BY USING NEURAL NETWORK WITH RADIAL BASIS FUNCTION FOR NEURAL FILTER DESIGN

Hung-Ching Lu  Ta-Hsiung Hung  Shian-Tang Tzeng
Department of Electrical Engineering
Tatung University
40 Chungshan North Road, 3rd section,
Taipei, 10451, Taiwan, Republic of China

Abstract: This proposed neural filter tasks full advantage of the architecture of the neural networks with radial basis functions (RBF) to deal with a series of corrupted images in this paper. There are a few neurons in the hidden layer to recognize the desired signal and get rid of undesired noise. The compared results show that the performance of this neural filter is much better than that of other widely used noises such as Gaussian white noise, salt and pepper noise, speckle noise, blurred image, and blur with above mentioned noises.

Key-Words: neural filter, radial basis function, neural network.

1 Introduction
The field of neural networks has a history of some five decades but has found solid application only in the past twelve years, and it is still developing rapidly. Neural networks, composed of many simple elements operating in parallel, have been trained to perform complex functions in a wide variety of fields of applications including pattern recognition, identification, classification, speech, vision, and control system [1]-[6]. With the rapid industrial development, the applications of neural networks are getting more and more important. Today neural networks can be trained to deal with problems that are difficult for conventional computers or human beings. However, there have been several recent works concerning the representation capabilities of multilayer feedforward neural networks [7]-[11].

The most widely used method, backpropagation learning algorithm, in the neural networks [12] was created by generalizing the Widrow-Hoff learning rule to multiple-layers networks and nonlinear differentiable transfer functions. And networks with basis, a sigmoid layer and a linear output layer, are capable of approximating any function with a finite number of discontinuities. Nevertheless, there exist a disadvantage in this learning algorithm that it has a slow learning speed and is apt to drop into the local error minimum, especially there are a lot of input/target pairs. The local error minimum that has been found may be satisfactory, but if it is not, a network with more neurons may do a better job. However, the number of neurons or layers to add not be obvious. Alternatively, we might run the problem using several different sets of initial conditions to see if they led to the same or different solutions. It goes without saying that this is a time-consuming process to do the same work. Since the simple backpropagation learning algorithm has been rarely used to deal with real problems in the word, we make use of radial basis function to handle this problems. Next, we will take full advantage of the architecture, introduced in Section 2, of radial basis network to design the neural filter because the network presents much better performance and execution speed. In Section 3, we employ the column processing method to a column vector. This proposed filter and the simulation results will be shown at the end of Section 4. At last, there is conclusion given in Section 5.

2 Radial Basis Networks
Advantages of radial basis function (RBF) networks over multilayer perceptions trained with the backpropagation algorithm include a more straightforward training process and a simpler network structure. They work best when many training vectors are available [13]-[14]. From the point of view, the radial basis networks are chosen instead of the backpropagation networks.

2.1 Network Architecture
A RBF network consists of an input layer of source nodes, a single hidden layer of nonlinear processing units, and an output layer of linear weights as depicted in Fig.1. Radial basis network can fit any function well with a finite number of discontinuities [15]-[20].

2.2 Radial Basis Function
The heart of a RBF network is the hidden layer that is defined by a set of radial basis functions. In this paper, we will confine our discussion to the use of Gaussian function on account of the most commonly used in practice. Using the terminology of Fig.1, we may describe input-output mapping performed by the RBF network as follows:

\[ y_i = \sum_{k=1}^{S_1} w_{ik} \varphi(u; t_k) + b_i, i = 1, 2, \ldots, S_2 \quad (1) \]

where \( S_1 \) and \( S_2 \) are the total number of neurons of the hidden layer and the output layer, respectively. The term \( \varphi(u; t_k) \) is the \( k \) th radial basis function that computes the “distance” between an input vector \( u \) and its own center \( t_k \); the output signal produced by the \( k \) th hidden units is a nonlinear function of the distance. The scaling factor \( w_{ik} \) in (1) represents a weight that connects the \( k \) th hidden node to the \( i \) th output node of the network. Finally the constant term \( b_i \) represents a bias. The \( \varphi(u; t_k) \) is defined by

\[ \varphi(u; t_k) = \exp \left( -\frac{1}{\sigma_k} \left\| u - t_k \right\|^2 \right), k = 1, 2, \ldots, S_1 \quad (2) \]

where \( t_k \) is the center, \( \sigma_k \) is the width, and \( \left\| u - t_k \right\| \) denoted the Euclidean distance between \( u \) and \( t_k \). Thus, substituting (2) into (1), we may formulate the input-output mapping realized by a Gaussian RBF network as follows:

\[ y_i = \sum_{k=1}^{S_1} w_{ik} \exp \left( -\frac{1}{\sigma_k} \left\| u - t_k \right\|^2 \right) + b_i, i = 1, \ldots, S_2 \quad (3) \]

Thus, the requirement is to select suitable values for the parameters of each of the \( S_1 \) Gaussian radial basis functions, namely \( \sigma_k \) and \( t_k \), \( k = 1, 2, \ldots, S_1 \), and solve for the weights of the output layer. Next, we will make use of the stochastic gradient approach for the design of a RBF network in this paper.

2.3 Stochastic Gradient Approach
The centers of the radial basis functions and all other free parameters of the network undergo a supervised learning process. The first step in the development of this supervised learning procedure is to define the instantaneous value of the cost function

\[ E(n) = \sum_{i=1}^{S_2} e_i(n)^2 \quad (4) \]

where \( e_i(n) \) is the error signal defined to be the difference between some desired response and the actual output of the network. Specifically, for the \( i \) th output neuron we may write

\[ e_i(n) = d_i - y_i(n) \quad (5) \]

where \( d_i \) is the desired response at the \( i \) th node of the output layer, \( y_i(n) \) is the output at the \( i \) th node of the output layer, and \( n \) refers to the number of iterations of the algorithm. Now, the free parameters that minimize \( E(n) \). The results of a stochastic gradient procedure aimed at this

The update rules for \( t_k(n) \), \( \sigma_k^2(n) \), and \( w_{ik}(n) \) are assigned different learning rate parameters \( \mu_\sigma \), \( \mu_\mu \), and \( \mu_w \), respectively. For the initialization of the stochastic gradient algorithm, the free parameters of the RBF network may be assigned a subset of values drawn at random from the training set in order to reduce the probability of converging to an undesirable local minimum on the error surface.
3 Column Processing
It is necessary for us to reformat an image into a column vector if we want to copy with the image with this proposed method. For the sake of convenience, there is a figure to illustrate the process of converting an image into a column vector as depicted in Fig.2. In order to obtain the adequate weights and biases, the column vector can be sent into the radial basis network throughout training. When the training is completed, the network is capable of dealing with the corrupted images. On the other hand, we also have to rearrange the handled column vector to form an image by means of the reverse direction. The handled image can be compared with the original one to see if the corrupted image is improved.

4 SIMULATION RESULTS
In this section we will employ this proposed method to design a high performance neural filter with radial basis network as mentioned previously. The initial weights and biases are chosen arbitrarily from the training set and these parameters are adjusted based on the observations of the corrupted image. In order to enhance the ability of copying with corrupted image, we have to choose two data sets for the training examples. One is the original image data without any noise and the other is the original one added Gaussian white noise with zero mean and 0.02 variance. These two data sets will be acted as the training examples [21]. There are some widely used noise images acted as the test image such as Gaussian white noise with zero mean and 0.05 variance, salt and pepper noise with 0.05 noise density, and speckle noise with 0.05 noise variance, blurred image, and blur with above mentioned noises. When the handling process of radial basis network has been finished, the handled column vector need to be rearranged to form an image. Thus, we are capable of comparing the corrupted image with the handled one to see whether the latter is better than the former or not.

Example:
The original size of considered image is 256×256. To begin with, we will make use of the column processing method as mentioned previously to convert the image into a column vector. The mean square error (MSE) and the mean absolute error (MAE) are indexes to measure the performance of each filter in this paper. In other words, the lower in the MSE value or MAE value, the better is the performance of filter. The definitions of MSE and MAE are given in the following:

\[
\text{MSE} = \frac{1}{M \times N} \left( \sum_{i=1}^{M} \sum_{j=1}^{N} \left( \frac{\text{handled image}[i][j]}{\text{original image}[i][j]} \right)^2 \right)
\]

\[
\text{MAE} = \frac{1}{M \times N} \left( \sum_{i=1}^{M} \sum_{j=1}^{N} \left| \frac{\text{handled image}[i][j]}{\text{original image}[i][j]} \right| \right)
\]

When the adequate weights and biases are determined by means of the learning algorithm as described in this paper, the completely trained neural filter has the ability to deal with the corrupted images. In this example, there are 512 neurons in the hidden layer after 3,000 iterations and 65,536 neurons in both input layer and output layer. The learning rate parameters, \(\mu_l\), \(\mu_s\), and \(\mu_w\), are assigned to be 0.01, 0.05, and 0.1, respectively. The test image with Gaussian white noise with zero mean and 0.05 variance. This corrupted image coped with neural filter, 3×3 adaptive Wiener filter, and 3×3 median filter, respectively. We know that the median filter is able to reduce the “salt and pepper” type noise [22]. The handled image of using median filter is better than that of using Wiener filter. Therefore, we have to compute the MSE or MAE to see which one the performance is better. After the computation, the performance of using proposed filter is better than that of using median filter. The computation results are shown in Table 2 (MSE) and Table 3 (MAE) with these three filters. From the above simulation results, this proposed filter is capable of dealing with another noise type well even though the noise were not seen before. This proposed filter does a better job.

5 Conclusion
In order to cope with the corrupted images, the handled images of using neural filter are much
better than that of using $3\times3$ adaptive Wiener filter and $3\times3$ median filter. It is difficult to deal with the blurred image by means of the Wiener filter and median filter. But the blurred images can be handled well by taking advantage of this proposed filter in this paper. On the other hand, blur with above mentioned three type noises can be also handled very well by means of the proposed filter. Therefore, the handled images show the potential ability of neural filter from the above simulation results.

Acknowledgment
This work was supported by the National Science Council of Republic of China, under contract NSC 89-2213-E-036-023.

References


---

Fig. 1 Radial basis neural network architecture

Fig. 2 Convert image into column vector
Table 1 Summary of the Stochastic Gradient Algorithm for the Design of RBF Networks

\[ y_i(n) = \sum_{k=1}^{S_i} w_{ik}(n) \phi(u(n); t_k(n)) + b_i \quad i = 1, 2, \cdots, S_2 \]
\[ e_i(n) = d_i - y_i(n) \]
\[ w_{ik}(n+1) = w_{ik}(n) \mu_{ik} e_i(n) \phi(u(n); t_k(n)) \]
\[ t_k(n+1) = t_k(n) + 2\mu_{ik} e_i(n) w_{ik}(n) \phi(u(n); t_k(n)) \frac{u(n) - t_k(n)}{\sigma_k^2(n)} \]
\[ \sigma_k^2(n+1) = \sigma_k^2(n) + \mu_{ik} e_i(n) w_{ik}(n) \phi(u(n); t_k(n)) \left\| u(n) - t_k(n) \right\|^2 \]

where
\[ \phi(u(n); t_k(n)) = \exp \left( - \frac{1}{\sigma_k^2(n)} \left\| u(n) - t_k(n) \right\|^2 \right) \]

Table 2 Mean Square Error

<table>
<thead>
<tr>
<th>Noise Type</th>
<th>Coped with Neural Filter</th>
<th>Coped with 3×3 Wiener Filter</th>
<th>Coped with 3×3 Median Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>9.1514</td>
<td>539.6330</td>
<td>615.3732</td>
</tr>
<tr>
<td>Salt &amp; Pepper</td>
<td>3.1208</td>
<td>469.4863</td>
<td>33.9054</td>
</tr>
<tr>
<td>Speckle</td>
<td>0.1537</td>
<td>74.8709</td>
<td>112.5793</td>
</tr>
<tr>
<td>Blur</td>
<td>64.8365</td>
<td>301.6153</td>
<td>295.3122</td>
</tr>
<tr>
<td>Blur with Gaussian</td>
<td>108.1380</td>
<td>827.6224</td>
<td>869.6763</td>
</tr>
<tr>
<td>Blur with Salt &amp; Pepper</td>
<td>95.3332</td>
<td>777.9419</td>
<td>300.1442</td>
</tr>
<tr>
<td>Blur with Speckle</td>
<td>67.2211</td>
<td>359.6611</td>
<td>348.1799</td>
</tr>
</tbody>
</table>

Table 3 Mean Absolute Error

<table>
<thead>
<tr>
<th>Noise Type</th>
<th>Coped with Neural Filter</th>
<th>Coped with 3×3 Wiener Filter</th>
<th>Coped with 3×3 Median Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>2.3574</td>
<td>17.8191</td>
<td>19.6985</td>
</tr>
<tr>
<td>Salt &amp; Pepper</td>
<td>1.3885</td>
<td>8.3455</td>
<td>2.3900</td>
</tr>
<tr>
<td>Speckle</td>
<td>0.3063</td>
<td>6.4575</td>
<td>8.0411</td>
</tr>
<tr>
<td>Blur</td>
<td>6.5273</td>
<td>11.4667</td>
<td>11.2812</td>
</tr>
<tr>
<td>Blur with Gaussian</td>
<td>8.3611</td>
<td>22.0061</td>
<td>22.9649</td>
</tr>
<tr>
<td>Blur with Salt &amp; Pepper</td>
<td>7.9350</td>
<td>15.6571</td>
<td>11.3201</td>
</tr>
<tr>
<td>Blur with Speckle</td>
<td>6.6445</td>
<td>13.4095</td>
<td>14.1869</td>
</tr>
</tbody>
</table>