^2 \qquad (4)$$

In the formula, $f$ is a function of $R^{m \times n} \to R^{m \times n}$ . It can be known from equations (3) and (4) that the purpose of denoising is to find a function $f$ as close as possible to $\rho^{-1}$ . The image-to-image mapping complexity is too high for $R^{m \times n} \to R^{m \times n}$ and it is difficult to learn $f$ directly for machine tens of thousands of pixels. So in practice we need to split a picture into several smaller picture blocks and learn the mapping $g$ from the picture block to the picture block. All the picture blocks are denoised by such a mapping $g$ , respectively. All of these picture blocks are re-aggregated in some way to form a picture after denoising, as shown below

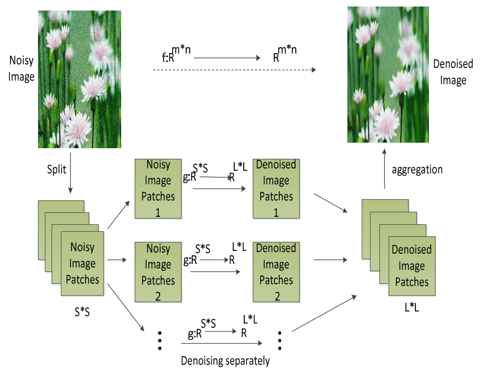

**Figure 2.** Image block based image denoising

In this way, we reduce the size of the problem and indirectly learn the image-to-image mapping $f$ by learning the mapping $g$ of image blocks to image blocks, making machine learning easier to learn $f$ .The framework for denoising based on deep learning is as follows

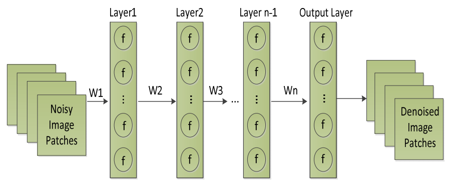

**Figure 3.** Image denoising frame

### 3.2 Image block splitting and aggregation

The most conservative method of splitting in this paper is to take out all the image blocks and then proceed to the subsequent process. Usually, the method of moving the window is used and the window moves in the picture at a specific step size and the image block in the window is taken out each time. The size of the step size is determined according to the size of the selected noise image block. We estimate the number of estimates for each target pixel point based on the side length $S$ of the noise picture in Figure 2 and the side length $L$ of the model denoised picture. In this paper, a 512 × 512 image is used. The side length of the image block before and after denoising is taken as 8 steps of 2[4].

For the process of re-aggregation, that is a pixel on the noise picture appears in multiple noise image blocks and denoised and how to obtain the final estimated value after multiple evaluations is obtained. In this paper, the averaging process is used to solve this problem. The average method usually selected in the process of re-aggregation is to directly calculate the arithmetic average, calculate the weighted average and calculate the Gaussian average. This paper uses the method of calculating the weighted average to calculate the final estimate. The reason for choosing to use it for calculation is that although one pixel on the noise picture is selected in many image blocks, the position appearing in each noise image block is different. According to Figure 2, the expected mean square error of the pixels appearing in the middle portion of the noise image block is smaller. Therefore, this paper adopts the method of weighted averaging, firstly the mean square error of each position of the image block is calculated and the weight of the estimated value is inversely proportional to the mean square error of the position in the image block.

$$p_{i,j}=\frac{\sum_{k=1}^{N}\frac{p_k^2}{e_{(x_k,y_k)}}}{\sum_{k=1}^{N}\frac{1}{e_{(x_k,y_k)}}} \quad (5)$$

Where $p_{i,j}$ represents the estimated pixel value of the noise picture after denoising on $i$ and $N$ represents the number of times a pixel appears in the picture block and $k$ represents the occurrence in the kth denoised picture block $(x_k, y_k)$ represents the position of the pixel in the $k$ th denoised picture block and $e_{(x_k,y_k)}$ represents the mean squared error of each position obtained by statistics.

## 3.3 Introduction to neural networks

Neurons are the basic building blocks of neural networks. An example of a neuron with four input parameters is shown in Figure 4. This neuron takes four input parameters $x_1$ , $x_2$ , $x_3$ , $x_4$ and intercept +1 as input values, and the output is

$$h_{w,b}(x)=f\left(w^T x\right)=f\left(\sum_{i=1}^{4}w_i x_i+b\right) \quad (6)$$

Where $w$ and $b$ are the parameters of the neuron corresponding to the weight of each input $f:R\rightarrow R$ is called the activation function. In this paper, the Rectified Linear Unit (ReLU) is used as the activation function. The neural network is to combine multiple neurons together. The input of some neurons is the output of other neurons to form a network, as shown in Figure 5. Here a circle is used to indicate the input of the neural network and a circle labeled "+1" is called a bias node. The leftmost layer of the network is called the input layer, the rightmost layer is called the output layer and the middle layer is called the hidden layer. In the figure 5, there are three input units (not counting units), two hidden layers and two output units. We use the back propagation algorithm to update the weight parameters of the adjusted neural network.

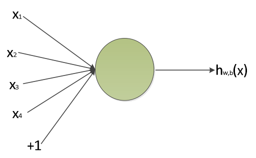

**Figure 4.** Single neuron

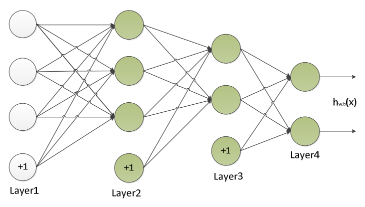

**Figure 5.** Neural Networks

## 3.4 Activation function

The ReLU[5] function formula is

$$f(x)=\begin{cases} x & if\ x>0 \\ zero & if\ x\le 0 \end{cases} \quad (7)$$

The ReLU function is a piecewise linear function. All negative values become zero and the positive value is unchanged. This operation is called unilateral suppression. Unilateral inhibition results in sparse activation of neurons in the neural network. Especially in the deep neural network model when the model increases the $N$ layer, the activation rate of ReLU neurons will be reduced by $2^N$ times.

## 3.5 Parameter setting and training method

a) Loss function

The experiments in this paper are based on the keras framework and the underlying backend of keras relies on TensorFlow. The loss function is one of the two parameters required to compile the keras model. In this paper, the mean square logarithm error is used as the loss function[6].

$$\varepsilon=\frac{1}{n}\sum_{i=1}^{n}\left(\log(p_i+1)-\log(a_i+1)\right)^2 \quad (8)$$

Where $n$ is the observed value of the entire data set, $p_i$ is the predicted value and $a_i$ is the true value.

b) Optimizer

The main advantage of Adam that after the offset correction, each iteration learning rate has a certain range which makes the parameters relatively stable[7]. The formula is as follows

$$m_t=\beta_1 m_{t-1}+(1-\beta_1)g_t \quad (9)$$

$$v_t=\beta_1 v_{t-1}+(1-\beta_1)g_t^2 \quad (10)$$

$$\hat{m}_t=\frac{m_t}{1-\beta_1^t} \quad (11)$$

$$\hat{v}_t=\frac{v_t}{1-\beta_2^t} \quad (12)$$

$$\varnothing_{t+1}=\varnothing_t-\frac{\alpha}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t \quad (13)$$

Where $m_t$ and $v_t$ are first-order moment estimates and second-order moment estimates for the gradient, respectively which can be considered as approximations to the expected $E[g_t]$ and $E[g_t^2]$ ; $\hat{m}_t$ and $\hat{v}_t$ are respectively for $m_t$ and $v_t$ The correction can be approximated as an unbiased estimate of the expectation.

According to the advice of the Adam algorithm proposer, the default value of $\beta_1$ is 0.9, the default value of $\beta_2$ is 0.999 and the default value of $\epsilon$ is $10^{-8}$.

c) Learning rate

A large value of the learning rate may make the neural network model learn faster and may also cause the neural network to diverge. After the experiment, the learning rate is set to 0.1.

d)Training method

This paper uses the gradient descent algorithm model to train in small batches. Small batch processing. Small batch processing is between the batch algorithm and the random gradient drop. The small batch processing algorithm selects fixed $n$ sample points from the training data set each time calculates the total error of the small batch of samples for the derivative of the weight matrix and then performs weight update, and each time after sweeping all sample points is called a cycle. The stochastic gradient descent algorithm can be regarded as a small batch processing algorithm with $n=1$ and the batch processing algorithm can be regarded as a small batch processing algorithm with $n=N$ ( $N$ is the number of training data). Compared with batch processing algorithms small batch processing has the advantages of fast update weight, online learning, less resources required and randomness to leave undesired local extreme points. Compared with random gradient reduction small batch processing. The error curve is more stable during training and is less affected by noise in the data. Considering that the current equipment has done a lot of optimizations on matrix operations, the speed of updating one weight of one sample point at a time and the weighting of one sample point at a time may be similar. Therefore, the training method of small batch processing has also been widely used in recent years. Based on the above considerations, in the work of this paper, the method of small batch processing is adopted and the parameter value is set to 128.

## 4 Evaluation index

The effect of image noise reduction can be evaluated by commonly used objective evaluation indicators. In this paper, Peak Signal to Noise Ratio (PSNR) is chosen to evaluate the performance index of the algorithm[8].

$$PSNR = 10\lg\left(\frac{255^2 \times M \times N}{\sum_{i=1}^{Row}\sum_{j=1}^{Col}\left[f(i,j)-g(i,j)\right]^2}\right) \quad (14)$$

Where $f(i,j)$ is the original image; $g(i,j)$ is the denoised image; $M$ and $N$ are the row and column, respectively. The larger the value of PSNR the better the filtering effect that is the smaller the distortion of the filtered image.

## 5 Experiment analysis

The size of the search area and the similar window in the NLM filtering algorithm has a great influence on the denoising effect. This paper reviews the relevant literature and tests it to determine that the selected search area is 8×8 and the similar window size is 4×4. This article uses 200 flowers as a dataset with a minimum image resolution of 1024 × 768 and a maximum image resolution of 1600 × 1200. The reason for selecting a flower picture set is that neural network based image denoising belongs to a denoising algorithm that learns the statistical characteristics of images. Selecting a specific type of image set can reduce the number of patterns that the network model needs to learn to a certain extent so that we can get better denoising effect for a specific type of image with a smaller network. If you need to get a more versatile model increase the type and size of the training data set and expand the network size. We used 180 of them for training data and 20 of them for testing.

For each picture in the training set, we extract 2000 image blocks so a total of 360000 image blocks are used as training samples. For the images in the test data set, we also select 2000 image blocks for each image, for a total of 40000 image blocks for verification set. The number of cycles for training iterations is set to 500. Experience has shown that this iteration number allows the network to converge to a nice local minimum for this simple structure. In this paper, we choose sigma = 20, sigma = 40, sigma = 60 to compare our model with Gaussian filtering algorithm, median filtering algorithm, bilateral filtering and traditional non-local mean filtering algorithm. The PSNR values of some test images of the three noise environments are listed in Table 1, Table 2 and Table 3.

**Table 1.** Denoisng result of different methods for sigma = 20 (PSNR/dB)

| Algorithm / Image | Noise image | Gaussian | Median | Bilateral | NLM | Algorithm |
|---|---|---|---|---|---|---|
| Img1 | 22.52 | 28.30 | 31.44 | 29.87 | **33.90** | 33.63 |
| Img2 | 24.57 | 31.79 | 32.84 | 33.1 | **34.33** | 32.76 |
| Img3 | 24.54 | 30.01 | **33.24** | 32.89 | 33.10 | 33.10 |
| Img4 | 23.63 | 29.31 | 31.96 | 31.51 | 33.70 | **33.92** |
| Img5 | 24.91 | 31.79 | **34.19** | 33.79 | 33.08 | 33.56 |
| Img6 | 24.14 | 31.19 | 33.67 | 32.31 | **34.29** | 32.13 |
| Img7 | 22.57 | 30.26 | 31.31 | 30.27 | 32.64 | **33.29** |
| Img8 | 24.40 | 32.27 | 33.32 | 33.09 | **34.00** | 32.87 |
| Img9 | 24.02 | 27.52 | 32.10 | 31.96 | **33.15** | 31.90 |
| Img10 | 22.96 | 31.38 | 31.07 | 31.30 | **36.35** | 34.60 |

Note: The data corresponding to the bold font is the best value for each indicator.

**Table 2.**Denoisng result of different methods for sigma = 40 (PSNR/dB)

| Algorithm / Image | Noise image | Gaussian | Median | Bilateral | NLM | Algorithm |
|---|---|---|---|---|---|---|
| Img1 | 16.93 | 27.45 | 26.74 | 21.98 | **30.35** | 30.13 |
| Img2 | 18.64 | 27.96 | 29.50 | 24.40 | 28.71 | **30.98** |
| Img3 | 18.66 | 27.11 | 29.12 | 24.42 | 28.21 | **29.73** |
| Img4 | 17.88 | 27.23 | 27.75 | 22.94 | **28.91** | 28.74 |
| Img5 | 18.95 | 27.76 | 29.62 | 25.03 | 28.41 | **30.65** |
| Img6 | 18.31 | 27.98 | 28.82 | 23.69 | 28.97 | **29.70** |
| Img7 | 17.22 | 29.12 | 26.77 | 21.73 | **31.11** | 30.22 |
| Img8 | 18.46 | 28.24 | 29.27 | 24.06 | **29.96** | 29.78 |
| Img9 | 18.25 | 25.77 | 28.09 | 23.63 | 28.28 | **29.80** |
| Img10 | 17.52 | 29.51 | 26.92 | 22.20 | **31.67** | 31.20 |

Note: The data corresponding to the bold font is the best value for each indicator.

**Table 3.**Denoisng result of different methods for sigma = 60 (PSNR/dB)

| Algorithm / Image | Noise image | Gaussian | Median | Bilateral | NLM | Algorithm |
|---|---|---|---|---|---|---|
| Img1 | 13.87 | 26.24 | 23.71 | 17.47 | 27.71 | **29.10** |
| Img2 | 15.22 | 25.00 | 26.19 | 19.79 | 28.33 | **30.55** |
| Img3 | 15.27 | 24.50 | 26.06 | 19.83 | 27.00 | **29.03** |
| Img4 | 14.63 | 25.01 | 24.82 | 18.83 | 25.71 | **27.83** |
| Img5 | 15.47 | 24.73 | 26.46 | 20.19 | 27.03 | **28.31** |
| Img6 | 14.98 | 25.23 | 25.65 | 19.35 | 25.90 | **27.36** |
| Img7 | 14.21 | 27.31 | 23.86 | 18.03 | 28.16 | **30.82** |
| Img8 | 15.06 | 25.21 | 26.01 | 19.50 | 25.48 | **28.02** |
| Img9 | 14.93 | 23.82 | 25.23 | 19.29 | 25.95 | **28.39** |
| Img10 | 14.45 | 27.31 | 24.05 | 18.30 | 28.17 | **30.52** |

Note: The data corresponding to the bold font is the best value for each indicator.

The NLM filtering algorithm exhibits a good denoising effect when sigma=20. In the test picture, the algorithm of this paper is better than the NLM filtering algorithm in only 4 of them. The algorithm of this paper is superior to the NLM filtering algorithm among the 13 images in sigma=40. At sigma=60, only one image in this paper is inferior to the NLM filtering algorithm. It can be seen that with the increase of noise intensity, the denoising effect of the proposed algorithm which is a good choice for denoising in high noise environment is better. For low-noise situations a larger training set is needed to train the network model for better results. For example, the larger multi-layer perceptron network trained in [9] also has good performance in low noise.

## 6 Conclusion

This paper analyzes the image denoising problem from the perspective of function and introduces the denoising idea based on image block denoising from the macro level. This article describes the framework for denoising using deep learning and discusses some of the parameter settings to be aware of when using this algorithm. Unfortunately there is not enough image smoothing. Although the peak signal-to-noise ratio of the image is very high after denoising, there is still some room for improvement in the processing of the background. This paper only verifies that the non-local mean algorithm combined with deep learning is effective and has better denoising effect than the traditional non-local mean algorithm. However, the training data set used in this experiment is limited if you want better denoising. The effect also requires more training data to train the network model.

## Acknowledgment

This work was supported by the Information+ Discipline Construction Project (5111823414) and the Science Research Level Improvement Project (5211823406) of Beijing Information Science & Technology University.

## References

1. Buades A, Coll B, Morel J M. A non-local algorithm for image denoising[C]// Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. IEEE, 2005:60-65 vol. 2.
2. Goossens B, Luong H Q. A fast non-local image denoising algorithm[J]. 2008, 6812:81210-81210.
3. Yan Nana. Research on Degraded Image Restoration Technology Based on Non-local Mean [D]. Qinhuangdao City, Hebei Province: Yanshan University, 2011.
4. Levin A, Nadler B, Durand F, et al. Patch complexity, finite pixel correlations and optimal denoising[C]// European Conference on Computer Vision. Springer, Berlin, Heidelberg, 2012:73-86.
5. umelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors.[J]. 1986, 323(6088):399-421.
6. Su Meihong. Research on loss function problem in machine learning [D]. Northwest University, 2015.
7. Ruder S. An overview of gradient descent optimization algorithms[J]. 2016.
8. Hore A, Ziou D. Image quality metrics: PSNR vs. SSIM[C]// International Conference on Pattern Recognition. IEEE, 2010:2366-2369.Hore A, Ziou D. Image Quality Metrics: PSNR vs. SSIM[C]// International Conference on Pattern Recognition. IEEE, 2010:2366-2369.
9. Harmeling S. Image denoising: Can plain neural networks compete with BM3D?[C]// IEEE Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, 2012:2392-2399.